MUSIC/SP

Version 5

Release 1

Administrator's Reference

April 1998

Ninth Edition (April 1998)

This edition applies to Release 1 of Multi-User System for Interactive Computing / System Product (MUSIC/SP) Version 5, and to all releases of this product until otherwise indicated in new editions or Technical Newsletters. MUSIC/SP Version 5 is published and licensed by McGill Systems Inc.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to: MUSIC Product Group, McGill Systems Inc., 550 Sherbrooke St. West, Suite 1650, Montreal, Quebec, Canada H3A 1B9. Fax: (514) 398-4488.

Preface

About this Book

This publication describes how to operate and maintain MUSIC/SP. It contains detailed instructions for using the system utility programs provided with MUSIC/SP. It is directed towards hardware and software support personnel, operators, systems administrators, and systems programmers. This version corresponds to MUSIC/SP Version 5, Release 1.

For installation procedures and routine maintenance, refer to the *MUSIC/SP Administrator's Guide*. This guide describes the menu-driven system administrator facility.

How this Book is Organized

This publication is organized into the following five parts, appendixes, and an index.

I. *Planning for MUSIC/SP*.
Part I provides information about hardware requirements. It contains information about how to run MUSIC/SP under VM.

II. *Operating MUSIC/SP*.
This part includes three chapters providing information about loading the system, the system console, and routine maintenance.

III. *Customizing MUSIC/SP*.
This part describes the methods for customizing MUSIC/SP to suit your installation. Some of the topics include: System Reconfiguration, Terminal Configuration and tailoring, and Job Submission.

IV. *MUSIC/SP System Utilities*.
This part includes a description of the system utility programs available on MUSIC/SP.

V. *Internals of MUSIC/SP*.
This part describes the details about the internal workings of MUSIC/SP. Some of the topics include: Region Sizes, File System, Load Library, and System Programming.

MUSIC/SP Publications

The following is a list of all the current MUSIC/SP publications. These hardcopy publications can be ordered through the MUSIC Product Group. Online versions (softcopy) of the user publications can be accessed with the MUSIC/SP command called "MAN".

- *MUSIC/SP Administrator's Guide* (April 1996), describes how to install and operate MUSIC/SP.
- *MUSIC/SP Administrator's Reference* (April 1998), describes the internals of MUSIC/SP; utility programs and supervisory commands; gives detailed storage estimates; and documents console messages.

- *MUSIC/SP User's Reference Guide* (October 1997), describes how to use MUSIC/SP; its command language; terminal and batch set up; and job processing using the various language processors.

- *MUSIC/SP Guide for New Users* (April 1996), introduces new users to the use of MUSIC/SP via an IBM 3270-type workstation. It describes the FSI (Full Screen Interface) menu facility. New users learn how to use many programs on MUSIC/SP for such tasks as editing and running programs.

- *MUSIC/SP Office Applications Guide* (April 1996), describes the features of the TODO (Time, Office, and Documentation Organizer) facility. This includes the scheduling function, spell checking, and MUSIC/SCRIPT (text processing).

- *MUSIC/SP Mail and Conferencing Guide* (April 1996), describes electronic mail on MUSIC/SP. This includes Mail Profile, Mail Directory, using POP clients, and conferencing programs.

- *MUSIC/SP Internet Guide* (April 1996), describes the programs available on MUSIC/SP that provide communication between users through electronic conferencing and discussion lists. Emphasis is placed on access to the Internet with programs such as TELNET (logging on other computers), FTP (File Transfer Protocol), WEB (World-Wide Web), RN (Newsreader), and GOPHER (document search and retrieval protocol).

  Note: A separate guide devoted to the WEB on MUSIC can be found at the following site: http://musicm.mcgill.ca

- *MUSIC/SP Campus-Wide Information Systems* (*CWIS*) *Guide* (April 1996), describes how to create and maintain a Campus-Wide Information System, Help facility, or Classified Ads facility; how to do full-text searching; and how to provide gopher access. MUSIC/SP's resources are used to provide online distribution of information to a wide audience.

- *MUSIC/SP Teacher's Guide* (April 1996), describes various MUSIC/SP facilities related to the academic environment. Emphasis is placed on communication between teacher and student and easy methods for learning how to use MUSIC applications.

- *MUSIC/SP Client/Server* (*MCS*) *Booklet* (April 1996) provides an overview of MCS. Full documentation is available on the MCS diskette.

- *MUSIC/SP Personal Computer Workstation User's Guide* (May 1994), describes the components of the Personal Computer Workstation (PCWS). It is intended for the novice or experienced user of a personal computer, who wishes to connect to MUSIC/SP or another host system. Note that documentation for *PCWS for Windows* is available on the PCWS diskette.

- *MUSIC/SP World-Wide Web Support*, (June 1997), describes how to produce Web documents ready to display on your Web browser or MUSIC's Web browser.

Trademarks

Part I. Planning

Chapter 1. Introduction

Overview

The basic information required to install and administer the MUSIC/SP system is contained in the *MUSIC/SP Administrator's Guide*. The ADMIN program provides a menu driven approach to system configuration and maintenance but offers little explanation about the tasks that are being performed automatically as the result of the various menu selections. This may be of benefit in a lot of situations but on occasion a more detailed knowledge of the system is required. This manual contains reference material that is of interest to the administrator or systems programmer who wishes to obtain more detailed background information on MUSIC/SP.

MUSIC/SP often runs as a guest operating system under VM/SP or VM/XA. In this case the hardware requirements discussed below, are provided by VM in the form of a virtual machine defined in the VM directory. Under VM, MUSIC/SP can share devices such as printers, tape drives, disks, and terminals with other virtual machines. *Chapter 2 - Running MUSIC/SP under VM* discusses configuration and performance issues specific to running the system under VM.

If required, MUSIC/SP can run without VM. In this case MUSIC/SP controls the entire processor and all the associated devices.

Hardware Requirements

Processor

MUSIC/SP will run on any processor supported by VM. Specifically this ranges from the IBM 9370 up to the IBM 30xx processing complexes.

Storage

The storage required to run MUSIC/SP depends on the number of active terminals. Allowing 1 megabyte for every 32 active terminals will provide enough storage for good performance in the typical situation. Additional storage can be used to enhance performance by reducing paging and swapping loads and making highly used applications memory resident. The pre-configured system requires 2 megabytes.

Channels

A multiplexer with channel address of 0 is required. The system console and printer are accessed through this channel. Terminals may also be on channel 0. At least one selector or block multiplexer channel is required to access the tape and disk units. Performance gains can be achieved if more than one channel is available for disks. MUSIC/SP uses paging and swapping techniques to manage user tasks and the I/O overhead involved in these functions can be reduced if it is spread over a number of channels.

Terminals are normally accessed through channel 0. If this is not possible, then they can be attached to channels 1 through 15 provided no tapes or disks are also on that channel.

Disk

MUSIC/SP requires at least 2 disk volumes, totaling at least 320 megabytes of space for a typical configuration. Any of the following disk devices can be used: 3310, 3330, 3340, 3350, 3370, 3375, 3380, 3390, 9332, 9335, 9345 or 0671. Logical volumes can be either dedicated devices or minidisks under VM. The pre-configured system can only be installed on 3350, 3370, 3380, 3390, 9332, 9335, 9345, or 0671 disks.

Tapes

One 2400, 3400, 9346, 9347, 9348, 8809 series tape drive, or 3480 cartridge tape drive is required for system generation and efficient system backup.

Tape Drives on IBM 9371 Processor

The following notes applies to tape drives on the IBM 9371 processor only. Tape drives on other models of the 9370 processor family do not have this requirement.

When running MUSIC/SP on the 9371, special care must be used when defining the address associated with the tape device. This is true for the 9346 (1/4 in) or the 9348 (1600/6250). The address selected MUST be on a unique channel. No other devices can exist on the channel with the tape unit. If you followed the sample configurations in this book, then channel 4 can be used.

When running under VM, this can be accomplished by specify the VIRTUAL address on the ATTACH or DEDICATE statement.

When running NATIVE (without VM), the physical address of the tape unit is specified via the Configuration Setup screen. Refer to the *9371 I/O Installation and Configuration Guide* (SA24-4220).

Unit Record Devices

Card Reader — A 2540, 2501 or 3505 card reader is supported. It is not required unless the installation requires the ability to read cards.

Card Punch — A 2540, 3525 or 1442 Mod II card punch is supported. It is not required unless the installation requires the ability to punch cards.

Printer — A 1403, 3211, 3203, 3262, 3289 or 5203 is required for the printer. An upper and lower case print chain (such as TN) is desirable when you wish to print MUSIC/SCRIPT documents. In addition, printers supported by the 3270 subsystem and asynchronous ASCII printers can be attached directly to MUSIC/SP for additional printer support. Under VM, MUSIC/SP has access to the VM system printers and any network printers run by RSCS.

Console — The system requires a printer-keyboard console such as a 1052, 3210, 3215 or the equivalent functions on an emulated console.

Transmission Control Units

Asynchronous terminals such as the IBM 2741, ASCII, and Personal Computers programmed to emulate asynchronous terminals, require a transmission control unit. 3270 terminals and terminals connected via a protocol convertor such as the 7171, do not require one. MUSIC/SP can use any of the following transmission control units: 2701, 2702, 2703, or 37xx (in emulator mode), Integrated Communications Adapter (ICA). The transmission control units such as the 2702, 2703, 3705 (or equivalent) must be plugged

correctly in order that IBM-type terminals function correctly under MUSIC/SP.

On 270x series, the ports to be used by IBM terminals must be specified as 2741 break type. MUSIC/SP supports 1050 terminals on the same type of line as the 2741 type.

On 37xx series, generate the lines to be used for IBM terminals as '1050' with 'UNITXC=NO'. Note that with older versions of the emulator program, you must specify 'UNITXC=NO' on each LINE macro as the specification on the GROUP macro was not sufficient. If no 1050s are to be used with the 37xx, then you may specify the terminal type as 2741 instead of 1050.

On 37xx series, specify the option 'CHECK=DCD' on the GROUP macro to ensure MUSIC/SP will be informed of unusual line disconnects.

ASCII (TTY) type lines should be specified such that the RETURN key is an end of line signal. The *transmit on* (X-ON) and *transmit off* (X-OFF) should also be line end characters. For example, when generating a 3705 emulator program, specify CHAREC=(XONOFF,B1) on the GROUP macro. If your control unit is NOT set to handle RETURN as the end of line character, then you must specify the RNA option on the MUSIC/SP TTY device specification card during MUSIC nucleus generation step (NUCGEN).

4331/4361 Communications Adapter Notes

Up to eight start/stop (s/s) terminals may be attached to the 4331 and 4361 processor's Communication Adapter (CA). IBM hardware RPQ 7S0276 is required to support ASCII s/s terminals on the CA. The following configuration options must be specified.

- PERM REQUEST TO SEND - YES
- READ INTERRUPT - YES
- WRITE INTERRUPT - YES
- UNIT EXCEPT SUPPRESS - YES

Options not mentioned here are terminal dependent, and should be set as required.

Disk Device Recommendations

When comparing disk devices it is particularly important to compare their average access time as well as their transfer rate. The device's capacity usually depends on the block size used. Just comparing their maximum capacity can be very misleading. Please refer to the device tables in the chapter "Direct Access Storage". The tables show device capacities for blocksizes 512 (used for files) and 4096 (used for swap and page data sets).

The following contains notes on each of the devices.

3380 This device has superior data transfer rate and access times. Its transfer time is particularly useful to handle the block-paging and swapping functions of MUSIC/SP. The Extended Capability Models of the 3380 line can have double the capacity of the earlier Standard Models. These higher capacity models, called the AE4 and BE4, are identified as the 3380 E in the above table.

Be careful when estimating its true data capacity on MUSIC/SP. Notice that the capacity of a 3380 device with 512-byte records is about one half that of the published maximum capacity. Make sure that the processing unit's channels are fast enough to handle the device.

3370 Although not as fast as a 3380, the 3370 is a good general purpose disk. It can be attached to the 4331 and 4361 processors. The 3370 is a FBA-type device. Although it is supported by MUSIC/SP and VM it may not be supported by the other operating systems that you may want to run under VM.

3375 This device is very similar to the 3370 except that it uses the CKD format rather than the FBA format the 3370 does.

3310 The 3310 is supported, but the 3370 offers much better performance.

3330 The 3330 is supported, but the 3370 offers much better performance.

3340 3340's are supported, but the other devices offer better performance characteristics. Avoid attaching them to the 4331 or 4361 processing unit through the built-in adapter as this may cause unexpected performance problems.

3350 3350's are supported but the 3370 or 3375 devices have a better performance.

933x These disks are part of the 9370 system and have similar characteristics as the 3370.

0671 These are the FBA type disks that can attach to the 9371 processor.

3380 Model AA4 Device Addressing Notes

This topic describes notes relating to the 3380 Model AA4. This discussion does not apply to the Extended Capability 3380 models.

On the 3380 model AA with two controllers, either controller can access any of the four internal paths to the drives. This allows each access mechanism to be accessible from either controller and any two of the internal paths can transfer data simultaneously. Normally, one would want the controllers to be on separate channels, so that the processing unit can access any two of the access mechanisms simultaneously so long as they are on different internal paths.

All MUSIC/SP disk I/O is queued at the channel level. That is, if there is an active request on a channel, MUSIC/SP will wait for it to complete, before doing anything else on that channel. Also MUSIC/SP will access a disk via only one address. Thus for each access mechanism, only one address may be specified to MUSIC/SP. Access mechanisms that are to be accessed simultaneously must have the addresses chosen such that they are on different channels.

To set this up one must know which access mechanisms are on different internal paths. Access mechanisms which share an internal path should be defined to MUSIC/SP as being on the same channel.

Access Mech	Path
0,1,8,9	0
2,3,10,11	1
4,5,12,13	2
6,7,14,15	3

Example

Suppose a 3380 model AA and B (access mechanisms 0-7) is connected to storage directors (3880s) 14x and 24x. Each access mechanism can be accessed by either the 14x or 24x address but you can only tell MUSIC/SP about one of them. Suppose that four disks are required on each channel, then the following addresses could be defined to MUSIC/SP.

- 140,141,142,143 for channel 1
- 244,245,246,247 for channel 2

This is only one of many ways the addresses could be assigned. A big mistake would be to define devices at 140 and 241. MUSIC/SP would attempt to access them as if they were on different channels whereas they actually share a common path in the 3380.

For more detailed information see *IBM 3380 Direct Access Storage Description and User's Guide* (GA26-1664).

FE Service Aids

Reporting Disk and Processing Unit Errors

MUSIC offers both the installation and the IBM Field Engineer (FE) information to determine whether system errors were caused by the software or hardware.

Serious hardware problem, such as uncorrectable equipment checks or permanent data checks are logged out on the console immediately and the console alarm is sounded. The error messages are documented in *Appendix A. Console Messages and Wait States Codes* of this manual.

MUSIC will also write an OBR record for disk and tape errors to VM's LOGREC file. These records may be printed using the VM EREP program.

The internal statistics log of the 3330 disk drive may be dumped with a utility program called BUFLOG. The use of this program is described later in this manual.

The hardware makes a distinction between *hard* and *soft* machine checks. Upon encountering soft machine checks, the system attempts to recover from the error and continue processing. When encountering hard machine checks or when soft machine checks have reached a maximum threshold value, the system will shut down immediately. The system administrator has the option of running the EREP program or reloading MUSIC. The last machine and channel check logs are maintained in main storage.

On a number of processing unit models, the hardware maintains the last checks internally, so it may not be necessary to immediately run the EREP program. Refer to the appropriate S/370 reference manual pertaining to processing unit operations.

E-Mail Discussion Lists

The LISTSERV software on BITNET provides a forum for you to discuss topics of interest with other computer users. MUSIC users can join MUSIC-L which discusses general MUSIC issues. A system administrator can join the list called MUG. The MUG list is a closed list and your subscription must be confirmed by the list owner (security issues related to MUSIC are sometimes discussed).

To join a list, send e-mail to the address of the list and include the SUBSCRIBE command as the text of the message. For example, to join the MUSIC-L list send mail to "LISTSERV@MARIST.BITNET" and include one line of text only, as follows:

```
SUBSCRIBE MUSIC-L John Smith
```

You will receive confirmation by e-mail about being added to the list.

Note: The MAIL facility offers a selection on the menu called "List Manager". This item provides an easy method for subscribing to discussion lists.

Chapter 2. Running MUSIC/SP Under VM

Overview - Running MUSIC/SP Under VM

This chapter contains information about the configuration and performance of MUSIC/SP running under VM.

(Unless otherwise mentioned, the term VM in this manual will be used to refer to VM/ESA and its predecessor products VM/SP, VM/XA, and VM/370.)

Although MUSIC/SP is running as a virtual machine under VM, it is important to remember that the single MUSIC/SP machine is supporting a large number of interactive users. MUSIC/SP is a guest operation system, not just another user. As such, the VM performance options, that apply to guest operating systems, should be applied to guarantee optimum performance to your MUSIC/SP users. This chapter outlines some of these key performance options and also gives specific details as to how MUSIC/SP should be configured as a virtual machine.

Configuration Notes

You must specify the REALTIMER, ECMODE, BMX, and VCUNOSHR options in the VM directory entry for the MUSIC virtual machine.

The terminals may be defined in the directory either by the DEDICATE or SPECIAL specification statements. When DEDICATE is used those terminals are permanently attached to the MUSIC/SP virtual machine and cannot access the other facilities of VM. When SPECIAL is given, each user must enter a DIAL MUSIC (or equivalent) command to be connected with MUSIC. Users who use 3767s and 2741s through the DIAL command must press carriage control immediately after VM has acknowledged that the line is dialed to MUSIC.

WARNING: Do not mix 3270s, asynchronous terminals, or unit record devices on the same virtual subchannel. The subchannel is indicated by the middle digit of the device address. For example, if a 3270 is defined on address 0E2 make sure that no other non-3270 device has an address in the range from 0E0 to 0EF. Failure to observe this rule will result in severe performance problems.

MUSIC and VM Performance Considerations

MUSIC can take advantage of any of the VM hardware assist options available on your processing unit. These assists can significantly reduce the VM overhead for MUSIC.

Always configure production MUSIC systems under VM in such a way that VM never has to page any part of MUSIC. This can be done by running MUSIC in a V=R region or by issuing the VM LOCK command to lock all of MUSIC's pages in storage. (MUSIC will work in a paged configuration, but the end users will typically complain of erratic response. This is caused by VM freezing the entire MUSIC system any time any user gets any page exception.)

The use of additional main storage for the MUSIC Fixed Linkpack Area (FLPA) will improve MUSIC's performance only if the same amount of real main storage is added to MUSIC. For example, suppose your

MUSIC system is configured as a 2MB virtual machine with all the pages locked in main storage and you want to add 2MB. Do not be tempted to run MUSIC as a 4MB virtual machine with only 2MB locked. In this case, you must run with all 4MB of MUSIC locked in storage.

Run MUSIC as a V=R machine for optimal performance. This will reduce the amount of processor time VM will need to support MUSIC, and as a result, will be able to spend more time servicing other machines. The V=R option does not give MUSIC any special performance over other virtual machines. During system initialization, when MUSIC detects that it is in a V=R region, it will automatically issue a /CP SET NOTRAN ON command so that CCW translation is bypassed.

Do *not* use the DEDICATED CHANNEL option under VM as it will increase the VM overhead. However, it is desirable for MUSIC to have exclusive use of the real channels that are used to access its disks. The use of a second or third real channel for swapping may improve MUSIC's performance even if this channel is shared with other virtual machines.

Some performance improvement will occur if you define each MUSIC disk device as being on a separate channel even when they are not. For example, using the virtual addresses of 130, 230, and 330 is preferable to defining them as 130, 131, 132. This is because MUSIC queries disk requests at a channel and VM may be able to queue the request at a device level. Note, however, that you cannot use this technique to give the appearance of multiple channels for swapping and paging when multiple channels do not really exist.

VM Commands that Effect Performance

A facility exists to issue commands to VM each time MUSIC is loaded. Refer to the topic "Modifying the System Catalog" in *Chapter 6 - System Reconfiguration* for further details. A number of commands can be issued at this time to enhance MUSIC performance.

The SET RUN ON command should always be issued to guarantee that the MUSIC virtual machine keeps running when it enters CP mode or the console screen fills up.

The use of the SET FAVORED and PRIORITY commands of VM will give MUSIC top service under VM. The SET FAVORED 99 option can be used to give MUSIC top service even though MUSIC may only require a fraction of the total processing unit time.

The SET STBYPASS command cannot be used with MUSIC/SP.

Detecting the Environment

MUSIC automatically detects the environment it is running in as explained at the beginning of this chapter. Certain flags are set at location 240 (hex) in MUSIC to reflect the environment it sees at IPL time.

Defining the Spooled Reader

To simulate the operation of a real card reader, it is necessary to use the VM command SPOOL RDR CONT for MUSIC's batch reader. This CONT option can be set up as one of automatic commands issued at MUSIC IPL time. See the topic "Modifying the System Catalog" in *Chapter 6 - System Reconfiguration* for more information.

The virtual reader (class M and A) for the VMREADX program must *not* have the CONT option. The batch reader must not be spooled class * (i.e. to read all classes) if VMREADX is used. Normally MUSIC's batch reader is spooled to read class J files.

Defining the Spooled Printer

MUSIC will normally automatically skip to a new page when the printer senses a channel 12 punch on the carriage tape. This function will normally not be simulated when the printer is spooled through VM. However, the VM spooled printer can be set up to accomplish this operation when the printer is defined to VM as a 3211. The virtual FCB (forms control buffer) must be loaded with an appropriate LOADVFCB command to VM. For example: "DEF 3211 E", "LOADVFCB E FCB MUS1". Refer to the appropriate VM publication. It is advisable to add the FCB specification to VM's nucleus *before* attempting to generate MUSIC under VM. An example of an FCB specification suitable for MUSIC is:

```
FCB MUS1,6,66,(4,1,64,12),1
```

Do not spool the virtual printer or punch with the CONT (continuous) option, since this would cause the automatic CLOSE command to be ignored when it is issued by MUSIC at the end of each batch job.

Defining Extra Unit Record Devices

Various MUSIC utility programs require that extra unit record devices be defined in the MUSIC directory. The devices need not be defined if the programs will not be used. SUBMIT, GSUB, OLDSUB, VMSUBM, and AUTOSUB **require** virtual punches on addresses 011 through 013. If more punches are defined, the programs should be configured to take advantage of them. AUTOPR and VMPRINT require a virtual printer. The default address for this is 017, but this can be changed if the programs are reconfigured. VMREADX requires a class M virtual reader on 018 and class A on 019. Consult *Chapter 8 - Job Submission and Retrieval Programs* for further information on the programs mentioned above.

Using Minidisks for MUSIC/SP Disk Volumes

VM *minidisks* can be used for MUSIC disk volumes. However, the installation should be aware of the following: (a) these pseudo devices cannot be accessed without VM, and (b) additional CPU overhead will be incurred as VM has to perform address translation on them. In most cases the CPU overhead should not exceed a few percent and so it may be acceptable to use minidisks in cases where your CPU is not loaded. Care should be taken when they are formatted to ensure that the number of cylinders on the minidisk is correctly specified to the MUSIC Disk Format Utility (FORMAT).

Minidisks for MUSIC do not have to start at CYL 0 on the real pack as is required by some other operating systems.

Minidisks can be used even if you are running V=R. VM will normally go through CCW translation for I/O to these devices. However, I/O to full pack minidisks will not go through CCW translation when accessed in full read/write mode. (*Full pack* means 808 cylinders for 3330-11, 404 for 3330-1, 698 for 3340 M70, 349 for 3340 M35, 555 for 3350, 959 for 3375, 885 for 3380, and 203 for 2314. It means 126016 blocks for 3310 and 558000 blocks for 3370 devices. Consult the coding the VM module DMKVSC near label TSTNDED for details.)

MUSIC Console Under VM

The console alarm used by MUSIC to alert the operator to unusual conditions under MUSIC will normally never sound under VM. The VM operating system will normally type the message `RRRR....RING....GGGG` instead. If this proves to be operationally undesirable, it may be changed by one of two techniques. One is to dedicate the real machine console to MUSIC and the other is to modify VM's supervisor to sound the real alarm if it is the MUSIC machine.

The operation of MUSIC's virtual console on a 3270 is not recommended for production operation. This is due to the fact the VM's 1052 simulation on a 3270 will cause the console to stop printing once the screen is full. This screen full condition will be cleared only after about 1 minute has elapsed. If the screen contains any message which sounded the alarm, then this automatic clear will not be done by VM. When the screen is full, the console will not be available to MUSIC to print any more messages.

If MUSIC discovers that it has a high-priority message to print (such as DROPPED, or I/O error etc), the system may loop until the message can be displayed. This situation can be avoided by running MUSIC in a disconnected state and having the console messages sent to the VM operator's console using the secondary operator facility. To set this up specify OPERATOR as the secondary operator parameter on the CONSOLE statement in the VM directory entry for MUSIC.

Once MUSIC is initialized it can be disconnected (/CP DISC command), and all console messages that would have gone to MUSIC's console will be sent to OPERATOR. Console commands can be sent to MUSIC while it is in this state using the SEND command.

You can also set it up so that MUSIC can be entirely controlled from OPERATOR and you never have to physically log on to MUSIC at all. Include an IPL statement in MUSIC's directory that will load from MUSIC's SYSRES pack when logged on. MUSIC can now be started from OPERATOR using the AUTOLOG command. If the data parameter of the AUTOLOG command is =AUTO, MUSIC will execute a quick startup and require no further operator intervention to get going.

The MUSIC CONSOLE utility can be used to view console messages from a priviledged userid.

Miscellaneous Notes

It will simplify the generation and operation of MUSIC under VM if the MUSIC virtual machine were given VM operator privileges.

The MUSIC IPLable main storage dump program will operate under VM, however certain parts of the resulting dump will not be valid. They are the CAW and a 4K area in the middle of the virtual storage used by the VM IPL simulator.

Sample VM Directory Entry for MUSIC

The following sample defines a virtual machine called *MUSIC* which, by default, has 4 megabytes (4096K) of main storage. All operator privileges are given to MUSIC here, though only the G class is mandatory. Even with the full set of VM privileges, the MUSIC terminal user will not be able to issue privileged VM commands. Class C or E is required to take advantage of all the MUSIC system's performance facilities under VM.

The 4MB of storage shown in this example will have to be increased to support many simultaneous users. Users of VM/ESA ESA option can specify storage sizes up to 64MB for virtual machines. VM/ESA 370 option and previous versions of VM were usually limited to 16MB as a maximum size.

Note that the console is defined as a 3215 device. It must not be defined as a 3270 (although a 3270-type terminal can be used). OPERATOR, the default VM userid of the system operator, is specified as the secondary operator. This allows MUSIC to be run disconnected and have the console messages sent to the operator's console.

The batch card reader, printer and punch are defined in this example as being spooled. This option can be dynamically changed to real devices as required. Consult the VM documentation for the procedure to do this.

The printer is defined as a 1403 in the example. It is best to define it as a 3211 as mentioned under the heading "Spooled Printer" above. Before defining it as a 3211, make sure that you have the VFCB defined. The definition as a 1403 will work sufficiently well during MUSIC installation.

The OPTION statement requires the REALTIMER, ECMODE and BMX specifications. The option VCUNOSHR is available with fix VM24061 to VM/SP Release 4. VCUNOSHR will improve MUSIC's performance and should be used if available. The VIRT=REAL option will improve performance as described in the above text. The DEDICATE statements are used here to attach the disk packs and teleprocessing lines for the exclusive use by MUSIC. The SPECIAL statement defines an address available for a 3270 to be connected to when the user dials MUSIC. The spooled punch devices 11 through 13 are those for the MUSIC SUBMIT and VMSUBM interfaces to use. The spooled printer at address 17 is for use by the VMPRINT and AUTOPR utilities. The class M reader at address 18 is used by the VMREADX program.

```
USER MUSIC MUSIC 4M 16M ABCDEG 5
OPTIONS REALTIMER ECMODE BMX VCUNOSHR
IPL 120
CONSOLE 01F 3215 T OPERATOR
DEDICATE 120 C00
DEDICATE 220 C01
SPOOL 00C 2540 READER J
SPOOL 00D 2540 PUNCH
SPOOL 00E 1403
SPOOL 011 2540 PUNCH
SPOOL 012 2540 PUNCH
SPOOL 013 2540 PUNCH
SPOOL 017 1403
SPOOL 018 2540 READER M
SPOOL 019 2540 READER A
SPECIAL 520 3270
SPECIAL 521 3270
SPECIAL 522 3270
SPECIAL 523 3270
etc...
```

Part II. Operating MUSIC/SP

Chapter 3. Loading the System

Initial Program Load (IPL)

The system initialization procedures require the mounting of all necessary disk packs. When the disks are properly set up, the operator initializes MUSIC by performing the IPL operation. Under VM, this is done by simply entering the command "IPL xxx" where *xxx* is the address of the MUSIC system residence disk pack normally called MUSICX. The system responds with various messages and questions as shown below.

No Messages?

If no messages appear, it could be caused by the MUSIC system not being able to communicate via the generated console address. If this is the case, you can get MUSIC to recognize the new address by depressing the REQUEST key on the console. (VM simulates this with the PA1 key on a 3270.)

Another reason for no messages is that the display console is working in 3270 mode. This is not supported by MUSIC. (VM can simulate the 3215 mode on a 3270, which is the usual mode of a virtual console under VM.)

Sample IPL of MUSIC

The following shows a sample IPL of MUSIC. The arrows (-->) identify the lines entered by the operator.

```
1--->ipl 140

     M066 MUSIC/SP, Level=xxxxxxxxxxx
     M077 Enter operator id or special options or HELP
2--->rwm
     12:51 PM MON MAR 04, 1985 Type 'OK' or new values
3--->ok
     12:51 M097 The following volumes are permanent:
     12:51 M105 MUSICX on unit 140 SYS
     12:51 M070 No disk drives are available for mounts.
     12:51 M139 Storage size  2048K. Pageable storage 1456K
     12:51 M140 MAXMPL=  6, MAXRRS= 232K, Num RCBs=   23
     12:51 M092 System initialization completing.
     12:51 M300 BATCH IDLE
```

Item 1 above is the operator entering the IPL command to VM.
Item 2 operator identification. (The system does not check this identification.)
Item 3 verification of the time and date.

A full explanation of the messages printed above can be obtained by consulting *Appendix A. Console Messages and Wait State Codes* in this manual.

Special Options

The following special options can be entered at the time the system is expecting the operator's initials. After taking note of the option, the system will prompt again for the operator's initials or for more special options. (Entering a blank line or =AUTO or a line that does not start with the "=" character allows the system IPL to continue beyond this point.)

(blank) Fast start up of the system. The prompt for the time and date checking will not be issued. The time and date information is obtained directly from VM and will be correct unless the values were entered incorrectly when VM was started.

=AUTO Fast start up of the system. This is the same as entering a blank line.

=NOTERM Means bring up the system but do not allow terminal users to sign on yet. Batch jobs can run. This NOTERM option can be used when the operator wants to run a MUSIC batch job when no terminals are on the system. It can also be used to allow just a few terminals on. This would be done by issuing a /ADD command for that terminal.

The operator can later allow all terminals to sign on by issuing the /ADD ALL console command.

=EDIT Specifies that the operator wants to make temporary changes in the system catalog. The use of this option is explained in the topic "Modifying the System Catalog" in *Chapter 6 - System Reconfiguration*.

=CATxxxx Instructs the system to use the alternate system catalog name of SYS1.MUSIC.CATxxxx instead of the default SYS1.MUSIC.CATALOG. The system catalog lists the names of all the system data sets, link pack area members, etc.

=RESET Erases all the accounting records on the system. This is normally only done when initializing the accounting data set for the first time.

=CONFIG Allows the operator to temporarily change the I/O configuration. This could be used if you wanted to IPL MUSIC on a backup processing unit. The MUSIC/SP starter system always requires the reconfiguration step as it has no I/O configuration defined. The use of this option is fully explained in the topic "Reconfiguring Temporarily at IPL Time" later in this chapter.

=LOADPRT Informs the system that MUSIC's printer is to be loaded with the appropriate FCB and UCB images. This is normally only used in the case that the printer is being directly controlled by MUSIC and is not being spooled through VM. The use of this option is fully explained in the topic "Initializing the Printer" later in this chapter.

=RDROFF Prevents MUSIC batch from selecting jobs from the internal reader. This is equivalent to the console command /RDR OFF.

Specifying Time and Date

MUSIC obtains the time and date information from the hardware's time of day clock and also from VM. It is unlikely that VM will have incorrect information unless these values were incorrectly entered when VM was started. In this case, you might want to re-IPL VM.

It is possible to change the time and date information at MUSIC IPL time. This new information will be used only within MUSIC and will not affect other virtual machines running under VM. To change the information, enter a non-blank operator id. The system will then issue a prompt showing the current time and date values. The operator then has to put in the correct information using exactly the same format. The time and date must be abbreviated by their first three letters. The following example shows how to change the time to 5pm.

```
        03:46 PM MON MAR 04, 1985 TYPE 'OK' OR NEW VALUES
----> 05:00
```

The time is normally entered using a 12-hour clock. To enter a time between 12:00 and 1:00 o'clock, either a.m. or p.m., enter it as 12:xx, not as 00:xx. Enter 12:00 AM for 12 noon but enter 12:01 PM for one minute later. (You may also use the 24 hour notation if you specify HR instead of AM or PM.)

When a new value is entered, checks are performed to ensure that all values are valid and that the day of the week entered coincides with the date.

Initializing the Printer

If the printer can only print upper case characters then the printer should be initialized with the FOLD option. (The FOLD option will print lower case characters as upper case characters.)

The printer should have the correct carriage tape or FCB (Forms Control Buffer) and the UCS (Universal Character Set) buffer image must match the print train in use. If the printer is spooled through VM, refer to *Chapter 2 - Running MUSIC/SP under VM* for details.

Printers, such as the 1403, which have a carriage tape, should have the tape punched with channel 1 and channel 12 punches to identify the top and bottom of the page respectively. When MUSIC detects a channel 12 punch it will automatically do a skip to channel 1 causing a *skip to next page* operation. Printers, such as the 3211, must have an appropriate FCB image loaded to accomplish the same function. This FCB image, along with the UCS image, which defines the character set, may be loaded during system initialization by operator commands, or while the system is running using the SETBUF utility program which is described later in this manual. The following details show how to load the appropriate buffer images during system initialization.

The operator must inform the system of the need to load the printer buffers by specifying the =LOADPRT option when prompted for the *Operator ID*. Subsequently MUSIC will prompt for further information as follows.

a) M135 Fold option on buffer load - yes/no (Default no)

- respond yes or no as required. The fold option will cause lower case characters to be printed as upper case.

b) M125 Enter FCB name

- this will not appear for printers which have a carriage tape.
- Valid responses are:

```
MUS6 - 6 lines per inch
MUS8 - 8 lines per inch
```

c) M130 Enter UCS name

- The UCS defines the printer's character set and is dependent on the type of printer and character chain used. Valid responses are listed below.

```
UCS BUFFERS IMAGES AVAILABLE ARE:

3203/1403  PRINTERS

AN    NORMAL AN ARRANGEMENT
HN    NORMAL HN ARRANGEMENT
PCAN  PREFERRED SET, AN
PCHN  PREFERRED SET, HN
QN    PL/I - 60 GRAPHICS
QNC   PL/I - 60 GRAPHICS
RN    FORTRAN, COBOL COMMERCIAL
YN    HIGH SPEED ALPHAMERIC
TN    TEXT PRINTING - 120 GRAPHICS
PN    PL/I PRINTING 60 GRAPHICS
SN    TEXT PRINTING - 84 GRAPHICS


3211 PRINTERS

A11   STANDARD COMMERCIAL
H11   STANDARD SCIENTIFIC
G11   ASCII
P11   PL/I
T11   TEXT PRINTING


3289 PRINTERS

F48   48 CHARACTER GRAPHICS
F64   64 CHARACTER GRAPHICS
F96   96 CHARACTER GRAPHICS
F127  127 CHARACTER GRAPHICS


3262 PRINTER

P48   48 CHARACTER EBCDIC
P52   52 CHARACTER AUSTRIA/GERMANY
P64   64/72 CHARACTER EBCDIC
P63   PREFERRED 64 CHARACTER EBCDIC
P96   96 CHARACTER EBCDIC
P116  116 CHARACTER FRENCH CANADIAN
P128  128 CHARACTER KATAKANA
```

Reconfiguring Temporarily at IPL Time

MUSIC's device addresses and terminal configuration can be respecified at IPL time. This feature is primarily used when the MUSIC starter system is used. Reconfiguration can also be done at a later date should

you require to IPL MUSIC on another processing unit.

The reconfiguration is only in effect for the current IPL and completely replaces the configuration specified at nucleus generation time. Permanent reconfiguration is done by using the nucleus generation procedure (NUCGEN program).

If only the console address needs changing then this reconfiguration step is not required. Simply press the console REQUEST key after the IPL is done.

The reconfiguration process is started by the operator specifying the special operator ID of =CONFIG. MUSIC will then prompt the operator for the required information. If you have no printer, reader, or punch, then enter --- for its address.

A typical dialogue is shown below. The lines entered by the operator are identified in this example by a -> character to the left of the line. Enter a blank line to end the disk/tape configuration section. The TERM control option is no longer used; enter a blank line when prompted.

```
   M066 MUSIC/SP, Level=xxxx
   M077 Enter operator id or special options or HELP
-> =config

   M115 Temporary system I/O reconfiguration. Enter all addresses
                                                 in form 'cuu'
   Enter console address
-> 01f
   Enter printer address or --- if none
-> 00e
   Enter reader address or --- if none
-> 00c
   Enter punch address or --- if none
-> 00d

   Disk/tape configuration:  Enter one device per line
   in format 'tttt-cuu' where 'tttt' is one of the following
   2314
   3340
   3330
   3350   (Native mode)
   2305   for 2305 Model 1
   2306   for 2305 Model 2
   F512   for 3310 or 3370 or 933x FBA device
   3375
   3380
   7TRK   for 7-track tape drive
   9TRK   for 9-track tape drive
   8809   8809 tape drive

-> 3380-247
-> 3380-143
-> 9trk-180
->

   Terminal configuration
   Enter terminal control option as 'xxxx'
   (Usual response will be a blank line
   See Operations Manual 'TERM' configuration card for full details.)
```

```
->

   Enter terminal info in format 'tttt-cuu'
   Where 'tttt' is one of the following terminal types
   Enter a blank line to terminate terminal configuration
   2741
   3767
   1050
   TELE  for tty-type terminals
   3270

-> tele-030
-> tele-031
-> 3270-0f0
-> 3270-0f1
->
   Configuration complete
   M077 Enter operator id or special options or HELP
-> de
```

Editing the System Catalog

During normal system initialization, the operator is given the opportunity to modify the MUSIC system catalog. If any errors are found within the catalog, the system will automatically invoke the catalog editing facility. The edited catalog is in effect for the current IPL of MUSIC only. Permanent changes to the catalog are made using the EDTCAT utility program described in *Chapter 17 - System Utility Programs*.

The catalog modification is carried out in several steps.

1. If the catalog is to be edited the operator must respond with =EDIT when prompted for the operator ID. This will cause the catalog editing procedure to be invoked later in the system initialization process.

2. The operator is asked which catalog cards are to be displayed. A blank line entered at this time will cause the editor to advance to the next step. A response of ALL will cause the entire catalog to be displayed. The operator may display a single entry of the catalog by entering the 8 character label field (cols 1-8) of the required card. This step continues until a blank line is entered.

3. This step allows the deletion of specific cards identified by their 8 character label field (cols 1-8). A blank line ends this step.

4. Step four allows the addition of lines to the catalog. Refer to *Chapter 19 - Direct Access Storage* for the format of these entries. A blank line is entered to end this step.

5. At this time the operator may repeat the whole procedure if more changes are to be made or end the editing function by entering a blank line.

Shutting Down MUSIC/SP

A specific procedure should be followed when operation of MUSIC is to be terminated for any reason, to ensure that: all records and files on the system disk packs are properly saved for the next run; all system

maintenance functions in progress are properly terminated; and users are notified of the shutdown.

- The /MESSAGE ALL command (with no other message text) can be used to issue a shutdown warning message. (See *Chapter 4 - The System Console* for full details.)

- Wait for the appropriate length of time given in the above warning message (e.g. 5 minutes). This lets users finish their work and sign off.

- Check that there is no batch job running, and no /PAUSE or tape mount requests pending. To do this, use the operator commands "/go" and "/batch".

- Shut down some BTRM sessions, specifically those for RDMAILER and SYSLG (the System Log Message Server). This is to avoid losing any log records. To do this you must know the TCB numbers of these BTRMs. You can get them by entering the commands "enqtab rdmail" and "enqtab syslog" in a terminal session. (Normally these TCB numbers will remain constant until you change MUSIC's terminal configuration.) To shut down the BTRMs, enter the operator commands "/reply n1 stop" and "/reply n2 stop", where n1 is the TCB number of RDMAILER and n2 is the TCB number of SYSLG. If there is more than 1 RDMAILER, issue the command for each. You should stop RDMAILER before SYSLG. Wait for the message "Sys Log Server Btrm is shut down" to appear.

- As soon as possible after the above step, enter the operator command "/stop". This shuts down MUSIC.

The system responds with:

```
ACCOUNTING FILE CLOSED. MUSIC IS SHUT DOWN
```

VM will often follow this message with one similar to the following:

```
CP ENTERED; DISABLED WAIT PSW xxxxxxxxxx
```

This VM message can be ignored.

Terminal Processing

The console operator normally need not be concerned with what users are doing at their terminals. In a few instances, however, the operator receives messages concerning individual remote terminals.

If an equipment malfunction occurs at a terminal, the system informs the operator by typing an I/O error message on the console and cancelling the terminal session. The line will normally be automatically re-enabled to allow its use by another user. A similar message occurs if the specified device address does not exist or is in the *not operational* state. Some I/O errors can be caused by users disconnecting or powering off their terminals. Consult "Terminal I/O Messages" in Appendix A for specific details about these messages.

If too many errors occur on a particular terminal or if the terminal had been explicitly dropped by the operator, then the system issues the message:

```
M201 TERMINAL nnn DROPPED, ADDR=xxxx
```

The console alarm is also sounded. This message indicates that the system has made the terminal unavailable for use. The operator can make the terminal available again via the "/ADD nnn" command after verifying the reason for the occurrence of the message.

The following is a brief summary of console commands that effect terminal processing.

/ADD	make a terminal available after it has been dropped.
/CANCEL	cancel the program running on a terminal.
/DROP	drop a terminal.
/FIND	find a TCB number based on user code.
/HALT	terminate current I/O and drop a terminal.
/MESSAGE	send a message to a terminal.
/RESET	reset a terminal session.

Batch Processing

In addition to terminal processing, jobs may be brought on cards or submitted from a terminal to run at the central site. These jobs run concurrent with terminal processing and are called batch jobs. The card reader, punch, printer, and console as well as tape and disk are used instead of remote terminal equipment.

The operator can read card jobs by placing the cards in the card reader and then readying the unit.

Jobs submitted from a terminal will be processed according to the submitted job class. Apart from the lack of cards, these batch jobs proceed in a similar fashion to ones on cards. Refer to the topic "Batch Processing with VM" for information that is specific to VM and the topic "Batch Processing Using the Internal Reader" for other batch related information.

An attempt will be made to start the card reader after system initialization has been completed. At other times, the reader will start when it becomes ready. Output may be on either the card punch or printer (or both), depending on the user program. In addition, system messages, referring to the job, that require operator intervention are sent to the console.

Batch's priority is normally set to run jobs only during those instants when no terminal needs service. The "/CTL B-HI" console command can be used to alter this priority.

MUSIC processes only one batch job at a time. That is, after reading one complete job, it will completely finish executing it before reading the next one. Printed or punched output from a batch job may start while the job is still executing. Also, once the job ends execution, the system will commence reading the next job even though the last job has not finished printing or punching.

When a batch job requests tape work files, or nonpermanent user data set volumes, the operator is informed when and where to mount the required volumes. The operator must not ready the volume(s) until the system requests them. The operator should not ready a tape drive until the tape has been successfully loaded (threaded). When processing is complete, the user-designated external labels are typed on the console for proper identification of those tapes to be saved.

MUSIC's printer must have an appropriate carriage tape or FCB loaded. It must also be loaded with the correct UCS image. See the topic "Initializing the Printer" in this chapter for details.

Punched output is selected into pocket P2 of the IBM 2540 Card Read Punch. If a punch check occurs, the card in error is selected into pocket P1. To continue processing, the punch feed must be emptied and the punch made ready. All cards in pocket P1 should be thrown away. A separator card is punched in front of each deck produced. This card is similar to the /ID card submitted with the job except it has the last eight columns punched as asterisks to facilitate identification.

The operator can cancel the current batch job once it has started execution by pressing REQUEST on the console and entering /CANCEL.

Batch Processing with VM

MUSIC Batch operating with VM, when VM has *spooled* reader, printer and punch devices, will appear to operate somewhat differently to that described above. The real devices will be controlled by VM and operate as described in the VM publications. MUSIC jobs read in from VM readers will require an appropriate VM USERID control card in front of them. Card decks punched from MUSIC will normally have VM separators in addition to the MUSIC one. Two blank cards will follow those punched by the user.

Output of MUSIC jobs spooled by VM will normally only start to print or punch after the entire job has finished. MUSIC will automatically issue VM CLOSE and SPOOL commands at the end of each job unless the /CTL NOVMCLOSE or /CTL NOVMSPOOL MUSIC console commands have been used.

MUSIC VM Reader Classes

If your installation allows terminal users to directly submit MUSIC batch jobs, then the following notes apply. (Each installation may change some of these conventions. For example, overnight class may be defined to be after 5pm.)

When a user submits a job through the MUSIC SUBMIT facility, a two character job class is specified. The first identifying the kind of job and the second when the job is allowed to run. The meaning of the first character is given below:

- A No special handling and no tape requests.
- S Job requires special handling such as special paper but does not require magnetic tape.
- T Job requires magnetic tape.

Specifying class S or T will cause the program to ask you to enter a message that will be sent to the operator just before the job is run.

The job will be held to be run overnight by specifying the *letter* O as the second character of the job class such as CLASS='AO'. (Overnight class should not be started before 6pm weekdays.) Specifying the second letter as 'A' will allow the job to be started at any time. Note that the default class is 'AO' and that the MUSIC charge rates are cheapest when the overnight class is eligible to run.

These two character classes will appear as 1 character VM reader file classes. The following correspondencies are made:

User Class	VM Reader Class
AA	J
SA	S
TA	T
AO	E
SO	F
TO	G

MUSIC will normally automatically run the reader class J. If reader files S and T appear, you should run them when you can. For example, to run class T you first make sure the tapes are available to MUSIC, then you can run all class T jobs by typing the VM command "SPOOL C CLASS T". After the tape jobs have been read you should reset the MUSIC reader to CLASS J.

Similar procedures should be followed for all the classes. Note that class E, F, and G must not be run before 6 pm weekdays. They can be run any time on Saturday and Sunday. (The VM command "Q RDR" may be of help to list the jobs waiting to run. The "Q RDR ALL" command will also list the MUSIC code of the submitter together with the MUSIC job name.)

Batch Processing Using the Internal Reader

Installations could use the *internal reader* to submit jobs to run at batch although most VM installations would normally use the VM spooled reader facility described above to perform this function. In either case, the user will use the SUBMIT facility.

After the MUSIC system has been initialized, the system will process the first job in the internal reader queue number 1, if any. After processing that job, the system will take the next one in the queue. Jobs from the internal reader will always be executed before those submitted on cards.

The /RDR operator command can be used to change the queue number if it is desired to run another class of jobs. This command can also be used to see what queues have jobs waiting for execution.

MUSIC Internal Reader Classes

If your installation allows terminal users to directly submit MUSIC batch jobs, then the following notes apply. (Each installation may change some of these conventions. For example, overnight class may be defined to be after 5pm.)

When a user submits a job through the MUSIC SUBMIT facility, a two character job class is specified. The first identifying the kind of job and the second when the job is allowed to run. The meaning of the first character is given below:

A No special handling and no tape requests.
S Job requires special handling such as special paper but does not require magnetic tape.
T Job requires magnetic tape.

Specifying class S or T will cause the program to ask you to enter a message that will be sent to the operator just before the job is run.

The job will be held to be run overnight by specifying the *letter* O as the second character of the job class such as CLASS='AO'. (Overnight class should not be started before 6pm weekdays.) Specifying the second letter as 'A' will allow the job to be started at any time. Note that the default class is 'AO' and that the MUSIC charge rates are cheapest when the overnight class is eligible to run.

These two character classes will appear as numeric internal reader numbers. The following correspondencies are made:

User Class	Internal Reader Class
AA	1
SA	2
TA	3
AO	4
SO	5
TO	6

MUSIC will normally automatically run the reader class 1. If reader files 2 and 3 appear, you should run them when you can. For example, to run class 3 you first make sure the tapes are available to MUSIC, then you can run all class 3 jobs by typing the MUSIC command "/RDR 3". After the tape jobs have been read you should set the MUSIC reader to class 1.

Similar procedures should be followed for all the classes. Note that class 4, 5, and 6 must not be run before 6 pm weekdays. They can be run any time on Saturday and Sunday. (The MUSIC command "/RDR Q" may be of help to see what queues have jobs in them.)

Allowing Internal Reader Submission

When MUSIC is first installed, the SUBMIT program is configured to submit jobs through VM. To change SUBMIT to use the internal reader, simply edit the file $PGM:OLDSUB and alter the INTRDR=FALSE parameter in the file to read INTRDR=TRUE.

Controlling Auxiliary Printers

328x type printers or ASCII printers may be defined to the system as auxiliary printers. When the system in IPLed, these devices are automatically *signed on* under the code $MON and the printer program started. Console commands which effect terminals also apply to these devices (/ADD, /CANCEL, /DROP, /RESET etc.). In addition, the following commands are provided to allow some operator control over these devices:

```
/REPLY n STOP                    /REPLY n CANCEL
/REPLY n GO                      /REPLY n HOLD
```

n	Terminal identification number of the printer.
STOP	Stop the printer.
GO	Resume or start printing.
CANCEL	Stop printing the current file and go on to the next one in the print queue. The file which was printing is deleted from the system.
HOLD	Stop printing the current file and go on to the next one in the print queue. The file which was printing is placed at the bottom of the print queue.

The OUTPUT and PQ commands may be issued from a MUSIC terminal to determine the status of various printers and the state of the output queue. If it is required to restart the program controlling the printer the best method is to use the /CANCEL command to stop the program and the /RESET command to restart the program when required.

Chapter 4. The System Console

Overview

After the MUSIC system is initialized, the processing unit console can be virtually left alone. Except for the processing of batch jobs, the system is operator-free. System errors that can cause problems will sound the console alarm, and the console operator can respond. When such problems arise or when system or terminal status information is desired, there are available a variety of system console commands. These commands can perform the following types of functions: controlling batch and terminal job execution, sending messages to terminals, enabling and disabling TP lines, and interrogating and changing internal switches and constants.

Some processing units, such as the S/370 125, will sound the console alarm when no processing unit activity has taken place for one minute. This is not to be taken as an error condition.

System Console Commands

The following describes the console commands that can be used from the processing unit console. It should be noted that all system responses shown are prefixed by the time of day in the form xx:xx. All references to terminals are by logical unit number. *terminal n* refers to the terminal identified by internal number *n.*

The operator is cautioned not to keep the console in read status for long periods of time. Otherwise, it is possible for a backlog of messages to build up to the point that the system will loop waiting for these messages to be printed.

The / (slash) is optional with the commands.

/ABEND n

Explanation**:** Forces a program check in the program in process on terminal *n.* Use the identification of B for batch.

System Response**:** `*OK`

/ADD n

Explanation**:** A request that terminal *n* be added. If this port is already enabled, the request is ignored.

System Response**:** `*OK`

Note: If /ADD ALL was entered, all nonactive terminals are added including those previous dropped.

/BATCH

Explanation**:** Shows the status of the job running at batch. It redisplays any outstanding /PAUSE statements and tape mounts. It also shows how long the job has been running or waiting for tape mounts,

etc. This time is in the format HH:MM for hours and minutes. Abbreviation: /BAT.

System Response**:** Status messages.

/CANCEL n (or /CAN n)

Explanation**:** The current job on terminal *n* is to be canceled. If *n* is omitted, the current batch job is canceled.

System Response**:** `*OK` (if the command was issued for a terminal).

/CP command-for-VM

Explanation**:** The text following the /CP command is sent to the control program of VM. VM commands are documented in IBM *VM/370*: *Command Reference for General Users* (GC20-1820) and *VM/370*: *Operator's Guide* (GC20-1806).

System Response**:** MUSIC responds with either a *OK message, or a message giving the VM error code in response to the command. In addition to this, the VM control program may also type a response on the console. Note that any command that requires a privilege class that the MUSIC virtual machine does not have will be rejected by VM and an error message printed.

/CTL B-HI

Explanation**:** Sets scheduling priority of batch jobs equal to that of terminal jobs, so they can compete for processing unit time on an equal basis.

System Response**:** `*OK`

/CTL B-LO

Explanation**:** Sets batch scheduling priority lower than that for terminal jobs. This is the default setting.

System Response**:** `*OK`

/CTL CD-ON

Explanation**:** User codes can be authorized such that they can be used on batch only with permission of the console operator. See the RESTR option of the CODUPD utility. This command allows these restricted codes to be used. This command has no effect on user codes that are not allowed on batch at all or on those that are allowed on batch with no restrictions.

System Response**:** `*OK`

/CTL CD-OFF

Explanation**:** Disallows batch-restricted job. This is the default setting. (See /CTL CD-ON description.)

System Response**:** `*OK`

/CTL NOVMCLOSE

Explanation**:** This command stops MUSIC from issuing a VM CLOSE command at the end of each batch job when the output devices are spooled. The VM SPOOL command for these devices will also not be performed. The /CTL VMCLOSE command is used to enable these operations again.

System Response**:** `*OK`

/CTL NOVMSPOOL

Explanation**:** This command stops MUSIC from issuing a VM SPOOL command at the end of each batch job when the output devices are spooled. (MUSIC normally issues a "SPOOL xxx TO SYSTEM" command at the end of batch jobs using VM spooled output devices.)

System Response**:** `*OK`

/CTL PRTCHK-ON

Explanation**:** Perform normal error correction on batch printer operations. This is the default setting.

System Response**:** `*OK`

/CTL PRTCHK-OFF

Explanation**:** Perform no correction on I/O error conditions (except printer not ready) on the batch printer. This command can be used while loading the UCS buffer (on printers so equipped) to ignore errors until the buffer is properly loaded.

System Response**:** `*OK`

/CTL PWCHK-ON
/CTL PWCHK-OFF

Explanation**:** These commands determine whether the contents of batch password cards are to be checked. The default is PWCHK-ON.

System Response**:** `*OK`

/CTL PURGE

Explanation**:** Allows files to be purged from batch via /PURGE jobs.

System Response**:** `*OK`

/CTL NOPURGE

Explanation**:** Restricts purging of files from batch. This is the default setting.

System Response**:** `*OK`

/CTL VMCLOSE

Explanation**:** This command will reset the action taken by the /CTL NOVMCLOSE command. MUSIC normally will issue a CLOSE command at the end of each batch job which uses VM spooled output devices.

System Response**:** `*OK`

/CTL VMSPOOL

Explanation**:** This command will reset the effect caused by the /CTL NOVMSPOOL command. MUSIC normally will issue a "SPOOL xx SYSTEM" command at the end of each batch job that uses VM spooled output devices.

System Response**:** `*OK`

/DAILY message text

Explanation**:** Display message text on all terminals at sign-on time.

System Response**:** `*OK`

Note: The message will continue to be transmitted until the system is loaded or until a /DAILY command with no message text is issued.

/DISABLE PUNCH

Explanation**:** All batch punched output is to be deleted. This enables the system to operate even if the card punch is not available. All output skipped by means of this command cannot be recovered. To re-enable, issue the /ENABLE PUNCH command.

System Response**:** `*OK`

/DROP n

Explanation**:** Terminal *n* is to be dropped. It will not be available for use unless added via the /ADD command. Note that the line will not be dropped until some sort of terminal activity occurs. For a terminal in read status, the line will not be dropped until the attention or return key is depressed. (see /HALT and /RESET).

System Response**:** `DRPNG n`

/DUMP nnnnnn

Explanation**:** Dump 32 bytes of main storage starting at the location designated. The address *nnnnnn* is virtual. Only locations in the first 4K of storage, or in the nucleus or above, can be specified.

System Response**:** `32 bytes of main storage as specified.`

/ENABLE PUNCH

Explanation**:** The batch punch is to be enabled. This is the default setting.

System Response**:** `*OK`

/FIND string

Explanation**:** The TCB number and device address of all users whose userid starts with the characters specified in *string* will be displayed on the console. *String* may be from 1 to 16 characters long.

/GET UCB cuu

Explanation**:** Display the storage location of the disk or tape UCB, and the volume name (if available). *cuu* is the device address.

System Response**:** `cuu UCB AT xxxxxx, VOL=vvvvvv`

/GO

Explanation**:** This command is used to resume batch job processing following a halt due to a /PAUSE statement. Use /NOGO to tell MUSIC not to run the job and to cause it to be canceled. See /PAUSE (M310 message) under "Batch Processing Messages" in Appendix A.

System Response**:** `*OK`

/HALT n

Explanation**:** This command has the same effect as /DROP except that it causes *any* current I/O activity to be immediately terminated. Normally the specified line will immediately be DROPPED. /HALT ALL will cause a /HALT command to be issued to all lines on which there is a currently signed on user.

System Response**:** `*OK`

/LOCATE module

Explanation**:** This command displays the address in main storage for the *module* name in the system nucleus module. Abbreviation: /LOC.

/MESSAGE n message text

Explanation**:** Send the message text to terminal *n.*

System Response**:** `*OK`

Note: If ALL is entered instead of *n*, the message is sent to all active terminals. If the message text is omitted, the text transmitted will be `SYSTEM IS SHUTTING DOWN IN 5 MINUTES.` At this time a DAILY message of `SYSTEM IS ABOUT TO SHUT DOWN` is also automatically scheduled. The operator is cautioned not to send another message until sufficient time has occurred for the first message to be sent. Otherwise, it is possible for the first user to receive the same

message that was intended only for the second user. In this case however, the second user will receive the correct message. Abbreviations: /MES, /MSG.

/NOGO

Explanation**:** This command is used to resume batch processing following a halt caused due to a /PAUSE statement. Use /GO to allow the job to be processed. Use /NOGO to tell MUSIC not to run the job and to cause it to be canceled. See /PAUSE (M310 message) under "Batch Processing Messages" in Appendix A.

System Response**:** `*OK`

/QUEUE (or /Q)

Explanation**:** Display the number of active terminals and the terminal currently being dispatched in a user region.

System Response**:** `CPU ccc ACTIVE nnn`

ccc The terminal number of the user currently resident in user region. A batch job appears as 000. If no user is currently being dispatched, this field is displayed as ---. The --- may also mean that all regions are waiting for I/O to complete.
nnn The number of active terminals (i.e. terminals signed on).

/RDR n or OFF or Q

Explanation**:** This command is used to control the internal batch reader. Specifying "/RDR 2" will cause batch to start processing batch internal reader queue number 2. Specifying OFF will stop the system from selecting any further jobs from the internal reader. Specifying Q will display the classes that have jobs waiting to run.

System Response**:** `*OK` or classes that have jobs waiting to run.

/REP nnnnnn hhhh
/REP nnnnnn 'cccc'

Explanation**:** The storage location *nnnnnn* is to be modified by the value of *hhhh.* Up to 44 bytes can be modified in one command. The h's are any hexadecimal digits and the number of characters must be even. For convenience, a comma may be inserted between pairs of h's as in the following example: 0700,4700,0000. Character data may be specified by enclosing it in single quotes. The address *nnnnnn* is virtual. Only locations in the first 4K of storage, or in the nucleus or above, can be specified. Alternate command name: /STORE.

System Response**:** `*OK`

/REPLY n xxxxxxxx

Explanation**:** Places 8 characters of information into the post code field (in the XTCB) of the specified terminal, and reactivates the program. The characters are taken from the *xxxxxxxx* parameter, extended with blanks if necessary. A terminal number of 0 can be used for Batch. This command can be used to communicate with the AUTOPR programs controlling any auxiliary printers. Abbreviation: /R.

System Response: `*OK`

/RESET n

Explanation: The user on the specified terminal will be signed off. A RESET command is also sent to VM. The line is then made available for subsequent users. If the terminal in question is a BTRM, running a background job such as AUTOPR or VMREAD, this command will restart the background job.

System Response: `*OK`

/SEARCH (or /SEA)

Explanation: Displays the TCB number, userid and real address of all sessions associated with the virtual terminal address "addr".

/STATUS (or /S)

Explanation: Display a list of active terminals. (The WHOALL or WHOACT utility programs provide this information and more.)

System Response: A list of active terminals is displayed. If more than 28 are active, only the first 28 are shown and an asterisk is placed at the end of the line indicating the overflow.

/STOP

Explanation: Shuts down the system immediately.

System Response: `12:54 ACCOUNTING FILE CLOSED. MUSIC IS SHUT DOWN.`

/STORE nnnnnn hhhh
/STORE nnnnnn 'cccc'

Explanation: The /STORE command is the same as the /REP command. Abbreviations: /STOR, /STO.

/SYSTOP

Explanation: Same as /STOP.

System Response: `ACCOUNTING FILE CLOSED. MUSIC IS SHUT DOWN.`

/TCB n

Explanation: The contents and location of the TCB for terminal specified by *n.*

System Response: `TCB nnnnnnnnnnnnnnn`
LOC xxxxxx

Where *n's* are hexadecimal display of first eight bytes of TCB. The first 4 characters are the TCB

number in hexadecimal and the next 4 are the physical line address associated with the TCB. The *x's* are location (hex) of TCB for terminal *n.*

/VARY cuu, OFFLINE

Explanation**:** This command is used to remove a tape unit from the list of available drives from which the system selects the unit to be used for the next tape job. Abbreviation: OFF for OFFLINE.

System Response**:** `*OK`

/VARY cuu, ONLINE

Explanation**:** This command is used to re-add a tape unit to the list of available drives from which the system selects units to be used for the next tape job. The command can be used to inform the system that a tape unit previously found in not operational state is now available for use. Abbreviation: ON for ONLINE.

System Response**:** `*OK`

/WHO n

Explanation**:** An inquiry to know who is (or was) on the terminal designated by *n.* (The WHOALL or WHOACT utility programs provide this information and more.)

System Response**:** `NNN UUUUUUU *`

where:

NNN	is the terminal number (TCB number).
UUUUUUU	is the userid. If this field is printed as |||||||, then someone is (or has) been connected to the line without signing on.
*	This character identifies that a user is currently active on this line.

Auxiliary Operator CONSOLE Facility

The CONSOLE utility allows a privileged user at a 3270-type terminal to view the most recent messages on the MUSIC/SP main operator console, and to enter MUSIC and VM operator commands. It is invoked by the command "console" (or "cons" for short).

The MUSIC console handler records the latest console activity in an 8K buffer in main storage. This includes MUSIC operator messages, tape mount requests, system error messages, and operator input (MUSIC commands only). The CONSOLE utility displays the contents of this buffer.

The display does not automatically show new console activity. You must refresh the display by pressing the ENTER key, in order to see any new console messages.

Privileges required:

FILES and CREAD are needed to display console messages. MAINT is needed to enter operator commands.

Screen Display

```
16:39 M402 SIGN ON: CCGM000, UAD=0A2, TCB= 17
16:40 M402 SIGN ON: CCFP000, UAD=0A3, TCB= 18
16:41 M306 /ID MUSJOB   CCFP 000 999 999 999
16:42 *OK
16:42 M402 SIGN ON: CCGM000, UAD=0A4, TCB= 19
16:43 M300 BATCH IDLE
16:52 M402 SIGN ON: CCEL000, UAD=0A2, TCB= 17
16:55 M408 CMD FROM  16 CCDE000: s
16:55   1   2   3   4   5   6  15  16  17  18
16:56 M405  16 CCDE000 VM CMD: q ti
16:58 M405  16 CCDE000 VM CMD: yyyy
16:59 M306 /ID CCDE1    CCDE 000 MAX 999 999
16:59 M300 BATCH IDLE
17:06 M405  18 CCFP000 VM CMD: q e
17:07 M405  18 CCFP000 VM CMD: q e
17:09 M408 CMD FROM  18 CCFP000: batch
17:09 M315 NO BATCH JOB ACTIVE
OpCmd: _

PF1-Help   3-Exit   7-Up    8-Down   9-Browse    10-Edit   12-Retrieve
ENTER-Refresh                       17:11  Auxiliary MUSIC/SP Console
```

Figure 4.1 - Console Screen Display

The top part of the screen displays part of the contents of the console buffer. When the program starts, the most recent console messages (the "bottom" of the buffer) are shown. PF keys 7 and 8 are used to page up or down in the buffer.

The field labelled OpCmd is an input area where you can type a MUSIC operator command. After typing the command, press ENTER to submit the command to MUSIC's console handler and display the new contents of the console buffer. A VM command (for the MUSIC virtual machine) can be entered by typing "cp xxx", where xxx is the VM command, for example "cp query time".

The output of a VM command is displayed temporarily at the top of the screen. If there are many lines of VM output, it is displayed in non-full-screen mode (unit 6). Press ENTER or PA2 when "More..." appears in the bottom right corner, to continue the display.

Some potentially destructive commands ask you to confirm the command by typing "yes" (or "y") in the command area and pressing ENTER. Type anything else to cancel the command.

The screen line following the command input area is used for displaying messages from the CONSOLE utility.

The bottom two lines show the definitions of the 3270 action keys (PF keys and ENTER), and show the time of day when the last action key was pressed.

Action Keys

PF1-Help — Displays HELP information on how to use the CONSOLE facility. This includes a brief description of some of the more common MUSIC operator commands.

PF3-Exit — Exits from the CONSOLE facility.

PF7-Up — Displays the previous page of console messages (further back in time).

PF8-Down — Displays the next page of console messages (further ahead in time).

PF9-Browse — Copies the entire console message buffer to file @CONLOG and then BROWSEs that file. This gives you the full power of the MUSIC Editor for displaying and searching.

PF10-Edit — Same as PF9, except the Editor is used, instead of Browse.

PF12-Retrieve — Each time this key is pressed, the previous input line is displayed in the input area. You can then modify the text, if you want, and press ENTER to re-execute the command. The last 5 input lines are remembered. Repeatedly pressing PF12 cycles through these lines.

ENTER — Pressing the ENTER key does several things:

1. It executes the operator command, if you have typed one in the input area (or retrieved a previous input line via PF12).
2. It refreshes the console messages, i.e. it gets and displays any new messages that have occurred since the last time the ENTER key was pressed. If there are no new messages, "No change" is displayed.
3. It moves to the bottom of the console messages, to display the most recent ones.

Some Common MUSIC Operator Commands

Refer to "System Console Commands" for a full description of these and other commands. *n* is a terminal (TCB) number, which you can get from various messages or by the FIND command. The minimum abbreviation is shown in capital letters.

ADD n — Adds terminal *n* if it has been dropped.

BATch — Shows the status of the current batch job, if any.

CANcel n — Cancels the job running on terminal *n*. Cancels the current batch job if *n* is not specified.

CP xxx — Executes a VM command *xxx*.

DUMp x — Displays 32 bytes of MUSIC's main storage at hex address *x*. See also STORE.

FIND uuu — Finds and displays terminal numbers for all userids that start with the string *uuu*. *uuu* can be 1 to 7 characters long. An asterisk (*) in the output means that the userid is signed on.

GO — Allows a batch job to execute after a /PAUSE statement in the job has suspended the job. See also NOGO. This is used to respond to an M310 message.

HALT n — Stops and drops terminal *n*. This can be used to terminate a BTRM job.

MESsage n text
MSG — Sends a message to terminal *n*.

NOGO — Rejects a batch job after a /PAUSE statement in the job has suspended the job. See also GO. This is used to respond to an M310 message.

Queue — Displays the number *n* of the terminal currently being executed, and the number *m* of signed on terminals, as: "CPU n ACTIVE m"

RDR Q/OFF/m	Displays information about the internal reader queue (RDR Q), or stops the internal reader from processing jobs (RDR OFF), or sets a new reader class number (RDR *m).*
Status	Displays the terminal numbers of the first few signed-on terminals.

STOre x tttt
REP
	Changes MUSIC main storage at hex address *x*. The new bytes are specified by *tttt* in hex or as characters in single quotes. See also DUMP. Examples:
	
	`STORE 81AB84 47F0`
	`STO 8E0 'MUSICX'`

STOP	Shuts down MUSIC.
VARY cuu OFF	Makes a tape device unavailable for MUSIC jobs. *cuu* is the virtual address of the tape drive.
VARY cuu ON	Makes a tape device available for MUSIC jobs. *cuu* is the virtual address of the tape drive.
WHO n	Shows the userid on terminal *n.* An asterisk (*) means that the user is signed on.

System Errors and Restart Procedures

A system error or an error caused by a machine malfunction may, depending on its severity, cause the system to shut down. For most disk errors, the system may be restarted. A more detailed discussion of the restart procedure can be found in *Chapter 3 - Loading the System*.

The procedures to be followed after a serious I/O error occurs will vary with the type of error and with installation standards. If the system can be restarted without errors, it is likely that the trouble was caused by a temporary hardware malfunction. If it is determined that a permanent error exists on a disk pack, it must be repaired. Backup copies may be restored. This procedure, however, loses any changes made since the backup was taken. Alternately, the MUSIC system programmer can be consulted. Often errors can be corrected without loss of data.

Taking A System Dump

If it is decided that the system should be restarted, the operator should note the console status indicators (PSW, system light, wait light -- see the appropriate processing unit operators' procedures). A main storage dump should be taken if the cause of the failure is not known. This can be done by loading from the disk pack labeled MUSIC1. A full storage dump will be placed onto a data set on that pack. After the system is restarted, this dump can be printed using the storage print utility (PRDUMP) described in *Chapter 17 - System Utility Programs*.

The dump program will normally print the message `DUMP OK` when it has finished. The message `DUMP DID NOT ALL FIT ON DISK` will be issued if the disk data set was too small to hold the dump. The portion that did fit will be correct. By default, the dump data set will hold 16 megabytes (16384K) of storage.

The dump program expects a console address of 01F or 009 working in 1052 mode. If this is not the case then no messages will print and a disabled wait state with PSW ending FF03 will result, although the dump

operation will not be affected.

The stand-alone dump program is placed on the MUSIC1 volume by the DMPGEN program. Refer to the DMPGEN job in file $GEN:DMPGEN.SAMPLEJOB for the required control statements. A DMPGEN job should be run whenever the size or location of the dump data set is changed. Note that the dump data set (normally PGMDMP on volume MUSIC1) must have only 1 extent.

Chapter 5. Routine Maintenance

Overview

The following is a guide to what programs and utilities should be run daily and weekly. They assume an environment where MUSIC is in full production use and that daily backup and accounting information are critical. Full documentation of the utilities are described in *Chapter 17 - System Utility Programs* of this publication. Techniques of dumping full packs are discussed later in this chapter.

Daily Activities

Perform the following steps on a daily basis at a time when the system is not usually busy.

- Run jobs to print statistical counters. These jobs invoke the COUNTS, WAITS, IOTIME and LIBSPACE programs. These outputs are useful for tuning MUSIC and for tracking the usage of Save Library space.
- At this point you can optionally re-load the system to reset the statistical counters and close the accounting file. Usually you should issue the "/MESSAGE ALL" command five minutes before shutdown to warn the users of the shutdown.
- Run the CODUMP utility to produce a backup copy of the code table. (Store this tape in a secure place as it contains the passwords for all MUSIC codes.)
- Run the session accounting program ACTDMP to produce accounting records for the day's activity.
- Run the NOWDOL utility to update the NOW$ field in each code record. This step uses the records produced by the ACTDMP utility.
- Run the MFARCH and MFCHEK utilities to dump to tape all files that have changed since the last backup operation.
- Run the DSARCH utility to dump to tape all UDS files that have the BACKUP attribute.

Once a Week Activities

The following should be run once a week in addition to the above. They need not all be done on the same night.

- Run MFACCT to produce Save Library accounting records to be used with your billing program.
- Run DSACT (DSACT1, DSACT2) to produce UDS accounting records to be used with your billing program.
- Dump entire disk volumes. These backups are used only in disastrous situations such as if an entire pack is dropped or otherwise made unusable. Dumping entire packs takes considerable longer than the

special MUSIC backup utilities. All packs should be dumped once a week for backup purposes. You must make sure to dump the packs that have MUSIC Save Library data sets at the same time. Dumping them on different nights means that the Save Library data sets cannot be used from the restored copies. Make sure that the MUSIC system is shut down while these dumps are taken.

The VM DDR utility can be used to dump all supported disk types. See the *MUSIC/SP Administrator's Guide* for a description of how to perform these full pack dumps.

- Run ELOG.CLEANUP to delete any empty editor log files. (The AUTOSUB utility can be used to automatically run this job.)

- The output from the LIBSPACE utility should be inspected to make sure that there is a sufficient amount of free space on the Save Library. Careful note should be made of the amount of space available in the largest 5 extents of each space as that might limit the saving of large files. Additional space can be added as explained under the heading "Adding Space to the Save Library" in *Chapter 6 - System Reconfiguration*.

Maintaining the User Code Table

The dumping and restoring of the user code table is accomplished using the MUSIC utility CODUMP.

The frequency of the dumping of the user code table is determined by the installation. It is also necessary to occasionally restore the code table as this serves to compress its index records and eliminate lost space due to deleted user codes. A detailed description of the use of this utility can be found in *Chapter 17 - System Utility Programs* of this manual.

Maintaining the News Facility (/NEWS)

The /NEWS command allows general users to obtain information about recent items of interest at your installation. This command runs the file $PGM:NEWS.DATA. To modify the message, simply edit this file and find the line starting with NEWS. Items added immediately after that line will be displayed first. The lines entered are displayed line by line as they exist on the file.

Broadcast Messages At Sign-on

It is possible to set up two files containing broadcast messages that will be displayed on the user's terminal at sign-on time. These messages are in addition to the one resulting from the /DAILY operator command. The two file names are $PGM:ALERT.FILE and $PGM:ALERT.NEWS. They can each contain an 80-byte 1-line message, with optional carriage control in column 1. The files must be public (PUBL attribute). If either file does not exist or is empty, the corresponding message is not issued. One of the files could be used by operations staff and the other by software support staff.

Typical usage would be to create a $PGM:ALERT.NEWS file, with a message such as "Type NEWS for info on ...", whenever an item is added to NEWS. Delete the file after a day or two.

Maintaining the HELP Facility

The HELP facility maintains a log file of request for topics that do not currently exist. This file can be examined by editing the file $HLP:@GO.NOINFOLOG. You may delete lines from this file once the requests have been noted. You can add extra HELP files. Consult the *MUSIC/SP Campus-Wide Information Systems* (*CWIS*) *Guide* for full details. This guide discusses IDP (Information Display Program) for creating and editing help facilities.

Use IDP to MUSIC's main help facility: $HLP:HELP. Edit the file $HLP:@GO.HOURS to update your installation's hours of operation. Edit $HLP:@GO.ROUTE to update your installation's route destinations. Edit $HLP:@GO.FORMS to update your installation's code numbers for special forms.

Maintaining the New User Automatic Userid Facility

This program front-ends the new user registration process. The putative new user is presented with four or five screens, each of which can contain explanatory text that the site can modify. Various operating modes such as no code allocation, code allocation with disabled access until validated, or immediate code allocation with notification of what has been done, are selectable via the parameter file. Anonymous e-mail to the system administrator can be sent if there is a problem that needs lookin at. Validation against a MUSIC data file is available (the file contains names and id information). Fix-ups to generate a unique userid are automatically done, and may also be disabled if this is not desirable.

This should be run on a userid such as NEWUSER with the CODES, LSCAN and FILES privileges, AUTOPR($INT:NEWUSER), NONCAN, NOMULTI, RESTR, MAX$(NOLIMIT).

The following files contain the text for message screens:

$INT:NEWUSER.MSG1	text for the initial welcome screen
$INT:NEWUSER.MSG2	text used to prompt for user information
$INT:NEWUSER.MSG3	text used to query for a password
$INT:NEWUSER.MSG4	text announcing success (may require validation depending on your site's parameter settings)
$INT:NEWUSER.MSG6	text used to indicate an error (user not allowed to register, for example)

The file $INT:NEWUSER.VALIDUSRS contains a list of users allowed to register. To use this list, specify the OTHERINFO VALIDATE option in the parameter file. The new user's name and "other-info" data will be matched against what is in the file. Only a successful match will allow the user to continue.

The file $INT:NEWUSER.PARMS contains the following keywords:

```
TITLE            <text> | OFF
NOTIFY           <e-mail address> | OFF
ALLOCATE_USERID  IMMEDIATE | DEFER
ACTIVATE_USERID  ON | OFF
GENERATE_USERID  SERIAL | NAME | RANDOM
USERID_PREFIX    <text>
CODEPARMS        <codupd commands>
```

```
PHONEINFO        REQUIRED | OPTIONAL
ADDRESSINFO      REQUIRED | OPTIONAL
OTHERINFO        REQUIRED | OPTIONAL | VALIDATE
FIXUPS           ON | OFF
TESTMODE   (Note: use ONLY for testing, not for production)
```

MUSIC/SP Administrator's Reference

Part 2 - Chapters 6 to 9 (AR_P2.PS)

Part III. Customizing MUSIC/SP

Chapter 6. System Reconfiguration

Overview of System Reconfiguration

The ADMIN facility provides a series of menus under the topic "System Support Functions" that allow you to perform most of the basic configuration functions. (The ADMIN Facility is described in the *MUSIC/SP Administrator's Guide*.) This chapter is intended to provide background information on what is going on behind the menus. The programs and commands used often require privileges and should be run from the $000 userid (or equivalent) with VIP set ON.

Modifying the System Catalog

The system catalog defines the names and volumes of system data sets, the members to be made resident in the link pack area, and VM commands to be executed automatically at IPL time. The catalog has to be modified if you are adding or moving system datasets, if you are adding or deleting modules from the FLPA or PLPA, or if you want to add some VM commands to be executed during initialization. Modifying the catalog is a two step process. First use the editor to make the changes to the file $GEN:SYSCAT, then use the EDTCAT utility to copy the catalog records to the catalog system dataset (SYS1.MUSIC.CATALOG). The new catalog takes effect the next time the system is loaded. If the system fails to initialize due to errors in the catalog, you can use the =EDIT IPL option to fix the catalog and allow the system to get going. See the description of the EDTCAT utility for more details.

Defining New Terminals

Terminals are defined by device cards in the NUCGEN job. The "System Configuration" function of ADMIN can be used to make routine additions and deletions to the I/O configuration, including the terminals you have defined. You can do the same thing without using ADMIN by editing the NUCGEN jobstream in the file $GEN:NUCGEN.JOB, running the jobstream and installing the generated nucleus. See the description of the NUCGEN utility for more details.

If you are adding a large number of new terminals you may also have to increase the size of some system data sets and allocate more resources to the system to maintain system performance. Consult *Chapter 19 - Direct Access Storage* to determine if the SWAP, PAGE, and SCRATCH datasets are large enough for the number you intend to define. Additional terminals usually means additional users. You may have to add Save Library data sets to provide file space for these users. Use the LIBSPACE utility to determine what free space is available in the existing Save Library.

Additional terminals increase the systems main storage requirements. Main storage is not only required for additional control blocks and buffers, but is also needed for extra page pool space to support the increased load. The actual amount of storage required is dependent on a large number of factors, but you should have at least 1 Meg for every 32 users. Extra storage will give better performance. You may also want to spread swapping and paging over a number of different channels. This is important in optimizing performance on systems with a large number of users.

Increasing the Number of DASD Channels

DASD channels are of the selector type and can only handle one I/O request at a time. If more than one channel is available, I/O operations can be done simultaneously on each channel. MUSIC/SP's performance can be improved by making more than one disk channel available. In order to take full advantage of this performance gain you must configure the system to use the available channels for both swapping and paging. Up to three channels may be used simultaneously for swapping and paging.

Allocate the additional swap and page data sets. *Chapter 19 - Direct Access Storage* has details as to the space and blocksize requirements. For optimal performance try to locate these data sets near the middle of the disk packs. This will minimize arm movement.

Use the FORMAT utility to format and initialize the new data sets. Add entries for the new data sets in the system catalog. The system will automatically use the new data sets the next time it is loaded.

Performance gains can also be made by balancing the channel load and placing high usage data sets on low usage disks. The IOTIME utility gives a breakdown of disk I/O requests by channel, DASD volume, and system data set. It is extremely useful in monitoring disk usage. The basic procedure for moving or enlarging a system data set is as follows.

- Allocate the new data set using different name.
- Format the dataset.
- Copy any required data.
- Modify the catalog to point to the new data set.
- Re-IPL the system and test that the system works.
- Delete the old data set after few days.

Care should be taken during the step that copies the data to make sure that the old data is not changed either during or after the copy process. The simplest rule is to make sure that there are no users on the system and do not create or delete any files. For data sets like PAGE, SWAP and SCRATCH this step can be skipped.

Increasing the Main Storage Size.

MUSIC/SP's performance can be improved by giving the system more main storage. This enables the system to keep more user tasks in storage and reduces both the paging and swapping overhead. For performance reasons it is very important that VM does not page MUSIC/SP, so all of MUSIC/SP's pages should be locked or it should be run in the V=R region. The simplest way to lock MUSIC/SP's pages is to include a VM LOCK command in the catalog so it will be issued automatically when the system is started. There are a number of NUCGEN parameters that tell the system how to use main storage.

The MAXCOR parameter defines the main storage available. Since the system will automatically figure out how much it has, most people find it convenient to set MAXCOR to a large value and have the system figure out the real value for itself.

The REGION parameter defines the maximum virtual storage area for the user region. The default of 1 Meg is suitable for most applications. Increasing this has no major effect on system performance but may require you to enlarge the PAGE data sets if a significant number of users take advantage of the larger region.

The MAXRRS parameter defines the maximum real storage that a user can get. It determines how large a program can get before the system will page it. It also determines the maximum size that is involved in a swap operation. Most programs run quite well using the default MAXRRS of 272K, but programs whose

virtual storage "working set" is significantly larger than MAXRRS may experience performance problems due to excessive paging. The setting of MAXRRS can have significant effects on system performance. Increasing MAXRRS will improve the performance of user programs with large storage requirements by reducing the paging overhead. This performance gain will be at the expense of the system in general, in terms of an increased swap load and reduction in the multi programming level (MAXMPL), unless sufficient real storage is added to the system to accommodate the increase in the MAXRRS. To increase MAXRRS by nnnK add at least nnnK*MAXMPL to the total main storage. If you do not have enough main storage satisfy the formula above then you can trade off a higher swap load for improved performance of large programs.

If you are running VM/ESA with the ESA feature you can define up to 64MB for MUSIC. The storage above 16MB is added to the storage available to user programs. This will reduce swapping and allow you to increase the MAXRRS parameter as described above. The storage above 16MB cannot be used for FLPA, RAM disk or trace tables. There is no extra overhead in running user programs above 16MB line.

Adding Space to the Save Library

Use the LIBSPACE utility to find the current amount of free (unused) space in the library. You can add to the space available by following the procedure given below.

To add more space to the Save Library, you must create additional library data set(s). There are from 1 to 180 Save Library data sets, named SYS1.MUSIC.ULnn (where nn=01, 02, ..). Each can hold up to approximately 57000K of user files. The data sets must be added in order. For example, if MUSIC was installed with 4 data sets (UL01 to UL04), the next to be added would be SYS1.MUSIC.UL05.

Note: It is possible to have up to 180 library data sets. The data set names from 100 on would then contain 3 digits instead of 2, e.g. SYS1.MUSIC.UL100.

The smallest recommended size for each library data set would be 16200 records, which can hold up to about 8096K of user files. The maximum size is 114,504 records, which can contain 57,248K of user files.

Older versions of MUSIC supported a maximum size of 16200 records. Each library data set can be a different size. Installations which have a large number of Save Library data sets will want to choose larger sizes for individual data sets, so that they will not exceed the maximum number of 180 data sets. Larger size means that individual user files within these data sets can be larger. For example, for the 16200 record size the maximum user file size would be about 16200 x 512 or 8 million bytes. The largest data set can accommodate a maximum file of about 57 million bytes.

A library data set can be increased in size at a later date by following the procedure in the next topic.

The following is an example of the procedure for adding a Save Library data set of a size of 16200 records. It is also described in *Chapter 19 - Direct Access Storage*. The example below assumes that UL01 through UL04 already exist and that UL05 is to be added. The device type is assumed to be 3350; the procedure is similar for other device types.

1. Calculate the number of tracks needed for 16200 records of 512 bytes each.

   e.g., 3350: 16200/27 = 600 tracks (round up)

2. Allocate the data set (SYS1.MUSIC.ULnn) and use the FORMAT utility to format it. First enter /VIP ON, then run the following job:

```
/FILE 1 UDS(%UL05) VOL(MUSICX) NEW
/ETC NTRK(600) BLK(512) BUFNO(0)
/LOAD IEFBR
```

Start the FORMAT program at a terminal and enter the following commands when prompted.

```
DEVICE=3350
VOLUME='MUSICX'
DSN='SYS1.MUSIC.UL05'
```

When the dataset has been formatted terminate the FORMAT program by responding to the prompt with /CANCEL.

3. Initialize that data set by running the following job.

```
/FILE 1 UDS(%UL05) VOL(MUSICX) OLD
/INC ULINIT
```

When prompted for an option enter LIB.

4. Add the following record to the system catalog. The file $GEN:SYSCAT normally contains the catalog.

```
ULIB05   U005 00 MNS 000 MUSICX SYS1.MUSIC.UL05
```

After changing $GEN:SYSCAT run the EDTCAT utility.

5. The new space will be available after the next IPL of MUSIC.

More than one library data set may be added at the same time, if desired.

Run the LIBSPACE utility to find the current number of ULnn data sets (4 in the above example).

The standard number of tracks in a ULnn data set (16200 512-byte records) for the common device types are:

```
3340 . . . .  1350 tracks
3330 . . . . . 810
3350 . . . . . 600
3375 . . . . . 405
3380 . . . . . 353
3310,3370  . . 507
933x . . . . . 507
```

Enlarging an Existing Save Library Data Set

The following procedure describes the steps required to enlarge an existing Save Library data set. You might want to do this if you have a large number of data sets and do not want to approach the maximum of 180 data sets.

In the example below we are enlarging data set SYS1.MUSIC.UL05 that exists on MUSICX. The larger one is on MUSICY.

1. Allocate the new larger library data set. Follow step 2 of the previous topic to allocate and format the new one.

2. Create the file CPYLIB that contains the following JCL:

```
/FILE 1 UDS(%UL05) VOL(MUSICX) SHR
/FILE 2 UDS(%UL05) VOL(MUSICY) OLD
/INC DSCOPY
```

3. Create the file DOCAT that contains the following:

```
/INC EDTCAT
/INC $GEN:SYSCAT
```

4. Create the file FIXMAP that contains the following JCL that points to the new larger data set:

```
/FILE 1 UDS(%UL05) VOL(MUSICY) OLD
/INC ULINIT
```

5. Edit the file $GEN:SYSCAT to point to the new volume the larger data set is on.

6. Make sure no one else is on the system and that you make no changes to the Save Library during the following steps. If you do have to make changes, simply restart the procedure from step 7.

7. Type "CPYLIB" to run the job in that file to copy the old data to the new data set.

8. Type "FIXMAP" to enlarge the bitmap on the new larger data set. Enter the option "XLIB" when prompted. (If you typed another option here you must return to step 7.)

9. Type "DOCAT" to put the new system catalog on line.

10. Re-IPL the system. After this point the new larger data set will be in use. If everything is right, then delete the old data set.

Reorganizing Save Library Free Space

Over time, the free (unused) space in each Save Library space may become randomly distributed throughout it. This means that MUSIC may not be able to obtain sufficiently large amounts of space from it. The allocation algorithm will use up to 5 extents from a single library data set to satisfy an initial request for file space. (Subsequently the file can grow to 16 extents and may span library data sets). For example, it might be that there is 200K free space but the largest 5 extents are 2K each. In this case only a 10K file can be allocated. This *fragmentation* condition can be spotted by looking at the output of the LIBSPACE program. Run the MFMOVE utility to eliminate this fragmentation. If your free space is often fragmented (every month or so) and users complain about being unable to allocate files, the problem is probably a lack of file space and you should consider adding some new datasets to the Save Library.

Enlarging the Code Table

The MUSIC code table is comprised of two data sets: the index (SYS1.MUSIC.CODINDX) and the code records (SYS1.MUSIC.CODTABL). The standard size table can hold about 20000 user code records. This

can be increased to about 50000 if desired. Increasing the code table size has no impact on system performance. The procedure is:

1. Use CODUMP utility to dump the old code table to tape.

2. Allocate and format new system data sets SYS1.MUSIC.CODINDX and SYS1.MUSIC.CODTABL, using files FMTCDX and FMTCOD with the FORMAT utility. After formatting, the only authorized code on the system is $000 (with password MUSIC). The index should contain at least 340 records (block size 4096). The code table data set should contain the desired number of records, up to a maximum of about 50000.

3. Use the CODUMP utility to restore the code table from the tape produced in step (2). Code $000 must be used for this job.

Note: Before doing step (2), it is a good idea to rename the old index and code data sets (using DSREN utility). Then if something goes wrong, you can IPL MUSIC and change the catalog to point to the old data sets if necessary. In this way you will always have a usable system. After step (3), the old data sets can be deleted.

Enlarging Main Storage Dump Data Set

The system main storage dump data set, called PGMDMP on the MUSIC1 volume, is set up to handle a storage size of up to 16 megabytes. Some installations may want to change its size. Consult the job in the file $GEN:DMPGEN.SAMPLEJOB for the commands that were used to create the original one. Simply delete the old UDS file and rerun the job specifying a different number of records. For example, use 16385 records to handle an 8 megabyte storage dump. Note that the dump data set must be only one extent. You can check this by using the $INFO option of the UTIL program.

Configuring the RAM-Disk

To improve system performance it is possible to define a area of memory to be used as a RAM disk area. During system initialization files are brought from disk into this area. Subsequent access to these files requires no I/O operations. If the right files are chosen a significant reduction can be made in the I/O overhead incurred in accessing high usage files. The RAM disk is read only. If a file in the RAM disk is changed the system automatically removes it from RAM and uses the modified version on disk until the next time RAM is loaded.

The RAMDLD utility loads the RAM disk area from the Save Library. It is called automatically by JOBONE at system startup time, but it can also be run on its own to re-load the RAM disk while the system is running. VIP must be set ON to run this program. You can run the program by issuing the RAMDLD command. Note that reloading the RAM disk on a running system may cause a few users to experience temporary problems if they happen to be accessing an old RAM file at that instant.

The file $PGM:RAMDISK.LIST contains a list of files that are to be loaded into RAM. If RAMDLD runs out of room in the RAM disk area it simply gives up. The size of this area is defined by the RAMDSK parameter in the NUCGEN. Note that once loaded this is a read-only RAM disk, so good candidates for this area are high access read only type files, like the execution files and load modules for common commands, programs and utilities.

The format of the $PGM:RAMDISK.LIST file is simply a list of fully qualified file names (Code,

subdirectory path, and name) that are to be loaded into the RAM disk area in memory. There should be only one file name per line and it must start in column 1. Lines starting with a "*" are ignored and can be used as comments. A sample $PGM:RAMDISK.LIST file is distributed with the system.

The RAMREP program produces a report on RAM disk utilization. It is useful in monitoring the performance of your RAM disk area. It is important to monitor RAM disk usage because if the files in RAM are not often used the memory assigned to them would be better used as part of the page pool or LPA.

The performance benefit will vary from system to system but, using a RAM disk area of 400-800K, you should be able to get between 15% to 25% of all open requests be for files in RAM.

To enable the RAM disk performance option take the following steps.

1. Determine how much memory to allocate for the RAM-Disk. An area of 512K is large enough to hold the files defined in the distributed RAMDISK.LIST file.

2. Define a RAM disk area in the NUCGEN (ADMIN 4 10 5).

3. Add any local high usage files to $PGM:RAMDISK.LIST.

4. When you next IPL the RAM-Disk will be activated automatically.

Configuring MUSIC Programs

There are two important files that configure many programs on MUSIC. They are:

$EML:MAIL.CONFIG
and
$TCP:TCPIP.CONFIG

These files contain important information specific to your site, and when these programs are run they affect many MUSIC programs on your system.

MAIL.CONFIG contains information about MAIL programs and programs that pertain to MAIL such as POP servers and MCS (MUSIC/SP Client/Server). Refer to *Chapter 9 - Electronic Mail Facility*.

TCPIP.CONFIG contains information about many servers that use TCP/IP, such as: TELNET, FTP, Gopher, Web, etc. Refer to *Chapter 16 - Configuring MUSIC for TCP/IP*.

Benchmark Programs

The benchmark programs (userid: $BMK) BM1 thru BM8 can be used to measure machine speed, mainly for the purpose of comparing the speed of the various machines that run MUSIC. They measure CPU processing speed (floating-point and integer instructions) and disk i/o speed. Since the programs should be run on a quiet MUSIC system, without any swapping or paging or competition from other jobs, they measure the speed of the hardware, not the efficiency of MUSIC.

The programs are:

BM1 Instruction speed for double-precision floating-point calculations, involving arrays. The program

should display the result X = 499461.038961036

BM2 Instruction speed for integer calculations, involving arrays. The program should display the result X = 871192

BM3 Disk speed for sequential read of 1000 512-byte blocks, 1 block at a time.

BM4 Disk speed for sequential read of 8000 512-byte blocks, 20 blocks at a time.

BM5 Disk speed for reading 1000 512-byte blocks, 1 block at a time, with a seek of a varying number of cylinders required for each read. The seek distance varies from 0 to 8000 blocks, with an average of about 4000 blocks.

BM6 Same as BM3, but writes instead of reads.

BM7 Same as BM4, but writes instead of reads.

BM8 Same as BM5, but writes instead of reads.

Note: Please refer to $BMK:BM.DOC for complete documentation on these programs.

Chapter 7. Terminal Configuration and Tailoring

Terminal Definition

Terminals and workstations are defined in the NUCGEN. The device address, the type of terminal and various connection options are specified. See the section on the NUCGEN program for details.

In addition to the information specified in the NUCGEN, the system module TRMCTL contains definitions of specific terminal types. Each terminal definition is assigned a unique name. To use a particular terminal definition, the user must specify this name during sign on or in the profile. These terminal definitions contain many parameters, such as line length, carriage return rate, tab and backspace characters, screen addressing, translate tables and other special characteristics.

All parameters for a terminal definition are specified as arguments of the TERMINAL macro instruction. New TERMINAL definitions are added to the TRMCTL module immediately following the existing ones.

The module TRMCTL contains sample TERMINAL specifications for several varieties of terminals. The conditional assembly statements will skip the generation of them. These extra specifications are used at McGill University.

Once changed, the object module for the new TRMCTL module is added to the NUCGEN generation module after the DEVEND statement. The nucleus generation step is then performed to produce a tape which is then IPLed to effect the change.

Some parameters described below are not valid for all the major classes of terminals supported by MUSIC. Each description specifies which of the generic terminal types honor each parameter. Descriptions with no terminal type explicitly mentioned are valid for all types. The major supported types are :- TTY, 2741, 1050, 3270.

When parameters call for the specification of terminal control characters, these are the codes as seen by the processing unit. In particular the codes seen from ASCII terminals differ from those described in the vendor's publications. Refer to topic "ASCII to S/370 Code" in this chapter for further information.

Terminal Macro Parameters

APL — This parameter indicates which type of APL character set is available on 3270 type terminals.

APL=DAF	Data analysis feature (3272)
APL=TEXT	APL/text feature (3274)
APL=NO	None (default)

BLANK — This parameter must be the hexadecimal value of a blank in terminal code.

BS — This parameter must be the hexadecimal value of a physical backspace character in terminal code.

BUFSIZ — Size, in bytes, of terminals built in buffer. This is usually used when transmitting data to a terminal at a line speed higher than the terminal can print. Periodically MUSIC sends a sequence to the terminal saying its buffer is full and wait for a response before continuing output.

BUFSEQ Two byte sequence used to inform buffered terminals that its buffer is full (see BUFSIZ).

CLEAR This parameter contains a three-character (hex) clear screen sequence to be used by MUSIC to clear TTY type screens. If not specified, MUSIC will not attempt to clear the screen.

CRONLY This parameter controls the ending sequence sent to a TTY terminal at the end of an output line which is not to be followed by a line feed. Normally this parameter points to a sequence containing a carriage return. Allowed values are the same as for READNL.

ENDNL This parameter controls the ending sequence sent to a TTY terminal at the end of an output line. Allowed values are the same as for READNL.

FASTBS This parameter may have a value of YES or NO. It indicates whether the terminal has a fast physical backspace. That is, one that does not require idle characters following it. It is valid only for TTY type terminals.

FFDELAY This parameter gives the number of idles to be transmitted after a forms feed operation. A forms feed will only be attempted if FORMS=TRUE is specified.

Note: This delay is usually related to line speed and the length of the forms.

FOLDNL This parameter controls the ending sequence sent to a TTY terminal at the end of the first line of folded output (that is, a line too long for the physical carriage length). Allowed values are the same as for READNL.

FORMS This parameter may have a value of YES or NO. It indicates whether the terminal is equipped for vertical forms control.

HEX This parameter contains a pair of hexadecimal bytes which will be used to translate input text for this terminal. After translation to EBCDIC, any occurrence of the first parameter will be replaced by the second value. The two values must be contained in parentheses and separated by a comma. This parameter may be overridden by a user profile or by the terminal command /CTL.

IDLE This parameter consists of one or two hexadecimal characters which act as an idle on this terminal type. That is, these characters will have no effect on the terminal output. If two values are to be used, they should be in parentheses, separated by a comma. This idle character is used only by TAB routine to know which characters do not cause the type head to move. The system always uses the character X'DF' to transmit idles to the terminal as this character is recognized by the transmission control hardware.

IDLES This parameter is not processed by the TERMINAL macro. It simply is allows a convenient method of recording the number of idles to be transmitted (as calculated by RETRAT, RETBAS, MINSIZ).

INBS This parameter contains the EBCDIC value of the character to be treated as a backspace during input. This parameter may be overridden by a user profile or by the terminal command /CTL.

INTAB This parameter contains the EBCDIC value of the character to be treated as a horizontal tab during input. This parameter may be overridden by a user profile or by the terminal command /CTL.

LF This parameter must be the hexadecimal value (in terminal code) of a character which causes a line feed function on the terminal.

LFDELAY This parameter gives the number of idles to be transmitted after a line feed operation (for example, on a double spaced print line).

LINE This parameter gives the physical line length of the terminal. It is used by the system to split (or fold) lines on TTY type terminals which cannot print a full 133 bytes per line. This parameter should be specified for all terminal types as it can be used by application programs to format output.

LINUM This parameter contains the number (in decimals) of consecutive lines of output which can be sent to a TTY screen before it goes into *MORE mode*. If not specified, the terminal will scroll normally. (Value 1-238)

MINSIZ This parameter is only valid for TTY terminals. It is used to specify the minimum length of a line transmitted to the terminal. If the text of a line is not long enough, idles are added before the CR/LF sequence. No idles are added after the CR/LF. If MINSIZ is specified, RETRAT and RETBAS must not be used.

MORE This parameter contains a five-character (hex) screen addressing sequence which is used to issue the `MORE...` message to a TTY type terminal when it enters *MORE mode*. Use DF as a fill character if the required sequence is less than five characters. This can occur when the number of lines per screen is exceeded or a page skip carriage control is issued. If not specified, no message is issued.

NAME This parameter consists of 1-8 alphanumeric characters. It is used by MUSIC users on the /ID and /RESTART commands and with the PROFILE program to specify terminal control types. The NAME parameter need not be unique. See the SPEED parameter for further details. This parameter is required.

NUMBER This is the internal number by which the system identifies the terminal specifications. All numbers must be unique. User defined specs have numbers starting at 25. The maximum number is 100. There must be a specification supplied for each generic terminal type in the system (TTY, 2741, 1050, 3270). Its NUMBER must be equal to the internal code for the terminal type (these are supplied in the original TRMCTL module). The NUMBER is placed in a user code table record when the TERM command of the user PROFILE program is used. For this reason, once a specification is added, it should not be removed (or have its NUMBER changed), unless you are certain that no user profiles reference it. This parameter is required.

PREP This parameter indicates whether a *prepare* command should be used in buffered terminal protocol and *more mode*. PREP=YES is the default. PREP=NO should be specified for terminals attached via a packet-switched network, since the prepare command has no effect in that case.

PRSEQ This parameter contains the bytes (in terminal code, in hexadecimal), to be transmitted to a TTY type terminal before input is read. It may contain codes to turn on paper or magnetic tape readers. It may be from zero to 3 8-bit bytes long.

READNL This parameter is a code number indicating the type of CR/LF sequence to be transmitted to a terminal after an input line is received. The values may be:-

0 = No sequence (except possible idles)
1 = Line feed
2 = CR/LF
3 = X-OFF, CR/LF
4 = CR

If any other sequences are required, the module TERMIO must be changed accordingly.

RETRAT
RETBAS — These parameters are used for TTY, 2741, and 1050 terminals to control the amount of time allowed for carriage return - line feed operations. Following the CR/LF (or equivalent), the system transmits to the terminal the specified number of IDLE characters. These characters have no visual effect at the terminal, but since they take time to transmit, they give the terminal time to perform its new-line function. The number of idles to be transmitted is calculated as follows:-

$$IDLE\_COUNT = \frac{LINE\_LENGTH * 16 + RETBAS}{RETRAT}$$

where LINE-LENGTH is the number of characters just types on the terminal, and RETRAT and RETBAS are the values of the corresponding parameters. By setting the parameters appropriately, many different timing characteristics can be generated. For example, by setting RETBAS to zero, the number of idles would be proportional to LINE-LENGTH. By setting RETRAT to a value greater than 133*16, the idle count would be independent of LINE-LENGTH. The line speed must be taken into consideration when determining these values. The maximum value for each is 32767. RETRAT and MINSIZ cannot both be specified as 0.

RETURN — This parameter must be the hexadecimal value (in terminal code) of a character which causes a carriage return or new line function on the terminal.

RTDELAY — This parameter gives the number of idles to be transmitted after a typical short carriage return operation. This delay may be used during password blankouts, and before messages immediately following terminal input. It is valid for TTY and 2741 type terminals. The maximum value for RTDELAY is 36.

SHIFT — This parameter may be the hexadecimal value of the upper case shift indication in the terminal code. It should be specified only for terminals which transmit and receive shift characters. IBM 2741 and 1050 terminals are examples of these. MUSIC uses this bit mask to minimize the sending of shift characters in cases where they will not really be needed, such as for blanks, tabs, backspaces and idles. A value of 00 indicates no optimization is to be done.

SPEED — This parameter gives the line speed (in bits/second) for which this specification is to be used. The value may be one of 110, 134, 300, 600, 1200, 1800, 2000, 2400, LOCAL, or ANY. LOCAL is used for 3270s only. If ANY is used, this terminal specification may be used at any line speed. The combination of NAME and SPEED must be unique. This parameter is required.

TAB — This parameter must be the hexadecimal value of a physical tab character in terminal code.

TABRAT
TABBAS — These parameters are used to calculate the number of idles transmitted following an output tab on a TTY, 2741, or 1050 type terminal. The formula is the same as for RETRAT and RETBAS except that the character count used is the number of spaces to be tabbed over instead of the line length. The value of TABRAT must not be zero.

TRAN — This parameter is used to set the translate table number (TTAB) in the TCB. If not specified, the default for that terminal type is used. (See module TRANTB).

TYPE — This parameter identifies the generic type of terminal being specified. It must be one of

TTY, 2741, 1050 or 3270. It is required.

UCONLY This parameter translates output to upper case letters.

3270 Emulation

Overview - 3270 Emulation

Most software running on IBM S/370 processors is designed to work on 3270 terminals. It is possible to connect ASCII devices such as PCs or terminals to IBM processors through protocol convertors that make the ASCII devices function like 3270 terminals. The following information is specific to the **7171 ASCII Device Attachment Control Unit** and the **9370 ASCII Subsystem.**

The 9370 ASCII subsystem is implemented as a line adapter card that plugs directly into the 9370 rack. The 7171 Control Unit attaches to a S/370 channel as would a local 3174 control unit. ASCII terminals, printers or PCs are attached using a standard RS-232 full duplex asynchronous interface. The terminals can be directly connected using null-modem cables or through switched lines and modems.

From the host point of view, the terminals are treated exactly like regular 3270 style terminals and they should be defined as such in the host I/O configuration tables. There are some features of these protocol convertors that the host software can take advantage of if it knows it is dealing with a protocol convertor. To do this special options have been added to the I/O configuration tables.

Configuration

VM's I/O configuration is defined in the module DMKRIO. To let VM know that a terminal is connected through a 7171 or through the ASCII Subsystem, the FEATURE=EMUL3270 option should be specified on the RDEVICE statement for the terminal. Specifying this does two things. It causes VM to automatically drop the line when the user logs off and it allows VM to give MUSIC correct information about the device. If you do not want VM to do the automatic line drop you can specify the E3270HLD feature on the RDEVICE macro. If the EMUL3270 feature is not specified, MUSIC will not recognize that the terminals are connected through a protocol convertor.

On MUSIC these terminals are defined as 3270 devices in the NUCGEN. If MUSIC is running under VM, the terminal features are passed to MUSIC from VM when the terminal connects to MUSIC. So if FEATURE=EMUL3270 is specified in the VM configuration, MUSIC will recognize the terminal as being on a protocol convertor. If MUSIC is running without VM, the 7171 option should be specified on the device statement for the terminal in the NUCGEN. The 7171 option is ignored if MUSIC is running under VM.

In addition to supporting 3270 emulation, terminals and PCs connected through the protocol convertor can support applications that use ASCII transparent mode such as PCWS file transfer and KERMIT. If the DIALUP option is specified on the device statement in the NUCGEN, MUSIC will also automatically issue a "host disconnect" command when the user signs off. This causes any switched line connections to be broken and frees up the line for subsequent users to dial in. If DIRECT is specified on the NUCGEN device statement the "host disconnect" sequence is not issued. The DIRECT and DIALUP options only effect terminals that connect through a protocol convertor, they have no effect on real 3270 terminals.

ASCII Transparent Mode Support

The 7171 and the 9370 ASCII Subsystem both support the "transparent" transmission of ASCII data. This feature is particularly useful when the capabilities of the ASCII device fall outside the realm of the 3270, such as in the areas of file transfer, vector graphics, text processing, and personal computing. In addition to supporting applications such as KERMIT that were designed to use transparent mode, MUSIC also supports applications that were written to run on native ASCII devices. In other words you can connect a terminal, PC or printer to MUSIC through the 7171 or ASCII subsystem and treat as if it were connected through regular ASCII asynchronous port.

MUSIC/SP's "ASCII Transparent Mode Support" intercepts the I/O requests at the channel program level and the commands and data transformed so that they perform the required functions. This is done at the lowest level of terminal support and is thus completely transparent to the application.

Terminals that connect to MUSIC in 3270 emulation mode can switch to ASCII transparent mode using the TOTTY command. Once this command is issued, protocol conversion is terminated and the terminal or PC functions as an asynchronous ASCII device. The user can switch back to emulation mode using the TO3270 command.

MUSIC ports can also be configured to start out in ASCII transparent mode when the terminal first connects. In this case the port appears as if it were a regular ASCII port. This is useful for special applications. It could be used, for example, to attach a dedicated ASCII printer or to define a group of ports for exclusive ASCII use. These must be defined as TTY devices in the MUSIC NUCGEN. The "7171" option must also be specified. For example,

```
TTY 0F0-0F3 DIALUP,1200,7171
```

The above example defines ports 0F0, 0F1, 0F2, and 0F3 as dialup 1200 baud ASCII lines. The baud rate, in this case 1200, is not used by MUSIC/SP to govern the transmission rate of the data, it is simply used to calculate the number of IDLES that should be inserted after certain control sequences, such as "LINE FEED", to give the terminal time to perform the control operation. Since most modern terminals do not require IDLES, the speed specified is generally of no consequence. The protocol convertor has automatic line speed detect and allows a variety of speeds on a particular port, so despite the fact that 1200 baud was specified in the NUCGEN, a terminal could successfully use the port at any of the allowed speeds.

Note: Although the protocol convertor can detect the speed of a terminal, it is not setup to pass this information back to MUSIC, so AUTOSPEED should never be specified in the NUCGEN.

Connecting with PCWS

There are two ways to use connect to the system through a 7171 or ASCII subsystem using PCWS.

PCWS using VT100 Emulation.

This is the simplest and in most cases the most efficient way of connecting your PC to MUSIC through ASCII lines. The protocol convertor ports should be defined as regular 3270 devices on both VM and MUSIC. Don't forget to specify FEATURE=EMUL3270 in the RDEVICE macro for VM. The user starts PCWS in VT100 emulation mode (ALT-V) and then connects to the protocol convertor. When prompted for a "TERMINAL TYPE", the user can use VT100 (standard monochrome) or VT100P (color). The PC can now be used as a regular 3270 terminal. (The VT100P terminal definition comes with MUSIC and must be installed in the protocol convertor). The following summarizes the procedure.

1. Start up PCWS in VT100 mode.
2. Connect to the protocol convertor and press Carriage Return.

3. Select a terminal type of "VT100" or "VT100P".
4. Wait for the VM logo to display and connect to MUSIC/SP as you would from a real 3270 terminal.

When the user starts a file transfer using XTMUS or XTPC the system automatically switches to ASCII transparent mode, does the file transfer and switches back. If you want to use native PCWS page mode instead of VT100 emulation, enter the command "TOTTY PCWS" to change to transparent mode and then press ALT V to switch out of VT100 emulation.

Native Page Mode PCWS

Originally PCWS was designed to connect PCs to MUSIC through native ASCII ports. It is possible to use native PCWS using the ASCII transparency feature of the 7171 or ASCII Subsystem, however since the overhead is quite high, VT100 emulation is better. However in circumstances where FULL duplex communication is not possible or very expensive (i.e. using packet switched networks), it is more efficient to use native PCWS on a line configured for ASCII transparency.

A group of lines are defined as TTYs with the 7171 option.

```
TTY 0F0-0FF DIALUP,1200,7171
```

The lines are dedicated to MUSIC in the VM directory. (For example, "DEDICATE 0F0 140" where 140 is the real address.)

The user starts PCWS in PAGE mode and then connects to one of the ports on the protocol convertor that is dedicated to MUSIC. When prompted for a "TERMINAL TYPE", the user can enter TYPETERM or PCWS. (The PCWS terminal definition comes with MUSIC and must be installed in the protocol convertor. It is equivalent to TYPETERM). After pressing the ENTER key the MUSIC sign-on message will appear. The following summarizes the procedure.

1. Start up PCWS in PAGE mode.
2. Connect to the protocol convertor and press Carriage Return.
3. Select a terminal type of "TERMTYPE" or "PCWS".
4. Press Carriage Return to get the MUSIC sign on message.

ASCII Transparency: Usage Notes

When ASCII transparency mode is used certain functions do not work exactly the same as they would through a regular asynchronous communications controller. Specifically in the handling of the BREAK key and PREPARE command.

BREAK

The BREAK key is treated as an error and if the user presses it the terminal will be disabled till the "Error Reset" sequence, (Ctrl-R), is entered. Due to this the BREAK key cannot be used to interrupt program execution as it is on regular ASCII lines. When connected in transparent mode the break function can be accomplished by pressing the key defined as the 3270 RETURN key. It would probably be convenient for your users if you modify the TYPETERM key definitions so that 3270 RETURN key is different from the CARRIAGE RETURN key that is normally used to enter lines in ASCII mode. During output the system may continue outputting for a few lines before issuing a "break time" read.

If you are using PCWS to connect to MUSIC in transparent mode, there is an option in PCWS to send a Carriage Return when pressing the Break key instead of a Break signal. Enter SETUP mode in PCWS and change the "Page Mode Break" option on MENU 3 on the MISCELLANEOUS SETTINGS panel.

PREPARE, Scrolling, and Pacing

MUSIC uses the PREPARE channel command in scrolling and pacing operations. The PREPARE command terminates on the receipt of ANY character from a terminal. Thus in scrolling, if the page is full MUSIC writes the `more...` message and issues a PREPARE command. The user can go to the next page by pressing any key. The protocol convertor does not transmit characters to the host as they are typed, but waits for a *line turnaround* character such as CARRIAGE RETURN before sending any data to the host. For most users this means that they should press "CARRIAGE RETURN" to go to the next page instead of CLEAR. MUSIC also uses this PREPARE command in handshaking with buffered terminals to prevent buffer overflow. In this case if the terminal cannot be configured to respond with a CARRIAGE RETURN when it has emptied it's buffer the alternative below should be investigated. *Turnaround* sequences other than CARRIAGE RETURN can be added to the terminal definition tables in the 7171 or ASCII subsystem.

XON/XOFF Pacing

Since the 7171 and the ASCII Subsystem communicate with the terminals in full duplex mode they can perform XON/XOFF line pacing automatically without intervention from the operating system. MUSIC uses the buffered terminal protocol mentioned above to perform the same function. Since the MUSIC approach involves more overhead and relies on the PREPARE command, it is recommended that if pacing is required you use the XON/XOFF support provided by the protocol convertor. When using this type of pacing it is important that MUSIC does not disrupt the protocol by transmitting XONs and XOFFs of its own. The default ascii terminal type (See type ASCII in the module TRMCTL) will avoid sending XONs and XOFFs. The PCWS terminal type will also avoid this.

Terminal Definition Tables for 7171 and ASCII Subsystem

MUSIC/SP provides three Terminal Definition Tables to enhance the ease of use of PCWS through the 7171 or ASCII Subsystem.

The first terminal type, "PCWS", is an alternative to the "TYPETERM" terminal type. It exists simply to remove any confusion for general users.

The second terminal type, "VT100P", should be used when connecting with PCWS in VT100 mode. The "P" is added to differentiate it from IBM's "VT100" terminal type. This TDT is added to take advantage of PCWS's extensions to the VT100 emulator, specifically colour and keystroke handling. It should NOT be used as a replacement for IBM's "VT100" terminal type. The terminal definition makes the keystrokes in VT100 mode quite similar to the keystrokes in 3270 mode of PCWS.

The third terminal type, "VT52P", should be used when connecting with PCWS in VT52 mode. Again, the "P" is added to specify that it is really designed for use with the PCWS VT52 emulator.

To install these terminal definition tables, download the files named $PCW:PCWS.TRM, $PCW:VT52P.TRM, and $PCW:$VT100P.TRM to a PC.

On the 7171 Add the terminal types "PCWS", "VT100P", and "VT52P" to the control file used for the 7171 terminal definition table generation. Re-link the image file for the 7171, and re-load the image file to your 7171. If you are not sure on how to add terminal definition tables, refer to the *7171 ASCII Device Attachment Control Unit Reference Manual and Programming Guide* (GA37-0021).

If you are using a 9370 ASCII Subsystem consult the *ASCII Subsystem Terminal Installation and Customization Guide* (SA33-1564) for information on how to IMPORT a 7171 .TRM file.

Terminal Translate Tables

The module TRANTB contains the terminal translate tables. The beginning of the module contains a table of pointers to the actual translate tables. Each entry in this table contains a table name, some flag bytes, and the addresses of three translate tables (input, printed output, punched output). If the translate table address is zero, no translation will be done. MUSIC uses the $$TTAB field of the TCB to index this table. $$TTAB is set to one of the system defaults when the line is enabled. When a user signs on, this default may be overridden by the value specified in the TRAN parameter of the TERMINAL macro for that specific terminal type. (Module TRMCTL). An example of this is the 3270A terminal type. Using this mechanism an installation may add special function translate tables which apply only to specific terminals.

MUSIC also provides support such that a user can dynamically change to another set of translate tables in mid-session. Each of the entries in TRANTB has an eight byte name. The user need only specify that name on a /TEXT command to use those tables. For example:

/TEXT ABC	Switch to the table set named ABC
/TEXT STANDARD	Go back to standard tables

There is the restriction that the *new* table set must be of the same type as the old one. (i.e. byte 9 of the new entry must match byte 9 of the old one). This avoids using tables that were not designed for a particular terminal type.

The MUSIC system, as distributed, does not contain any extra translate tables. For this reason the form of the /TEXT command documented above, is NOT documented in the *MUSIC/SP User's Reference Guide*.

Enhanced ASCII Support in MUSIC

The following describes the flexible ASCII terminal support provided in the MUSIC system. This support not only enhances the performance of terminals in the operational area of speed and accuracy but also in the user's eyes in terms of usability.

As previously mentioned the module TRMCTL contains a number of control blocks which have information pertinent to the operation of terminals. These control blocks contain information such as speed, idles required, line length, special control sequences, and tab chars. They are generated by the TERMINAL macro. Each one has an associated name, which the user may specify on the /ID command or in the user profile.

There are some recent additions to this TERMINAL macro which were implemented specifically to address the problems posed by ASCII screens and buffered terminals. These features have also turned out to be quite useful in setting up communication protocols with a wide variety of mini and micro computers.

Controlled Scrolling

One major problem encountered when using an ASCII screen is trying to list a long file. Once the screen fills up old data just disappears (scrolls off the top) as new data is added. To avoid this three parameters were added to the TERMINAL macro. The number of lines on the screen, a clear screen sequence, and a screen address for a `MORE...` message must be specified. When the line count is exceeded or a "next page" carriage control is issued, MUSIC issues the `MORE...` message indicating that the current page is full. This is referred to as *MORE mode*. The terminal user, having read what is on the terminal's screen, tells the system that the next page is wanted by pressing the return key. (Other keys will work). MUSIC will then

resume outputting by clearing the screen and starting again from the top.

LINUM=nn

This is the number (in decimal) of consecutive lines which MUSIC will send to the terminal. When this number is exceeded MUSIC enters *MORE mode*. Normally LINUM should be one less than the actual number of lines on the screen since one line is required by the MORE message. Installations using IBM 3270s may wish to set LINUM to 22 on their ASCII screens for compatibility purposes, since MUSIC uses 22 lines for output on 3270s. During a terminal session the user may change the value of LINUM using the /CTL LINES=nn command.

MORE=aabbccddee

This five character sequence plus the text `MORE...` is sent to a terminal when LINUM is exceeded or when a *next page* carriage control is encountered. This sequence is sent to the terminal untranslated, so it should be specified in raw ASCII characters. (Refer to topic "ASCII to S/370 Code" in this chapter for details.) Usually this sequence is a screen addressing sequence used to position the MORE message at an appropriate location on the screen. The lower right hand corner is a good place. If a terminal has no screen addressing capabilities there are two alternatives. Specifying idles as the screen addressing sequence will cause the `MORE...` message to appear on the next line. If the sequence is specified as zeros, or is not specified, MUSIC will send a BELL character to the terminal to inform the user of the *MORE* condition.

CLEAR=aabbcc

This three character sequence (in raw ASCII) is used to clear the screen after *MORE mode* and in the event a clear screen carriage control (X'70') is sent.

These three parameters may be specified in various combinations. For example if only LINUM was specified then MUSIC would ring the terminal alarm and wait when the line count was exceeded. No `MORE...` message, or clear sequence would be sent in this case. If only MORE was specified the line counter would never overflow, thus *MORE mode* would only occur if a next page carriage control was encountered. If CLEAR is omitted MUSIC simply never attempts to clear the screen.

Buffered Terminals

Some terminals have a built in buffer. Characters arrive from MUSIC at a given rate and are stored in this buffer while the terminal processes them in its own time. There are two advantages to this scheme:

1. Some functions at a terminal take a substantial amount of time. (Forms feeds, setting special features, plotting, etc..). Normally MUSIC would have to allow idle time for these events to take place. If the terminal is buffered MUSIC can send these time consuming sequences without idles, knowing that the terminal can save subsequent data in its buffer and catch up later on. Generally the terminal can perform much faster since it knows exactly when it can continue with the next operation. Take for example the case of a forms feed. MUSIC would have to allow enough idles for an entire page to be fed through since it doesn't know how many lines are left until the next page. A buffered terminal would continue the output immediately the forms feed was completed, regardless of how long it took.

2. Buffered terminals can be run at a higher speed than the same terminal without the buffer feature. Data can be sent to the buffer at say 1200 baud, while the terminal is printing at 300 baud. This requires some protocol to avoid overflowing the buffer. A printer rated 300 baud must be able to handle *worst case* data at that speed, meaning that it can usually exceed that speed for the average case.

Because of hardware limitations, MUSIC supports terminals in half duplex mode where communication can only be in one direction at a time. Therefore the only protocol is to have MUSIC stop periodically when it considers the terminal's buffer is almost full, and wait for the terminal to give the go-ahead to continue when it is empty. Two new parameters have been added to the TERMINAL macro to allow this.

BUFSIZ=nnnn

This decimal number indicates the size of the terminal's buffer. MUSIC counts the characters sent to the terminal. When this count exceeds BUFSIZ, a special sequence (BFSEQ) is sent to the terminal and MUSIC waits for some response before sending the next line.

BFSEQ=aabb

This is the two character sequence MUSIC sends to inform a terminal that its buffer is full, or near full. MUSIC will not send any more data until the terminal responds to this.

Note that the terminal must be capable of following this protocol. When it detects the special sequence in its buffer it should automatically send an acknowledging sequence back to MUSIC indicating that it is ready to accept more data. MUSIC will accept any character a valid response.

Some terminals use what is called the X-ON/X-OFF protocol. This only works on full duplex asynchronous lines which are normally not supported on most IBM telecommunication controllers. There is a technique to overcome this problem on MUSIC if the terminal has an answer back drum. The technique is to have MUSIC put in a sequence to trigger the drum as the last piece of information sent to buffer just before it fills up. First consult the terminal owner's manual to find out how to set the answer back drum to be a single DEL character. (Other non-printable characters might work as well.) Then generate a MUSIC terminal type using the options BFSEQ=DFAD and an appropriate BUFSIZ parameter.

Some examples of the use of the above parameter can be found by looking at the source of the module TRMCTL under the $SYS code.

Applications to Mini/Micro Computers

Both the *MORE mode* and *buffered terminal* protocols have interesting applications in communication with mini/micro computers. Generally speaking the most common way to communicate between MUSIC and a mini is to program the mini to *look* like an ASCII terminal. Problems arise however when the mini requires *think time* to process the data it receives from MUSIC. This situation occurs most often in the area of file transfer, when the mini requires time to transfer data to cassette tape, floppy disk or some other medium. Both protocols described above can be used to stop outputting every once in a while and give the *terminal* time to respond. This can be done by having a terminal type defined with LINUM=1. MUSIC enters "MORE mode" after each line of output and sends a `MORE...` message to the terminal. The mini, on recognizing the *MORE* sequence sends a character when it is ready for the next line. A more sophisticated case might be a mini with an input buffer of a given size programmed to use buffered terminal protocol.

Regardless of whether any of the above protocols are used one of the key questions, in programming mini/micro computers to talk to MUSIC, is: "What control characters are involved?"

The first thing is that data is transferred in *reverse ASCII*. That is ASCII as documented on your 370 Reference card or the manual for your particular terminal, but all bits are reversed and a parity bit added in the low order end. (See topic "ASCII to S/370 Code" in this chapter for details.)

The next thing is the control characters at the end of each data line.

1. When MUSIC wants data (a read), it sends a *read prompt* to the terminal. This can be up to 3 bytes long and is defined by the PRSEQ parameter in the TERMINAL macro for that terminal type. By default this is a DC1 character (X-ON).

2. To send data to MUSIC simply send the data followed by a carriage return character (CR). The data may be up to 80 bytes long in general or up to 250 bytes long for conversational reads (read 9). One should keep in mind that each conversational read requires a full time slice, thus if large volumes of data are to be transferred it would be more efficient to use spooled input (SYSINR) which, though it accepts only 80 byte records, does not require user region service for every record.

3. When MUSIC sends data to the terminal it is usually terminated by a sequence of control characters which is dependent on the operation. These control sequences may be defined in the TERMINAL macro. Examples are: What to send:- after a line (ENDLN), after a folded line (FOLDLN), after a read is completed (READLN), etc.. These sequences are usually a combination of carriage return, line feed, and X-OFF (CR/LF/DC3). The most common are:

after a read	CR/LF
after a complete line	DC3/CR/LF
after a folded line	CR/LF

   For example take a terminal in *Go mode when a user types /users.

a) terminal sends	/users CR
b) MUSIC sends	CR/LF
c) MUSIC sends	001 USER SIGNED ON DC3/CR/LF
d) MUSIC sends	DC1 (read prompt)
e) the terminal may send the next line.	

   MUSIC, by default, translates reverse ASCII to EBCDIC on input and vice versa on output. Sometimes it may be required to bypass this translation and send/receive raw characters directly from a program. On output the X'41' carriage control can be used to prevent translation. On input, calls to the subroutines NOTRIN and TRIN can be used to turn translation off and on.

ASCII to S/370 Code

Many installations will want to tailor their MUSIC system to support some special ASCII terminal features. Often these features are triggered by sending some special sequence of characters. It is therefore important to understand how the characters are translated by MUSIC and the hardware so that the appropriate controls will be correctly sent.

The bit patterns received by the control unit are assembled into bytes before they are sent to the processing unit. ASCII codes are made up of 7 character bits plus one parity bit. These ASCII character bits are labeled from the left as 7, 6, 5, 4, 3, 2, 1. The control unit reverses the order of these bits and has the parity bit being the right most bit after inversion. For example the letter W is 1010111 in ASCII. The control unit reverses the bit order to 1110101. If the character was sent with even parity, then a right most bit of 1 would be added to form the string 11101011 which would be sent to the processing unit as the hex bytes EB. MUSIC's input translation table will then convert this to E6 which is the character W in EBCD code. The reverse process occurs on transmission.

MUSIC has a facility to bypass the MUSIC translate table on output. That is done by using the special hex carriage control of 41 as documented in the *MUSIC/SP User's Reference Guide*. The transmission control unit's translation cannot be bypassed. This is usually of little concern. Note, however, that the hex character of DF is recognized by the control unit as an idle character. It never transmits an idle character down the

line.

The following table shows each of the 128 ASCII codes and how they are converted by the transmission control unit. The three last columns show the byte stored in main storage depending on whether even, odd or mark parity is used. (Mark parity is the parity bit always on.)

CHAR	ASCII	EVEN	ODD	MARK
NUL	0000000	00	01	01
SOH	0000001	81	80	81
STX	0000010	41	40	41
ETX	0000011	C0	C1	C1
EOT	0000100	21	20	21
ENQ	0000101	A0	A1	A1
ACK	0000110	60	61	61
BEL	0000111	E1	E0	E1
BS	0001000	11	10	11
HT	0001001	90	91	91
LF	0001010	50	51	51
VT	0001011	D1	D0	D1
FF	0001100	30	31	31
CR	0001101	B1	B0	B1
SO	0001110	71	70	71
SI	0001111	F0	F1	F1
DLE	0010000	09	08	09
DC1	0010001	88	89	89
DC2	0010010	48	49	49
DC3	0010011	C9	C8	C9
DC4	0010100	28	29	29
NAK	0010101	A9	A8	A9
SYN	0010110	69	68	69
ETB	0010111	E8	E9	E9
CAN	0011000	18	19	19
EM	0011001	99	98	99
SUB	0011010	59	58	59
ESC	0011011	D8	D9	D9
FS	0011100	39	38	39
GS	0011101	B8	B9	B9
RS	0011110	78	79	79
US	0011111	F9	F8	F9
SP	0100000	05	04	05
!	0100001	84	85	85
"	0100010	44	45	45
#	0100011	C5	C4	C5
$	0100100	24	25	25
%	0100101	A5	A4	A5
&	0100110	65	64	65
'	0100111	E4	E5	E5

CHAR	ASCII	EVEN	ODD	MARK
(	0101000	14	15	15
)	0101001	95	94	95
*	0101010	55	54	55
+	0101011	D4	D5	D5
,	0101100	35	34	35
-	0101101	B4	B5	B5
.	0101110	74	75	75
/	0101111	F5	F4	F5
0	0110000	0C	0D	0D
1	0110001	8D	8C	8D
2	0110010	4D	4C	4D
3	0110011	CC	CD	CD
4	0110100	2D	2C	2D
5	0110101	AC	AD	AD
6	0110110	6C	6D	6D
7	0110111	ED	EC	ED
8	0111000	1D	1C	1D
9	0111001	9C	9D	9D
:	0111010	5C	5D	5D
;	0111011	DD	DC	DD
<	0111100	3C	3D	3D
=	0111101	BD	BC	BD
>	0111110	7D	7C	7D
?	0111111	FC	FD	FD
@	1000000	03	02	03
A	1000001	82	83	83
B	1000010	42	43	43
C	1000011	C3	C2	C3
D	1000100	22	23	23
E	1000101	A3	A2	A3
F	1000110	63	62	63
G	1000111	E2	E3	E3
H	1001000	12	13	13
I	1001001	93	92	93
J	1001010	53	52	53
K	1001011	D2	D3	D3
L	1001100	33	32	33
M	1001101	B2	B3	B3
N	1001110	72	73	73
O	1001111	F3	F2	F3

CHAR	ASCII	EVEN	ODD	MARK
P	1010000	0A	0B	0B
Q	1010001	8B	8A	8B
R	1010010	4B	4A	4B
S	1010011	CA	CB	CB
T	1010100	2B	2A	2B
U	1010101	AA	AB	AB
V	1010110	6A	6B	6B
W	1010111	EB	EA	EB
X	1011000	1B	1A	1B
Y	1011001	9A	9B	9B
Z	1011010	5A	5B	5B
[	1011011	DB	DA	DB
\	1011100	3A	3B	3B
]	1011101	BB	BA	BB
↑	1011110	7B	7A	7B
_	1011111	FA	FB	FB
`	1100000	06	07	07
a	1100001	87	86	87
b	1100010	47	46	47
c	1100011	C6	C7	C7
d	1100100	27	26	27
e	1100101	A6	A7	A7
f	1100110	66	67	67
g	1100111	E7	E6	E7
h	1101000	17	16	17
i	1101001	96	97	97
j	1101010	56	57	57
k	1101011	D7	D6	D7
l	1101100	36	37	37
m	1101101	B7	B6	B7
n	1101110	77	76	77
o	1101111	F6	F7	F7
p	1110000	0F	0E	0F
q	1110001	8E	8F	8F
r	1110010	4E	4F	4F
s	1110011	CF	CE	CF
t	1110100	2E	2F	2F
u	1110101	AF	AE	AF
v	1110110	6F	6E	6F
w	1110111	EE	EF	EF
x	1111000	1E	1F	1F
y	1111001	9F	9E	9F
z	1111010	5F	5E	5F
{	1111011	DE	DF	DF
split bar	1111100	3F	3E	3F
}	1111101	BE	BF	BF
~	1111110	7E	7F	7F
DEL	1111111	FF	FE	FF

Chapter 8. Job Submission and Retrieval Programs

Overview of Job Submission

This chapter contains information regarding the internals and setup of the various programs involved in the submission of batch jobs and processing output from these jobs. The SUBMIT command is used to submit jobs to MUSIC batch or other virtual machines, job output can be retrieved and inspected using the OUTPUT command, and the PRINT command is available to schedule printing.

The following is a brief summary of the programs covered in this chapter:

- SUBMIT - Submit a job to batch

- OUTPUT |
- PRINT |
- AUTOPR |- Process printed output
- VMREADX |
- $ROUTING |

- VMSUBM - Spool a reader file to VM

- VMPRINT - Spool a print file to VM

Customizing SUBMIT

The SUBMIT program can be used to submit jobs to MUSIC batch or to any other batch machines accessible through VM. The SUBMIT program can be invoked from either *Go mode or the Editor through the SUBMIT command. Details of the SUBMIT command, the /INFO statement and the parameters involved are documented in the *MUSIC/SP User's Reference Guide*.

Setup Requirements

The SUBMIT program requires that spooled punches on addresses 011, 012, and 013 be defined in the VM directory. Information about which system a job is submitted to, job control statements, tag information, and the like are contained in special *model-JCL* files stored under the code $JCL. Files already exist on the distributed system to enable submission to MUSIC batch. If batch jobs are to be submitted to other batch machines, model-JCL files must be set up for that purpose.

Setting Up Model-JCL Files

The model-JCL for the SUBMIT command is contained in Save Library files under the code $JCL. The file names are of the form "$JCL:JSUB.procname", where *procname* is the processor name that the user would specify on the /INFO statement or the *TO* parameter of the SUBMIT command. With the exception of the model-JCL to submit a job to MUSIC batch, these files must be set up by the installations system support personnel.

There are basically four parts to the model-JCL file. A system definition statement containing information

about the system to which the batch job is to be sent, a symbol table describing the parameters that are substituted into the JCL, and the template statements for header and trailer JCL. These are described in detail after the example below. Figure 8.1, which is from the file $JCL:JSUB.MUSIC and is the model-JCL used when submitting a job to MUSIC, is intended to serve as a reference.

```
----+----1----+----2----+----3----+----4----+----5  COLUMN NUMBER

*        JM0                                                    <--system
&CODE          01 C  01 16 05 =CODE                             <--
&SUB           01 C  00 08 04 =SUB                                |
&USERID        04 C  01 16 07 =USERID                             |
&TIME          01 N  03 03 00000010 00000001 00000999             |
&PAGES         02 N  03 03 00000500 00000001 00000999             |symbol
&CARDS         02 N  03 03 00000000 00000000 00000999             |table
&PW            02 CR 01 08 03 =PW                                 |
&CLASS         02 C  02 02 02 AA                                  |
&ROUTE         01 R  01 08 06 =ROUTE                              |
&FORMS         02 C  01 08 00                                     |
&COPIES        03 N  01 03 00000001 00000001 00000255             |
&MSG           01 C  01 50 00  Enter message for operator       <--
&/
/ID MUSJOB     &USERID &TI &PA &CA R=&ROUTE C=&COPIES F=&FORMS <--
/PAUSE &MSG                                                       |
/PASSWORD=&PW                                                     |JCL
&/                                                                |
/END                                                              |
&/                                                             <--
************
```

Figure 8.1 - Sample Model-JCL File

1. SYSTEM DEFINITION STATEMENT (1ST STATEMENT)

   This statement describes the system to which the job is to be submitted.

SPOOLID	COL 1-8	VM logon ID of system to receive the job.
SPOOL CLASS	COL 10	VM spool class to which the data is submitted. For submitting to a MUSIC system, this defines the VM class (normally J) corresponding to submit parameters CLASS(AA). The other possible values of the CLASS(xx) parameter are SA, TA, AO, SO, and TO, which automatically use VM classes S, T, E, F, and G respectively.
MUSIC SYSTEM	COL 11	The character M should be coded if the target is a MUSIC system.
ACCESS	COL 12	See note 1.
EXPAND	COL 13	If the character X is coded, any /INCLUDE statements in the submitted files will be expanded.
TAGINFO	COL 16	If non-blank, a VM tag command will be issued. This is used by RSCS to send jobs to various systems.

2. SYMBOL TABLE (ONE LINE PER SYMBOL)

   SUBMIT allows symbolic parameter substitution in the model-JCL. These parameters can be

overridden by the user when the appropriate keyword parameter is specified on the /INFO statement or SUBMIT command. Symbolic parameters are described one per statement.

&	COL 1	
SYMBOL	COL 2-13	Name of the symbol.
MIN ABBR.	COL 15,16	Number of characters for the minimum abbreviation.
TYPE	COL 18	C-Character, N-Numeric, R-Route info.
	COL 19	R-required, F-Fixed (see note 3).
MIN LENGTH	COL 21,22	Minimum length of value.
MAX LENGTH	COL 24,25	Maximum length of value.
TEXT LENGTH	COL 27,28	Actual length of value (char only).
DEFAULT TEXT	COL 30-80	See note 2.
DEFAULT VALUE	COL 27-34	(Numeric only).
MIN VALUE	COL 36-43	(Numeric only).
MAX VALUE	COL 45-52	(Numeric only).

3. HEADER JCL

   This is separated from the symbol table by a &/ statement. The symbolic variables which are to be replaced are coded as a "&" followed by the valid abbreviation of one of the symbols defined above. If the symbol is not followed by a comma, a bracket, or a blank, it must be terminated by a period.

4. TRAILER JCL

   This is separated from the header JCL by a &/ statement. It is output after all the user data has been sent.

Notes

1. Setting Access restrictions.

   The access restriction field can be set to a value 0 through 8. Zero indicates that anyone can use the procedure. If a value of 1 through 8 is specified, the SUBMIT checks that the appropriate JCL access option is set in the user profile. These options can be set in the user profile using the keywords JA1 through JA8. For example, in order to submit a job using model JCL with an access code of 4, the option JA4 must have been specified in the user's profile for the sign-on code.

2. Default text.

   The default text field can contain the actual default text or one of a number of special keywords described below. The text length field must specify the exact length to be used. A length of zero indicates that there is no default text regardless of anything in the text field.

=USERID	the user's sign-on userid
=CODE	the user's file ownership id
=SUB	the user's subcode
=ROUTE	the user's default output destination (from ROUTE table)
=PW	the user's batch password (from code table)

3. Symbol type.

   A symbol can be defined as *character* or *numeric* or *route information* by placing a C, N, or R in column 18. Character data is checked only for a valid length. Numeric data must fall within the range that has been specified. Also leading zeros will be omitted during substitution, within the limits defined

for the number's length. If a symbol is designated as R (routing information), SUBMIT will verify if the location specified is valid, if the ROUTING table is present in the link pack area.

A symbol can be designated as required by putting an R in column 19. If the user does not specify a value for this symbol on the SUBMIT command or /INFO statement, SUBMIT will prompt the user to enter a value. The text for the prompt is taken from the default text field. The text length field should be set to zero in this case to indicate that no default value is present and that the actual data appearing in the text field is to be used as a prompt. The first character of the prompt is used as a carriage control. If the carriage control is specified as "?", the response area will be blanked out in a manner similar to the password area at sign-on time. Only character variables can be designated in this way.

A symbol can be designated as fixed, (i.e. the user can't change it) by putting an F in column 19. This is usually used with the special default such as =CODE and =ROUTE so that SUBMIT will put in the appropriate value and at the same time disallowing the user from changing it.

Processing Print Files and Batch Output

Output sent back to MUSIC from batch jobs, or created by the PRINT command, is managed through a single *output queue* by a group of utility programs. The internals of this *output queue* and the utility programs are detailed in this topic. Figure 8.2 illustrates the various components.

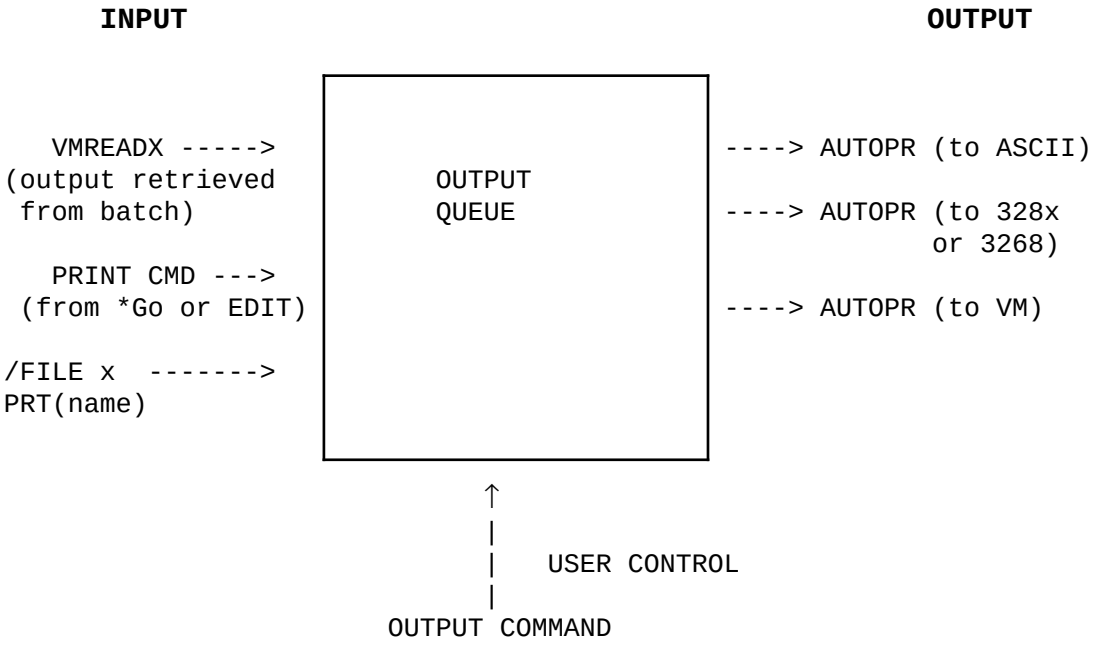


Figure 8.2 - Output Queue Components

The PRINT command can be used from *Go mode or the editor to schedule a file for printing at one of the printers defined to the system. This command creates a *print* file and places an entry into the output queue indicating who owns it and where it is to print.

The VMREADX program can also create print files. Operating systems such as CMS, VSE, VS1, MVS, and other MUSIC systems, can be set up to send job output back to a MUSIC system. Output to be processed by VMREADX should be spooled to MUSIC as class M files. VMREADX puts this output into *print* files and updates the output queue indicating that it is to be held for the user.

The OUTPUT command allows a user to inspect any files that are being held for the user or are waiting to print. The user can then purge these held files or schedule them to be printed at any of the printers defined to the system. Supported printers include local 3268 and 328x series printers, local or remote ASCII printers (with or without keyboards) and VM virtual spooled printers. The printers are driven by a service program AUTOPR which scans the output queue for files to print and sends the appropriate data stream to the printer in question.

Central to the entire subsystem is the OUTPUT QUEUE file which is effectively a list of all *print* files in the system. Each entry in this list contains the owner's sign-on code and the location code of the printer for which the file is destined. The actual output data is kept in separate Save Library files. These files are all stored under a special user-ID of $PRT and do not count against the owner's allocation.

System Components

There are six major components in this system. Later on, each will be described in detail, indicating what must be done to get the various components working in the required environment.

OUTPUT QUEUE	Contains a list of all *print* files in the system. (In file $PRT:PQ).
OUTPUT command	Allows user to inspect and print any output that is being held for the user. (Source in $PGM:OUTPUT.S).
PRINT command	Allows user to schedule the printing of a Save Library file at one of the printers on the system. (Source in $PGM:PRINT.S).
PQ command	Allows user to get a list of what files are scheduled to print on a given printer. (Source in $PGM:PQ.S).
AUTOPR program	Runs as a BTRM or AUTO-SIGNON program and drives the printers. A device dependant routine is called to do the actual I/O. (Source in $PGM:AUTOPR.S).
VMREADX program	Runs as a BTRM program and puts any print file spooled via VM by other operating systems into the output queue. (Source in $PGM:VMREADX.S).

Output Queue

The OUTPUT QUEUE is kept in a Save Library file ($PRT:PQ). Each Q entry corresponds to a logical record in the file. Since direct access is used, the file has record format F. Each entry contains information as to who owns the file and where it is to print. The actual data portion of the print file is kept in a separate Save Library file whose name is derived from the entry's position in the OUTPUT Q file. For example the data for 309th entry in the OUTPUT Q is kept in the file $PRT:PQ.00309. The first record in the file (entry 0) is maintained as a pointer to the next free entry. When this pointer reaches the last entry, it is reset to point to the beginning of the file. This has the effect of making the OUTPUT Q logically a circular file. The last entry in the file MUST contain only FF's (hex). Free entries contain 00's (hex). All access to the OUTPUT Q is done via the subroutine QSCANX and its entry points. (See $PGM:QSCANX.S for details.) The format of each used entry is described in file $MCM:PQENT.M.

Enlarging the OUTPUT Q

The OUTPUT Q file distributed with the system has room for about 2000 entries. This should be sufficient for most installations. The file can easily be enlarged using the editor by inserting empty entries (i.e. entries containing X'00's) just BEFORE the LAST entry. The maximum number of entries that should be allocated

in the OUTPUT Q file is 99999. The following is an example to add 1000 empty entries:

```
*Go
edit $prt:pq;last;up;xin;insert 00;ain;dup 999;save
```

Note: This can ONLY be done when NO ONE else is accessing the Q. (i.e. VMREADX and AUTOPR programs must be cancelled prior updating the print queue file.)

Output Management Facility

The OUTPUT Management Facility allows you to inspect the batch output queue from a terminal. To use this facility enter "output" in command mode. For a description of this facility press F1 (help) after the facility is invoked or refer to the *MUSIC/SP User's Reference Guide*.

The PRINT Command

The PRINT command is used to send a file to a printer. The user's file is copied to a print file and an entry placed in the output queue. The print file will have a logical record length of 133 with carriage control added if required. The output will be printed whenever the service program (AUTOPR) handling the particular printer gets around to it. The format of the print command is as follows:

```
PRINT filename R(location) CC
```

filename is the name of the file to be printed. This is the only parameter that need be specified. Routing information can be given via the R parameter where *location* is the printer location name. CC should be specified if carriage control characters are provided by the user. If the installation has provided a ROUTING table in the Link Pack Area, the PRINT program will use it to set the default location (printer name), and also to verify that any user specified *location* is valid.

Configuring the AUTOPR Program

This program runs the auxiliary printers. A copy of the program is set up to run on each of the printers defined on the system. It scans the MUSIC print queue until it finds some output routed to its printer, prints the output, and deletes it from the queue. The program must always be run under the code $MON unless special authorization for some other code is granted through the Routing Table. AUTOPR requires the FILES and SYSCOM privileges. If it is run from the file $PGM:AUTOPR, these are granted as program privileges, otherwise the code running the program must have these privileges.

The program can be used to drive dedicated printers on specific ports. These could be ASCII or 328x/3268 or 3262 (models 3 and 13) devices. These devices should be set up to be automatically signed on using the SIGNON parameter on the device specification for that address in the NUCGEN jobstream. The auto sign-on feature will attempt to sign on the code $MONsss, where the subcode *sss* corresponds to the device address of the terminal or printer in question. Before the printer will function, the above code and subcode must be allocated in the code table, and the appropriate AUTOPR program set as the auto-program for that code. The code must be assigned the privileges FILES, MAINT, LSCAN, and SYSCOM, and must have unlimited time: PRIME(NL), NONPRIME(NL), DEFTIME(NL), BATCH(0), PW(*NOLOGON). See the topic "Defining BTRMs and Auto Signon" in *Chapter 22 - System Programming* for a more detailed

discussion of the auto sign-on feature. 328x/3268 printers should be specified as 3270 devices in the NUCGEN. ASCII printers should be specified as TTYs. Examples are given below.

The program can also be used to send the print file from the queue to a user as mail. The recipient userid/ nickname is determined from the FORMS field for this output job, the ID=xxxx (where xxxx is an e-mail address or nickname or userid) in the TAG field for the route name MAIL in the system routing table, or is the owner of the output data. This can be used to send AUTOSUB output as mail to a user. No physical terminal is required, and the AUTOPR program is set up to run as a BTRM type terminal in the NUCGEN jobstream. On a BTRM, the system will attempt to sign on the code $MONsss, where the subcode sss corresponds to the device address specified for the BTRM in the NUCGEN. Before output data can be sent to users as mail, the above code and subcode must be allocated in the code table, and the appropriate AUTOPR program set up as the auto-program for that code. See the topic "Defining BTRMs and Auto Signon" in *Chapter 22 - System Programming* for a more detailed discussion on BTRMs.

The program can also be used to spool the print files from the queue to the VM system printer, RSCS, or other virtual machines. The *location* information is used to route the file to the appropriate VM printer. Each location name maps to a specific VM USERID. If the USERID is RSCS the location information is passed to RSCS via the print file's TAG. No physical terminal is required, and the program is set up to run as a BTRM type *terminal* in the NUCGEN jobstream. One such *terminal* can service ALL the different VM printers. On a BTRM, the system will attempt to sign on the code $MONsss, where the subcode *sss* corresponds to the device address specified for the BTRM in the NUCGEN. Before the printer will function, the above code and subcode must be allocated in the code table, and the appropriate AUTOPR program set up as the auto-program for that code. See the topic "Defining BTRMs and Auto Signon" in *Chapter 22 - System Programming* for a more detailed discussion on BTRMs. When you generate the MUSIC system a BTRM running AUTOPR is setup on address 010 to spool print files to VM using a virtual printer on address 017.

The program reads two parameter lines from unit 5. The first one is shown below. (The second one depends on the entry given on the TYPE parameter.)

LOCS='name1','name2',...	The location name(s) defined for the particular printer. Printers can be assigned more than one name. Different printers can also be assigned the same name. For example if two printers were to service the same terminal room they should be given the same location name.
WAIT=nn	The number of seconds to wait if it finds that a print file is busy and there is nothing else to print.
TYPE='xxxx'	The type of printing device. This can be... -328X (328x/3268 type printers) -3286 (3286 model 1, no CR or FF support) -TTY (ASCII async printer) -VM (a VM spooled printer) -MAIL (used for ROUTE MAIL support)
USERS='c1','c2'...	A list of up to 100 user codes that are to be authorized to send files to this printer. If a user is not authorized the print file is just deleted. If not specified all users can send files to this printer.

When TYPE='VM', a second parameter statement must be included describing the virtual printer(s) to be used. Normally the VMID and CLASS parameters need not be specified, since this information is taken from the Routing Table entry for the named printer.

UNITS=Zuad1,Zuad2,...	Unit address(es) of the virtual printer(s). Normally one printer is sufficient. These printers must also be defined in MUSIC's VM directory as virtual 1403 devices and must not have the same address as the batch printer which is defined in the NUCGEN. The "Z" indicates that the address is given in

	hexadecimal. The default is Z17 (hex 17) and is not the same as BATCH.
VMID='id1','id2',...	The VM USERIDs of the virtual machines to which the location names correspond. These must be entered in the same order as the printer location names specified in the LOCS parameter. Files routed *name1* will be spooled to *id1*, files for *name2* will be spooled to *id2* etc. By default *id1* is SYSTEM, the VM system printer, and the rest are set to RSCS. The values specified in VMID are used only if this information is not available from the Routing Table. The values specified in CLASS are used only if this information is not available from the Routing Table.
CLASS='c1','c2',...	VM print classes corresponding to the USERIDs specified in VMID. The default is class A.

When TYPE='3286', the second parameter statement must be included describing the length and width of the forms.

LINLEN=n	The number of characters in a line. The default is 132.
PAGELN=n	The number of lines on a page. The default is 66.

Examples

When creating the auto-program files for AUTOPR great care must be taken in entering the parameters, especially the quotes and the commas. It is recommended that you sign on to the code and verify that the auto-program starts up all right, before attempting to run the real printer. (To sign on to that userid, you must specify a password other than "*NOLOGON".)

1. In this example AUTOPR is set up to spool output to the VM system printer as class A print files.

   - Allocate the code $MON SC(010) specifying AUTO1 as the auto-program, and privileges FILES, MAINT, LSCAN, and SYSCOM, and unlimited time: PRIME(NL), NONPRIME(NL), DEFTIME(NL), BATCH(0), PW(*NOLOGON).

   - Set up the file $MON:AUTO1 to contain the following.

     ```
     /INCLUDE AUTOPR
     LOCS='SYSTEM',WAIT=20,TYPE='VM'
     VMID='SYSTEM',UNITS=Z17,CLASS='A'
     ```

   - Define a BTRM on address 010 by adding the following device card to the NUCGEN.

     ```
     BTRM 010
     ```

     Note: These are set up by default when you install MUSIC.

2. In this example AUTOPR is set up to spool output to RSCS printers REMOTE1 and REMOTE2 as class B print files. Output for CENTRAL is spooled to the VM system printer as class A files and output for CMS1 is sent to the VM user CMS1 as class X files. The files are spooled via a virtual printer on address 017.

   - Allocate the code $MON SC(010) specifying AUTO1 as the auto program, as in example 1.

- Set up the file $MON:AUTO1 to contain the following.

```
/INCLUDE AUTOPR
LOCS='REMOTE1','REMOTE2','CENTRAL','CMS1',WAIT=20,TYPE='VM'
VMID='RSCS','RSCS','SYSTEM','CMS1',
UNITS=Z17,CLASS='B','B','A','X'
```

- Define a BTRM on address 010 by adding the following device card to the NUCGEN.

```
BTRM 010
```

- Add the names REMOTE1, REMOTE2, CENTRAL, and CMS1 to the ROUTING table.

3. A 1200 baud ASCII printer on address 047 which has location name of ROOM112.

   - Allocate the code $MON SC(047) specifying AUTO47 as the auto program, as in example 1.

   - Set up the file $MON:AUTO47 to contain

```
/INCLUDE AUTOPR
LOCS='ROOM112',WAIT=20,TYPE='TTY'
```

   - Add the following device card to the NUCGEN jobstream and do a NUCGEN. ASCII printer should always be specified as DIALUP if they have no keyboard, since if DIRECT is specified MUSIC waits for an interrupt from the keyboard before starting output.

```
TTY 047 DIALUP,1200,SIGNON
```

   - Add the name ROOM112 to the ROUTING table.

4. A 3287 printer on address 0DC which is known as PRINTER6.

   - Allocate the code $MON SC(0DC) specifying AUTODC as the auto program, as in example 1.

   - Setup the file $MON:AUTODC to contain

```
/INCLUDE AUTOPR
LOCS='PRINTER6',WAIT=20,TYPE='328X'
```

   - Add the following device card to the NUCGEN.

```
3270 0DC DIRECT,SIGNON
```

   - Add the name PRINTER6 to the ROUTING table.

5. In this example AUTOPR is set up to send mail to user $000 if the route is MAILX and to the output data owner if the route is MAIL.

   - Allocate the code $MON SC(010) specifying AUTO1 as the auto-program, as in example 1.

   - Set up the file $MON:AUTO1 to contain the following.

```
/INCLUDE AUTOPR
LOCS='MAIL','MAILX',WAIT=20,TYPE='MAIL'
```

   - Define a BTRM on address 010 by adding the following device card to the NUCGEN.

```
BTRM 010
```

- Add the names MAIL and MAILX to the ROUTING table. These entries would be defines as follows:

```
ROUTE NAME=MAIL,TYPE=MUSIC,CLASS=A
ROUTE NAME=MAILX,TYPE=MUSIC,CLASS=A,TAG='ID=$000'
```

6. The AUTOPR utility, used for printing files and job output, has support for the HP LaserJet 4Si printer. It is considered a TYPE=VM printer, and is specified by the value 1 in the new XTYPE parameter. Here is an example of AUTOPR control statements for handling a system printer name and two HP 4Si printers called HP and HP2:

```
/INC AUTOPR
LOCS='SYSTEM','HP','HP2',TYPE='VM'
VMID='SYSTEM','RSCS','RSCS',
UNITS=Z17,CLASS='A','A','A',
XTYPE=0,1,1
```

AUTOPR Notes

AUTOPR is usually run under the code $MON. It can be run from another user code if the code is authorized by the system administrator in the ROUTING table. The program will only process files that have been sent to the user's printer and will not allow access to other print files in the system. This mechanism allows users to operate their own printers. However, since regular users do not have access to the auto sign-on feature, this technique is limited to ASCII printers with keyboards.

The following are the most common causes of problems with AUTOPR.

- The userid $MONsss is not correctly defined in the code table. This can be checked by signing on to the code from a terminal and verifying that the auto-program starts correctly and does not produce error messages. (To sign on to that userid, you must specify a password other than "*NOLOGON".) Make sure that sss is defined as the subcode: ADD $MON SC(sss) etc.

- The parameters specified for AUTOPR are incorrect. This can be checked by signing on to the code from a terminal and verifying that the auto-program starts correctly and does not produce error messages.

- The BTRM or auto-signon terminal is not correctly defined in the NUCGEN. Check that the device address specified in the NUCGEN matches the subcode assigned to $MON.

- The program appears to be running but no files are printing. If printing via VM, check that a virtual printer is defined, in addition to the batch printer, whose address matches the UNITS parameter. This printer should be defined as a virtual 1403. If printing to an ASCII or 328x printer, check that the device is correctly connected to the system.

Configuring the VMREADX Program

VMREADX processes VM spool files that are send to the MUSIC/SP virtual machine. Print files are put into the output queue. Reader files are put into a reader queue where they are processed by MUSIC/SP's MAIL Facility.

VMREADX replaces the older VMREAD program. The major difference between the two programs is in their handling of electronic mail from outside the system. VMREAD delivers mail in the old MEMO format. VMREADX uses the MAIL Facility. In this respect the two programs are not compatible. If, during some conversion period, you wish to use the MEMO program to receive mail from outside the system, use VMREAD instead of VMREADX. Do not specify the RDRCOD parameter if VMREAD is used. Other parameters are the same.

The following information pertains to VMREADX's processing of print files. The *jobname* field in the Q entry is set from the FILENAME field of the spool file. This is important, as MUSIC bases the ownership of a print file on the first four characters of this *jobname*. VSE and VS1 can be set up to set this FILENAME field automatically from the assigned jobname. Under CMS, SPOOL and CLOSE commands can be used to set this name field. If a route destination of MUSIC is specified for a MUSIC batch job, MUSIC will automatically spool the output back to itself, setting the name field directly from the user code of the batch job. Other systems like MVS may have to be slightly modified to use the NAME parameter when closing VM print files. If the FILENAME field is blank, *NO-OWNER is used. The print file will by default be added to the print queue with a printer location name of HOLD, indicating that the file is being held for inspection by the user. If the spool file's FILETYPE field matches a MUSIC printer location name the file will be sent directly to the printer.

A copy of VMREADX must be run for each spool file class you wish to process. Normally two copies of VMREADX are required, on for class M and one for class A. Class M is used for print files and mail. Class A is used for mail.

No physical terminal is required, and the program is setup to run as a BTRM type *terminal* in the NUCGEN jobstream. On a BTRM, the system will attempt to sign on the userid $MONsss, where the subcode *sss* corresponds to the device address specified for the BTRM in the NUCGEN. The userid $MONsss should be allocated in the code table with the appropriate VMREADX program set up as the auto-program. VMREADX requires that the code have the MAINT, FILES, and SYSCOM privileges, and unlimited time: PRIME(NL), NONPRIME(NL), DEFTIME(NL).

The program reads the following parameters from the input stream.

SYSCOD='$PRT'	Code under which the print files are to be stored.
RDRCOD='$VMR'	Code under which the reader files are to be stored.
DELAY=nnn	Number of seconds to wait between output scans. Since VMREADX wakes up automatically when a file arrives this can be set to a large number.
READER=Zaddr	Unit address of the virtual reader VMREADX will use ("Z" means hexadecimal). This reader should be defined in MUSIC's VM directory.
DAYS=nnn	Number of days that output is to be kept before it is automatically deleted. This applies only to print files.
MSGS=n	Output unit number for messages from this program or 0 meaning send messages to the operator console. Normally n would be 0 or 6.
TRACE=n	Used in a debugging situation to set the tracing level. If n is 0 no tracing is done.

Example

The following example is how VMREADX is set up by default after MUSIC has been installed. The BTRM on address 011 is used for VMREADX. By default VMREADX looks for a virtual reader on address 018. Note that the BTRM address is in no way related to the address of the virtual reader.

- The userid $MON SC(011) should be allocated with AUTO2 specified as the auto-program, and with privileges FILES, MAINT, LSCAN, and SYSCOM, and with unlimited time: PRIME(NL), NONPRIME(NL), DEFTIME(NL).

- The file $MON:AUTO2 should contain

```
/INCLUDE VMREADX
SYSCOD='$PRT',RDRCOD='$VMR',READER=Z18,DAYS=7,DELAY=600
```

- The NUCGEN job should have a BTRM defined on address 011.

- The VM directory entry for MUSIC/SP should have a class M reader defined on virtual address 018.

Notes

Any print file spooled to MUSIC class M will be processed by the VMREADX program and placed in the MUSIC output queue. The DIST field specified on the CLOSE command for the print file will be used to indicate file ownership. The FILETYPE field can optionally be specified on the close command to send the file directly to one of MUSIC's printers. Systems wishing to send output to MUSIC need only fulfill these conditions.

CMS — Issue the appropriate SPOOL and CLOSE commands, for example:

```
SP PRT CONT
... Write output to printer ...
SP PRT MUSIC CL M NOCONT
CLOSE PRT DIST userid NAME jobname prtname
```

where *jobname* is to be the jobname associated with the MUSIC print file, *prtname* (optional) is the name of a MUSIC printer, and *userid* is the first 8 characters of the MUSIC file ownership id (the userid without the subcode, if any). If *prtname* is omitted, the file is held in MUSIC's output queue.

VS1 — If VS1 is generated with the VM HANDSHAKING option, it will automatically close any print file using the JOBNAME as the name field on the CLOSE command. Generate a virtual printer to handle the output to go to MUSIC. This printer can be setup to process only a special SYSOUT class, or associated with a specific location. The user would send output to MUSIC by specifying the special SYSOUT class or printer location in the JCL. When the VS1 machine logs on, the appropriate VM commands should be issued to spool this printer to MUSIC Class M. The VM DIST (distribution) field should be set to the first 8 characters of the MUSIC ownership id (the userid without the subcode, if any).

VSE — If VSE is generated with the VSE-VM/370 Linkage enhancements, it will automatically close any virtual print files. As with VS1, a specific printer should be chosen for MUSIC output and the appropriate commands issued when VSE is logged on to spool this printer to MUSIC Class M.

The following are the most common causes of problems with VMREADX.

- The code $MONsss is not correctly defined in the code table. This can be checked by signing on to the code from a terminal and verifying that the auto-program starts correctly and does not produce error messages.

- The parameters specified for VMREADX are incorrect. This can be checked by signing on to the

code from a terminal and verifying that the auto-program starts correctly and does not produce error messages.

- The BTRM is not correctly defined in the NUCGEN. Check that the device address specified in the NUCGEN matches the subcode assigned to $MON.

- VMREADX appears to be running but the spool files are not being processed. Check that the address specified in the READER parameter matches that of a virtual reader and that the reader is spooled to correct class (M or A).

- The reader files disappear but nothing shows up in the output queue. Check that MUSIC's batch reader, (the one defined in the NUCGEN), is spooled to class J. If it is spooled to class M, A or *, the files will be processed by batch rather than VMREADX and deleted since they are not valid batch jobs.

Setting up the Routing Table

The load library module $ROUTING must be set up to provide a table of valid printer location names. When this module is included in the LPA, programs such as SUBMIT, PRINT and OUTPUT, use this table to set default locations and to verify that any user specified location is valid. If it is removed from the LPA, no checking can be done and users can specify any eight character sequence as a printer name.

The subroutine ROUTE (see $SUB:ROUTE.S for details of calling sequence) is provided to access this information. It has three functions: to return the default printer name for a specific terminal; to verify if a printer name is valid; and to determine if a user is authorized to run the AUTOPR program for a specific printer.

The $ROUTING module has provision for a system default location, usually the machine room printer. There is also a table of printer names versus REAL terminal address range. This can be used to set a default printer name for terminals in a particular terminal room or building, such that when a user submits a job, output is automatically returned to the location associated with the terminal from which the job was submitted. Printer location names which are not associated with particular terminals must also be included in this list with a dummy address range (X'FFFF'-X'FFFF') if they are to be recognized as valid by the ROUTE subroutine.

The REAL address of the terminal is used in the $ROUTING module. For remote 3270's, this is the line address of the control unit, thus all terminals on that control unit will have the same real address. If a terminal is connected via VM PASSTHRU, the *logical device* address assigned by PASSTHRU is used. This may be different each time the terminal connects via PASSTHRU.

This REAL address is contained in the RDEV field of the TCB and is listed by the WHOALL program. The real address was chosen over the virtual (MUSIC's) address due to the fact that the virtual address can be different, each time the terminal *dials* in from VM.

The default table defines SYSTEM and MUSIC as valid printer names. Output sent to SYSTEM will be sent to VM's system printer. Output sent to MUSIC will be placed on hold in the output queue for inspection by the user via the OUTPUT command. If other printer names are required they must be added to the routing table. The module containing the table is generated from assembler language source by the following procedure.

1. Edit the file $PGM:$ROUTING.S to fill in the appropriate address ranges and routing information. The comments in the source explain the table format. All locations to which printed output can be directed should be included in the module.

2. Assemble the module by executing the file. The object deck will automatically be saved.

3. Linkedit the module by executing the file $PGM:$ROUTING.LKED.

4. Place the new table in the load library by executing the file $PGM:$ROUTING.LD.

5. The new routing table will be loaded in to the LPA at the next IPL.

The system utility program ROUTETABLE can be used to display the contents of the routing table. See the program description in *Chapter 17 - System Utility Programs*.

Chapter 9. Electronic Mail Facility

Overview of the Mail Facility

The MAIL Facility allows you to send and receive electronic mail to and from other computer users. To use this facility enter "MAIL" in command mode. For a description of how to use this facility press F1 (help) once the facility is invoked or refer to the *MUSIC/SP Mail and Conferencing Guide*.

Configuring MUSIC's MAIL System

This section is intended for system programmers and administrators who are responsible for configuring the MUSIC Mail program so that it can communicate with systems outside of MUSIC. If MAIL is to be used only internally, within a MUSIC system, the configuration process is still required to set up the RDMAILER BTRM to handle recurring, postdated and forwarded mail, but you can ignore the information about external mail, VM, RSCS, SMTP, and mailers.

Although communicating with the variety of outside mail networks is in principal quite simple, the large number of options available makes choosing a particular set of options a daunting task for those unfamiliar with current electronic mail technology. On MUSIC/SP, the MAIL.CONFIG program is used to set the options for the MAIL and RDMAILER programs. A MAILER.PROFILE file should be created and one or more VMREADX programs are setup to read incoming mail from VM. On VM, depending on the configuration, mailer tables may require adjusting, RSCS may require modification, or SMTP may require configuration changes. After describing the options available in configuring the Mail Facility, emphasis is placed on specific configurations that are most common. These configurations are:

1. MUSIC/SP and CMS on a single processor.
2. Existing network connection with modification to RSCS.
3. Existing network connection to the Internet with SMTP configuration changes.
4. Using a VM mailer.

The Postmaster

Each site that runs electronic mail should designate an individual to act in the capacity of postmaster. In general, the postmaster should monitor mail usage and keep up to date on issues concerning electronic mail. This can include the development, dissemination, and enforcement of a mail code of conduct, as well as knowledge of the MUSIC Mail facility, mail networks, and list-servers.

The postmaster must, on a daily basis (or as required), monitor and when appropriate, respond to mail delivered to the postmaster code ($EMP). This is where notification of mail delivery problems and other general inquires about mail and user addresses, from both internal and external mail users, is deposited. Mail sent to "postmaster" will be placed in the $EMP mail box. A useful procedure is to make the individual's (postmaster) userid a surrogate for $EMP. This will allow viewing and sending of mail on behalf of the postmaster userid (code), without having to sign on to it.

The Postmaster Mail Filter program can be set up to automatically handle mail in the postmaster's mailbox. This program can unsubscribe users from mailing lists and perform a number of other tasks. See the topic Mail Filter Program later in this chapter.

The Postmaster is known by the MUSIC/SP Mail facility by a number of names or userids. All mail addressed to the following userids is delivered to the postmaster: postmaster, postmast, system, mailer, maint, support, root, and all userids starting with $EM.

The postmaster should carefully monitor the file usage by the mail system. This can be done with the MSTAT program (described later). It is recommended that the MAIL.CLEANUP program be run monthly. However, monitoring may suggest a more frequent schedule.

The postmaster should review the following topics and help determine the site's requirements.

- Review this chapter on the Mail facility so as to have a broad understanding of the components and flow of the mail facility and other related issues.
- Review RDMAILER and MAIL parameters with consideration to customizing it for your site.
- Review and become familiar with using the MAIL program, with the expectation that you will have to consult and educate your user community on mail usage.
- Review the items on the ADMIN 4 5 Configure/Administer The Mail Facility menu.

- The setting of prime time and the allowed size of mail text files to be send to remote sites during prime time.
- The system wide expiry date. This is a default amount of days used to set the mail's expiry date. It is very useful in keeping file usage down.
- User-specific limits for expiry date to override the system wide expiry date on a user or code table TYPE basis.
- User-specific mail item limits which limit a user to viewing a number of mail items and forcing the user to delete some of the viewed items before more items from the mailbox are presented.
- A site email policy and add it to the MAIL FAQ item on Mail's main menu (item A).
- Review the delivery of autoforwarded, postdated, and recurring mail since these items are given to RDMAILER for delivery (ie they reside in $EMD's mailbox until they are delivered, expire or are expired). You may need to deal with these types of mail.

Mail System Components

There are a number of components that work together to provide access to electronic mail networks. Some of these are part of MUSIC/SP, others run under VM.

Figures 9.1 and 9.2 later show the relationships between the components.

MAIL

The MAIL program is the user interface to the electronic mail system. Internal mail is managed through a system of mailboxes, text files, and distribution files. By default, these files are maintained on the userids $MBA to $MBG, and $MDAA to $MD99. The mailboxes for individuals have file names of the format $MB?:Buserid where ? can be A to G. The text and distribution files have rather cryptic names that have significance only to the mail system.

When mail is sent outside the system, MAIL transfers the file to a special outbound mailbox designated for outside mail. This mailbox has the file name $MBE:B$EMD or $MBD:B$EMD000 depending on whether the namelist parameter PCODE is set to $EMD or $EMD000. Note that outbound address checking is not

done by MAIL, so errors that occur in delivering the mail are not reflected to the user right away.

Similarly, mail automatically forwarded to another userid either local or remote, or mail that is postdated or recurring, is also transferred to the special outbound mailbox.

BIGMAIL

BIGMAIL is a version of MAIL with a 3000K region size.

RDMAILER

The RDMAILER program runs in the background as a BTRM and is responsible for sending mail outside the system and also for receiving mail from the outside. It also handles postdated mail, recurring mail and forwarded mail. RDMAILER controls the special outbound mailbox. When it finds an item in this box it verifies the destination address and spools the file to the appropriate service machine on VM for ultimate delivery. Usually SMTP, RSCS, or a mailer virtual machine provides this service. If RDMAILER determines that it can't deliver the mail it sends an error to the original sender, and depending on the type of error, will also send a note to the *postmaster*, which by default is the userid $EMP. In the case where a VM mailer or SMTP is being used to deliver the mail, RDMAILER will usually leave the verification of the destination address up to the VM mailer or SMTP respectively.

Mail coming from the outside eventually arrives on MUSIC/SP as a virtual reader file. The program VMREADX (see below) places these files into the reader queue used by RDMAILER. RDMAILER uses information from the queue entry, the file's tag, and the file's content, to determine where the file is from and to whom it should be delivered. If a file can be delivered to a local user, an entry is added to the appropriate mailbox. If the mail cannot be delivered, a message is sent to the sender if possible, and depending on the severity of the problem, the postmaster is informed.

The operation of RDMAILER is influenced by the parameters specified to the MAIL.CONFIG program and by the contents of the file $EML:MAILER.PROFILE. MAIL.CONFIG defines the local environment: the local time zone, the node name of the MUSIC/SP machine, the name(s) of the RSCS machine(s), and the local node name of the processor. $EML:MAILER.PROFILE defines the external electronic mail network. It is the same format as the mailer profile used by the Columbia VM Mailer on BITNET. This is described later.

From 1 to 10 auxiliary RDMAILER BTRMs can be configured to deliver mail received from the outside (ie incoming mail processing). See the sections Configuring Mail and Mail Administration for further information on this topic.

VMREADX

VMREADX runs in the background as a BTRM. It reads in virtual reader and print files spooled to MUSIC/SP from other virtual machines running under VM. It puts incoming reader files into MUSIC/SP's reader queue under the userid $VMR. It is from this reader queue that RDMAILER gets the incoming mail. Incoming print files are put into the output queue under the userid $PRT. One consequence of the design is that VMREADX can only process one class of files at a time, so if you have mail arriving on both class A and M, (not an unusual situation), you will have to run two copies of VMREADX, one to process class A and one for class M.

Note: VMREADX is the enhanced version of the VMREAD program used with the original version of MAIL. When VMREADX is installed, the BTRM that runs the old VMREAD program should be disabled by deleting its $MONxxx userid or removing the BTRM specification for VMREAD from the NUCGEN. (xxx - BTRM device address.)

RSCS

RSCS is the component of VM that transfers electronic mail between physical machines in a network of VM computer systems. One drawback is that the transfer mechanisms of RSCS assume a two level addressing scheme of USERID/NODE where the NODE represents a particular CPU, and USERID is the VM logon ID of the virtual machine. No allowances are made for the fact that the virtual machine, for example MUSIC/SP, can support hundreds of users. Two solutions are available.

1. Operating above the RSCS transport mechanism is a standard electronic mail format in which the sender and recipient are identified in the envelope and text of the mail itself. Thus, a remote site need only use RSCS to deliver mail to the MUSIC/SP virtual machine. RDMAILER figures out who the recipient is from the mail envelope. In this case there is no need for any modifications to RSCS. This method is recommended for new MUSIC/SP sites. Those that are already defined on BITNET may find it more convenient in the short term to keep using method 2 (below) rather that redefine their system to the network.

2. McGill offers some suggested modifications to RSCS to allow the MUSIC/SP virtual machine to appear as a distinct NODE in the RSCS network. Sites who have applied these modifications to allow their MUSIC/SP system to communicate with BITNET and other networks can leave them in place and replace MEMO with MAIL without changing their sites definition in the various tables that describe the network.

VM Mailers

A number of programs are available to provide network electronic mail services to VM processors through an RSCS network. Most support an electronic mail format that consists of BSMTP commands and addresses comprising the mail envelope and RFC822 headers in the mail text itself. This standard allows a variety of networks to exchange electronic mail.

If your site runs a mailer program, the MUSIC/SP MAIL system can take advantage of the services it provides. RDMAILER will simply send any outbound mail to the VM Mailer. The mailer verifies the recipient address, provides the appropriate envelope and routing, and passes the file to RSCS. If a problem occurs, the mailer will send a message to the sender on MUSIC describing the problem. In order for MUSIC/SP to use the mailer's facilities, you must usually identify the MUSIC/SP virtual machine to the mailer in some way.

The Columbia VM Mailer is very popular on BITNET. It uses a "MAILER PROFILE" file to describe the network topology. The VM node and userid of the MUSIC machine must be entered in the "INCOMING:" section of this file, if the mailer is to accept mail from MUSIC/SP for delivery to the network. An entry must also be included in the "OUTGOING:" or "DOMAINS:" section of the file if the mailer is to deliver mail to the MUSIC/SP system. Since other network nodes must also have this information, you should contact the network coordinators (BITNIC) to have your MUSIC/SP node defined to the network. There are a number of advantages to using the VM mailer as the gateway to the outside world, the most important being that you only have to maintain one set of network topology tables.

If you do not run a mailer service machine on VM, RDMAILER can provide all the required services.

SMTP

SMTP is the Simple Mail Transfer Protocol client and server virtual machine on VM that transfers electronic mail to/from other SMTP machines in the Internet. SMTP is a part of the VM TCP/IP program product (5735-FAL).

SMTP can be set up to communicate with MUSIC/SP's RDMAILER program. This allows MUSIC/SP users

using the MAIL program to exchange mail with others on the Internet.

There are a number of things to consider in this setup. See the Sample Configurations section later and the file $TCP:TCPIP.SMTP.DOC for further details.

This setup does not use the TCP-to-RSCS Gateway interface for SMTP.

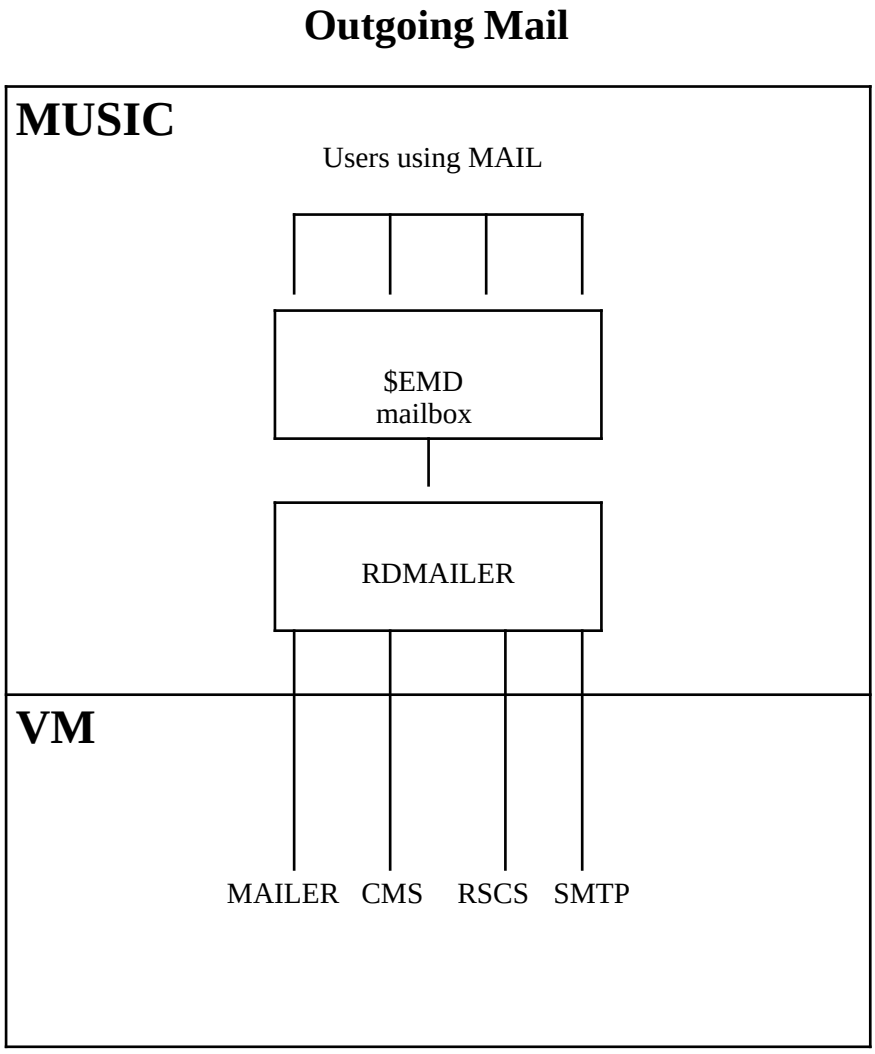


Figure 9.1 - Information flow for outgoing mail.

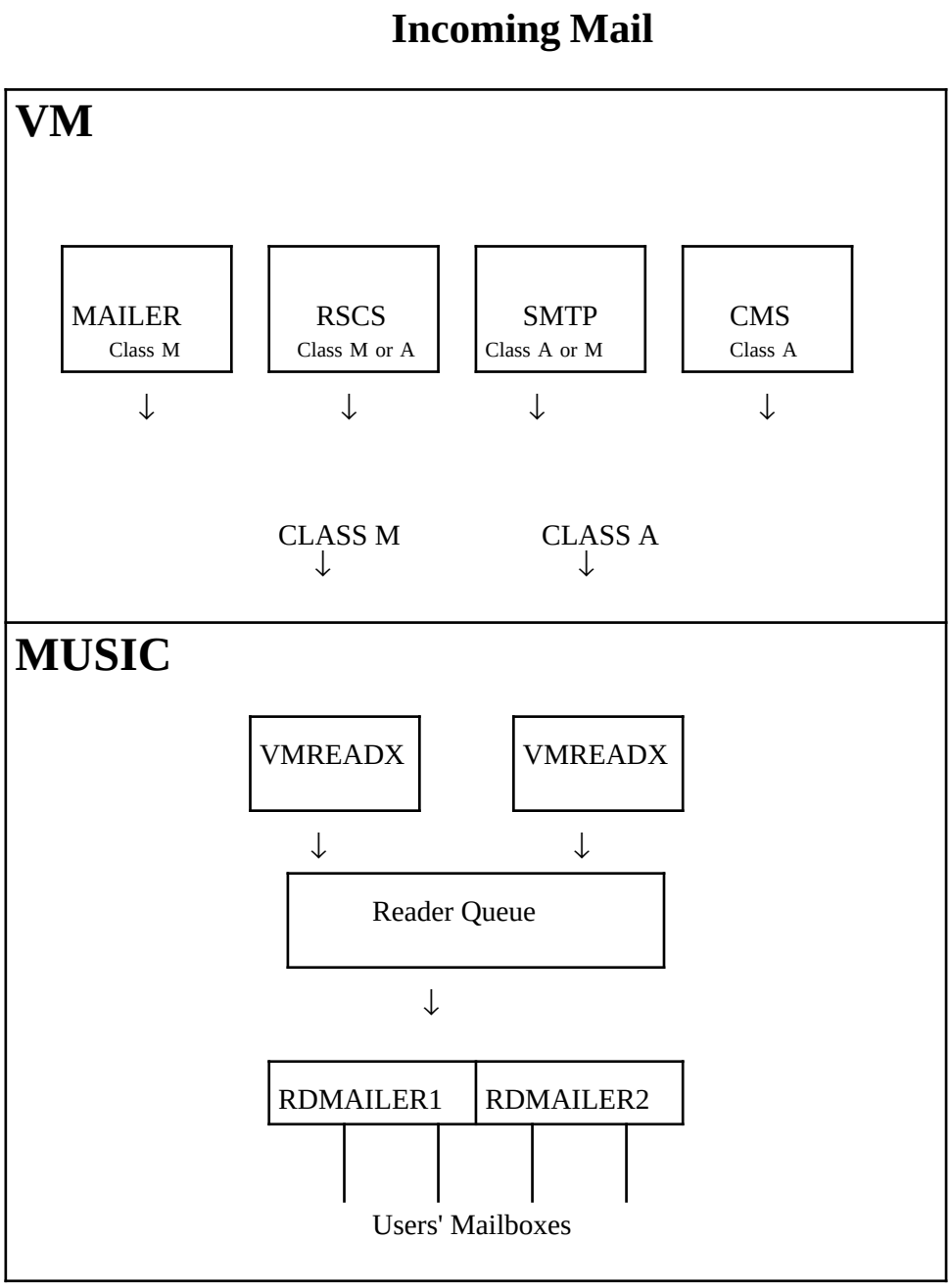


Figure 9.2 - Information Flow for Incoming Mail

SENDFILE

SENDFILE is used to send a copy of a file to another MUSIC or CMS user on your computer, or on other computers that are connected to yours via RSCS. Sendfile does not currently use the nickname file. Therefore, you will have to specify the exact userid and VM node name. See the SENDFILE command in the *MUSIC/SP User's Reference Guide* for usage details.

Files sent to MUSIC via SENDFILE are received via MAIL. The mail item has a subject line specifying it is a sendfile.

GETMINFO

GETMINFO invokes the GETMAIL utility program with the namelist parameter INFO=TRUE. This program gets a list of information for the user's incoming mail items and stores this list to a file.

GETMINFO has an entry in the system LOOKUP table with the FILES and CODES privileges to access the user's mailbox which is not stored on the user's code.

At program termination, the end of job return code is set via CALL EOJ(x).

GETMAIL

GETMAIL invokes the GETMAIL utility program with the namelist parameter INFO=FALSE. Thus, this program gets your incoming mail and stores the text in a file.

This program has an entry in the system LOOKUP table with the FILES and CODES privileges to access the user's mailbox which is not stored on the user's code.

At program termination, the end of job return code is set via CALL EOJ(x).

SENDMAIL

SENDMAIL invokes the SENDMAIL utility program which is the fast-track method to send a piece of mail. SENDMAIL requires that the text you wish to send already exist in a file.

This program has an entry in the system LOOKUP table with the privileges FILES, LSCAN, CODES, and SYSCOM to do the mail send.

At program termination, the end of job return code is set via CALL EOJ(x).

$EML:PREMBK

PREMBK is a user friendly, front end to the Mailbook program that allows users to enter a mailbook file name. Then it calls the Mailbook program. PREMBK is invoked from the MAIL main menu (option 7) if a file name is not entered. If a file name is entered with the option, the Mailbook program is invoked directly.

MAILBOOK

MAILBOOK invokes the MAILBOOK utility program which allows you to view your mail logs (also known as notebooks, mailbooks or folders) that you have created previously by either copying your outgoing mail to a file or "XLOGDELing" your incoming mail to a file. You can enter the file name of the mailbook when you invoke the program. If you do not, the PREMBK program will be invoked where you can enter a mailbook filename. The MAILBOOK program can also be invoked from the MAIL main menu (option 7). The MAILBOOK program user interface has the same look and feel as the MAIL program.

The MAILBOOK program is configured with MAIL.CONFIG. See the section "Configuring Mail" for further details. The MAILBOOK program can be used as a general viewer for files which have a separator line (73 equal signs) between items. The MAILBOOK program can also read mail list digests where the mail item is composed of a number of individual mail items. The namelist parameters COPY=t or f, DELETE=t or f, and POST=t or f can be used to control those functions within the program. COPY allows or disallows copying the item to a file. DELETE allows or disallows deleting the item from the mailbook file. POST allows or disallows all send functions of the MAILBOOK program. POST=t also sets

DELETE=f. Setting POST=t would make this a POST only mailbook.

MAIL.CLEANUP

The MAIL CLEANUP program, $EML:MAIL.CLEANUP, is designed to delete mail text and distribution files that have expired. See the topic "Running the MAIL.CLEANUP Program" for more details.

MAILBOX

This program displays the mailbox filename for a given userid. If the userid is not given, the mailbox filename for the signed-on user is displayed. ADMIN 4 5 7 can be used to run this program. The program is invoked manually as follows:

```
MAILBOX userid
```

$EML:DIRECT

The DIRECT program, MAIL main menu option 5, allows you to create, change, and remove entries from your private directory file, @NAMES. This file is used by the MAIL facility to enable you to refer to private nicknames without having to remember userids.

$EML:DIRECT.PRINT

The DIRECT.PRINT program is used by the DIRECT and DIRECT.PUBLIC programs to create a readable copy of the directory in the file @NAMES.LISTsss which the user can print on the printer of their choice. If the userid has a subcode "sss", then it is suffixed to the file name. Otherwise, the file name is @NAMES.LIST.

$EML:DIRPUBL

DIRPUBL is a public program that allows users to browse the public directory. MAIL main menu option 6 invokes this program.

$EML:DIRECT.PUBLIC

$EML:DIRECT.PUBLIC program allows you to create, change, and remove entries from the public directory file, $EML:@NAMES. This file is used by the MAIL facility to enable you to refer to public nicknames without having to remember userids.

It is important to understand the search order of the nickname files. When a user sends mail to someone, the MAIL facility attempts to resolve the name given for that person. The name is resolved by searching the user's nicknames file, then searching the public nicknames file, and then trying the name as a userid (ie the system code table). If the name is still not resolved to a userid(s), the user is told that the name is invalid.

$EML:DIRECT.PUBLIC works exactly like the Mail Directory program, DIRECT, also known as the private directory program. Thus, the description of that program in the Mail and Office Applications Guide can be used as a reference.

There is no default public directory. You must create one for your site using $EML:DIRECT.PUBLIC if you wish to create public nicknames.

$EML:DIRECT.PUBLIC requires the FILES privilege to run. ADMIN 4 5 5 can be used to run this program.

$EML:URL

The URL program is invoked when a URL hotspot in MAIL or MAILBOOK is processed. The URL program parses the URLs and calls the appropriate client program to process the URL. The MAIL help topic URL gives further details of URL support and the Mail facility.

$EML:FMAIL

FMAIL is a version of the MAIL program that exits the MAIL program when a user returns to the MAIL main menu from any other MAIL function. This program can be used to present a tailored view of the MAIL program. See the MAIL Administration section for the topic "How to create your own MAIL menu" for further information.

$EML:LM

LM, also known as LIST MANAGER, is a facility that allows users to manage their subscriptions to BITNET and Internet mail lists. MAIL main menu option 8 invokes this facility.

XTELL

The XTELL command is the same as the TELL command except that the message is displayed on the user's screen even if the user has suppressed messages by issuing the /MESSAGE OFF command.

Configuring MAIL

1. Allocate BTRMs to run VMREADX and RDMAILER

Add the following BTRM specifications to the NUCGEN job and create a new nucleus.

```
BTRM 013-017
```

This adds five BTRM's, three for RDMAILER and two for VMREADX. The virtual addresses shown (013-017) need not be used. Just make sure that the ones you choose do not correspond to any real devices and that they match the addresses later specified in the MAIL.CONFIG program.

2. Allocate a Virtual Reader

Allocate a virtual reader for each of the VMREADX programs by adding the following entries to MUSIC's VM directory.

```
SPOOL 18 2540 READER M
SPOOL 19 2540 READER A
```

The above example uses 18 as a class M reader and 19 as a class A reader. Note that the addresses of the virtual readers must NOT be the same as the BTRMs.

3. Run MAIL.CONFIG

Before running this program cancel any running BTRM programs. If this is not done you may get "FILE IN USE" messages when the MAIL.CONFIG attempts to generate the MAIL program files. These files include:

- $EML:MAIL
- $EML:MPROF
- An auto program file for RDMAILER
  0 to 10 auto programs for auxiliary RDMAILERs
- Two auto program files for VMREADX
- $EML:RDMAILER
  0 to 10 $EML:RDMAILERnn for auxiliary RDMAILERs
- $EML:GETMAIL
- $EML:GETMINFO
- $EML:SENDMAIL
- $EML:MAILBOOK
- $EML:MAIL.CLEANUP
- $EML:PHONEX
- $EML:DMSDDL
- $PGM:TELL
- $PGM:XTELL
- $TDO:SCHED
- $TDO:MEET
- $TCP:POPD
- $TCP:POPPASSD
- $MCS:MCSH
- $EML:QPUTI
- $EML:NETCNV

Fill in the fields on the Configuration screen to indicate all the Mail parameters for your site. Each field (Mail parameters) is described below.

ACCESS — The programs that use this parameter are: MAIL, MPROF, RDMAILER, GETMAIL, GETMINFO, SENDMAIL, MAILBOOK, and PHONEX. This parameter defines who can use the MAIL program. Enter a name for the file containing the authorization table. The default file name is $EMD:MAIL.AUTHOR and it allows access for all. For information about the authorization table layout see the topic "Authorization Table".

AFWDON — The AFWDON=f parameter can be used to not allow users to set the "Forward mail to" field in their Mail Profile. The "Forward mail to" field is used in the MAIL facility to automatically transfer mail sent to this userid to the given addresses in this field. This parameter can be used when the site wants all verification of sent mail to be done on a system different from the local MUSIC system. AFWDON=t allows the "Forward mail to" field to be set. AFWDON=t is the default. Note that SNOOP/VIP users can override this feature.

ALIAS
MUSNOD — The programs that use this parameter are: MAIL, GETMAIL, GETMINFO, MAILBOOK, MPROF, RDMAILER, SENDMAIL, PHONEX, SCHED, and MEET. Enter alternate names (up to 4) for your MUSIC system domain name. The name variable can be up to 132 characters in length. There is no default. This allows your MUSIC system to receive mail addressed to it, under the name specified in MUSNOD or one of those specified in ALIAS1-4. These names would have to be domain names registered with the Internet, BITNET, or known to a mailer that has been designated to resolve domain names for your MUSIC system. For example, if the domain name was MUSICX.DOMAIN.NETWORK, then the alias could be given as MUSICX or MUSICX.DOMAIN.NETWORK.

ALTSYS — The ALTSYS=f parameter can be used to not allow users to set an alternate systemid in their Mail Profile by using the F2:Alt System or ALTSYS command. ALTSYS=t allows an alternate system to be set. ALTSYS=t is the default. Note that SNOOP/VIP users can override this feature.

BITNET — The RDMAILER program uses the BITNET parameter to send email to any remote sender using a local BITNET email address. This parameter can aid a site in its migration from BITNET to Internet email addressing by advertising the change to these senders. BITNET=T sends email. BITNET=F is the default and it does not send email.

When BITNET=T, RDMAILER executes the file $EML:BITNET to send mail when it receives mail destined to a local BITNET email address. The body of the mail is the contents of the file $EML:@NOTICE. Either of these files can be tailored to meet the site's needs.

CLASS — The CLASS parameter for RDMAILER specifies the spool class for the virtual punches used for spooling mail text to other systems, such as VM. The default is class M.

CODTBL — The CODTBL=t parameter can be used to force the MAIL program to use the CODE TABLE NAME field as a profile name if a profile name is not present in the mailbox. The default is CODTBL=t. This parameter is used by the MAIL, MPROF, MAILBOOK, SENDMAIL, and RDMAILER programs.

CONSOL — When true, messages about POPD, POPPASSD, and RDMAILER activity will be sent to the MUSIC console. The default is T.

COPY — COPY=t allows users to copy files both as a function of MAILBOOK and with the VIEW store command. If COPY=f, the COPY select code does not appear in the list, and both the copy function in MAILBOOK and the VIEW store command are not allowed. The default is COPY=t.

DBCS — The DBCS=t parameter is used for double byte character set support within the MAIL program. The default is DBCS=f.

DEADX — The programs that use this parameter are: MAIL, GETMAIL, GETMINFO, and SENDMAIL. This parameter removes dead records from mailboxes. Enter a percentage between 0 and 100. If the percentage of dead records in the mailbox is higher than your value, then the mailbox is compressed to remove the dead records. DEADX=20 is the default.

EPRIME — The RDMAILER program uses this parameter to define the end of prime time. The default is EPRIME='7 pm'. The time can be 1 to 12 characters long. RDMAILER will not release large mail files during prime time. The SPRIME parameter specifies the start of prime time and SIZE specifies the length for large files.

ERRFIL — The programs that use this parameter are: GETMAIL, GETMINFO, MAIL, MAILBOOK, and SENDMAIL. Enter a name of a file for storing the error messages for each program. The default file name is $EML:MAIL.MSGS.

EXPIRE — The RDMAILER program uses this parameter to set the expiry date for inbound mail. Enter the number of days to be added to the date of the mail received. The MAIL and SENDMAIL programs use this parameter to set the default expiry date for mail sent. Enter the number of days to be added to the current date. The MAIL, GETMAIL, and GETMINFO programs also use EXPIRE to expire mail in the mailbox older than the current date plus the expire value. Enter the age value for which you want mail kept in the mailbox. The default is 365 (days) or one year. It can take on values of 0 (no limit set) to 36500 days.

When EXPIRE is set to 0, a default expire date of one year after the mail release date is assumed, however the user can always change this value. A user or a group of users can be assigned a value to override this default. See the documentation on the mail authorization file for further information to set up overrides.

EXPIRE can also take on values between -1 and -1216 which represent the number of 30 day months. Thus if EXPIRE=-3, EXPIRE would take on the value of 90 days.

GATRSC The GATRSC parameter is used to define a domain name that can be added to any email address ending in .bitnet or with only a one part domain name that is 1 to 8 chars long. This can be used by a site to have mail sent to email addresses on BITNET automatically converted to an Internet email address. Then it is delivered to that Internet host which can deliver to the BITNET node. For example, assume the site had only Internet connectivity and its domain name was "somewhere.edu" and it wanted to allow its users to have easy access to BITNET email addresses. If the site defined GATRSC as "INTERBIT.CREN.NET", users could send mail to user@nodename. The email address would be transformed to user%nodename@INTERBIT.CREN.NET, and RDMAILER would deliver the mail to "INTERBIT.CREN.NET".

Some sites may not want to configure GATRSC as described above. With VM's SMTP, if a user sends mail to user@address, SMTP can send it to user@address.domain_name, where domain_name is defined within the SMTP profile on VM. If this method is used at the site, GATRSC should be left blank.

INSWCH The INSWCH parameter defines the number of incoming mail items that RDMAILER will process consecutively before giving outgoing mail processing a chance. The range is 5 to 1000. The default is INSWCH=25.

KILFIL The RDMAILER program uses the filename defined by this parameter as a kill file. The default filename is $EML:KILLFILE. The contents of the kill file are defined in the ADMIN facility, option 4 5 B "Update the RDMAILER kill file". The kill file contains email addresses of senders. When RDMAILER encounters an incoming mail item from a sender that matches something in the kill file, RDMAILER deletes the mail item without taking any other action. In other words, the mail item is killed and not delivered.

Note: Kill file processing can slow down incoming mail delivery by RDMAILER. It is best to keep the kill file to the minimum number of entries required.

KILSIZ This parameter defines the number of characters you want to set aside for kill file entries. The default is KILSIZ=16000 allowing for 200 entries of 80 characters each. Note that because KILSIZ is given as a character count, you must define KILSIZ for the maximum number of characters you expect in the kill file. For example, if you have 100 entries in the kill file, KILSIZ=8000 is required (i.e. 100*80=8000).

LOG The programs that use this parameter are: MAIL, POPD, POPPASSD, RDMAILER, and SENDMAIL. This parameter is used to keep a log of all mail transactions coming from and going to MUSIC. The information is written to the log file by the LOG server BTRM. Set to T if logging is to take place. The default is F. See the MAIL Logs section under "MAIL Administration" for further information.

MAILER This parameter is used with the RDMAILER program. It specifies a 1 to 8 character name for the VM mailer. VM's SMTP can be the VM mailer so use SMTP's virtual machine name here. There is no default value. Leave this field blank if a VM mailer or VM's SMTP is not being used. This field and the MNODE parameter must be specified if a VM mailer or VM's SMTP is being used as the mailer.

MAXRCD MAXRCD defines the number of Received: headers RDMAILER will allow. This protects against network and local mail loops, bouncing the rejected item to the local postmaster and the sender. If the number is negative, only the Received: headers above the body of the mail will be counted. If the number is positive, all Received: headers regardless of placement in the mail text (provided that it begins in column 1) will be counted. The allowed values are -5 to -99 and 5 to 99. 0 means do not check. The default is -30.

MAXRS MAXRS defines the number of re-sents RDMAILER will allow. This protects against network and local mail loops, bouncing the rejected item to the local postmaster and the sender. If the number is negative, only the resent statements above the body of the mail will be counted. If the number is positive, all resent statements, regardless of placement in the mail text (provided that it begins in column 1), will be counted. The allowed values are 5 to 99. The default is 15.

MBDEL MBDEL=t allows users to delete files as a function of MAILBOOK. If MBDEL=f, the DELETE select code does not appear in the list, and the DELETE function in MAILBOOK is not allowed. The default is MBDEL=t.

MNODE This parameter for RDMAILER specifies the RSCS node name of the processor running the VM mailer or VM's SMTP if it is being used as a VM mailer. This 1 to 8 character name may or may not be the same as VMNOD. Leave this field blank if a VM mailer or VM's SMTP is not being used. This field and the MAILER parameter must be specified if a VM mailer or VM's SMTP is being used as the mailer.

MYNODE This parameter is used by SENDFILE, TELL, XTELL, and NETCNV to set the dummy RSCS node name of your MUSIC system. The default is MUSIC.

NUMRDM NUMRDM is used to represent the number (0 to 10) of auxiliary RDMAILER BTRMs configured for processing incoming mail. RDMAILER must know this in order to determine which entries it should handle in the reader queue. The default is NUMRDM=0, which signifies one RDMAILER BTRM and it handles both incoming and outgoing mail. NUMRDM=1 signifies one RDMAILER for outgoing mail, and one RDMAILER for incoming mail. See the topic on multiple RDMAILER BTRMs in the MAIL Administration section for further information. The NUMRDM parameter is also used by the VMREADX btrms, POPD, MCSH, and QPUTI.

PERIOD The PERIOD parameter for MAIL.CLEANUP deletes mail text and distribution files. Enter the number of days (1 to 365) before the current date for which a mail file will be deleted if it was created on or before this new date. For example, if PERIOD=20 and today's date is 30NOV90, then all mail files created on or before 10NOV90 are deleted. The default is PERIOD=0 which means to delete the file only if the file's expiry date is before the current date.

POST POST=t is meant to be used as a "post only" mailbook (ie send operations are not allowed). Setting POST=t also forces MBDEL=f (i.e. mailbook delete). If POST=f, send operations are allowed on the MAILBOOK items. The default is POST=f.

PSTMST This parameter sets the postmaster userid for MAIL, MAILBOOK, RDMAILER, MPROF, and SENDMAIL. The postmaster userid is where notices of problems and bad or undeliverable mail is sent. The default is "$EMP". This userid must be different from PCODE.

Some sites may have PSTMST set to "$EMP000". This is okay and you do not need to change it to "$EMP". If you do change it from "$EMP000" to "$EMP", you must reconfigure the MAIL facility (ie run Mail Config option 3 to create the executor files) and you must rename the file "$MBF:B$EMP000" to "$MBC:B$EMP" before the MAIL facility is used by any of your users or the RDMAILER BTRM is restarted. This is the postmaster's

mailbox that is used by the MAIL facility.

QFILE This parameter for RDMAILER and QPUTI specifies the name of the reader queue for accessing incoming mail. The default is "$VMR:RQ".

RCLASS Sets the incoming mail class(es) that RDMAILER is to process. This corresponds to the VM spool class of the incoming mail. The default for RCLASS is 'M','A'.

RDREG This parameter defines the REGion size for the RDMAILER BTRMs. The default is RDREG=1024. This parameter allows for sites to define a larger region to accomodate more kill file entries and/or mail items in core.

RELAY The RDMAILER program uses the RELAY parameter to allow or disallow mail to be relayed through your site. RELAY=f disallows mail relaying. The default is RELAY=t. Mail relaying is a mailing method where someone outside your site uses your site to send mail to someone else not at your site. This method is used in mail spamming.

RMTALL The programs that use this parameter are: MAIL, MPROF, RDMAILER, and SENDMAIL. The RMTALL=t parameter can be used to force the sending of all outgoing mail via RDMAILER to another system or remote mailer that will do the send. This can be used when the site wants to verify all sender/recipient combinations on a system different from the local MUSIC system. The normal send method is for MAIL and RDMAILER to verify sender/recipient combinations. In the case where the remote system is not found in the MAILER PROFILE, RDMAILER sends the mail to MAILER at MNODE. See MAILER and MNODE namelist parameters for a description of these parameters. If RMTALL=t, all mail sent by the MAIL program will be put into the post office mailbox for delivery by the RDMAILER program. RDMAILER will send this mail to MAILER at MNODE. If RMTALL=f, all mail sent by the MAIL program will be sent the normal way. RMTALL in MPROF controls whether the user is allowed to use the ALTSYS support to change his/her default systemid. RMTALL=t disallows ALTSYS support, and RMTALL=f allows ALTSYS support. RMTALL=f is the default in all programs.

RMTERR This parameter is used by RDMAILER. When RMTERR=t, the default, bad mail is returned to the sender. When problems occur during the delivery of incoming remote mail, postdated mail, recurring mail, autoforwarding mail, or outgoing remote mail, the sender would be notified. These include bad address, invalid recipient, and other syntax errors.

RSCS This parameter for RDMAILER and SENDFILE specifies your local RSCS virtual machine. This node name of RSCS running on VM can be 1 to 8 characters. The default is "RSCS". If RSCS is not used, specify blanks.

Note: You should list here the names of all RSCS machines you have running on your VM system. This includes any dummy node names related to your implementation of the "RSCS mods" that McGill provides.

SCLASS This value is used internally for RDMAILER to indicate outgoing mail. It has nothing to do with the VM spool class used for outgoing mail (see CLASS). The default is "O". There is no reason to change this value.

SIZE This parameter for RDMAILER specifies the number of lines the mail text file can be in order to be sent during prime time. Prime time is defined by the SPRIME and EPRIME parameters. Mail text that exceeds this will be held until non-prime time is reached. The default is 13108 lines or 1 megabyte of data. SIZE of less than 100 lines will be rejected.

SLEEP Specify a positive integer for setting the sleep or delay value. RDMAILER will remain idle during the specified number of seconds before looking for new work. The default is 600

seconds (10 minutes). Since the programs that feed RDMAILER automatically wake it up, this value need not be set very low.

SMTP — Eight character node name of SMTP running on VM. If you do not have SMTP running under VM, leave this blank. This parameter is used by the RDMAILER program.

SNDTYP — This parameter for RDMAILER sends the MUSIC mail to the default mailer defined by the MAILER parameter. One of the following protocols can be entered:

BSMTP (default)
BITNET
DEFRT
LOCAL

SNOOP — Where 'privilege' is one of the following 'FILES', 'CODES', 'MAINT', 'LSCAN', 'DREAD', 'CREAD', 'VIP', 'INFO', 'SYSMNT', 'SUPV', or 'SYSCOM'. If a privilege is given, all users with this privilege can bypass the surrogate support test and look at anyone's mailbox. They just type the userid in the FOR() field. The default privilege to bypass the surrogate support is 'VIP'. If this value is changed, remember that the user must have at least FILES privilege since the GETMAIL program must write to the mailbox. The Mail Profile facility supports this feature with the FOR name command. This parameter is used by the MAIL, GETMAIL, GETMINFO, MPROF, and SENDMAIL programs.

SPRIME — The RDMAILER program uses this parameter to define the start of prime time. The default is SPRIME='7 am'. *time* can be 1 to 12 characters long. RDMAILER will not release large mail files during prime time. The EPRIME parameter specifies the end of prime time and SIZE specifies the length for large files.

UNITS — This parameter for RDMAILER specifies the device addresses of virtual punches used for spooling mail text to other systems. These devices are normally defined in the VM directory for MUSIC. Up to 10 virtual punch addresses (specified in HEX) can be given. Care should be taken in entering the "Z" to indicate a hexadecimal address. The default is 11 and 12.

VMNOD — This parameter of RDMAILER and SENDFILE specifies the node name of your processor in the VM RSCS network.

ZONE — The programs that use this parameter are: MAIL, MAILBOOK, DMSDDL, RDMAILER, and SENDMAIL. It defines the local time zone. *zone1* is the time zone to use from the start of the year until *date1* or from *date2* until the end of the year. *zone2* is the time zone to use from *date1* until *date2*. Valid values for *zone1* or *zone2* are EST, EDT, CST, CDT, MST, MDT, PST, PDT, UT, or -12 to +12 to give GMT +-12. The dates are in format DDMMM. The defaults are 'EST','EDT','01MAY','01OCT'

Typically *zone1* is the standard time zone, while *zone2* is the day light savings time zone. Some sites may choose to set *zone1* and *zone2* to the same value. *date1* is the day *zone2* is to come into effect. *date2* is the day *zone1* is to come into effect.

4. Special Namelist Parameters

Listed below are a number of namelist parameters that should no longer be used or that are available for the appropriate programs but are not available within the MAIL CONFIG facility.

FCODE — The FCODE parameter sets the file userids for the following programs: MAIL, MAIL.CLEANUP, RDMAILER, and SENDMAIL. The file userid(s) is the userid on which all mail distribution list and text files are stored. Only the first three characters of FCODE are used for the userid. The fourth and fifth characters used for the five character userid are

automatically generated. The default is "$md?", where files are generated on userids $mdAA to $md99, 1296 combinations. This allows mail files to utilize upto 1296 save library index segments or a lesser value, whichever is available. See the MAIL Administration section later for a discussion of this parameter and the Save Library index.

The FCODE parameter should not be changed from the default. The MUSIC system has been coded to skip UCR "accounting" on any userid starting with "$md", thus reducing the number of I/O operations on the system. Changing the FCODE parameter will increase the I/O load on system.

HOLDIN The HOLDIN parameter can be used to create an RDMAILER BTRM which will process only outgoing mail (ie HOLDIN=t). The default is HOLDIN=f.

HOLDOT The HOLDOT parameter can be used to create an RDMAILER BTRM which will process only incoming mail (ie HOLDOT=t). The default is HOLDOT=f.

MYNUM MYNUM is used to indicate which number within NUMRDM does this RDMAILER represent. For example, NUMRDM=5 and MYNUM=2 indicates that there are 5 RDMAILERs configured for processing incoming mail, and this one is number 2. MYNUM is used to determine which entries to handle in the reader queue. See the topic on multiple RDMAILER BTRMs in the MAIL Administration section for further information.

PCODE This parameter is used by most of the mail facility programs: GETMAIL, GETMINFO, MAIL, MAILBOOK, MPROF, RDMAILER, and SENDMAIL. It defines the outgoing mailbox for RDMAILER. The default is $EMD.

Some sites may have PCODE set to "$EMD000". This is okay and you do not need to change it to "$EMD". If you do change it from "$EMD000" to "$EMD", you must reconfigure the MAIL facility (ie run Mail Config option 3 to create the executor files) and you must rename the file "$MBD:B$EMD000" to "$MBE:B$EMD" before the MAIL facility is used by any of your users or the RDMAILER BTRM is restarted. This is the post office mailbox that is used by the MAIL facility.

5. Modify the $EML:MAILER.PROFILE file

You must modify the file $EML:MAILER.PROFILE to suit your particular needs. This file is used by RDMAILER to determine where to send outbound mail. If you intend to use a VM mailer or SMTP to handle all of the outbound mail, this file is NOT required. This file is required when you use RDMAILER as your MAILER. The format used in the Columbia VM Mailer's "MAILER PROFILE" file is used. RDMAILER uses the OUTGOING and DOMAIN sections. If for some reason you require the MUSIC mail system to have knowledge of the entire BITNET topology, the easiest way to create this file is to copy the mailer profile file from VM. Otherwise, you will have to get the files XMAILER NAMES and DOMAIN NAMES from LISTSERV@BITNIC and run $EML:MAILER.PROFILE.MK to make this file. See the description of the $EML:MAILER.PROFILE.MK program later in this section. Normally, however, the file only contains local additions and exceptions.

If you have Internet connectivity but not BITNET connectivity (ie you are running SMTP on VM and not a VM mailer), you do not need to modify the $EML:MAILER.PROFILE file. This way all outgoing mail will be sent to your SMTP virtual machine when it is defined in the MAIL.CONFIG MAILER parameter.

When a user sends mail externally the address is specified as:

```
userid@system
```

RDMAILER uses $EML:MAILER.PROFILE to verify that the "system" exists, to find out how it should get

it there, and to determine what format the mail should be in. If RDMAILER cannot find an entry for the "system" specified, it will pass the mail off to SMTP or a VM mailer if one has been defined. Otherwise the mail is sent back to the sender with an error indicated.

The program $EML:MAILER.PROFILE.MK can be used to make the file $EML:MAILER.PROFILE. See the description of this program under "Mail Utility Programs" for more details.

A sample $EML:MAILER.PROFILE is given below.

```
OUTGOING:
; Alias name     Node     Userid   Exit     Type Parm
; ----------     -------- -------- -------- ---- --------
MCGILL1          MCGILL1  ?        DEFRT    1
MCGILLM          MCGILL3  MUSIC    BSMTP    3
MCGILL2          MCGILL1  MAILER   BSMTP    3
END OUTGOING
; Table of mail domains that I can send to.

DOMAINS:
; Domain name    Node     Userid   Exit     Type Parm
; ----------     -------- -------- -------- ---- --------
MCGILL.CA        MCGILL1  MAILER   BSMTP    3
END DOMAINS
```

Figure 9.3 - Sample Mailer Profile

Note: ADMIN 4 5 4 can be used to edit the file $EML:MAILER.PROFILE. Make your changes using the MUSIC/SP Editor, save your changes, and restart the RDMAILER BTRMs to have the edited file put into use.

The sample shows an OUTGOING list and a DOMAINS list. These lists are headed by the "OUTGOING:" or "DOMAIN:" identifiers and terminated by an "END" statement. Lines that start with ";" are comments. The other lines describe individual nodes in the network. The first column is the node or domain name that is specified by the user sending the mail. The second column is the RSCS node name of the processor where the mail should be sent. This can be different from the first column. The third column is the VM USERID of the virtual machine that is to receive the mail. Normally this would be the VM mailer service machine. If a question mark is specified here, the USERID specified by the sender is substituted.

The fourth column defines the type of mail the recipient is capable of handling.

DEFRT — No mailer is being used. The mail is sent directly through RSCS to the user at the remote site.

BITNET — The receiving mailer can handle RFC822 format mail. The mail is sent to the mailer, who distributes it locally based on the RFC822 headers.

BSMTP — The receiving mailer can handle BSMTP format mail. The mail is sent to the mailer, who distributes it locally based on the BSMTP addresses. BITNET members are required to support BSMTP. This should be the default for non-local entries in this file.

Subsequent columns are not used by RDMAILER but are present in the mailer profile used by the Columbia VM Mailer.

Using the sample in Figure 9.3, if mail was sent to JOHN@MCGILL1, RDMAILER would send the file to userid JOHN on the processor identified to the RSCS network as MCGILL1.

Mail for JOHN@MCGILLM would be sent to the virtual machine MUSIC on the processor MCGILL3. Assuming this is a MUSIC machine, RDMAILER would distribute the mail to the user JOHN.

Anything sent to ...@MCGILL2 would be sent to the the userid MAILER at MCGILL1, regardless of the userid specified. The mailer would handle delivery to the specific user(s).

The domain table is interesting in that an exact match on system names is not required. For example, mail sent to JOHN@VM1.MCGILL.CA would be sent to MAILER at MCGILL1. It would be up to this mailer to figure out the VM1 part.

6. Create a Site Mail Profile

Modify the site Mail Profile if your site has a requirement to create a standard mail profile for all new mailboxes. If a site Mail Profile is not created, new users do not have any Mail Profile options preset and they must set the options themselves.

ADMIN 4 5 3 "Update site Mail Profile" can be used to create/modify a site Mail Profile that is copied to all new mailboxes at creation time. The Mail Profile facility is used to create the site Mail Profile. The site Mail Profile is stored in $EMM's mailbox. Only the Mail Profile from $EMM's mailbox is copied to new mailboxes at creation time. The Name and Email Id fields of the site Mail Profile are not copied to the new mailboxes.

This facility can be used to set any Mail Profile options that should be copied to new mailboxes. For example, the site wants mail sent to be copied to a file automatically. On the "Create and Send Mail Options" screen, simply set the "Copy the mail to be sent to a file" to Y, set a filename where to save the mail to be sent (e.g. MAIL.LOG), and set Append to this file to 'Y'. Filenames specified should probably not be userid prefixed. That way when the Mail Profile in the new mailbox is actually used, the filenames all default to the userid of the new mailbox.

Sample Configurations

Single processor network

In this case mail cannot be sent outside of the processor, but it is possible for mail to be exchanged by MUSIC systems and CMS users running on the processor. There are a number of considerations.

MAIL.CONFIG:

- Since there is no RSCS, SMTP, or VM mailers the fields RSCS, SMTP, MAILER, and MNODE should be left blank.

- Choose a node name for your MUSIC machine(s) and your VM processor by setting MUSNOD and VMNOD. These can be chosen somewhat arbitrarily but should have some meaning to your users. For example MUSIC and VM might be good choices.

$EML:MAILER.PROFILE:

- Define the VM node name in OUTGOING: section of the MAILER.PROFILE file as a DEFRT type node.

- Define any other MUSIC systems as BITNET nodes.

Example: This defines VM and a second MUSIC system called MUSICX.

```
OUTGOING:
; Alias name     Node     Userid   Exit     Type Parm
; ----------     -------- -------- -------- ---- --------
VM               VM       ?        DEFRT    1    TRUNCATE
MUSICX           VM       MUSICX   BSMTP    3
END OUTGOING
; Table of mail domains that I can send to.
DOMAINS:
; Domain name    Node     Userid   Exit     Type Parm
; ----------     -------- -------- -------- ---- --------
END DOMAINS
```

Figure 9.4 - Defining VM and MUSICX

On VM:

CMS users can send mail to MUSIC by spooling a punch file containing the text to MUSIC as a class A or M file. Suitable sender/receiver addresses should be included at the top of the file to identify the sender and the recipient of the mail.

```
To:   USR1@MUSIC
From: USR2@VM
```

The VM NOTE command is the normal method of sending electronic mail in CMS. If the MUSIC userid is used on the NOTE command, NOTE will attempt to send the mail to a CMS user with that userid. If the address of "userid AT MUSIC" is specified, NOTE will try to send the file to RSCS. Even if you are running RSCS, this will fail since RSCS can only send the files outside the system. One solution would be to modify NOTE to handle mail for MUSIC as special and spool it directly to MUSIC instead of RSCS. A simpler solution is to modify the VM file SYSTEM NETID so that the NODEID is VM and the NETID is MUSIC.

Existing RSCS node

McGill provides a modification to RSCS to allow a MUSIC site to look like a node in an RSCS network. The application of this modification is described in the file $EML:RSCSV13.MODS for RSCS V1R3, $EML:RSCSV2.MODS for RSCS V2R2 or V2R3, and $EML:RSCSV3.MODS for RSCS V3R1. ($EML:RSCSV2.TELLMODS and $EML:RSCSV3.TELLMODS describe the required modifications for MAIL plus those required for intersystem TELL support.) The MAIL system can function with this mod in place, so you can switch from MEMO to MAIL without requiring everyone in the network to update their tables to reflect your change.

MAIL.CONFIG:

- Set the RSCS parameter(s) to be the name(s) of your RSCS machine(s).

- Set VMNOD to be the node of your processor in the RSCS network.

- Set MUSNOD to be the node assigned to your MUSIC machine. This is the same as the dummy node name specified in the modification to RSCS.

- If you want to use your VM mailer to handle outbound mail on MUSIC's behalf, set MAILER and MNODE to the appropriate values. If you have a VM mailer service machine it will be to your advantage to do this, since it eliminates the need to maintain the full MAILER.PROFILE tables on MUSIC.

See the following section if this is the case.

- If you want to use your SMTP virtual machine to handle outbound mail on MUSIC's behalf, set MAILER to SMTP and MNODE to the appropriate value.

$EML:MAILER.PROFILE:

- If you are not using a VM mailer or SMTP, the MAILER.PROFILE must contain a list of all the nodes and domains in the network that you wish to send to.

- An up-to-date copy of this information can be obtained from those administering the network.

  The program $EML:MAILER.PROFILE.MK can be used to make the file $EML:MAILER.PROFILE. See the description of this program under "Mail Utility Programs" for more details.

On VM:

External users can send mail to MUSIC using the same techniques that they use to send to any other node in the network. Local CMS users will have no trouble using the CMS MAIL program to send mail to MUSIC. Local CMS users can also send mail to MUSIC by spooling a punch file containing the text to MUSIC as a class A or M file. A suitable sender/recipient address should be included at the start of the file.

```
To:   USR1@MUSNOD
From: USR2@VMNOD
```

In the example above MUSNOD is the RSCS node reserved for your MUSIC system and has nothing to do with the logon ID of the MUSIC virtual machine.

The VM NOTE command has some problems when used by local CMS users. If the VM NOTE command is used and an address of "USER AT MUSNOD" is specified, NOTE will send the file to RSCS since it knows nothing about where MUSNOD is. RSCS will reject the file since MUSNOD is not on its list of external nodes. The NOTE command could be modified to handle the local MUSIC node as a special case and spool the file directly to MUSIC bypassing RSCS. An alternative is to create a special exec that: temporarily creates a SYSTEM NETID file on the user's minidisk pointing to MUSIC instead of RSCS; issues the NOTE command; then gets rid of the SYSTEM NETID file. In this way, NOTE will send the mail directly to the local MUSIC system instead of to RSCS.

Using a VM Mailer

Since the software to maintain the network configuration resides on VM and since a variety of mail handling service machines run on VM, it makes sense that rather than duplicate these functions in the MUSIC system, the MUSIC mail system would take advantage of these services wherever possible.

MAIL.CONFIG:

- Set the RSCS parameter(s) to be the name(s) of your RSCS machine(s).

- Set VMNOD to be the network node name of your processor.

- Set MUSNOD to be the node assigned to your MUSIC machine.

- Set the MAILER and MNODE parameters to point to your local VM mailer. This need not be on the same physical processor as your MUSIC machine.

$EML:MAILER.PROFILE:

- This file should be empty except perhaps for some destinations that for some reason you do not want to go through the mailer.

On VM:

Your VM mailer should be able to handle BSMTP headers since MUSIC uses them to route the mail. Make the appropriate changes on VM to add the MUSIC node to the network topology tables. If you are using the Columbia VM Mailer this involves updating the MAILER PROFILE file. Add an entry for the MUSIC virtual machine to the "INCOMING:" list so the mailer will accept mail from MUSIC as valid. Also add an entry for your MUSIC node in the "OUTGOING:" or "DOMAINS:" list so the mailer will deliver mail to the MUSIC system. Other nodes in the network that want to communicate with your MUSIC system must make the same changes to their MAILER PROFILE files. Normally this distribution of MAILER PROFILE information is coordinated by a central site. (In the case of BITNET, it is node BITNIC.) See the description of the $EML:MAILER.PROFILE.MK program later for further details on obtaining this distribution information if you do not receive it automatically.

Using SMTP

MUSIC MAIL takes advantage of the services SMTP offers and does not need to duplicate these services.

SMTP Configuration on VM:

- Set the MAILER configuration statement for the SMTP virtual machine for your MUSIC system to send all mail to MUSIC. The MUSIC/SP MAIL facility can handle NETDATA conversion. Use the following statement:

```
MAILER MUSIC NEW LOCAL RSCS NOUNKNOWN                    (TCP/IP 2.0)
MAILER MUSIC NETDATA SOURCEROUTES LOCAL RSCS NOUNKNOWN (TCP/IP 2.2)
```

MAIL.CONFIG:

- Set the RSCS parameter(s) to be the name(s) of your RSCS machine(s). No RSCS modifications are required if you are using only SMTP.

- Set VMNOD to be the network node name of your processor.

- Set MAILER to be either the VM mailer program if you are running one, or the SMTP virtual machine name if you are only running SMTP. This parameter is used by outgoing mail.

- Set MNODE to the respective node name of MAILER parameter as defined above. This need not be on the same physical processor as your MUSIC machine.

- Set the SMTP parameter to the SMTP virtual machine name that can deliver mail to MUSIC. This parameter is used for incoming mail.

- Set MUSNOD and ALIAS1 to 4 to be the fully qualified domain names, FQDN, representing your MUSIC system. You MUST also include in this list the node name of your VM system.

$EML:MAILER.PROFILE:

- If you are running SMTP and/or a VM mailer, this file should contain just comments and no entries. In this case, all mail is automatically sent to either the VM mailer or SMTP if there is no match with an entry in this file.

On VM:

External users can send mail to MUSIC using the same techniques that they use to send to any other Internet user.

Local CMS users can also send mail to MUSIC when they use the NOTE and SENDFILE execs distributed with VM TCP/IP program product as replacements for the CMS versions of these execs.

If you are using a VM mailer, it can be configured to communicate with the SMTP virtual machine also.

Passing Commands to RDMAILER

Commands can be passed to any RDMAILER BTRM via the ATTENTION program. While an RDMAILER BTRM is running, it can be told to perform a number of things as well as change the status of certain variables. If your site is running more than one RDMAILER BTRM, you must know the "ENQNAM" of the RDMAILER BTRM with which you want to communicate before you can send it a command. The "ENQNAM"s of RDMAILER BTRMs that handle incoming mail are RDMAILnn, where nn is the namelist parameter MYNUM. The "ENQNAM" of the RDMAILER BTRM that handles outgoing mail is RDMAILER. If you are running only one RDMAILER BTRM for incoming and outgoing mail, the "ENQNAM" is RDMAILER.

By typing "attention ENQNAM" from *Go, where ENQNAM has the value as described above, you can enter the following commands:

HOLD x — Hold the processing of the class specified. If *x* is not specified then all classes are held. *x* can be any of the values specified in RCLASS or SCLASS. HOLD IN/OUT is also supported to do all classes.

STOP — Terminates RDMAILER.

END — Terminates RDMAILER.

CANcel — Terminates RDMAILER.

Quit — Terminates RDMAILER.

RCn x — Sets the character specified by x to be RCLASS, the remotely originating mail class. n is a number from 1 to 5.

Go x — Restarts class x. If no class is specified then all RCLASSes and SCLASS are restarted. GO IN/OUT is also supported to do all classes. If the HOLDIN or HOLDOT namelist parameter is set to TRUE, the GO command is ignored.

SIze n — Sets the file size in lines allowed to be sent during prime time. n must be a positive integer greater than or equal to 100.

SLeep n — Set the integer n as the SLEEP value. n is a positive integer representing time in seconds.

Log ON/OFF — Set logging to on or off.

Next x — Process the next entry in class x. Processing of the other class will be carried on normally. If x is not specified then any RCLASS and SCLASS will be processed, whichever is encountered first in the queue.

Note: NEXT is only valid after a HOLD command is issued. STep x is the same as NEXT.

SClass x	Set the class for sending locally originating mail to the character specified by x.
DOMains	Re-read the domains file and rebuild the list incore.
Show	Prints out the namelist parameters and RDMAILER options and the status of current settings and values.
KILFIL	Displays the kill file entries RDMAILER is currently using. This may not correspond to the entries in $EML:KILLFILE.

VMREADX Parameters

VMREADX handles the initial processing of inbound mail and replaces the VMREAD program. When converting from the old MEMO program to MAIL, VMREAD should be disabled by deleting the autoprogram that is set up to run VMREAD. By default this ran on BTRM address 011.

MAIL.CONFIG automatically allocates the codes and creates the files necessary to run VMREADX. In case there is reason to change the defaults, the program parameters are described below. They should be entered in namelist format as shown in the sample files that are given.

READER	The device address of the virtual reader
DAYS	Number of days to keep old print files
DELAY	Number of seconds to delay before looking for work
TRACE	Print trace and debug messages.
MSGS	Unit number for messages (0 is console)
SYSCOD	Code for print queue ($PRT)
RDRCOD	Code for reader queue ($VMR)
MYNODE	Set to the dummy RSCS node name of your MUSIC system (MUSIC)
NUMRDM	Number (1 to 10) of RDMAILER BTRMS configured for processing incoming mail. (NUMRDM=1)
RWDLY1	Number of seconds to wait between wakeups of RDMAILER after having finished processing all VM reader files and before going to sleep. (RWDLY1=30)
RWDLY2	Number of seconds to wait between wakeups of RDMAILER after processing each punch file. (RWDLY2=60)

The following autoprogram files show recommended setting for these parameters.

File: $MON:A014 (handle the class M reader)

```
/INC VMREADX
READER=Z18,DAYS=20,DELAY=60,TRACE=0,MSGS=0,
NUMRDM=2
```

File: $MON:A015 (handle the class A reader)

```
/INC VMREADX
READER=Z19,DAYS=20,DELAY=60,TRACE=0,MSGS=0,
NUMRDM=2
```

Note: VMREADX replaces all the functions of VMREAD so you should delete the VMREAD BTRM or at least disable it's autoprogram.

See the topic "Running Multiple RDMAILER BTRMs" in the Mail Administration section for further information on this topic.

SMTP Notes

The MUSIC file $TCP:TCPIP.SMTP.DOC provides the necessary information to help you configure SMTP to work with the MUSIC MAIL facility.

If you are connected to both BITNET and the Internet and don't mind or prefer your mail arriving via the Internet link, you should send in an update to your node definition to BITNIC informing them of your domain name and that you no longer require nameserver support for your node. See the file DOMAIN GUIDE at LISTSERV@BITNIC for further details.

If this is your situation, and you do not have a VM MAILER, and you are using $EML:MAILER.PROFILE with the BITNET XMAILER NAMES and DOMAIN NAMES files that are sent every month, then you should change those nodes in the DOMAIN NAMES section of the mailer profile that have been defined to deliver their mail to SMTP at INTERBIT to your SMTP machine. Note that you should only change those nodes which have :interconnect.YES or :interconnect.MX. You can find the interconnect tag for the nodes in the raw DOMAIN NAMES file you are sent each month.

Special File Names

The following special files are used by MAIL. The default PCODE value of $EMD is used where appropriate.

$EML:DIRPUBL.MSGS	contains the DIRPUBL messages. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $EML:DIRPUBL.MSGS.S by the $EML:MSG.BLDR program.
$EML:DIRPUBL.MSGS.S	contains the raw text for the DIRPUBL messages. This file has a VC record format and a record length of 512. It is a text-only file, and it can be edited. This file is used by the $EML:MSG.BLDR program to produce the $EML:DIRPUBL.MSGS files. This file is normally offline and may have to be restored from the source tapes if use is required.
$EML:MAIL.MSGS[@]	contains the MAIL messages. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file has National Language variants E (Spanish), F (French), P (Portugese), K (Kanji-Japanese) supported through the [@] placeholder in the file name. These files are produced from $EML:MAIL.MSGS[@].S by the $EML:MSG.BLDR program.
$EML:MAIL.MSGS[@].S	contains the raw text for the MAIL messages. This file has a VC record format and a record length of 512. It is a text-only file, and it can be edited. This file has National Language variants E (Spanish), F (French), P (Portugese), K (Kanji-Japanese) supported through the [@] placeholder in the file name. These files are used by the $EML:MSG.BLDR program to produce the $EML:MAIL.MSGS[@] files. These files are normally offline and may have to be restored from the source tapes if use is required.
$EML:PROFILE.MSGS	contains the MPROF messages. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $EML:PROFILE.MSGS.S by the

	$EML:MSG.BLDR program.
$EML:PROFILE.MSGS.S	contains the raw text for the MPROF messages. This file has a VC record format and a record length of 512. It is a text-only file, and it can be edited. This file is used by the $EML:MSG.BLDR program to produce the $EML:PROFILE.MSGS file. This file is normally offline and may have to be restored from the source tapes if use is required.
$EML:VIEW.MAIL.COMI	contains a list of the commands available to VIEW for incoming mail.
$EML:VIEW.MAIL.COMO	contains a list of the commands available to VIEW for outgoing/acks/unreceived mail.
$EML:MAILER.PROFILE	contains the MPROF messages. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $EML:PROFILE.MSGS.S by the $EML:MSG.BLDR program.
$EML:MAILER.TEMPLATE	serves as the template for $EML:MAILER.PROFILE. This is used by the program $EML:MAILER.PROFILE.MK.
$EMD:MAIL.AUTHOR	contains the authorization table for send operations and user/type specific expiry dates and mail item view limits. (The PCODE parameter can be used to specify an alternate code other than $EMD.) See the section "Authorization Table" below for information.
$MB?:Buserid	contains the user mailbox. $MBA to $MBG are used to store mailboxes. Use the MAILBOX program to determine the filename of a user's mailbox. See the description of the MAILBOX program under the section "Mail Utility Programs" in this chapter.
$MB?:Ouserid	contains the user overflow mailbox. $MBA to $MBG are used to store mailboxes. The overflow mailboxes contains the mail records that could not be added to the mailbox at delivery time. Invoking MAIL, GETMAIL, GETMINFO, copies the overflow mailbox to the real mailbox.
$MBG:B$EMM	contains the site Mail Profile. System Administrators can use ADMIN 4 5 3 "Update site Mail Profile" to update the Mail Profile for this special mailbox. The Mail Profile from this mailbox is copied to all new mailboxes when they are created.
$EMD:@MAILLOG.yyyymmdd	contains a log of all mail sent and received, and the RDMAILER progress and error messages if LOG=t. *yyyymmdd* is the date the log was started. This is the default mail log filename. This log file is not created if LOG=f. The log file is created by the MUSIC LOG server BTRM. See the MAIL Logs section under MAIL Administration for further information.
$MD??:cccc	are files that contain mail text and distribution files. *cccc??* is a 6 character string based on the date and time of day. See the MAIL Administration section later for further information on these filenames and the Save Library index.
@NAMES	is the name of the file of nickname information created and maintained by the DIRECT program.

$EML:@NAMES	is the name of the file of public nickname information created and maintained by the DIRECT.PUBLIC program.
$EML:MEDITOR	MAIL editor
$EML:MEDITOR.SHOWTEXT	MAIL editor showtext. Users can override the MAIL editor showtext by creating their own MEDITOR.SHOWTEXT file.
MEDITOR.USERCMDS	user MAIL editor commands file. These commands are the last editor commands issued. As an example this file can include commands to redefine F1/F13, F5/F17, or the show text.
$EML:MMSG.MAC	MAIL editor macro to put out a message when the text of the mail item cannot be merged into the text to be resent/forwarded.
$EML:MBKEDITOR	MAILBOOK editor
$EML:MBKMSG.MAC	MAILBOOK editor macro to put out a message when the text of the mailbook item cannot be merged into the text to be resent/forwarded.
$EML:MEDHELP.MAC	MAIL/MAILBOOK editor macro to invoke the MAIL/ MAILBOOK editor help
$EML:SAVE.MAC	MAIL/MAILBOOK editor SAVE macro
$EML:SEND.MAC	MAIL/MAILBOOK editor SEND macro
$EML:SIG.MAC	MAIL/MAILBOOK editor SIGnature macro
$EML:SUSPEND.MAC	MAIL editor SUSPEND macro
$EML:FMRGEDX	Editor for merging in the original mail headers in with the forwarded mail text (MAIL)
$EML:FMBKEDX	Editor for merging in the original mail headers in with the forwarded mail text (MAILBOOK)
$EML:FMEDX.MAC	Editor macro for putting the original mail headers onto the forward mail text (MAIL)
$EML:FMBKEX.MAC	Editor macro for putting the original mail headers onto the forward mail text (MAILBOOK)
$EML:MKNICK	REXX exec used by the MAIL MAKENICK function
$EML:PCEDIT	REXX exec to create a file using an PC editor and transferring the file to MUSIC
$EML:MAGFIL.SUPPORT	describes the "File to be executed when mail arrives" field on the MPROF General Options screen.
$EML:NAMDIR.MAC	Editor macro used by the DIRECT and DIRECT.PUBLIC programs.
$EML:NORECEIVE	contains a list of valid userids that are not allowed to receive mail. The mail

	is returned to the sender as if the userid did not exist. This file is used by RDMAILER. The userids are listed one per line. Wild card characters * and ? are allowed in the userids. RDMAILER must be restarted when this file is changed. This file can be modified by using ADMIN 4 5 6 "Update noreceive file".
$EML:MAIL.MAKE	Exec to remake the MAIL facility programs. This can be run after you have applied source changes and have recompiled/reassembled the source. This program can be run from ADMIN 4 5 8 "Remake Mail facility after local mods".
$EML:MAIL.CHGVNM	Program required to change all of the mailboxes to the new format required for MUSIC/SP 2.4 from the earlier version of the mailboxes.
$EML:MAIL.FILE.COUNT	output of the MSTAT program appended here
$EML:MBK.VIEW.CMDS	contains a list of the commands available to VIEW for MAILBOOK
$EML:MAIL.IND.DESC	describes the IND common block used in the MAIL facility
$EML:MAIL.LRMAIL.DESC	describes the LRMAIL common block used in the MAIL facility
$EML:MAIL.LRPROF.DESC	describes the LRPROF common block used in the MAIL facility
$EML:MAIL.LRSEND.DESC	describes the LRSEND common block used in the MAIL facility
$EML:MAIL.MAILNM.DESC	describes the MAILNM common block used in the MAIL facility
$EML:MAIL.MDFILE.DESC	describes the MAIL distribution file as used by the MAIL facility
$EML:MAIL.MLIST.DESC	describes the incore copy of the MAIL items as used by the MAIL facility
$EML:MAIL.NICKS.DESC	describes the nicknames file records as used by the private and public directory programs
$EML:MAIL.SUBS2.DESC	describes the low level mailbox routines (SUBS2)
$EML:MAILBOX.DESC	briefly describes the mailbox, and some of the underlying structure of the files used in the MAIL facility
$EML:MAIL.SEND.RC	describes return codes when sending mail
$EML:MAIL.SUBS2.RC	describes the return codes of the low level mailbox routines (SUBS2)
$EML:MPROF.MENU[@]	contains the MPROF menu. This file has National Language variants E (Spanish), F (French), P (Portugese), K (Kanji) supported through the [@] placeholder in the file name.
$EML:RDMAILER.SMTP.DOC	contains a description of how to get RDMAILER and SMTP to work together for a BITNET site. You should also refer to the file $TCP:TCPIP.SMTP.DOC.
$TCP:TCPIP.SMTP.DOC	describes how to configure the MUSIC MAIL facility with SMTP.

$EML:RSCSV13.MODS	contains suggested modifications for RSCS V1.3 for intersystem email communications
$EML:RSCSV2.MODS	contains suggested modifications for RSCS V2 for intersystem email communications
$EML:RSCSV2.TELLMODS	contains a full description of the suggested modifications for RSCS V2 for intersystem email and tell communications
$EML:RSCSV3.MODS	contains suggested modifications for RSCS V3 for intersystem email communications
$EML:RSCSV3.TELLMODS	contains a full description of the suggested modifications for RSCS V3 for intersystem email and tell communications
XMAILER_NAMES_filename	file obtained from BITNIC and used as input into the $EML:MAILER.PROFILE.MK program to make RDMAILER's $EML:MAILER.PROFILE.
DOMAIN_NAMES_filename	file obtained from BITNIC and used as input into the $EML:MAILER.PROFILE.MK program to make RDMAILER's $EML:MAILER.PROFILE.
$EML:INTEREST.GROUPS	contains a copy of the Internet list of lists used by the List Manager facility.
$EML:INTEREST.WIDX	contains the index required for the search engine in the List Manager facility.
$EML:KILLFILE	contains the kill file entries RDMAILER uses to match against senders of incoming mail items.

Other Temporary files used by MAIL

@MAIL.NEW.tcbn, @MAIL.REPLY.tcbn, @MAIL.FORWD.tcbn	are files used by the editor where the user types in the mail text. These should normally be deleted at send termination. It is purged before the editor is called if it exists.
&&TEMP	contains a log of results of the send mail operation. If any errors occur, this file is normally viewed to show the user the errors that occurred. This file is created as needed as a temporary file.
@MAIL.SUPSND.tcbn	used by the MAIL where the user has suspended the sending of a mail item. After the mail item has been stored in the user's mailbox, the file is deleted.
@DPUBL.KEYS	contains the user's function key definitions for the DIRPUBL program.
@MAIL.KEYS	contains the user's function key definitions for the MAIL program.
@NAMES.LISTsss	generated by the $EML:DIRECT.PRINT program to hold a readable copy of the directory. The user would print this file on the printer of their choice. If the userid has a subcode "sss", then it is suffixed to the file name. Otherwise, the file name is @NAMES.LIST.

@PROFILE.KEYS	contains the user's function key definitions for the MPROF program.
@PROFLOG.tcbn	contains the log file for the MPROF program.
@PHMSG.tcbn	contains the PHONEX message text created to send to the recipient.
@SMEXEC.tcbn	is the SENDMAIL exec created by PHONEX to send the message to the recipient.
@SENDFILE	contains a copy of the file sent using the SENDFILE program. It is purged at the end of execution.
@Gtcbn	is used by GETMAIL and GETMINFO when information has to be down-loaded to a PC file.
@MBQ.tcbn	contains a skeleton of the MAILBOOK item that is created to pass to the user's defined MAIL editor.
@@@MBK.MAC	is a file created by MAILBOOK to do the edit required for the automatic addition of the item's text to the new text for resend and forward functions.
@MBK.tcbn	contains a copy of the PC file that is used as the mailbook file for the MAILBOOK program .
@PCX.tcbn	file created by MAILBOOK when you send a PC file as the mail text.
@MAIL.PCX.tcbn	file created by MAIL or SENDMAIL when you send a PC file as the mail text.
@CURMBKtcbn	contains a copy of the current MAILBOOK item when you are performing an operation on an item.
@CURMAILsss	contains a copy of the current MAIL item when you are answering, replying to, forwarding, transferring, or resuming an item, or when the digest feature is invoked. If the userid has a subcode "sss", then it is suffixed to the file name. Otherwise, the file name is @CURMAIL.
@MCtcbn	file created by MAIL, SENDMAIL, or MAILBOOK when copying only the item's text to a PC file or when copying to a mailbook which is a PC file.
@MNtcbn	file created by MAIL, SENDMAIL, or MAILBOOK when the recipients are listed in a PC file.
@MDtcbn	is used by MAIL to build a display of the mail item's distribution list.
@MSUBLsss	contains the users subscription information from the LIST MANAGER facility, MAIL main menu option 8. If the userid has a subcode "sss", then it is suffixed to the file name. Otherwise, the file name is @MSUBL.

Authorization Table

Normally, the MAIL Authorization table is updated using ADMIN 4 5 2. See the *MUSIC/SP Administrator's Guide* for further details regarding this update procedure. The data records in the file

$EMD:MAIL.AUTHOR define two functions. The first type of data records define who can send mail and to whom they can send. The second type of data records defines an expiry date and a limit to the number of mail items presented for viewing for specific users, groups of users, or type of users.

Type 1 Data Records (Send Controls)

The format is free, but each line must contain three strings:

```
sender id     1-16 characters
receiver id   1-132 characters
system id     1-132 characters
domain        1-132 characters (optional)
```

An entry that starts with "*" in column 1 is a comment and is not read into the table.

The receiver id is the receiving user's userid, either given as a userid or as a full email address (user@system). By giving the receiver id as a full email address (user@system), the special names for system id as given below (*ANY and *NONE) can be used to allow or limit full email address access. Wild characters ? and * are supported within the full email address specification.

For the system id, there are several special names that can be used.

*NEVER The specified sender is not allowed to access the MAIL facility at all.

*NONE The specified sender-receiver combination is not allowed at all.

*LOCAL The specified sender-receiver combination is allowed only on this copy of MUSIC.

*OTHER The specified sender-receiver combination is allowed to send mail to systems other than MUSIC systems. They are not allowed to send to any MUSIC system.

*ANY The specified sender-receiver combination is allowed to send mail to any system, whether MUSIC or not. This option means no checking is done on the system id.

The domain name, if given, must be one of the domain names specified in MUSNOD or ALIAS1-4. This domain name will be used to generate the userid/system address for the sender ID. In this way, a user or class of users can appear to be at a different domain. A "*" here means that the domain name to be used is MUSNOD. Not specifying one is treated as having entered a "*". If the domain name is not one of the above, it is defaulted to MUSNOD.

The character "?" can be used as a "wild" character in both the sender and receiver ids. A "?" in the sender or receiver id in the table is considered to match any character in the corresponding position of an actual id.

The use of an "=" means that the relative position in both the sender - receiver id must match.

The '*' can be used to indicate any characters. For example cc* would match any code starting with cc. Analagous to this is cc?????, where the last 5 chars of the code can be anything. Special care should be used in mixing '*' and '=', since they can undo each other and expanding '*' would mess up the absolute position of the '='. So that a pattern of "==*" is acceptable, the pattern "*==" is not.

Wild chars '*' and '?' are allowed on the system name as well.

Examples:

```
*---SEND ID--RECEIVE----------SYSTEMID---------DOMAIN NAME
```

```
ABCD*          *           *NEVER
ABCD           =           *NONE
A123           *           *NONE
ABCD*          LISTERV*    *NONE
 *              *           *ANY
```

The first line is a comment line just as appears in the file $EMD:MAIL.AUTHOR.

The second line does not allow any userid starting with ABCD to access the MAIL facility.

The third line allows the userid ABCD to access the MAIL facility, but ABCD cannot send mail to him/herself.

The fourth line allows the userid A123 to use the MAIL facility, but A123 cannot send mail to anyone.

The fifth line allows any userid starting with ABCD to use the MAIL facility, but they cannot send mail to any email address starting with LISTSERV.

The sixth line allows anyone access to the MAIL facility and they can send mail to any email address.

Type 2 Data Records (User Specific Controls)

The format is free, but each line must start with a closing parenthesis, ")", and can contain 2 or 3 other parameters:

```
userid     1-16 chars
  or
T=type     integer value

days       integer value
items      integer value (optional)
POP Freq   integer value (optional)
```

An entry that starts with "*" in column 1 is a comment and is not read into the table.

The first parameter after the ")" can either be a userid or a T=type (i.e. userid type). The userid supports the wildcard characters "?" and "*". A "?" is considered to match any character in the corresponding position of the userid. A "*" is considered to match all or no characters in the corresponding position of the userid. For the userid type support, if the type given here matches the type assigned to the userid in the code record, a match is made. The CODUPD program can be used to assign a type to a userid or group of userids. The userid type supports values from 0 to 255. Zero is the default type value in the code record if a type has not been assigned previously. See the MUSIC Save Library file $COD:ID.TYPES for suggested settings of the type field in the code record.

The days parameter represents an expiry date for the specification. This sets a local expiry date for the specified user(s) or userid types, overriding the system expiry date (i.e. the MAIL.CONFIG EXPIRE parameter). This allows you to set an expiry date for a group of users to be a shorter or longer time than the system default. Valid values are 0 to 36500. Zero is the default value. Zero allows the user to set any expiry date upto the maximum allowable by the Mail facility of 36500 days. This parameter has the same properties as the MAIL.CONFIG EXPIRE parameter. This rule is enforced for surrogate support when you are sending mail for someone else on their behalf.

The days parameter is also used to expire mail items in a user's mailbox. If a piece of mail is older than is allowed by the expiry date setting, it is deleted from the user's mailbox. The mail facility filters expired mail items from the mailbox and does not present these items for selection when a user wants to view mail.

Instead, these items are placed in a list and they are deleted from the mailbox when the user exits viewing mail, refreshes the mailbox, or exits the program. 500 expired mail items can be deleted from the mailbox in any one instance in this way. If more expired items exist in the mailbox, they are not presented for viewing, and they will be deleted the next time mail is viewed. This process is repeated until there are no further expired mail items in the mailbox.

Note: When you send mail to a number of local recipients, the expiry date set for the item is the maximum of the expiry dates of all the recipients and the sender, and not the expiry date of the sender alone.

The items parameter is an optional parameter that represents the number of mail items which will be presented to a user for viewing even though there are more items in the mailbox. This forces the user to manage his/her mailbox since they have to delete mail items presented to view the other items in the mailbox. Valid values are 0, 12 to 32767. Zero is the default value. Zero allows the user to view all items in the mailbox without imposing a limit on the number of mail items presented.

The POP Freq parameter is an optional parameter that defines the frequency with which a user can gain access to his/her mailbox via a POP mail client. Some POP mail clients allow a user to check for mail as frequently as desired. Sometimes, even a one minute interval is allowed. Usually, a ten minute interval is sufficient. Everyone is sharing the network resources. Frequent mail checking can increase the network load and thus reduce the network throughput for everyone.

Setting this value for the user in this table, allows access to the mailbox via a POP mail client only so frequently. Eventually, the user will get the message and set the frequency to check for mail in their POP mail client to what is allowed by the system.

The value given here for a specification sets a local POP frequency for the specification, overriding the system POP frequency (i.e. the MAIL.CONFIG POPD POPFRQ parameter). This allows you to set a POP frequency for a group of users to be a shorter or longer time than the system default. Valid values are 0 to 1440. A value greater than zero represents minutes. Any value given must represent one day or less of time. Zero allows the user unlimited access to her/his mailbox via POP. This value has the same properties as the MAIL.CONFIG POPD POPFRQ parameter.

A message is returned to the POP mail client indicating the time to wait before access to the mailbox is granted. The Eudora POP mail client shows the to wait. Other POP mail clients may not show this information.

Examples:

```
) ABCD* 30 100
) T=0 30
) T=1-255 61 0
```

The first line allows any userid starting with ABCD to set an expiry date of no more than 30 days in advance of the send date, and limits mail item presentation to the first 100 items in the mailbox.

The second line allows any userid with a userid type 0 to set an expiry date of no more than 30 days in advance of the send date. Since the items parameter is not given, these users are not imposed with a limit on the number of mail items presented to them when they view their mail and presents all items.

The third line allows any userid with a userid type between 1 and 255 to set an expiry date of no more than 61 days in advance of the send date. It also does not impose a limit on the number of mail items presented to them when they view their mail and presents all items.

Running the MAIL.CLEANUP Program

The MAIL CLEANUP program, $EML:MAIL.CLEANUP, is designed to delete mail text and distribution files that have expired. By using the expiry date encoded in the tag of these files, the program deletes the file if it's expiry date is in the past.

MAIL CLEANUP can be run when people are using the MAIL system. Also it can be cancelled at any time, since it does not touch the mailboxes and cause contention with mailbox access. File errors encountered while this program is running are reported in the program output, and the program proceeds. The program output gives a status line every one thousand files examined, and a summary is printed at the end of the report. Sample output is shown later in this section.

It is recommended that the MAIL CLEANUP program be run on a monthly basis (or more often if a site chooses).

This program requires the LSCAN and FILES privileges to run.

The program $EML:CHKMBOXES described in the MAIL Utility Programs section can be run to obtain statistics on the mailboxes.

How the EXPIRY DATE and TIME are set on Mail files

Mail text and distribution files have an associated expiry date and time. When a user sends mail either via the SENDMAIL or MAIL programs, an expiry date is set for the sent mail. When either program is invoked, a default expiry date is established for the user. The user can override his/her default expiry date at send time and create a shorter expiry date for the sent mail. The user's default expiry date is assigned from an entry for the user in the mail authorization file, $EMD:MAIL.AUTHOR. If that does not exist, the MAIL CONFIG program EXPIRE parameter is used.

When RDMAILER receives mail to deliver to a local user or users, it uses the maximum expiry date of all recipients for the mail's expiry date. This expiry date is taken from the mail authorization file for each recipient, or from the MAIL CONFIG program EXPIRE parameter if an entry doesn't exist for the recipient just as described above. Postdated and Recurring mail are handled differently. The user is allowed to set any delivery date for either of these mail types. The expiry date set for postdated mail is the delivery date plus the expiry date allowed. The expiry date set for recurring mail is the day after the last delivery of the recurring item. Each mail item sent from the recurring mail item establishes its an expiry date allowed as each item is treated as an "original" send. As stated above, the expiry date allowed is taken from the mail authorization file for each recipient, or from the MAIL CONFIG program EXPIRE parameter if an entry doesn't exist for the recipient.

Note: When you send mail to a number of local recipients, the expiry date set for the item is the maximum of the expiry dates of all the recipients and the sender, and not the expiry date of the sender alone.

Guidelines for the PERIOD parameter

The MAIL CONFIG program PERIOD parameter is used by the MAIL CLEANUP program. The PERIOD parameter represents the number of days before the current date for which a mail file will be deleted if it was created on or before this new date. For example, if PERIOD=20 and today's date is 30NOV90, then all mail files created on or before 10NOV90 are deleted. The default setting for PERIOD is PERIOD=0 which deletes the file only if the file's expiry date as encoded on the file's tag is before the current date. This is the preferred method for running MAIL.CLEANUP. This allows the Mail facility programs to determine the expiry date to set for the mail to be sent, and it allows MAIL.CLEANUP to honour the set expiry date.

Since mail items in the mailbox are expired and removed from the mailbox by the Mail facility programs independent of the MAIL.CLEANUP program, it is okay to have the value used to set the expiry date for mail sent to be a value less than the PERIOD value for the MAIL.CLEANUP program. The items are removed from the user's mailbox by the Mail facility programs prior to the mail text file being deleted off of the system by MAIL.CLEANUP. The user never sees an item for which the mail text file does not exist. If the reverse is done and the PERIOD value for the MAIL.CLEANUP program is less than the value used to set the expiry date, the user may see an item for which the mail text file does not exist. Then the item can only be deleted by the user.

If your site's mail policy changes and you must change the value used to set the expiry date for the user or instead decide to change the PERIOD value, it may turn out that the adjustment may allow the user to view an item for which the mail text file does not exist. This item can only be deleted by the user. This situation may arise since mail already received by the user has an associated expiry date that was assigned by the previous scheme. Depending on how the change is made, the Mail facility programs may not be able to expire the mail items before they are presented to the user.

MAIL CLEANUP Program Listing

The following is a listing of the MAIL CLEANUP program, $EML:MAIL.CLEANUP.

```
/SYS NOPRINT,REG=512,TIME=MAX
/FILE 1 N($EML:MAIL.FCLEAN.OUT) NEW(REPL) LR(133) SP(500) DEF
/LOAD XMON
CLEANUP N($EML:MAIL.FCLEAN.LMOD)
PERIOD=0
```

The following is a listing of the file created by a sample run of MAIL CLEANUP, $EML:MAIL.FCLEAN.OUT.

```
Mail Cleanup MON APR 18, 1994 05:13:58
using parameters FCODE=$MD? PERIOD=  0

The date used for comparisons in this program is WED OCT 20, 1993
 (ie current date - period)
05:13 so far files,totspace,dels,delspace=        0        0        0
05:14 so far files,totspace,dels,delspace=     1000     3672       14
05:15 so far files,totspace,dels,delspace=     2000     7554       34
05:16 so far files,totspace,dels,delspace=     3000    11046       49
05:16 so far files,totspace,dels,delspace=     4000    14482       71
05:17 so far files,totspace,dels,delspace=     5000    18318       86
05:18 so far files,totspace,dels,delspace=     6000    21910      101
05:18 so far files,totspace,dels,delspace=     7000    25678      123
05:19 so far files,totspace,dels,delspace=     8000    29318      143
05:20 so far files,totspace,dels,delspace=     9000    32776      158
05:20 so far files,totspace,dels,delspace=    10000    36368      171
05:21 so far files,totspace,dels,delspace=    11000    40172      184
05:22 so far files,totspace,dels,delspace=    12000    44126      195
05:22 so far files,totspace,dels,delspace=    13000    47764      207
05:23 so far files,totspace,dels,delspace=    14000    51696      218
05:24 so far files,totspace,dels,delspace=    15000    55128      232
05:24 so far files,totspace,dels,delspace=    16000    59272      252
05:25 so far files,totspace,dels,delspace=    17000    63044      275
05:26 so far files,totspace,dels,delspace=    18000    67058      291
05:27 so far files,totspace,dels,delspace=    19000    70492      305
05:27 so far files,totspace,dels,delspace=    20000    74024      314
```

```
Total number of mail data files =     20250
Total space of mail data files =     74918 K
Number of files deleted=       316
Space of files deleted=       978 K
JOB TIME  325.31 SERVICE UNITS
```

MAIL Utility Programs

MAILBOX

This program displays the mailbox filename for a given userid. If the userid is not given, the mailbox filename for the signed-on user is displayed. This program can be run from ADMIN 4 5 7. The program is invoked as follows:

```
MAILBOX userid
```

$EML:CHKMBOXES

This program looks at all of the mailboxes on the system and generates a statistics report for these mailboxes. Mailboxes are flagged in the report if they are larger than a certain size, have not been used for a number of days, have no corresponding userid, have a disabled userid, have a userid that has not been used for a number of days, have an overflow mailbox, or have encountered an error in the code table lookup.

The namelist parameter BIGBOX flags mailboxes in the report that are larger than BIGBOX K. The default value for BIGBOX is 300 (ie 300K). The namelist parameter NDAYS flags mailboxes in the report when either a mailbox has not been accessed in NDAYS days or a userid has not been accessed in NDAYS days. The default is 60 days.

The job output is written to units 6 and 1. Unit 1 is defined as file @CHKMBOXES.OUT. The program requires the privileges FILES, LSCAN, and CODES to run.

The following is sample job output from this program.

```
 -- CHKMBOXES --    MON APR 18, 1994   09:05
 Options:  BIGBOX=   300K   NDAYS=    60
$MBB:BGUEST                2K         BOXOLD               IDOLD
$MBB:B$000000           1584K  BIG                                OFLOW
$MBG:BJUNK                10K                       IDOFF  IDOLD
<stuff deleted here for brevity >

 Total number of mailbox files:       144
 Total space of mailbox files:       7790 K
 Number of extract errors:              0
 BIG     Number of large mailbox files:                   6
 BOXOLD  Number of old mailbox files:                    47
 NOID    Number of mailboxes with no userid:              0
 IDOFF   Number of mailboxes with userid disabled:        2
 IDOLD   Number of mailboxes with old userid:            52
 OFLOW   Number of mailboxes with overflow:               1
 IDERR   Number of userid lookup errors:                  0
```

MAILBOX.FIX

From time to time, a user's mailbox will get damaged. This damage usually is an inconsistency in the mailbox control record and the true number and types of mail records in the mailbox. This is usually caused by editing the mailbox and manipulating the records and filing the corrupted version.

MAILBOX.FIX is used to re-align the control information in the mailbox with the actual contents. The program is invoked as follows:

```
MAILBOX.FIX userid
```

where *userid* is the userid of the corrupted mailbox.

MSTAT

MSTAT is used to produce a snapshot of the current utilization of mail files. These are the files that are used by the mail facility to store mail text and distribution files. This program can aid in deciding when to schedule MAIL.CLEANUP.

The program is invoked as follows:

```
MSTAT
```

This program also generates a histogram in the file $EML:MAIL.FILE.COUNT.

UCRADD

This program is used to either display or generate UCR records for a code series. This program is automatically invoked by the configurator program to generate the ucr records for the FCODE series, and by MSTAT to display the UCR's.

To display the UCR's for FCODE of $MD?, FMIN of 1 and FMAX of 7 use:

```
UCRADD $MD 1 7 GET
```

To add the UCR's:

```
UCRADD $MD 1 7 SET
```

To delete the UCR's:

```
UCRADD $MD 1 7 DEL
```

The MUSIC system no longer updates the UCR for files described by the FCODE, FMIN, and FMAX namelist parameters as of MUSIC/SP 3.1 when FCODE=$MD?. Thus the get feature of this program can not return accurate information.

$EML:MAIL.MAKE

This program is a REXX exec that runs the various steps required to make the various Mail facility programs. It checks to make sure that the required object decks are online before running the link-edit for any of the programs. It can run the various steps required to make the load modules for MAIL and put them into the LPA. A system reIPL is required to put the new version of the MAIL program into production.

$EML:MAIL.MAKE can also relink the following programs: SENDMAIL, RDMAILER, GETMAIL, PROFILE, and MAILBOOK. All of these programs except for RDMAILER are put into production immediately after the link-edit is run as the link-edit recreates the load module. The RDMAILER BTRM must be restarted for it to use the new load module. This program can be run from ADMIN 4 5 8 "Remake Mail facility after local mods".

$EML:MAILER.PROFILE.MK

This program makes the mailer profile, $EML:MAILER.PROFILE, for RDMAILER. This program should only be used by sites which rely exclusively on RDMAILER to deliver the mail. In this case RDMAILER does not send all of its mail to a MAILER for delivery (ie the RDMAILER options MAILER and MNODE are not used). This program can be called with or without parameters. You are prompted for the parameters if you do not specify the parameters when the program is invoked.

The program can be invoked as follows:

MAILER.PROFILE.MK XMAILER_NAMES_filename DOMAIN_NAMES_filename

- XMAILER_NAMES_filename is the MUSIC filename of the XMAILER NAMES as distributed by BITNIC. This file must be in its plain text format, ie not CMSDUMP or NETDATA format.

- DOMAIN_NAMES_filename is the MUSIC filename of the DOMAIN NAMES as distributed by BITNIC. This file must be in its plain text format, ie not CMSDUMP or NETDATA format.

$EML:MAILER.TEMPLATE serves as the template for $EML:MAILER.PROFILE. If you have any local mods, eg a MUSIC test system, you can define it in the template and it will be copied into the mailer profile.

You should request from BITNIC that the XMAILER NAMES and DOMAIN NAMES files be sent to you automatically when they are updated. If necessary, you can "GET XMAILER NAMES" and "GET DOMAIN NAMES" from LISTSERV@BITNIC.

This program stores the new mailer profile as MAILER.PROFILE on the user's account. The user must copy the file to $EML:MAILER.PROFILE to replace the previous version of this file. Also the RDMAILER BTRM must be told to use the new file. This can be done by issuing the MUSIC console command "/reply tcbn DOM" where tcbn is the RDMAILER BTRM tcb number.

The file $EML:MAILER.PROFILE can be edited manually using ADMIN 4 5 4 "Update Mailer Profile".

$EML:INTEREST.BUILD

This program makes the index for the search engine required in the List Manager facility. It runs an ITSBLD job with input $EML:INTEREST.GROUPS and produces the index file $EML:INTEREST.WIDX.

See the description of the List Manager facility in the Mail Administration section below.

$EML:MSG.BLDR

This message builder program takes the raw text message files and creates a message file usable by the Mail facility. The produced file has a couple of important characteristics: it contains record displacement information as hexadecimal values in the file, and it is created with the public attribute. The files created by MSG.BLDR and used by the MAIL facility should all be put in the RAM disk. The program is invoked as follows:

```
$EML:MSG.BLDR input_msg_file output_msg_file
```

So, for example, if you had changes you wanted to make to the Kanji version of the MAIL messages file, make the changes to the raw text messages file $EML:MAIL.MSGSK.S using the Editor with language set to Kanji, and then run the following:

```
$EML:MSG.BLDR $EML:MAIL.MSGSK.S $EML:MAIL.MSGSK
```

After the build is done, you should then reload the RAM disk if $EML:MAIL.MSGSK is in the RAM disk.

The raw text message files are normally stored offline and may have to be restored from the source tapes if use is required.

$EML:MSG.CHECKER

This program can be used to verify the output of the $EML:MSG.BLDR program is correct. MSG.CHECKER takes the output from MSG.BLDR as input and produces a file identical to MSG.BLDR input. This validates the build operation. The program is invoked as follows:

```
$EML:MSG.CHECKER input_msg_file output_msg_file
```

Don't specify the same filename for the output_msg_file as the input_msg_file for the MSG.BLDR program. You will destroy the real message file for the MAIL facility.

So, for example, if you had made changes to the Kanji version of the MAIL messages file and have already run MSG.BLDR and you want to check the output messages file, you would run the following:

```
$EML:MSG.CHECKER $EML:MAIL.MSGSK MAIL.MSGSK.S
```

After the program is done, you should then do a compare on the output file MAIL.MSGSK.S and $EML:MAIL.MSGSK.S to guarantee they are equivalent. The compare is done as follows:

```
COMPARE $EML:MAIL.MSGSK.S MAIL.MSGSK.S
```

The raw text message files are normally stored offline and may have to be restored from the source tapes if use is required.

$EML:UNRES.CK

This program is a REXX exec that checks the unresolved external references generated by the link-edits of the MAIL facility (i.e. running $EML:UNRES.CK). It runs a compare job of the expected unresolved external references which are stored by the exec in the file @UNRES1, against the unresolved external references the link-edit generated which are stored by the exec in the file @UNRES2. If the files do not compare equal, a new unresolved external reference has been introduced by the link-edit and this may indicate an error in the link-edit.

This exec is called by $EML:MAIL.MAKE for each link-edit step performed to allow the system administrator to verify that the link-edit step worked.

$EML:NETCNV

The $EML:NETCNV program converts a file in NETDATA format created by the SENDFILE program into a readable file. The input filename is accepted as a program parameter, and it is automatically replaced with

the converted file.

The program is invoked as follows:

```
$EML:NETCNV input_filename
```

$EML:QPUTI

The $EML:QPUTI program adds a RFC821/822 compliant mail file to the MUSIC reader queue so that it can be delivered as mail by RDMAILER to the recipients. The program takes an input filename as a parameter. The file is deleted after it is added to the reader queue. The program is invoked as follows:

```
$EML:QPUTI input_filename
```

$EML:MAIL.BROADCAST

This facility can be used by system support personnel to broadcast a mail message to a large number of users. This broadcast facility delivers a mail message to each recipient. See the topic "Mail Broadcast Facility" later in this chapter.

The program is invoked as follows:

```
$EML:MAIL.BROADCAST
```

MAIL Administration

End of Year/Semester Cleanup

Your site may delete userids from the code table once in awhile, for example when a student has graduated from your institution. Since the mailboxes are not stored on the user's userid, but on system userids, you may also want to delete the mailboxes associated with these deleted userids. The way to handle this situation is to:

1. delete the mail text and distribution files
2. delete the userid from the code table
3. delete the mailbox

The mailbox does not contain the mail text per se, but it contains filenames where the mail text is stored. So the mail text and distribution files, which are also stored on system userids, must be deleted as done in step 1 above or by the MAIL CLEANUP program.

Notes:

1. The MAIL CLEANUP program only looks at the mail files on the system and does not examine mailboxes. So the MAIL CLEANUP program can be run to delete the mail files, but it does not have the power to remove the files unconditionally from the system. It only deletes mail files either when the expiry date of the file is in the past when the PERIOD parameter is set to zero, or when the creation date of the file is before the current date minus PERIOD days if the PERIOD parameter is not set to zero. So, the mail files could remain on the system until their expiry date has past. One approach to remove the user's mail files immediately is to follow the instructions given in the subject below entitled "How to clean up a mailbox quickly" before you remove the userid from the system.

Less work will be required at the end of the semester when MAIL CLEANUP is run regularly. MAIL CLEANUP purges expired mail text and distribution files from the system, returning the freed space to the system. A year's worth of accumulation can translate to a lot of purging to do at the end of the year.

2. Step 2 and 3 above can be done in one step by using the DELMAILBOX parameter of the CODUPD program's DELETE command. See the CODUPD program description for further details.

3. If you choose not to do steps 1, 2 and 3 together, you can do these three steps individually. First run a GETMINFO job to get all of the items from the user's mailbox and delete them. Then delete the userid from the code table. Then run MAILBOX userid (where userid is the userid whose mailbox you want to delete) to display the filename for the mailbox. Since the mailboxes are stored on system userids, using this program is the method required to determine the filename for a mailbox. The last step is to purge the file representing the mailbox.

   The following is a sample GETMINFO job which deletes all of a user's mail. Turn VIP ON before running the following GETMINFO job. The VIP privilege gives you access to the user's mailbox without signing on to the user's account.

   ```
   GETMINFO JUNK DISCARD ALL DELETE FOR(userid)
   ```

4. See the special subject below entitled "Mailing List Mail, Subscriptions, Unsubscriptions, and Digests" for information regarding this topic and end of semester issues.

Mailing List Mail, Subscriptions, Unsubscriptions, and Digests

If your system is connected to BITNET or the Internet, your users may be allowed to subscribe to mailing lists and receive list mail. Typically, a user receives more mail than they send and mailing list subscriptions can generate a tremendous increase in incoming mail traffic on the MUSIC system. This increase in demand can impact your MUSIC system, so it must be monitored.

Subscriptions

Users can subscribe to a mailing list directly after they find out how to subscribe to it, or they can use MAIL's main menu item 8, the List Manager facility, to subscribe to a list. The List Manager facility uses a list of lists from the Internet to allow users to subscribe to the lists therein. These subscriptions to lists done via the List Manager facility are stored in the file @MSUBL. If the userid has a subcode "sss", then it is suffixed to the filename (ie @MSUBLsss).

Your local mail policy may allow restrictions in the handling of list mail. An appropriate entry in $EMD:MAIL.AUTHOR could disallow a user or users from receiving external list mail. Further, you could add an appropriate entry in $EML:NORECEIVE to not allow a user or users to receive any external mail.

Unsubscriptions

After you remove a userid from the system, there is a chance that mail from a mailing list will continue to be sent to the removed user. In this case, RDMAILER will automatically return a delivery error message to the sender in a special format recognized by the BITNET LISTSERV and LMAIL programs. When either of these programs receives this error message, the userid is automatically removed from the list. Other mailing list manager programs may not support this feature and you may have to ask the list manager to remove the user or users from the list. If a user has used MUSIC's List Manager to subscribe to various lists, the file @MSUBL or @MSUBLsss as described above exists on the user's account. This file contains subscription information that could be used to unsubscribe a user from these lists.

The postmaster's mailbox will receive a copy of the delivery error message returned to the sender when a mail recipient is unknown locally as would be the case when a userid is removed from the system.

There is a requirement of the LIST serviced by the BITNET LISTSERV program to unsubscribe a user automatically when it receives this special delivery error message sent to its list by RDMAILER. The LIST must be set up with the "Auto-delete =Yes,Semi-Auto" feature. This feature is not the default when the list has the "Validate= All" feature set. Suggesting this to the list owner would be beneficial to him/her as they would not have to look at all the "No such user" messages they receive. They would be dealt with automatically. Also, MUSIC users refusing mail from a list or lists cause a special delivery error message to be sent to the list. These users are dealt with in a fashion similar to the case just described where a recipient does not exist (ie the userid has been removed from MUSIC) and that intended recipient would be removed from the list if the LIST has the described feature enabled.

The postmaster can also manually unsubscribe a user from a mail list. This is convenient if the automated process described above has not removed the user from the mail list. When an invalid or disabled local user is not removed from a mail list, the mail sent by the list is returned to the sender and a copy is given to the postmaster. When the postmaster is viewing this item, the UNSUB command can be used to manually unsubscribe the user from the mail list. See the topic "Unsubscribing a User from a Mail List" in the Mail Filter Program section below for further information.

When the postmaster's incoming mail is being viewed, the UNSUB command can be used to unsubscribe a user from a the oria mail list

Digests

Mailing List Digests are a convenient way to follow a list of interest and get numerous posting as one mail item. The digest can be viewed with the MAIL DIGEST feature within VIEW.

Digest subscriptions are network friendly. You receive one mail item and not a multitude of individual items that make up the digest. Thus digest delivery demands less system resources.

MAIL and the MUSIC Save Library

With heavy use of the MAIL system, you may find that your Save Library index is heavily populated. Sometimes, the index's auxiliary blocks may be used more frequently. This may slow down access to the Save Library in general. The MFINDEX program can be used to display information about the Save Library index.

If you find that access to the Save Library is slow (this could be manifested by the use of a lot of auxiliary blocks) you may want to increase the number of segments in your Save Library index. See the NEWINDEX program for further details on enlarging the index.

The MAIL facility is coded to allow 1296 userids for mail text and distribution files. The generated userids for use with these files are $MDAA to $MD99, which is 36*36 or 1296 combinations. As such, the MAIL facility can take advantage of upto 1296 segments in the Save Library index even though the number of segments must be a prime number. So the Mail facility functions without change as the number of segments in your Save Library index changes.

The FCODE namelist parameter does not appear within the MAIL CONFIG facility, although it gives the starting part of the userids ($MD) used for the mail text and distribution files. It should not be changed as system changes have been made to accomodate this fact to give a performance improvement. System performance may degrade if it is changed to something other than $MD.

Adding MAIL.CLEANUP to your AUTOSUB job

The file $ADM:AUTO.EMAILCLN defines a set of jobs to run MAIL.CLEANUP. This set of jobs shuts down RDMAILER, runs MAIL.CLEANUP, then restarts RDMAILER. You could add this job to the BTRM that runs AUTOSUB by using ADMIN 4 9. One idea would be to add an MSTAT job before and after the MAIL.CLEANUP job to product a snapshot of the current utilization of mail files.

MAIL Logs

The default mail log file is $emd:@maillogyyyymmdd, where yyyymmdd is the date the log was started. The log contains records for all mail sent and received, and it also contains progress and error message records generated by RDMAILER. The entries in the mail log from RDMAILER are prefixed with an asterisk. The log file is not created if the MAIL CONFIG parameter LOG is set to F.

The mail log records are generated by the MAIL, RDMAILER, and SENDMAIL programs. These log records are passed to the MUSIC LOG server BTRM with the application name MAIL. The MUSIC LOG server matches the received record's application name, MAIL, versus the application name as defined in the LOG server's definition file $PGM:SYSLG.DEFINE. It then writes the log records to the log file associated with application name. The default mail log file as defined in the LOG server's definition file is $EMD:@MAILLOG. With the default DAILY specification given, the mail log file used by the LOG server is $EMD:@MAILLOGyyyymmdd. You can change the filename or make the logs MONTHLY if so desired.

These logs can take up a lot of space on your system. You may want to delete the older files periodically. Also, it may be advisable to give the DAILY specification for the mail log if your site has considerable mail traffic.

For further information on the MUSIC LOG server BTRM, see its description in this manual.

How to clean up a mailbox quickly

You may find a need to delete mail items from a mailbox quickly. As an example, a user has been away for awhile and forgot to sign off of a automatic distribution list. Now his/her mailbox is too large and MAIL complains that it needs "x more bytes of storage to run". A simple way to delete mail enmasse from a mailbox is to use the DISCARD option of the GETMINFO or GETMAIL program. You can use the GETMINFO or GETMAIL program's FOR parameter to discard mail from someone else's mailbox provided you have the privilege associated with the program's SNOOP option. See the description of MAIL.CONFIG for details on the SNOOP option.

Examples:

1. This example deletes all mail from the user's mailbox that was received from the NOVELL list.

```
GETMINFO JUNK DISCARD ALL DELETE SELECT FROM NOVELL
```

2. This example deletes all mail from the user's mailbox that was rece ived before 01APR93. It also shrinks the user's mailbox to the absolute minimum file size required.

```
GETMINFO JUNK DISCARD ALL DELETE SHRINK SELECT BEFORE 01APR93
```

Space used by the Mail Facility

A user may find that he or she has received too much mail and is getting a message from MAIL that it "requires at least x more bytes of storage to run". This message results because MAIL is using the allotted region size to keep an incore copy of the mail items. The region size in $EML:MAIL dictates this value. Each mail item requires 136 bytes. So increasing the region size 100K would provide the user with the ability to see about 750 more mail items.

$EML:BIGMAIL is a version of $EML:MAIL with a region size of 3000K.

How to load a Conference automatically with mail received

You can load a Conference directly with mail you receive. First you must copy the mail to a mailbook. This can be done by using the MAIL XL command or COPY command with the MAILBOOK option set to Y (yes), or by using the GETMAIL program. After the mailbook is created, you append the mailbook to the conference. This can be done manually by going into the conference and appending the mailbook to the conference. You can do this automatically by using the following command:

```
CONF conference_name APP topic mailbook_filename NOH
```

where *conference_name* is the name of the conference and *topic* is the topic within the conference which you want to add to mailbook_filename is the filename of the mailbook to add to the conference.

This automatic function is allowed for conference owners or any userid with the FILES privilege. All other non-privileged users must use the manual method to append mail to the conference topic.

In the following example, all of the mail that has been received from the NOVELL list is put into a mailbook and then that mailbook is appended to the NOVELL topic within the LANS conference.

```
GETMAIL NOVELL.LOG ALL DELETE SELECT FROM NOVELL
CONF LANS APP NOVELL NOVELL.LOG NOH
```

See the topic "Merging Information into Topics" in the help for the Conferencing facility for further details.

How to have mail received handled automatically

There is a special feature within the Mail facility called MAGFIL support that allows you to have your incoming mail handled automatically. This feature is only available for users with the VIP privilege. If you turn on the VIP privilege and then invoke the Mail Profile facility, you will see another field on the General Mail Options screen. The field is called "File to be executed when mail arrives for you". Enter a filename in this field. This file could be a job that runs GETMINFO or GETMAIL.

There are many uses for this support. You could have mail from a particular person logged and deleted automatically. You could have mail from a automatic mailer or listserv logged and deleted or added to a conference automatically.

The mail text and distribution filenames are passed to the program designated in the "File to be executed when mail arrives for you" field when it is executed.

See the file $EML:MAGFIL.SUPPORT for further details.

Setting up a MAIL route for AUTOPR

The MUSIC system can be configured with a ROUTE destination of MAIL. This means that output from a job or a print file can be routed to you as mail. One use for this is to have output from the AUTOSUB job routed to you as mail. This way you can check your mail for the job output instead of looking for printed output.

This function helps a site move to a paperless office.

See the topic "Configuring the AUTOPR Program" for further details.

List Manager Facility Functions

The List Manager facility allows users to manage their subscriptions to BITNET and Internet mail lists. MAIL main menu option 8 invokes this facility.

The List Manager uses a file found on the Internet for the browse function from which users can view the available lists for subscription. This file is known as the Internet "interest groups" or "list of lists". It does not contain all of the mail lists on the Internet and BITNET. It is up to the individual mail list owner(s) to send a request into the "list of lists" coordinator for inclusion into this master list. This list is available via anonymous ftp on sri.com in the netinfo directory as file interest-groups. It may also be obtained through email by sending a message to mail-server@sri.com with "send netinfo/interest-groups" in the body of the message.

This file should be saved as the public MUSIC file $EML:INTEREST.GROUPS. The MAKPUBL program can be used to make the file public.

After this file has been saved public, you must run $EML:INTEREST.BUILD to make the ITS index required for the search engine within the List Manager facility. See the previous section "Mail Utility Programs" for a description of this program.

If you do not want to make this facility available to your users, rename the file $EML:LM.NOTAVAIL to $EML:LM. This will replace the List Manager with a pop up window telling users this facility is not available. The file attributes on $EML:LM should be public (PUBL) and execute only (XO). The MAKPUBXO command can be used to make a file public and execute only.

Each user has a file @MSUBLsss where their subscriptions to mail lists made via this facility are kept. If the userid has a subcode "sss", then it is suffixed to the file name. Otherwise, the file name is @MSUBL.

Changing the Expiry Date of Incoming Mail

A privileged user can reset the expiry date on incoming mail for any user. This privileged user must possess the privilege given in the Mail facility SNOOP parameter or the user must have the VIP privilege. When you are looking at the user's incoming mail, enter an "E" (expire) select code beside the mail item and a menu is presented to change the expiry date and time associated with the mail item.

This function can be applied to any item that is displayed that has not had its mail files deleted by the MAIL CLEANUP program or has not been deleted by the user to preserve that item.

When a mail item has expired, it is not displayed to the user and it is deleted from the user's mailbox upon exit or a refresh.

How to Retract a Delivered Message

There are two ways to retract a message delivered locally that should not have been delivered or was delivered by accident. If the item was sent with acknowledgements on, and the copy of the mail item on the outgoing mail list has not been deleted, you could expire the item to some date in the past. This updates the other local boxes. The other method requires the VIP privilege. So turn on VIP and then enter MAIL. You would then delete the item from the recipient's mailbox by typing their userid in the Mail For field, locating the item in their incoming mail list, and deleting it. You must look at the mail for all recipients and perform this operation.

How mail items and files get deleted

Typically, a user deletes a mail item received and it is removed from the mailbox and the associated mail files are deleted if this user is the only recipient of the mail or is the last recipient to delete the mail. This deletion or tag reduction is done by the MUSIC LOG server BTRM for the MAIL program. GETMAIL and GETMINFO do their own deletions.

However, what happens when a user does not manage their mail? This is where MAIL CLEANUP becomes important. It can delete mail files off of the system based on expiry date or creation date depending on how it is run, but it does not modify the mailboxes. This task is done by the MAIL facility programs themselves, removing expired mail items from a user's mailbox automatically when the program is invoked. For instance, when a user invokes the MAIL program, the mail is examined before presentation to determine if any mail has expired. If there is mail to expire, it is not presented to the user in the incoming/outgoing mail list. It is removed 500 items per invocation from the mailbox when the user exists the MAIL program. Thus, in a sense, the mailbox management is done by the users themselves.

When expired mail is removed from the mailbox, it is the actual mail item that is removed. The mail text and distribution files may have already been purged from the system by a previous run of the MAIL CLEANUP program or they will be deleted on the next run.

How does the expiry date get set on a mail item

Each user has a default expiry date, defined as the current date plus some value, usually a number of days. This value can be the system default expiry date which is defined by the EXPIRE namelist parameter. If an entry in the mail authorization file defines a different value (ie retention period) for the user, then this value is used in preference to the system default expiry date.

When mail is sent, the expiry date set is the maximum expiry date of all of the recipients' expiry dates. If the sender has acknowledgements turned on, the sender is also considered a recipient and its expiry date must be considered for the maximum expiry date. Acknowledgements are turned off by the system if the mail is sent offsite or is autoforwarded.

When mail is sent, an expiry date can be set on the Sending Mail Options screen. The displayed expiry date is the default for the sender. The expiry date displayed on this screen is a user override. If it is changed, it will be used instead of the maximum expiry date for all of the recipients. If the sender is a VIP user or the userid possesses the privilege defined by the SNOOP namelist parameter, the sender can set any expiry date in this override area.

The Mail facility limit to any expiry date is 100 years forwards from the current date.

Adding your site's mail policy to Mail main menu item A, MAIL FAQ

If your site has a mail policy, you may want to add it to MAIL's FAQ (Frequently Asked Questions) item on Mail's main menu. The MAIL FAQ text can be found in the file $HLP:@EM.FAQSCR. You could either edit the file and make your changes or you could use the IDP program to update the file as this file is a help file. If you were to use IDP, you would run "IDP E $HLP:EMHELP", then select the topic FAQSCR by positioning the cursor on the topic and pressing F5 to edit the topic. Save your changes and your done. Don't forget to add a title line to the menu for the mail policy.

How to Create Your Own MAIL Menu

You can create your own MAIL menu and include or exclude items from the distributed MAIL main menu. The program $EML:FMAIL uses the MAIL namelist parameter EXMAIL for this purpose. EXMAIL allows any MAIL subfunction to be invoked and exit MAIL totally when the subfunction is done. For example, if you run FMAIL 1, Incoming Mail would be presented. When you exit from this menu, you return to wherever you were when you invoked FMAIL 1. The following is a sample BBS MAIN.MENU file that could be a MAIL main menu.

```
)title Mail Facility
 Place the cursor on an item and press ENTER or RETURN.

   +?MAILSEND-? Create and Send Mail
   +?INMAIL  -? Read Incoming Mail
   +?OUTMAIL -? Outgoing Mail
```

The following is the sample BBS TOPICSX file associated with the above MAIN.MENU file.

```
05  TOPICS      TOPICSX
09  MAIN.MENU
08  MAILSEND    |FMAIL 2
06  INMAIL      |FMAIL 1
07  OUTMAIL     |FMAIL 3
```

Restricting MAIL Access

There are a number of things you can do as the system administrator to curtail abuse or restrict access to mail by a user.

1. Delete the userid's files, the code itself and the mailbox with CODUPD. This is the most severe action that can be taken.

2. Disable the userid. All mail sent to this user is returned to the sender with the message "Unknown userid". Even though the userid is disabled, a user with the VIP privilege can inspect the mailbox and delete items if that is so desired.

3. Do not allow the userid to access the mail program, send mail offsite or to a particular person or place. This is accomplished by either adding an entry in $EMD:MAIL.AUTHOR barring the user from mail access, or sending mail to this person or place. You can also add an entry in $EML:NORECEIVE to disallow a person from receiving mail from the outside.

4. Reduce the number of mail items that a user can view as an aid in helping the user manage their own mailbox. Add an entry in $EMD:MAIL.AUTHOR for the user with this limit set to some number.

5. Reduce the retention period of mail for this user. This aids the user to manage their own mailbox. Add

an entry in $EMD:MAIL.AUTHOR for the user with this limit set to some number.

Running Multiple RDMAILER BTRMs

RDMAILER can be configured to run as more than one BTRM. Up to 10 auxiliary RDMAILER BTRMs can be configured to handle incoming mail. One RDMAILER BTRM is available for outgoing mail. The MAIL.CONFIG program can be used for this configuration. Additional BTRMs are used to handle increases in incoming mail, like mailing list mail.

Before you reconfigure the number of RDMAILERs you run, make sure you have generated a MUSIC nucleus with enough BTRM addresses before you add the new BTRMs. The BTRM statement in $GEN:NUCGEN.JOB is used for this purpose.

Also, after running the MAIL.CONFIG program to reconfigure your setup, you must shutdown and restart the VMREADX BTRMs and all of the RDMAILER BTRMs. Only then will the BTRMs use the new configuration. A MUSIC shutdown will have the same affect.

RDMAILER can be configured in a number of ways. Here are two sample configurations.

- It can be configured as only one BTRM, handling both incoming and outgoing mail. This configuration is suggested if your site uses mail infrequently.

- It can be configured as two incoming BTRMs and one outgoing BTRM. The two incoming RDMAILERs both share the work of emptying the reader queue, and delivering the mail. This configuration is suggested for moderate mail usage.

There are some important features that RDMAILER uses when the MAIL system is configured to run more than one RDMAILER. First, the outgoing RDMAILER never handles incoming mail. Its duty is only outgoing mail, along with postdated, recurring, and autoforwarded mail.

The incoming RDMAILERs all handle only incoming mail and never outgoing mail, except it handles the autoforwarding for mail which it is trying to deliver. This reduces the autoforwarding demands on the outgoing BTRM. Thus each incoming RDMAILER has a different PCODE definition for the post office mailbox. The different PCODEs are generated automatically by the MAIL.CONFIG program.

Incoming RDMAILER's duty is to handle entries in the reader queue. Each incoming RDMAILER is assigned an id number n such that each RDMAILER only looks at those reader queue entries in the queue for it. So if there are two incoming RDMAILERs, then RDMAILER 1 processes entries 1,3,5,7,..., and RDMAILER 2 processes entries 2,4,6,8,... Thus, RDMAILER must know two factors, how many incoming RDMAILERs are running, and which one is it. Then it can process the right entries from the reader queue. VMREADX must also know how many incoming RDMAILERs are running so it can wakeup the correct RDMAILER when there is work to be done. The wakeup uses the "ENQNAM" namelist parameter for RDMAILER "$SRDMAILnn" where nn is 1 to 10. It is automatically generated by the MAIL.CONFIG program. You should not change this value, otherwise VMREADX will never wakeup RDMAILER. RDMAILER does have a builtin wakeup mechanism in case it never gets woken up.

You may find that the incoming RDMAILERs cannot keep up with the influx of MUSIC reader files or reader queue files. This can be determined by a buildup of MUSIC reader files, or a message on the MUSIC console indicating that VMREADX is delaying for "n" minutes as the reader queue is full. Before adding another incoming BTRM, make sure that all the VMREADX and RDMAILER BTRMs are functioning correctly. Also, the buildup could be temporary, and it may not warrant an additional BTRM. If the buildup persists for intermittent periods throughout the day, addition of one or more incoming BTRMs will alleviate the problem.

When a RDMAILER BTRM stops

If a RDMAILER BTRM stops for any reason, a message is written to the MUSIC console and a tell message is posted to the userid indicated by the NOTIFY parameter in the MAIL CONFIG facility. This is normal if you have shut down RDMAILER by the standard method (i.e. the console command /REPLY tcbn STOP, where tcbn is the tcb number of the BTRM running RDMAILER). When RDMAILER stops any other reason, this action notifies you that something has happened.

To verify that a RDMAILER BTRM has stopped, issue the MUSIC console command /REPLY tcbn HELLO, when tcbn is the tcb number of the BTRM running RDMAILER you suspect has stopped.

Incoming mail processing stopped

When a RDMAILER BTRM processing incoming mail stops, it is usually caused by invalid mail headers in the mail item it was trying to deliver. The first step in restarting the RDMAILER BTRM is to determine which reader file RDMAILER was processing when it stopped. The previous topic in this section entitled "Running Multiple RDMAILER BTRMs" gives basic information on how to determine which reader queue entry RDMAILER was processing when it stopped. Each reader queue entry has an associated file which is the actual spool file. Entry number one in the reader queue is used as a pointer to the next available entry in the reader queue. Thus entry two in the reader queue represents the first entry handled by a RDMAILER BTRM. The file $VMR:RQ.00001 is associated with entry two, as is $VMR:RQ.00002 with entry three and so on. The first entry in the reader queue that the stopped RDMAILER BTRM is to handle is typically the offending entry. Now that the entry has been identified, you can delete the offending reader file associated the entry, rename the reader file to another file name on your userid, and restart RDMAILER, or you can edit the file, fix the addressing, file the changes, and restart the RDMAILER BTRM.

Another situation can arise if the RDMAILER BTRM is not restarted. The VMREADX BTRMs read spool files and add entries into the reader queue that represent the files it has read. An incoming RDMAILER BTRM processes these entries in the reader queue, delivers the mail and frees up the entries so they can be reused by the VMREADX BTRMs. If RDMAILER is not running, it is not freeing up entries in the reader queue for reuse and there are less entries available for the VMREADX BTRMs to use. If all of the entries are used, the VMREADX BTRMs cannot do their job and they must shutdown too. When there are no entries available for the VMREADX BTRMs to use, they attempts to process spool files 10 times, delaying a number of minutes each time. If after 10 tries the reader queue is still full, they shut down and the spool files are not processed. Each time a VMREADX BTRM shuts down because the reader queue is full, it copies the current spool file it is processing to $VMR:VMFILE.nnnnn, where nnnnn is a number. These files can be placed back into the reader queue once an item becomes available for use by the $EML:QPUTI program. Note that when a VMREADX BTRM delays or stops because the reader queue is full, messages are written to the MUSIC console indicating what has happened.

Outgoing mail processing stopped

When a RDMAILER BTRM processing outgoing mail stops, it is usually caused by bad addressing in the mail item it was trying to deliver. The first step to restarting the RDMAILER BTRM is to identify which outgoing mail item the RDMAILER BTRM was processing when it stopped. You must look at RDMAILER's mailbox to find the mail item. This mailbox is the mailbox associated with the userid defined in the PCODE namelist parameter for that RDMAILER BTRM. If your userid possesses the VIP privilege and you have turned it on, or your userid has the privilege given in the MAIL CONFIG facility SNOOP parameter, you can look at this mailbox by typing the PCODE userid in the MAIL For field of the MAIL program. Look at the incoming mail items in PCODE's mailbox for the offending mail item. (The items are incoming mail because they are sent by users to the RDMAILER BTRM for delivery.) Note that postdated and recurring mail are delivered by the RDMAILER BTRM, so the first items in the mailbox may not be the item sought. Now that the item has been found, you can delete the offending mail item and restart RDMAILER, transfer the item to another mailbox where it can be looked at, and restart RDMAILER, or you can edit the

file, fix the addressing, file the changes, and restart the RDMAILER BTRM. While viewing the mail item, the ECHO NAME command is helpful in determining the name of the file.

Delivering SENDFILE mail items

The SENDFILE program can be used by either a VM, MUSIC, or BITNET user to send a file to a MUSIC user. SENDFILE uses the NETDATA format to encode a file. Although rare, a SENDFILE mail item can appear in the postmaster's mailbox that has not been decoded to a readable file. When this happens, the mail item has two identifiable characteristics. The first line of the mail body starts with the character string *\INMR01, and has similar strings in the first few lines. These strings are the NETDATA control record markers. The second characteristic is that the mail item subject is of the format "Sendfile: x y a".

The steps described below can be used to convert a mail item from its NETDATA format into a readable format, and then deliver the mail item to the intended recipients.

Steps:

1) Use the GET command when viewing the mail item to store it in the file name given by the mail subject.
2) Run "$EML:NETCNV filename" to convert the file to a readable format. The $EML:NETCNV program is described under "Mail Utility Programs".
3) Run "$EML:QPUTI filename" to have RDMAILER deliver the mail. The $EML:QPUTI program is described under "Mail Utility Programs".

Mail and Public Directory Editor work file

The personal and public mail directories used by the Mail facility are updated using the MUSIC editor and the REXX macro facility. A directory can reach a size where a message appears telling the user that it cannot insert a new record into the file or that F3 or the file command cannot be used. The message appears because the Editor work file is full and it must be increased in size. Three programs, $EML:DIRECT, $EML:DIRECT.PUBLIC, and $EML:MKNICK use the common REXX macro $EML:NAMDIR.MAC to update the appropriate directory. All three programs have an Editor work file defined as /FILE 1 within the program. Increase the number of records (NREC) value on the /FILE 1 statement to increase the Editor work file.

Creating a site Mail Profile

You can modify the site Mail Profile if your site has a requirement to create a standard mail profile for all new mailboxes. If a site Mail Profile is not created, new users do not have any Mail Profile options preset and they must set the options themselves.

ADMIN 4 5 3 "Update site Mail Profile" can be used to create/modify a site Mail Profile that is copied to all new mailboxes at creation time. The Mail Profile facility is used to create the site Mail Profile. The site Mail Profile is stored in $EMM's mailbox. Only the Mail Profile from $EMM's mailbox is copied to new mailboxes at creation time. The Name and Email Id fields of the site Mail Profile are not copied to the new mailboxes.

This facility can be used to set any Mail Profile options that should be copied to new mailboxes. For example, the site wants mail sent to be copied to a file automatically. On the "Create and Send Mail Options" screen, simply set the "Copy the mail to be sent to a file" to Y, set a filename where to save the mail to be sent (e.g. MAIL.LOG), and set Append to this file to 'Y'. Filenames specified should probably not be userid prefixed. That way when the Mail Profile in the new mailbox is actually used, the filenames all default to the userid of the new mailbox.

Overflow Mailboxes

On occasion, mail cannot be added to the mailbox. The MAIL facility makes an attempt to deliver the mail instead of returning it to the sender. In this case, the mail is appended to a overflow mailbox for the interim. Invoking any Mail facility program that reads incoming mail copies the overflow mailbox to the real mailbox and updates the mailbox counters. The GETMAIL, GETMINFO, SENDMAIL, and POPD programs all do this function, as does reading incoming mail in the MAIL program or refreshing the incoming mail list.

The overflow mailbox has the same filename as the real mailbox, except the first character in the filename is O and not B. For example, if the real mailbox was $MBC:BABCD, the overflow mailbox filename would be $MBC:OABCD.

RDMAILER Kill File

RDMAILER has the facility to delete any incoming mail item from a defined mail sender without delivering the mail item. This feature is commonly called a kill file and it is used to control mail spamming and mail bombing.

ADMIN 4 5 B "Update RDMAILER kill file" can be used to create/modify the kill file. The file $EML:KILLFILE is used to contain the kill file entries which are a list of email addresses. When RDMAILER finds incoming mail from any email address given in the kill file, RDMAILER deletes the incoming mail without delivering it and reports what it has done. The email addresses are listed one per line. Wild card characters * and ? are allowed in the addresses. RDMAILER can automatically detect when the kill file has been updated and it reloads it automatically.

The RDMAILER namelist parameter KILFIL defines the file $EML:KILLFILE for kill file processing.

The RDMAILER namelist parameter KILSIZ can be used to define the the number of characters you want to set aside in core for kill file entries. The default is KILSIZ=16000 which allows for 200 entries of 80 characters each. Since KILSIZ is given as a character count, you must define KILSIZ for the maximum number of characters expected in the kill file. For example, if there are 100 entries in the kill file, KILSIZ=8000 is required (i.e. 100*80=8000). If KILSIZ is set to a value greater than 16000, the MAIL.CONFIG parameter RDREG must be adjusted to its current value plus the difference in K between the new and old KILSIZ value. RDREG is the region size to use for RDMAILER. If a larger requirement for the kill file is set, then the region size for RDMAILER should be set higher by the same amount.

Note: Kill file processing can slow down incoming mail delivery by RDMAILER. It is best to keep the kill file to the minumum number of entries required.

Changing the Release Date of Mail

Any user can send postdated mail. Postdated mail can be sent by setting the release date and/or time on the Sending Mail Options screen of the Create and Send Mail function to something in the future, and then sending the mail. When the mail is sent, it is given to the outgoing RDMAILER BTRM for delivery. RDMAILER will deliver the mail when the release date and time have been reached.

The user can change the release date and time on the mail after the user has sent it if acknowledgements were turned on when the mail was sent. In this case, a copy of the mail appears in the user's outgoing mail list. The select code R-Release allows the user to change the release date and time. When the release date and time are changed, they are updated in the copy in RDMAILER's mailbox. If the updated date and time represents the past, RDMAILER delivers the mail. Note that if RDMAILER has already released the mail and delivered it, the release date and time cannot be changed.

Prior to RDMAILER releasing the mail and delivering it, if acknowledgements were not selected when the

mail was sent, the only copy of the mail resides in RDMAILER's mailbox waiting to be delivered. A privileged user can change the release date and time on the incoming mail item in RDMAILER's mailbox. This privileged user must possess the privilege given in the Mail facility SNOOP parameter or the user must have the VIP privilege. When you are looking at $EMD's incoming mail, enter an "S" (releaSe) select code beside the mail item and a menu is presented to change the release date and time associated with the mail item. As above, if RDMAILER has already released the mail, it cannot be found in RDMAILER's mailbox as it has already been delivered.

This function can be useful to correct entry errors in the release date and time of a mail item.

Mail Broadcast Facility

This facility can be used by system support personnel to broadcast a mail message to a large number of users. This broadcast facility delivers a mail message to each recipient. There may be other methods more suitable to announce something to your user community. Here is a brief list of some other methods of announcement to consider:

- MUSIC console command /MESSAGE ALL message_text
- MUSIC console command /DAILY message_text
- place text in file $PGM:ALERT.FILE and it is displayed at signon
- place text in file $PGM:NEWS.DATA, a one line entry in $PGM:ALERT.NEWS, and it is displayed when user enters MUSIC command NEWS
- create a popup window in FSI main menu so users see message text when the autoprog FSI is invoked after the user signs on

This facility uses the Mail Utility Program $EML:QPUTI to add one or more RFC821/822 compliant mail files to the MUSIC reader queue. The mail files are then delivered as mail by RDMAILER to the recipients. The FILES and SYSCOM system privileges are required to use the Mail Broadcast facility.

The Mail Broadcast facility has both a full screen interface for running the facility interactively and a batch mode which can be used by the MUSIC/SP automatic job submission facility.

In interactive mode, the Mail Broadcast facility can create an edit session for the user to create the recipient list and text to send if no filename is given for either field.

Any email addressing errors that occur while trying to deliver the mail broadcast are send back to the user as a mail message. In this case, the user is the email address designated as being the mail broadcast sender.

Mail Broadcast Facility versus the MAIL program

There are some differences between sending a mail broadcast using this facility and using the MUSIC/SP MAIL program or SENDMAIL program.

- the Mail Broadcast facility is faster
- the Mail Broadcast facility does not resolve nicknames so nicknames cannot be used with it
- both SENDMAIL and the Mail Broadcast facility can be used for the automatic submission of jobs (These jobs are run in BATCH mode.)

Help is provided when the program is run.

MAIL Filter Program

The postmaster's mailbox can be a busy place and the burden on a human to deal with the mail can be very time consuming. The postmaster mailbox filter program can be used to deal with the mail automatically, eliminating or reducing the burden with the postmaster mail.

The postmaster filter program's main purpose is to automatically unsubscribe disabled or deleted local user-ids from the mailing lists for which they still receive list mail. Users refusing mail from mailing lists are treated the same way as a disabled or deleted userid by the filter program. They are unsubscribed from those mailing lists for which they are refusing mail.

The filter program includes the following features:

- ability to track unsubscribes for the previous month
- ability to remember unsubscriptions done for the current week and not reissue the unsubscribe for that week
- sourceroute handling
- mass mailer deletion
- mail gateway handling

Where does Postmaster Mail come from?

The postmaster's mailbox can receive mail from two possible sources. It can be sent mail directly or by the MUSIC Mail facility's message transfer agent RDMAILER, which sends the postmaster a copy of the returned mail when a delivery failure occurs.

When the postmaster is sent mail directly, it is typically a request for help. However, there is a special case when the postmaster is sent mail by an automated process that is reporting an error. The postmaster's email address is used for the Errors To: field of the MUSIC Mail facility's autoforward feature. Thus any autoforwarded mail delivery errors are sent to the postmaster. This setting permits the postmaster to correct the problem and allow the mail to be delivered.

The delivery errors RDMAILER sends to the postmaster can be either mail sent to an invalid user or any other type of problem.

When mail is sent to an invalid user, the mail is returned to the sender and a copy is sent to the local postmaster. So when the MUSIC system administrator disables or removes a userid, the user may still receive mail from mailing lists. Since the userid is disabled or removed, it is now invalid. So the postmaster mailbox gets a copy of the returned mail.

Running the Postmaster Mailbox Filter Program

Consult the section below entitled "First-Time Setup" before running the filter program for the first time.

The program can be run either automatically or manually. To run automatically, the MUSIC system administrator must set up an AUTOSUB job for the postmaster filter program. The file $ADM:AUTO.PSTMST defines a job to run the postmaster filter program. You could add this job to the BTRM that runs AUTOSUB by using ADMIN 4 9. It can be set up to run automatically every day of the week at a given time by

MUSIC's automatic submission facility. It can also be set up to run more than once a day.

When the program is run, the program statistics are appended to the file $EMP:PSTMSTSTATyyyymm, where yyyymm represents the year and month. The statistics are also displayed with the program output as is the actual actions taken by the program. When the job is set up to run automatically, it can be configured to have the program output sent to you via email by using R=MAIL on the ID statement in the job.

The program looks at all items in the postmaster's mailbox and takes action on what it is programmed to handle. It can deal with "mail gateway temporary send errors" and "mass mail senders". Also, it can unsubscribe users from identifiable mail lists. Items the program has taken action on are deleted after whatever processing that is required for the item is done.

For those items the program cannot take action on, the program does not mark these items in the mailbox as read (ie old). So if you read the postmaster's mail, all items are marked new, and you cannot tell which items were left by the filter program and which items arrived after the program was run. It may be convenient to run the filter program manually just before reading the postmaster's mail. Then, only the current mail will be visible.

The program can be run manually on any userid that meets the following requirements:

1) the userid must have the SYSCOM, FILES, and LSCAN privileges

2) the userid must be allowed as a surrogate for the postmaster's mailbox or the userid must have the privilege defined by the SNOOP namelist parameter of the MUSIC Mail facility. Your MUSIC system administrator can provide this information.

The program is run manually by typing at *Go:

```
$EML:PSTMST.UNSUB
```

What does the Filter Program do?

The filter program looks at all items in the postmaster's mailbox. There are a number of cases that the program can handle: mail sent to the postmaster, mail sent by mass mailers, mail sent by identifiable mailing lists, mail sent by identifiable automated mail list managers, chain mail, and friendly mail destined for an invalid user.

1) Mail sent to the postmaster

   Mail sent by an individual requesting help is not acted on by the filter program.

   Mail gateways can report a temporary condition where the mail could not be sent for "x" time but will be retried for "y" days. This mail does not signify a failure but simply represents a delivery status notification (DSN). This type of mail is deleted from the postmaster's mailbox by the filter program. The file $EML:UNSUB.GATEWAYS defines the mail gateways recognized by the filter program and what criterion is used to indicate the DSN which can be ignored.

2) Mail sent by a mass mailer

   A mass mailer or spammer is someone who has sent mail to a large number of addresses. The postmaster mailbox will receive the mail for any invalid userids who were a part of the mailing. The filter program can deal with mass mailers statically and dynamically.

   In the static method, mass mailers can be identified when you are reading the postmaster's mail. The email addresses associated with these mass mailers can be added a static list. This list is used by the

filter program in future runs to delete mail sent by any email address within this static list. The static list is the file $EML:UNSUB.SPAMMERS.

In the dynamic method, the postmaster filter program identifies mass mailers that have sent mail to a number of invalid users. It deletes the mail from these senders once an initial number of mail items has been surpassed.

See the topic "Mass Mailers" below for further details.

3) Mail sent by identifiable mailing lists

When a userid is disabled or removed by the MUSIC system administrator, the user may still receive mail from mailing lists. Since the userid is disabled or removed, it is now invalid. When mail is sent to an invalid user, the mail is returned to the sender and a copy is sent to the local postmaster. When the filter program encounters an item from an identifiable mailing list, it sends a unsubscribe request to the mailing list manager on behalf of the user, and deletes the item from the postmaster's mailbox.

The filter program can identify a mail list if the mail is sent by listproc, listmanager, majordomo, list-serv, mailserv, maiser, mxserver, owner-, -owner, -request, -error, -errors, -admin, hpcnews@newsmaster.tgc.com, automail@power.globalnews.com, membership@match.com, info.apple.com, mailbase@mailbase.ac.uk, or wsmith@wordsmith.org.

The filter program can deal with unsubscribe requests that require a confirmation and subscription renewals. Some mailing list owners may have made these features available with the mailing list they administer.

4) Mail sent by identifiable automated mail list managers

Mail sent by any of the senders listed below is examined by the filter program to determine if the mail was sent in response to the unsubscribe request and it is deleted if it matches that criteria. These senders are:

```
*REPLY*, *POSTMAST*, *DAEMON*, *MAJORDOMO*, *SUPPORT*, *AGENT*,
*MAILBASE-ADMIN*, *LISTPROC*, *LISTSERV*, *MAISER*
```

5) Chain mail

When you are reading the postmaster's mail, you may notice that someone has sent chain mail to one or more local users. Some institutions have a Code of Conduct that forbids chain mail. If a local user is sending chain mail to invalid local user email addresses, you can have the filter program send this user the appropriate section of your Code of Conduct as a warning. The file $EML:UNSUB.CHAIN contains definitions for the variables required to support chain mail identification in the original subject of the mail item. If the originator address matches chain mail address that you trap, the warning is sent and the item is deleted. If the addresses don't match, the item is just deleted.

6) Friendly mail destined for an invalid user

Mail is often sent to invalid local users and winds up in the postmaster's mailbox because it could not be delivered. If the original mail subject contains any of a list of friendly words or phrases, e.g. hello, then the mail can be deleted. The list of friendly words/phrases, which when matched by the filter program allow it to delete the mail, is defined in the file $EML:UNSUB.FRIENDLY.

Otherwise the mail is left in the postmaster's mailbox.

Filter Program Statistics Data

The filter program statistics data is appended to the file $EMP:PSTMSTSTATyyyymm when the program is finished. It is also displayed in the program output.

The following defines each of the displayed fields:

Items in Postmaster mailbox	number of items in the mailbox when the program is started
Items read today	number of items actually examined today
Items deleted	total number of items deleted from mailbox
Errors	total number of errors encountered during processing, including mail fetch errors and unsubscribe request send errors
Total potential unsubscribes	number of total possible unsubscribes done, including unsubscribes sent and matches with previous unsubscribes
Unsubscribes sent	number of unsubscribes sent this run
Matches with previous info	number of unsubscribes already done within the last week
Entries reread to match	number of mail items that had to be reread to reach the bounced mail delimiter line
Entries sent to postmaster	number of items sent directly to the postmaster that were not acted on
Autoforward resolve errors	number of items in mailbox that were autoforward resolve errors
Subject not signifying a rejection	number of items that had the letters UNDELIVER in the subject
Mail gateway messages deleted	number of items sent to the postmaster by mail gateways that were temporary send errors
Friendly messages deleted	number of items which were deleted because the mail subject matched friendly phrases
Chain mail messages encountered	number of items identified as chain mail and deleted
Renewals deleted	number of items that represented subscription renewals that were deleted
Confirm unsubscription sent	number of items which required a confirmation of an unsubscribe request and the confirmation request was sent
Sendfile mail items delivered	number of sendfile items that were decoded and delivered
BCCs deleted (potential spammers)	number of items that had a BCC: RFC822 header and were deleted. Spammers use this method to suppress the mail list.
Total spams detected and deleted	

number of items sent by mass mailers that were deleted

Total unique dynamic spams detected
number of unique addresses detected where at least "newspam" mail items from this email address were found in the postmaster mailbox. "newspam" is a variable defined in the filter program. The default is newspam=5.

Spam count for spamidn — number of items sent by mass mailer n as defined in $EML:UNSUB.SPAMMERS that were deleted

Dynamic spam detected n times for address:
number of times where "n" mail items from this email address were found in the postmaster mailbox. "n" is at least "newspam" in size. "newspam" is a variable defined in the filter program. The default is newspam=5.

Since the filter program can track more than one mass mailer, if any mass mailers are detected during the run, the final number of items detected for each mass mailer is displayed with the statistics.

Filter Program Actual Actions Output

The filter program displays actions it has taken when it is run. These actions include:

- reporting the number of items found in the mailbox
- adding an entry to the WEEKDAY file and incore cache to record the unsubscribe request (see further information below)
- reporting an item has been previously unsubscribed
- reporting an item that has been sent by a mass mailer
- reporting any error that occurred during the run

Other Features of the Filter Program

The current week's unsubscribe requests are stored in the files $EMP:WEEKDAY.n, where n is 1 to 7 representing Sunday to Saturday. When the program is run on Wednesday, the unsubscribe requests done for that day are stored in $EMP:WEEKDAY.4. An unsubscribe request is issued once for the week. Then, for the week following the request, any mail from the mailing list sent to the same user is simply deleted from the postmaster's mailbox without resubmitting the unsubscribe request.

When the entries in the WEEKDAY files are older than one week, the filter program copies them to the file $EMP:MONTH. This file contains the previous 4 weeks of unsubscribe requests. Also, the filter program keeps only the previous 4 weeks of requests in the file $EMP:MONTH. All entries in that file older than 4 weeks are deleted.

The file $EMP:MONTH is not used by the filter program. However, sometimes the user will continue to receive mail from the mailing list despite your efforts to unsubscribe them. This file can be used as a reference tool to find out when the unsubscribe request was sent. The previous request information stored in this file identifies that it was sent. However, since the user is still receiving mail from the mailing list, it appears that the unsubscribe did not take place. The unsubscribe request needs to be resent either automatically or manually, and possibly sent to another address. The unsubscribe request is resent automatically the week after the first request was sent by the filter program if the user is still receiving mail from the list. The topic

below "Unsubscribing a user from a mail list when viewing the postmaster's mail" describes how to resend the unsubscribe request manually.

First-Time Setup

The following things must be done before the program is run for the first time:

1) the userid where the program is run must have the SYSCOM, FILES, and LSCAN privileges

2) the userid must be allowed as a surrogate for the postmaster's mailbox or the userid must have the privilege defined by the SNOOP namelist parameter of the MUSIC Mail facility. Your MUSIC system administrator can provide this information.

3) edit the file $EML:UNSUB.DEFINES and set the variables "mysite" and "sourcerouteid" to agree with your system settings.

4) edit the file $EML:UNSUB.GATEWAYS and define your local mail gateways or set gateways to zero if you have no local gateways. Also set the gateway mail criteria allowing deletion of the mail.

5) edit the file $EML:UNSUB.CHAIN and set the variables required for chain mail trapping.

6) run ADMIN 4 5 1 3 to reconfigure the Mail facility. The program $EML:QPUTI has been added to the configuration process and it must be created.

Components

$EML:UNSUB.DEFINES	local definitions required by the filter program
$EML:UNSUB.GATEWAYS	defines gateways and the criteria used for the filter program gateway matching
$EML:UNSUB.SPAMMERS	defines the known mass mailers
$EML:UNSUB.CHAIN	defines the variables used to trap chain mail
$EML:UNSUB.FRIENDLY	defines the friendly phrases used to match against the subject of friendly mail that can be deleted
$EML:UNSUBCL	program that manages the unsubscribe requests and moves requests from the $EMP:WEEKDAY.n files to the $EMP:MONTH file, and deletes requests from the $EMP:MONTH file
$EML:GETMINFO	program used to get a list of the items in the postmaster's mailbox
$EML:GETMAIL	program used to delete mail items from the postmaster's mailbox
$EML:QPUTI	program used to put the unsubscribe request in the MUSIC reader queue so that RDMAILER can deliver it to the intended recipient

Mass Mailers

As described above in the topic "What does the Filter Program do?", under item "2) Mail sent by a mass mailer", mass mailers can be dealt with in a static and/or a dynamic way.

1) Static method
When you are reading the postmaster's mail, you may notice that someone has sent unsolicited mail to a number of invalid local userids. Instead of deleting all of the items manually, the filter program can be set up to delete these items for you. If the email address of the mass mailer is appended to the file $EML:UNSUB.SPAMMERS, future filter program runs will delete all mail from any mass mailer listed in the file $EML:UNSUB.SPAMMERS.

Email addresses can be appended to the file $EML:UNSUB.SPAMMERS manually or automatically. The instructions at the top of the file $EML:UNSUB.SPAMMERS document the steps required to manually append an email address to the file. The changes must be saved before the new mass mailer addresses are used by future runs of the filter program.

The MUSIC Mail facility's SPAMMER command can be used to automatically append the pointed to email address to the file $EML:UNSUB.SPAMMERS. The email address is not appended to the file if it is already in the file. The SPAMMER command can be entered only while viewing the postmaster's mailbox or for a user possessing the CODE privilege defined by the MAIL program SNOOP namelist parameter or the VIP CODE privilege.

The file $EML:UNSUB.SPAMMERS used to track the mass mailers can either be used in a static fashion, never changing, or in a dynamic fashion (i.e. being updated whenever a new mass mailer is encountered). With the growing escalation of mass mailers on the Internet, the dynamic use of this file is of great advantage to the postmaster.

It is up to the person running the filter program to update this file, either by adding an entry for a mass mailer when one is identified by reading the postmaster's mail, or by deleting a mass mailer when one has instituted an aging policy for entries in this file.

2) Dynamic method
The postmaster filter program can be set up to identify mass mailers dynamically as it runs. The filter program "newspam" variable defines the number of times a sender email address can occur before it triggers the dynamic spam detection function. The default is newspam=5 (i.e. identify 5 mail items from the same sender before deleting subsequent mail items from this sender). Setting newspam=0 turns off dynamic mass mailer detection. When more than "newspam" mail items have been identified from the same sender, further items from this sender are deleted.

When the filter program runs, if a mail item is not acted on by any other action, it is examined as a candidate for mass mailing. A list of fifty dynamic mass mailers is maintained while the program runs. An age and least recently used algorithm is used to maintain this list. An entry with at least "newspam" mail items encountered is always kept.

Note that "newspam" mail items are left in the mailbox and are not deleted by the filter program. These items can be deleted manually. However, it may be convenient to leave these items in the mailbox to catch subsequent mass mailings by this sender in the coming days. You may also find it convenient to add these sender email addresses to the file $EML:UNSUB.SPAMMERS so that future filter program runs will automatically delete items from these senders.

A list of the dynamic mass mailers is printed at the end of the program report.

Unsubscribing a User from a Mail List

The MUSIC Mail facility's UNSUB command, which can be entered only while viewing a mail item in the postmaster's mailbox, can help the postmaster with the manual task of unsubscribing a user from a mailing list. This manual unsubscribe process requires the tailoring of the unsubscribe request by the viewer. When an item has been identified by the person reading the postmaster mail as list mail which should be acted upon, and the original local recipient was either invalid or refusing mail from the list, then the unsubscribe

request can be done.

After the UNSUB command is entered, an edit session is presented with a prototype unsubscribe request displayed. Care must be taken to preserve the format given, otherwise the request sent when the changes are filed may cause other mail system problems. The SITE and LISTX editor macros can be used to set the variables "site" and "list" to their respective correct value as determined by examining the file being edited.

The prototype unsubscribe request presented is an RFC821/RFC822 compliant email message. It is very important that the structure remain as presented and any edification must pay attention to dangling commas for the RFC822 To: field.

Some mass mailers suggest a method different from that described above to have an email address removed from their mass mail list. The method is usually described within the received mail and it requires sending a reply to the original mail item with the word "remove" in the subject. Be forewarned that by sending the remove reply, you are in fact registering for mass mail. Mass mailers are using this method of false advertising to get a list of valid email addresses which they can use in the future.

MAIL Exits

A number of exit routines are available in the Mail facility for use. These routines, EXROUT, EXDPRT, EXRADR, and EXDCDR allow the site to tailor various aspects of MAIL.

The EXROUT exit routine is a replacement for calls to the MUSIC routetable to determine routes available for the user.

The EXDPRT exit routine can be used as a substitute for MAIL's printer display feature that allows a user to select the printer for printing his/her mail.

The EXRADR exit routine allows the site to have a selection menu for recipients as input when the user enters "?" in a recipient field in MAIL.

The EXDCDR exit routine allows the site to decode encoded mail when the item has been selected but before the user's selected operation is performed on the mail item.

The following lists the Mail programs which are exit aware, and lists which exits are available in which programs.

```
$EML:MAIL     - EXROUT,EXDPRT,EXRADR,EXDCDR
$EML:MAILBOOK - EXROUT,EXDPRT,EXDCDR
$EML:MPROF    - EXROUT
```

In order to install the exit procedures, you must do the following:

1. Replace the file $EML:MAIL.exit_name.S with your file.

2. Compile/assemble your exit routine into the file $EML:MAIL.exit_name.OBJ.

3. The program $EML:MAIL.MAKE can be used to make sure that any routines (.OBJ) exist on disk before the appropriate Mail program(s) are link-edited. (Note that .OBJ files are commonly not resident - you might have to use ADMIN 3 5 to restore these files.) Once satisfied, re-link the appropriate link-edit link-edit by running $eml:MAIL.MAKE and selectively running the link-edit for MAIL, MAILBOOK, and/or PROFILE with this program.

Note that the exit object decks are listed in the file $EML:MAIL.EXITS, $EML:MBK.EXITS, and $EML:PROFILE.EXITS. These files are included in the link-edit for their respective program.

The program $EML:MAIL.MAKE reviews the linkage editor output after the link edit is done. If there are no errors, then your new version may be in production. Note the the MAIL program requires a system reIPL or a SYSUPDATE to put the new LPA module for mail into production.

Description of Mail Exits

The EXROUT Exit

This exit routine is offered as an alternative to "CALL ROUTE(2,ROUTENAME,IRC)".

The EXROUT exit is invoked in MAIL, MAILBOOK and MPROF as

```
CALL EXROUT(IRC,ROUTE)
```

where

```
IRC:    integer return code
        -1=this exit routine is not active
           MAIL,MAILBOOK,MPROF will call route
         0=route okay as checked by exrout
       <>0=something wrong with the route
           MAIL,MAILBOOK,MPROF will post an invalid route message.
ROUTE:  8 character routename
```

The EXDPRT Exit

This exit routine is offered as an alternative to the MAIL/MAILBOOK PRTGET routine invoked by "CALL PRTGET(ROUTENAME)".

The EXDPRT exit is invoked in MAIL and MAILBOOK as

```
CALL EXDPRT(IRC,ROUTE)
```

where

```
IRC:    integer return code
        -1=this exit routine is not active
           MAIL and MAILBOOK will call PRTGET.
        0=exit routine did processing, use route on return.
ROUTE:  8 character routename returned back to MAIL/MAILBOOK
        padded with blanks.
        If non-blank, MAIL/MAILBOOK will place the route name
        into the printer name field on the print screen.
```

The EXRADR Exit

This exit routine is called when a "?" is placed in any of the address fields to send mail (ie TO or CC fields). The substitution address is returned as the same field passed.

The EXRADR exit is invoked in MAIL as

```
CALL EXRADR(IRC,RADR)
```

where

```
IRC:    integer return code
        -1=this exit routine is not active
           MAIL is continue processing "?" as an email address.
        -2=no substitution address was given
           This means that from your selection panel no address was
           selected.  In this case the "?" gives invalid userid.
        0=resolve radr
        <>0=something wrong in processing
            MAIL will post the error message IRC found in the mail
            mail messages file $EML:MAIL.MSGS)
RADR:   132 character email address returned to caller to resolve
            (must be padded with blanks)
```

The EXDCDR Exit

This exit routine provides a means for mail to be decoded after a mail item has been selected but before the user's selected operation is performed on the mail item.

The EXDCDR exit is invoked in MAIL as

```
CALL EXDCDR(IRC,MFILE)
```

where

```
IRC:    integer return code
        -1=this exit routine does no processing.
         0=exit routine did processing.
MFILE:  64 character mail text file passed from MAIL.
```

MUSIC/SP Administrator's Reference

Part 3 - Chapters 10 to 16 (AR_P3.PS)

Chapter 10. TMENU - Tailoring the User View

Overview of TMENU

TMENU is the name of the program on MUSIC for creating tailored user views of MUSIC. This facility is available for making your own menus, or tailoring existing menus to better suit your needs and those of your users. Some of the available features include filtering the commands to which users of a particular view have access, signing off from MUSIC when the menu is terminated, and executing your own programs. A sample MUSIC view for programmers, called PROG, can be modified to suit your installation's needs. Programs that are used often can be added to the menu or items on the menu can be exchanged for other programs offered at your installation. For example, figure 10.1 which lists PROG.MENU, the companion menu for PROG, can be added to or altered so that when the sample program PROG is executed, the facilities you defined are available to the users of this view.

Making a Menu

The menu is used to provide four functions:

1. provides TMENU with the menu items and descriptive text.
2. provides TMENU with the names of program files to be executed as required by the user.
3. provides labels for the menu items for menu selection.
4. provides for default parameters to be passed to the respective program.

```
)+ - PROG
)% -  PROGRAMMER'S MENU  -
1. EDIT     - edit an existing file
2. EDIT     - edit a new file
3. LIBRARY  - look at your library
4. PASSWORD - change your sign-on password
5. PROFILE  - change your code options
6. NEWS     - list latest news
)1 EDITOR
)2 EDITOR-NEW
)3 TODO.LIB
)4 PROFILE>NEWPW
)5 PROFILE
)6 NEWS/
```

Figure 10.1 - Listing of PROG.MENU

The General Format of the Menu

The menu is made up of four types of specification lines and is divided into two logical components. The first component describes the menu items (selection codes) and the descriptive text. The second component describes the actual program names and their parameters if any. The program names and the menu items are connected by the *option code*. For example, item 'H' in Figure 10.2 has both a descriptive entry in component one as well as a program name and parameters definition in component two.

```
)+                            <--------  Menu start flag
)% -  TITLE OF MENU  -        <--------  Menu title line

1. Item One                   <--------
2. Item Two                           |  Option codes
P. Profile, print $ remaining         |  and descriptions
H. Help                       <--------

)1 PROG1                      <--------  Corresponding
)2 PROG2                              |  file and
)P PROFILE/PRINT$                     |  parameter
)H HELP                       <--------  specifications
```

Figure 10.2 - Sample Menu for User View Tailoring

)+ is the marker that indicates that the file being processed is a menu file. It must be the first line of a menu file and appear only once. An optional 1-16 character menu name can be placed on this line. This text will be placed on the extreme right of the title line of the menu.

)% is the title marker. The title marker provides a title for the menu posted on the screen. A blank must follow the "%" character. If more than one title line is present, the last one will be used.

The title itself will be centered by the menu formatter of TMENU, therefore do not attempt to center it. The standard that should be followed is to place a dash (-) before and after the title text separated by one or two blanks. Example:

)% - title -

The title will be centered and padded by dashes.

---------- title ----------

xx. this is the one or two character combination that the user will know the item by. This becomes the item option code or name.

The option code character(s) must be directly followed by a period and a blank (.).

General Format of the Option Specification Line.

```
Type 1

   xx. OPTION  -  Description
```

xx.	is the 1 or 2 character option code followed by a period and a blank.
OPTION	is the option name (usually in upper case).
-	is an optional delimiter.
Description	is the option description.

```
Type 2

   xx. Description
```

xx.	is the 1 or 2 character option code followed by a period and a blank.
Description	is the option description.

Guidelines for Specifying Option Lines

Option Code

- The option code can be numbers or characters except period and close parenthesis.
- Do not use roman numerals, arabic numerals are more familiar and more easily distinguishable. When a short number of items are used letters indicative of the function performed by the item is appropriate, for example, 'P' for Profile and 'H' for Help.
- If letters and numerals are to be mixed, it is best to place all the numeric option lines in sequential order followed by the letter option codes.
- The option lines should be in the order of greatest anticipated usage (not withstanding the previous point). This will facilitate usage.

Option Name

- The option name is not always used or appropriate, however when this format is used it should be in upper case. (Type 1)

Option Description

- The option description should begin with a capital letter.
- The description should be a statement with correct syntax. Statements that seem to be questions should

be avoided.

```
        1. Submit a File
        11. Submit a File
        H. Help on File
        1. HELP  -  Get help on file usage
```

Figure 10.3 - Sample Option Specification Menu

Program Specification Lines

Besides the program name and the default parameters (program options), two other types of information can be included on the program specification line. The first is whether the user's parameters are to override, not override, or be appended to the default parameters. The second is whether the program is to be an 'always' program, a noncancelable program, or just an ordinary program (that is neither of the first two characteristics).

```
  )xx programYparametersZ
```

)xx	The required close parenthesis indicates a program specification line. *xx* is a 1- 2-character item name or option code. There should be as many of these lines as there were item specification lines. These lines define the files to be executed and the default parameters if any.
program	is the file name to be scheduled for execution.
Y	is one of 4 optional delimiters (| + - >).
parameters	is the optional parameter list.
Z	is one of 7 optional program types (blank % ! " ? / <).

Defining Parameter Processing and Passing

The parameters are separated from the program name by one of four special characters. These characters are shown in Figure 10.4.

```
      |  vertical bar   - user if specified, otherwise
                          the default parm
      +  plus sign      - user parm prefixed to defaults
      -  minus sign     - user parm appended to default
      >  greater than   - user parm ignored
```

Figure 10.4 - Parameter Processing Flags for User View Tailoring

Each of these serves to delimit the program name from the parameters as well as to indicate the parameter

handling option.

The following defines and describes these four delimiter characters.

| The vertical bar indicates that any user specified parameters are to override the default parameters. That is either the user's parameters or the default parameters are used and not both.

+ The plus sign indicates that any user specified parameters are not to override the default parameters. Rather the user parameters are to be prefixed to the default parameters. That is both specifications are passed. The two parameter lists will be separated by a blank.

- The minus sign indicates that any user specified parameters are not to override the default parameters. Rather the user parameters are to be appended to the default parameters. That is both specifications are passed. The two parameter lists will be separated by a blank.

> The greater than sign indicates that any user specified parameters are to be ignored. Only the default parameters are passed.

Defining Program Types

One of five characters can be used to describe the program type. These characters and their functions are given in Figure 10.5.

```
" " no type flag     -  "nonalways"/cancellable
 %  percent sign     -  "always"/noncancellable
 !  exclamation mark -  "nonalways"/noncancellable
 "  double quote     -  "always"/cancellable
 ?  question mark    -  menu file
 /  slash           -  MUSIC command
 <  less than sign  -  TMENU command
```

Figure 10.5 - Program Type Flags for User View Tailoring

Any one of these can appear as the last character of the parameter list to indicate the program type. These program characters, when specified, must be appended to the parameter list on the program specification line. It will be converted to a blank and is not considered as part of the parameter list. If no parameter list or *parameter option* character (delimiter) is required or used the *program type* character can be appended to the program name.

The scan for the program type is done from right to left. Therefore the *program type* characters can appear as part of the parameter list provided that a program type character is used at the end of the parameter list.

The following defines and describes these five characters.

" " A blank or no program specification indicates that a program is to be a cancellable nonalways program. When no program type character is specified " " is the default.

% The percent sign indicates that a program is to be a noncancellable always program.

! The exclamation mark indicates that a program is to be noncancellable and nonalways.

" The double quote mark indicates that a program is to have the *always* attribute and is to be cancellable.

In brief an *always* program is one that is automatically reinvoked after it has completed running. As well if it scheduled a program via a NXTPGM call, at the completion of the scheduled program the always program is reinvoked.

? The question mark indicates that the program name is really a menu file.

/ The slash indicates that this is a MUSIC command string. Any commands listed with filter are not allowed here. See the TMENU FILTER option.

< The less than sign indicates that this is a TMENU command.

Note: Usage of the program types ", !, and % dictates that the program filenames given with these specifications are limited to 22 character filenames. All other program filenames, menus, or TMENU commands given abide by the normal MUSIC system rules and limits.

```
)3 GORK PARMS! <-default parms are overridable, the program
                  is noncancellable and nonalways.

)3 GORK!       <-default parms are overridable, the program
                  GORK is noncancellable and nonalways.

)3 GORK+PARMS% <-user parms will precede default parms, the
                  program is cancellable and always.

)3 GORK-PARMS  <-user parms will follow default parms, the
                  program is cancellable and nonalways.

)3 GORK>PARMS" <-no user parms, default parms only and gork
                  is an "always" and noncancellable program.

)3 GORK>PARMS  <-no user parms, default parms only, gork is
                  cancellable and nonalways.

)3 GORK?       <-the file name is a menu to be displayed,
                  and not a program.

)3 NEWS/       <-This indicates that news is a MUSIC command
                  and will be executed as such.

)3 END<        <-indicates that the user will be returned to
                  *Go or the calling program.
```

Figure 10.6 - Sample Program Specification Menu

Invoking TMENU

After a menu has been created you must set up an exec file for it. For example, the following is the exec file required for PROG:

```
/INC TMENU
MYNAME='PROG',FMENU='PROG.MENU',STACK='@PROG.STACK'
```

TMENU Parameters

ACCESS='filename'

Where *filename* is the name of the file containing a list of userids and access levels. Userids in the file can include wild characters. The access level can be any number from 0 to 9999999 but access is granted only for the value 1. If a level is not included with a userid entry then no access is assumed.

When a user requests a menu facility, his userid is matched against the ACCESS file (if it exists). The first userid in the file that matches his userid determines whether he will have access or not.

Note: If the ACCESS parameter is not specified, then all userids have access to that menu facility.

Example:

```
* This is a sample access file
ccfp 0       Keep this guy off this facility
????xxx 0    His friends cannot use it either
??????? 1    Everyone else can
```

In this example the userid *ccfp* (ccfp 0) and all userids of the length 7 and ending with *xxx* (????xxx 0) are not allowed access. All other userids have access (??????? 1). Notice that comments can be included by using an asterisk (*) in column one or they can be appended after the userid and access level.

CAL=t or f

Specify either t or f to indicate true or false respectively. If CAL=t is used a calendar is posted on the right of the menu. If CAL=f is used no calendar is displayed. When CAL is not specified the default is CAL=t.

CANCEL=t or f

When CANCEL=t is specified the user can return to *Go by PA1, /CANCEL ALL, or successive END commands or PF3's. However when CANCEL=f, /CANCEL is rejected and normal termination of TMENU results in a sign-off. In other words CANCEL=f prevents users from returning to *Go. The default is CANCEL=t.

ENDCMD='filename'

Where filename is the name of the file to be executed when the user exits the facility. ENDCMD is honoured only when CANCEL=t. filename could be an accounting program that records usage of the facility.

ERRFIL='filename'

Where *filename* is the name of a file that contains the messages for the program. The default file name is $TDO:TMENU.MSGS[@]. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file has National Language variants E (Spanish), F (French), P (Portuguese), K (Kanji-Japanese) supported through the [@] placeholder in the file name. These files are produced from $TDO:TMENU.MSGS[@].S by the MAIL message builder program, $EML:MSG.BLDR. This program is described in the Mail chapter in this manual. The $TDO:TMENU.MSGS[@].S files are normally off-line and may have to be restored from the source tapes if use is required.

FILTER=t or f

When FILTER=f, any command string which is not an option code or a command used by TMENU is passed to the MUSIC command processor. When FILTER=t, the option codes, the commands used by TMENU, and only the commands and/or programs listed below the parameter line are allowed. The default is FILTER=f.

A number of commands processed by the MUSIC terminal scanner can not be issued from TMENU. These commands are: /COMPRESS, /CTL, /DISCON, /NS, /PAUSE, /PROMPT, /REQUEST, /RUN, /STATUS, /TIME, /TEXT, /TABIN, /TABOUT, /USERS, /VIP, and /WINDOW.

The commands can be entered with any abbreviation up to the minimum abbreviation as specified by an integer to its right. If no minimum abbreviation is specified the command will have to be entered in its entirety as specified. Before the command string is passed the abbreviated name is expanded to its full length. For example, the following abbreviations would be allowed for this list of commands:

Command	Abbreviations
PURGE 2	purg, pur, pu
OUTPUT 3	outpu, outp, out
SUBMIT 2	submi, subm, sub, su
PRINT 3	prin, pri

If no commands and/or program file names are specified, then the "/" command is not allowed at all. This is the method for barring the use of the "/" command. For example, in the file ABC:

```
/INC TMENU
MYNAME='ABC',FILTER=T
```

ABC will not allow the use of the "/" command and will use the default menu of TODO.MENU as a first menu. The calendar will be posted on each menu screen. The stack file will be @TMENU.STACK.tcb where *tcb* is the current TCB number.

FMENU=menu — Where *menu* is the name of the first menu file to be displayed. When no menu file is specified the default menu, TODO.MENU[@] is used. This file has National Language variants E (Spanish), F (French), P (Portuguese), K (Kanji-Japanese) supported through the [@] placeholder in the file name.

FS=t or f — When FS=t and TMENU is invoked on a 3270-type terminal, automatic full screen support is provided. On a non full screen terminal such support is never allowed.

When FS=f, no matter what terminal type you are using, only non full screen support will be available. The default is FS=t.

KEYFIL='filename' — Where *filename* is the name of a file that contains the site definitions for the PF keys used in the program. Lines in this file have the following format: "PFnn x" where *nn* is a number from 1 to 24, and *x*, the definition, can be from 1 to 50 characters.

Users can store their own PF key definitions for later use by entering a definition for any PF key. These definitions are stored in the file *USR:TMENU.KEYS at the program termination for use when the program is invoked again. If this file is available when the program begins, then these definitions are used and the site definitions are not. Lines in the file *USR:TMENU.KEYS have the same format as the lines in the KEYFIL filename.

LOG=t or f — The LOG=t parameter can be used if you wish to keep a usage log. LOG=f is the default.

LOGFIL='filename' — Where *filename* is the name of a file that is used to log information from the program. The default file name is *USR:@TMENULOG.tcb, where *tcb* is the

current TCB number.

MDELAY=n — Where *n* is a value indicating the delay between attempts at finding out if mail is waiting. When mail is detected, a mail waiting message is posted. When *n* is a positive integer it is taken as representing minutes. When *n* is negative it is taken as seconds. To stop all mail checking, use a value for *n* that is greater than 24 hours such as -86400 secs or 1440 mins. This will prevent the posting of even the initial mail waiting message at the start of the program. The default is MDELAY=-300 or MDELAY=5 (five minutes).

MYNAME='filename' — Where *filename* is the name of the file. In our example it would be PROG. If this parameter does not specify the file name in which it is stored TMENU will not make the file an *always* program.

REMS=t or f — When REMS=t, TMENU will automatically display any reminders for the current day. You can set reminders and query for the day's reminders with the REMIND command or the use of PF2. The default is REMS=t.

When REMS=f the REMIND facility will not be available, and the REMIND command and PF2 will be ignored. The default is REMS=t.

STACK='filename' — Where *filename* is a 1-46 character name of a file to be used as the stack file. The tcb number is appended to it as follows: *filename.tcb*. The stack file is used to store a trace of menus and programs that have been called. In this way the program can return to the correct menu as it drops back through the stack. When no stack file is specified the default "@TMENU.STACK.tcb" is used, where *tcb* is the TCB number.

Note: Use JOBONE or utility program FILE.DELETE to delete stack files that have accumulated from abnormal termination of the program.

Built-in Functions

Command	Description
CANCEL	exit the current menu.
END	exit the current menu.
EXEC	pass the string to MUSIC as a command (abbr. /). See the FILTER option.
GETREM	post message reminders for today or no reminders.
HELP	provide help on how to manipulate the menu screen.
KEYS	post the KEYS screen to enter definitions for the PF keys.
MAIL	post message mail waiting or not waiting.
OFF	terminate the program and sign off.
PFnn	show the definition of PFnn, where *nn* is a number from 1 to 24.
PFnn x	set the definition of PFnn to *x*, where *nn* is a number from 1 to 24. The definition can be from 1 to 50 characters long.
=x	return to the first screen and process *x*.
REMIND	view reminder file (abbr. REM).
RETRIEVE	display the previous command in the command area. Up to 5 commands can be recalled.
*	echo the previous command.
*-n	Display the previous nth command entered in the selection area, where *n* is a number from 0 to 4.

Function Key Definitions

Key	Definition
F1/13	provide help on how to manipulate the menu screen.
F2/14	check today's reminders.
F3/15	terminate the current menu.
F6/18	post message mail waiting or not waiting.
F12/24	displays the previous command in the command area Up to 5 commands can be recalled.
PA1	terminate the current menu.
ENTER	process the command area.

Chapter 11. Editor and REXX Considerations

Editor Considerations

The file used for executing the Editor is $PGM:EDITOR. It must have the public and execute-only attributes since the Editor requires certain privileges. You may wish to change this file to redefine the default region size, work file size, function key definitions, and macro library, or issue commands during the edit startup. Bear in mind that this file contains the system-wide defaults. If an individual user has special requirements, they can be met by creating a personal Editor file for that user, rather than changing the defaults for everyone.

There is a similar file, $PGM:BROWSE, used by the BROWSE command.

Region Size

The default region size of 400K is adequate for editing all files. Certain editor macros may require additional memory.

Work File

The EDITOR requires a work file approximately twice as large as the file being edited. The standard size of 6000 records of 128 bytes each allows you to edit a file about 4500 80-byte records. Generally this is sufficient for most users and, when it is not, a personal Editor file with a larger work file solves the problem. The work file space is allocated from the system scratch dataset (SYS1.MUSIC.SCRATCH). If space is limited in this dataset, you may wish to define a smaller work dataset for the default Editor. The BIGEDIT command (file $PGM:BIGEDIT) uses a larger work file.

Commands and Function Keys:

Editor commands can be placed after the /LOAD EDIT statement. These will be issued each time the Editor is started. This is a useful place to define any local function key defaults that are different from those distributed with the system. In order to be consistent with the standard Editor, a "TOP" command should be issued last, to position the line pointer and place the cursor in the command area.

Editor Configuration Switches

There are some option bits at displacement 000C in Load Library member EDITOR. They can be REPed in order to customize some characteristics of the editor. In the distributed editor, all the bits are off. For compatibility with other MUSIC sites, it is recommended that you not change these bits. However, the choice is yours. The bits are defined near the beginning of module EDITIO ($EDT:EDITIO.S).

The bits are as follows:

Bit x'80' If this bit is on, the editor will start with an empty file (as if the NEW option was specified), if the specified file does not exist. If this bit is off, the editor displays an error message and exits if the file does not exist and NEW was not used.

Bit x'40' If this bit is on, the automatic TEXT UC done by the editor in some cases when it starts, is

suppressed.

Bit x'20' If this bit is on, the automatic TEXT LC done by the editor in some cases when it starts, is suppressed.

Other bits: Reserved.

Use the SYSREP utility to change the bits, by running the following job:

```
/INC SYSREP
LIB='SYST'
NAME EDITOR
VER 000C xx
REP 000C yy
```

where xx is the current hex value of the byte and yy is the new hex value of the byte. The first time you do this, xx is 00. After the first time, xx is the current hex value of the byte (the previous yy). yy is the sum of the bit values you want to be on. For example, to set bit x'80' only, yy is 80. To set bits x'80' and x'20', yy is A0. To turn off all the bits, yy is 00.

Notes:

1. The change will take effect at the next IPL of MUSIC.

2. These option bits apply to all edits for all users.

3. They are valid for MUSIC/SP 2.2 and all later releases.

4. You need to reapply the REP after any service fixes that replace the EDITOR member in the Load Library, and after upgrading or reinstalling MUSIC.

5. The displacement 000C will not change.

Editor Macros and Multi Tasking

Editor macros can be written in REXX. When a command is entered that is not a standard editor command it is passed on to the REXX interpreter. The interpreter scans MACLIB PDS for the macro. If found, the macro is read into memory. Editor commands in the macro are passed back to the editor for execution. These may in turn invoke other macros recursively.

If a macro is not found in MACLIB, the command is executed as a MUSIC/SP command concurrently with the edit session, using the multi-tasking facility. Using this, a number of files can be edited at the same time and the multi-session function keys used to switch between edit sessions. The following commands effect the editor's processing of macros and commands.

REXX on/off

Used to enable or disable macro processing.

CMDS on/off/none/macro

Used to control whether multi tasking execution is allowed. There are a number of sample macros on the userid $EDT. They are not documented and are only intended to serve as samples for users who wish to write their own macros. Complete documentation on writing Editor macros is included in the *MUSIC/SP User's Reference Guide*.

KEYS	- define program function keys
BROW	- browse a file while editing
FE	- edit certain fields in a file
GET	- get a new file for editing

REXX Considerations

The file $REX:REXX contains the default control statements for running the REXX interface. This file must be public. You should specify a region size that is suitable for the execution of any REXX programs that are available for general use at your site. Mixing REXX programs with different region sizes specified, can cause errors. To avoid problems, those writing REXX programs should not specify a region size in the programs themselves, but should use a /INCLUDE REXX statement at the beginning of the programs to pick up the system default region size.

The default is set at 1024K which is sufficient to run the Full Screen Interface (FSI) and other REXX applications that come with the system. If an individual requires more memory, create a private file called REXX in the user's library that specifies a larger region. Note that well behaved programs that start with "/INCLUDE REXX" will, in this case, pick up the region size from the user's private REXX file.

When a MUSIC/SP command is issued from a REXX program, the program is terminated, the command executed and the program restarted from where it left off. Because the restart is done from a work file and not the original file, any program privileges assigned to the original file are lost. Components of REXX programs that require privileges should be written as separate commands and invoked as required from the main program.

Chapter 12. TODO Facilities

REMIND Facility

The REMIND program allows you to set reminders of a future event. Reminders appear on the screen when you request the TODO facility.

Program Parameters and Authorization Tables

The executor file for REMIND, $TDO:REMIND, contains some program parameters (in MUSIC namelist format, separated by commas) which you may need to change.

ACCESS='filename' Where *filename* is the name of the file containing a list of userids and access levels. Userids in the file can include wild characters. The access level can be any number from 0 to 9999999 but access is granted only for the value 1. If a level is not included with a userid entry then no access is assumed.

When a user requests a menu facility, his userid is matched against the ACCESS file (if it exists). The first userid in the file that matches his userid determines whether he will have access or not.

Note: If the ACCESS parameter is not specified, then all userids have access to that menu facility.

Example:

```
* This is a sample access file
ccfp 0       Keep this guy off this facility
????xxx 0    His friends cannot use it either
??????? 1    Everyone else can
```

In this example the userid *ccfp* (ccfp 0) and all userids of the length 7 and ending with *xxx* (????xxx 0) are not allowed access. All other userids have access (??????? 1). Notice that comments can be included by using an asterisk (*) in column one or they can be appended after the userid and access level.

CANCEL=t or f When CANCEL=t is specified the user can return to *Go by PA1, /CANCEL ALL, or successive END commands or PF3's. However when CANCEL=f, /CANCEL is rejected and normal termination of REMIND results in a sign-off. In other words CANCEL=f prevents users from returning to *Go. The default is CANCEL=t.

ERRFIL='filename' Where *filename* is the name of a file that contains the messages for the program. The default file name is $TDO:REMIND.MSGS. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $TDO:REMIND.MSGS.S by the MAIL message builder program, $EML:MSG.BLDR. This program is described in the Mail chapter in this manual. The $TDO:REMIND.MSGS.S files are normally off-line and may have to be restored from the source tapes if use is required.

FILTER=t or f When FILTER=f, any command string which is not an option code or a command used by REMIND is passed to the MUSIC command processor. When FILTER=t,

the option codes, the commands used by REMIND, and only the commands and/or programs listed directly below the parameter line are allowed. The default is FILTER=f.

A number of commands processed by the MUSIC terminal scanner can not be issued from REMIND. These commands are: /COMPRESS, /CTL, /DISCON, /NS, /PAUSE, /PROMPT, /REQUEST, /RUN, /STATUS, /TIME, /TEXT, /TABIN, /TABOUT, /USERS, /VIP, and /WINDOW.

The commands can be entered with any abbreviation up to the minimum abbreviation as specified by an integer to its right. If no minimum abbreviation is specified the command will have to be entered in its entirety as specified. Before the command string is passed the abbreviated name is expanded to its full length. For example, the following abbreviations would be allowed for this list of commands:

Command	Abbreviations
PURGE 2	purg, pur, pu
OUTPUT 3	outpu, outp, out
SUBMIT 2	submi, subm, sub, su
PRINT 3	prin, pri

If no commands and/or program file names are specified, then the "/" command is not allowed at all. This is the method for barring the use of the "/" command. For example, in the file ABC:

```
/INC REMIND
MYNAME='ABC',FILTER=T
```

ABC will not allow the use of the "/" command. The stack file will be @REMIND.STAK.tcb, where *tcb* is the TCB number.

FS=t or f

When FS=t and REMIND is invoked on a 3270-type terminal, automatic full screen support is provided. On a non full screen terminal such support is never allowed.

When FS=f, no matter what terminal type you are using, only non full screen support will be available. The default is FS=t.

KEYFIL='filename'

Where *filename* is the name of a file that contains the site definitions for the PF keys used in the program. Lines in this file have the following format: "PFnn x" where *nn* is a number from 1 to 24, and *x*, the definition, can be from 1 to 50 characters.

The user can store their own PF key definitions for later use by entering a definition for any PF key. These definitions are stored in the file *USR:REMIND.KEYS at the program termination for use when the program is invoked again. If this file is available when the program begins, then these definitions are used and the site definitions are not. Lines in the file *USR:REMIND.KEYS have the same format as the lines in KEYFIL filename.

LOG=t or f

The LOG=t parameter can be used if you wish to keep a usage log. LOG=f is the default.

LOGFIL='filename'

Where *filename* is the name of a file that is used to log information from the program. The default file name is *USR:@REMINDLOG.tcb, where *tcb* is the current TCB number.

MDELAY=n — Where *n* is a value indicating the delay between attempts at finding out if mail is waiting. When mail is detected, a mail waiting message is posted. When *n* is a positive integer it is taken as representing minutes. When *n* is negative it is taken as seconds. To stop all mail checking, use a value for *n* that is greater than 24 hours such as -86400 secs or 1440 mins. This will prevent the posting of even the initial mail waiting message at the start of the program. The default is MDELAY=-300 or MDELAY=5 (five minutes).

MYNAME='filename' — Where *filename* is the name of the file. In our example it would be ABC. If this parameter does not specify the file name in which it is stored REMIND will not make the file an *always* program.

STACK='filename' — Where *filename* is a 1-46 character name of a file to be used as the stack file. The tcb number is appended to it as follows: *filename.tcb*. The stack file is used to store a trace of menus and programs that have been called. In this way the program can return to the correct menu as it drops back through the stack. When no stack file is specified the default "@REMIND.STAK.tcb" is used, where *tcb* is the TCB number.

Note: Use JOBONE or utility program FILE.DELETE to delete stack files that have accumulated from abnormal termination of the program.

Schedule and Meet Facilities

The SCHEDULE program allows you to organize your agenda on a daily or monthly basis. It can be used to schedule conference rooms or equipment items also.

The MEET program allows you to schedule a meeting for a group of attendees, conference rooms, and equipment items.

The CONFLIST program allows you to create a conference room or equipment item and gives authorization to a number of users. (See the *MUSIC/SP Office Applications Guide* for a user description of this program.)

All three programs share the same schedules (files) when they are used.

The SCHEDULE and MEET programs are configured with the MAIL.CONFIG program. See the MAIL.CONFIG program description in this manual for further details.

Program Parameters and Authorization Tables

SCHEDULE Parameters

The executor file for SCHEDULE, $TDO:SCHED, contains some program parameters (in MUSIC namelist format, separated by commas) which you may need to change.

CONFCD='userid' — A file holds the conference rooms and equipment items information as given by the CONFLIST program. CONFCD tells the program the userid where this file is stored. The default for CONFCD is $TDO.

ALIAS='name'
MUSNOD='name' — These are the names of the local MUSIC system; example MUSNOD='MUSIC'. See MAIL parameters under the topic "Configuring MAIL" in *Chapter 9* -

Electronic Mail Facility for a detailed description of this parameter.

MEET Parameters

The executor file for MEET, $TDO:MEET, contains some program parameters (in MUSIC namelist format, separated by commas) which you may need to change.

The program parameters are as follows:

ACCESS='filename' Where *filename* is the name of the file containing a list of userids and access levels. Userids in the file can include wild characters. The access level can be any number from 0 to 9999999 but access is granted only for the value 1. If a level is not included with a userid entry then no access is assumed.

When a user requests this facility, his userid is matched against the ACCESS file (if it exists). The first userid in the file that matches his userid determines whether he will have access or not.

Note: If ACCESS='filename' is not specified, then all userids have access to this facility.

Example:

```
* This is a sample access file
ccfp 0       Keep this guy off this facility
????xxx 0    His friends cannot use it either
??????? 1    Everyone else can
```

In this example the userid *ccfp* (ccfp 0) and all userids of the length 7 and ending with *xxx* (????xxx 0) are not allowed access. All other userids have access (??????? 1). Notice that comments can be included by using an asterisk (*) in column one or they can be appended after the userid and access level.

CONFCD='cod1','cod2','cod3','cod4','cod5'
Where *cod1, cod2, cod3, cod4, cod5* are valid MUSIC userids. These userids are used to access their respective conference rooms/equipment items files, better known as @CONFEQUIP. This can be used to have different department's conference rooms/equipment items accessible for meeting scheduling from one program. The default CONFCD userid is $TDO.

DEFTM1=n This parameter specifies the default BEGIN TIME for the program MEET. DEFTM1=900 (9:00 am) is currently specified for this program. This default BEGIN TIME is used when the user intends to schedule a meeting for a day other than the current day. If the current time is prior to 700 for the current day, then DEFTM1=900 is used. If it is later than 700 for the current day, then the MEET program inserts a BEGIN TIME equivalent to the next hour plus one hour. The value of *n* must be greater than or equal to 1 and less than or equal to 2400.

Example 1

If the current time is 1015 when a user is scheduling a meeting, then the BEGIN TIME will be 1200 on the screen.

Example 2

If the current time is 610 (6:10 am) when a user is scheduling a meeting, then the BEGIN TIME will appear as 900 (DEFTM1=900).

DEFTM2=n

This parameter specifies the default END TIME. This time appears on the screen unless the user specifies an END TIME. DEFTM2=1700 is currently specified for this program. The default END TIME is also used for multiple day meetings as the END TIME for all days of the meeting except the last day. The value of *n* must be greater than or equal to 1 and less than or equal to 2400. The value of DEFTM1 must not be greater than the value of DEFTM2.

Example 1

If the current time is 1401, then the default END TIME will appear as 1700 on the screen. (The BEGIN TIME will be 1600.)

Example 2

If the current time is 1800 on April 2nd, then the date changes to the next day and the values for both DEFTM1 and DEFTM2 appear on the screen:

```
Meeting Information
  Begin date    ===> 03APR86 (ddmmmyy)
  End date      ===> 03APR86 (ddmmmyy)
  Begin time    ===> 900____ (hhmm)
  End time      ===> 1700___ (hhmm)
  Time required ===> __ (hours) __ (minutes)
```

ERRFIL='filename'

Where *filename* is the name of a file that contains the messages for the program. The default file name is $TDO:MEET.MSGS. The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $TDO:MEET.MSGS.S by the MAIL message builder program, $EML:MSG.BLDR. This program is described in the Mail chapter in this manual. The $TDO:MEET.MSGS.S files are normally off-line and may have to be restored from the source tapes if use is required.

KEYFIL='filename'

Where *filename* is the name of a file that contains the site definitions for the PF keys used in the program. Lines in this file have the following format: "PFnn x" where *nn* is a number from 1 to 24, and *x*, the definition, can be from 1 to 50 characters.

Users can store their own PF key definitions for later use by entering a definition for any PF key. These definitions are stored in the file *USR:MEET.KEYS at the program termination for use when the program is invoked again. If this file is available when the program begins, then these definitions are used and the site definitions are not. Lines in the file *USR:MEET.KEYS have the same format as the lines in KEYFIL filename.

LOG=t or f

The LOG=t parameter can be used if you wish to keep a usage log for the program MEET. LOG=f is the default.

LOGFIL='filename'

Where *filename* is the name of a file that is used to log information from the program. The default file name is *USR:@MEETLOG.tcb, where *tcb* is the current TCB number.

ALIAS='name'
MUSNOD='name'

These are the names of the local MUSIC system; example MUSNOD='MUSIC'. See MAIL parameters under the topic "Configuring MAIL" in *Chapter 9 -*

Electronic Mail Facility for a detailed description of this parameter.

STACK='stackname' A 1-46 character name of a file to be used as the stack file. The TCB number is appended to it as follows: *stackname.tcb*. MEET establishes an always program environment. The stack file is used to restore the environment when the *always* program MEET is re-entered. The default stack name "@MEET.STACK.tcb" is used when no stack name is specified.

Note: Use JOBONE or utility program FILE.DELETE to delete stack files that have accumulated from abnormal termination of the program.

CONFLIST Parameters

The executor file for CONFLIST, $TDO:CONFLIST, contains some program parameters (in MUSIC name-list format, separated by commas) which you may need to change.

ACCESS='filename' Where *filename* is the name of the file containing a list of userids and access levels. Userids in the file can include wild characters. The access level can be any number from 0 to 9999999 but access is granted only for the value 1. If a level is not included with a userid entry then no access is assumed.

When a user requests a menu facility, his userid is matched against the ACCESS file (if it exists). The first userid in the file that matches his userid determines whether he will have access or not.

Note: If the ACCESS parameter is not specified, then all userids have access to that menu facility.

Example:

```
* This is a sample access file
ccfp 0       Keep this guy off this facility
????xxx 0    His friends cannot use it either
??????? 1    Everyone else can
```

In this example the userid *ccfp* (ccfp 0) and all userids of the length 7 and ending with *xxx* (????xxx 0) are not allowed access. All other userids have access (??????? 1). Notice that comments can be included by using an asterisk (*) in column one or they can be appended after the userid and access level.

ACODE='cod1' Where *cod1* is the MUSIC userid used to find the conference rooms and equipment items file, better known as @CONFEQUIP.

ERRFIL='filename' Where *filename* is the name of a file that contains the messages for the program. The default file name is $TDO:CONFEQUIP.MSGS The file has a VC record format and a record length of 1536. It is not a text-only file, and it should not be edited. This file is produced from $TDO:CONFEQUIP.MSGS.S by the MAIL message builder program, $EML:MSG.BLDR. This program is described in the Mail chapter in this manual. The $TDO:CONFEQUIP.MSGS.S files are normally off-line and may have to be restored from the source tapes if use is required.

Special File Names

The following special files are created by SCHEDULE, MEET, and CONFLIST at various times.

@MMMYY.SCHED. actual user's calendar of events for month MMM given the last two digits of the year YY. A separate file exists for each month. The last 12 months are kept on the system.

@MMMYY.NAME monthly schedules for conference room or equipment item NAME, MMM and YY being the month and last two digits of the year. A separate file exists for each month. The last 12 months are kept on the system.

@AUTHSCHED list of users authorized to look at, add, or update the schedule.

$TDO:@SCHEDMEET.TABLE
file used to control the access to schedules when using the MEET program. Enqueuing and dequeuing are done with this file.

@CONFEQUIP list of conference rooms and equipment items. 125 rooms and items can be accommodated. This file contains the conference room or equipment item name, an administrator userid, a description of the room or item, a list of users authorized to access this item.

@TMENULOG.tcb log file used in TMENU

@REMINDLOG.tcb logfile used in REMIND

@MEETLOG.tcb log file used in MEET

@TMENU.STACK.tcb
stackfile used in TMENU

@REMIND.STAK.tcb
stackfile used in REMIND

@MEET.STACK.tcb stackfile used in MEET

Installing a System Word Dictionary in the PLPA

A system word dictionary can either reside in the PLPA (pageable link pack area) or in the link pack area itself. This topic describes how to prepare a dictionary and install it in the PLPA.

When a dictionary is in the PLPA it is available to any user who has access to the system. However it does not occupy the user region. In this way many users will share the dictionary while each will run in a substantially smaller region than otherwise would be required.

MUSIC is distributed with a system word dictionary in the PLPA called DICT1. References will be made to this dictionary in order to illustrate how the installation is done. You should note that this procedure need only be followed if you want to add a new system dictionary or modify DICT1.

This procedure must be performed by system support personnel.

Initial Preparations

The SPELL algorithm requires that all words in the system dictionary be in lower case. Only letters of the alphabet (a-z) are permitted. Any special characters are translated out. Therefore when preparing a new list of words or modifying an old one, please remember to remove special characters. For example if you want to have *didn't* in your system dictionary you need to have the two entries "didn" and the letter "t". Spell ignores the "'" (quote) and sees two words "didn" and "t". In this way it will not flag *didn't* as an error. In addition numeric text is also ignored. For example if "pf" were an entry in the system dictionary then "pf1", "PF12", or "pf" would all be correct spellings.

You should also make sure that each letter of the alphabet is present in the dictionary, since the occurrence of any single letter is a correct spelling. This also ensures that abbreviations of the type A.B.C. are spelled correctly since the periods are translated out.

Several steps have to be performed to install a system dictionary in the PLPA. These are:

1. compute the PLPA storage required
2. process and compress the dictionary
3. compile the dictionary
4. link edit the dictionary
5. place the dictionary's name in the system catalog
6. install the dictionary in the PLPA

Descriptions of these procedures follow below.

Note: The raw word files for the system dictionary DICT1 are

$TDO:WORDS.A,
$TDO:WORDS.B,
$TDO:WORDS.C,
... to
$TDO:WORDS.Z,
and
$TDO:WORDS.APROPER,
$TDO:WORDS.BPROPER,
$TDO:WORDS.CPROPER,
... to
$TDO:WORDS.ZPROPER.

1. Compute the PLPA storage required

After you have prepared the list(s) of words for your dictionary, it is necessary to compute the number of total characters or bytes it will occupy in the PLPA. This is done by processing the list(s) through the BYTES program. This program will report the total number of lines it read (words in the dictionary), the average length of each word, and the total bytes the words would occupy if totally compressed and contiguous.

It is important that you pay attention to this step and the results it produces. You may have to modify source programs in later steps to accommodate very large dictionaries. The programs to be used later, PROCESS and COMPDICT, are setup to handle a dictionary of 830,000 bytes.

The BYTES program can be invoked as follows:

```
/INC BYTES
/INC words
```

where *words* is a file name containing the dictionary. Any number of files can be included here. A file called $TDO:BYTES.RUN1 has the run used for DICT1.

2. Process and Compress the Dictionary

After you have verified that the total number of characters in the dictionary is less than or equal to 830,000 you can proceed to running the PROCESS program. If however you have more than 830,000 bytes, you must modify $TDO:PROCESS.S and $TDO:COMPDICT.S.

In $TDO:PROCESS.S change the statements:

```
REAL*8 DICT8(102500)
LOGICAL*1 DICT1(820000)
```

to fit your needs. DICT1 must be a multiple of 8 and DICT8 must be dimensioned as the dimension of DICT1 divided by 8. Failure to ensure this will cause compiler errors.

In COMPDICT change the statements:

```
LOGICAL*1 ALL(965000)
LOGICAL*1 DICT(960000)
```

to fit your needs. The 5000 discrepancy between ALL and DICT must be maintained in any change you make.

In addition you should make sure that the /SYS REG=1000 in $TDO:PROCESS and $TDO:COMPDICT is incremented accordingly.

If you made changes to PROCESS.S or COMPDICT.S you must execute and relink them. Files $TDO:PROCESS.LKED and $TDO:COMPDICT.LKED should be used to generate a new load module. You can now continue with the processing of the dictionary.

PROCESS performs several functions and provides information required by SPELL. The first function is to translate out any non-alphabetic character in a word, the word is right justified and converted to lower case. Then each word is analyzed to determine the frequencies of character pairs as they occur and stored in $TDO:PAIRS.DICT. This information is used later by SPELL to aid in finding alternate spellings when words are not found in the dictionary. The total list is then sorted. The sorted list is converted to a compressed and contiguous list of characters and stored in $TDO:COMPRESS.DICT. In $TDO:MARKERS.DICT is stored the byte addresses for the dictionary, later to be used for accessing the list. The file $TDO:PROCESS.RUN1 has the run for DICT1.

PROCESS requires a region of 1000K bytes, and because of the extensive processing of each word, it will require a great deal of processing unit time. This process time is in the order of 1200 su for 830,000 bytes.

The following is a list of $TDO:PROCESS.

```
/SYS REG=1000,TIME=50
/FILE LMOD N($TDO:PROCESS.LMOD) SHR
/FILE 1 UDS(----TEMP) NREC(125000) LR(32) VOL(MUSIC3) NEW DELETE
/FILE 2 RDR
/FILE 3 UDS(----TMP3) NREC(80000)  VOL(MUSIC3) NEW DELETE
/FILE 8 N($TDO:MARKERS.DICT) NEW(REPL) DEF
/FILE 10 N($TDO:COMPRESS.DICT) NEW(REPL) SP(300) RLSE DEF
/FILE 11 N($TDO:PAIRS.DICT) NEW(REPL) SP(100) DEF
/LOAD EXEC
```

```
PROCESS
```

3. Compile the Dictionary

After the dictionary has been successfully processed by the PROCESS program, it can be compiled into an object deck. COMPDICT will take the $TDO:PAIRS.DICT, $TDO:MARKERS.DICT, and $TDO:COMPRESS.DICT and generate an object deck. The file $TDO:COMPDICT.RUN1 was used to generate DICT1.

The following is a listing of $TDO:COMPDICT.

```
/SYS REG=1000,TIME=MAX
/FILE LMOD N($TDO:COMPDICT.LMOD) SHR
/FILE 1 N($TDO:COMPRESS.DICT) OLD DEF
/FILE 2 N($TDO:MARKERS.DICT) OLD DEF
/FILE 3 N($TDO:PAIRS.DICT) OLD DEF
/FILE 7 N($TDO:DICT.OBJ) NEW(REPL) SP(500) DEF
/LOAD EXEC
COMPDICT
```

4. Link Edit the Dictionary

In order for the dictionary to be placed into the PLPA it must be link edited into a load module. The file $TDO:DICT1.LKED was used to generate DICT1.

The following is a listing of $TDO:DICT1.LKED

```
/SYS TIME=MAX,REG=200
/FILE LKEDWORK NR(16000)
/FILE LMOD NAME($TDO:DICT1.LMOD) NEW(REPL) SP(1000)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
/INC $TDO:DICT1.OBJ
 NAME DICT1
```

This step produces the file $TDO:DICT1.LMOD. This is the file that is to be loaded into the PLPA. Note that the object deck it requires is $TDO:DICT1.OBJ that was produced by $TDO:COMPDICT.RUN1.

5. Place the Dictionary's Name in the System Catalog

To install a dictionary in the PLPA, edit the system catalog and add a line like the following:

```
----+----1----+----2----+----3
RESPGM64 DICT1    PLPA 300000
```

Make sure that the number for the RESPGM keyword is unique. If RESPGM64 is already being used by some other module, change the 64 to a number not currently being used.

Once you have modified the catalog, you must run the EDTCAT utility as follows:

```
/INC *COM:EDTCAT
/INC xxxxxx
```

Where **xxxxxx** is the name of your system catalog. (Most installations use the name "$GEN:SYSCAT" for the system catalog. You need the VIP privilege on your userid to execute this utility. When you next IPL MUSIC, the dictionary will be accessible from the PLPA.

6. Install the Dictionary in the PLPA

The $TDO:DICT1.LMOD will be placed into the PLPA using the file $TDO:DICT1.PUTSYS. The following is a listing of $TDO:DICT1.PUTSYS

```
/FILE 1 N($TDO:DICT1.LMOD)
/INC LDLIBE
LIBE='SYST',IN=1
NAME='DICT1',RENT=T,REPL=T
```

As a final step the name DICT1 has to be entered in the system catalog and becomes available at the next IPL. If it is not listed here it will remain inaccessible even though it has been properly installed.

Accessing the Dictionary

In order to access a specific system dictionary you must set up a file as follows:

```
/INC SPELL.PROG
name
```

Where *name* is a 1-8 character name corresponding to a system dictionary in the PLPA. If name is not specified it defaults to DICT1. If the file from above were called SPELL2 then the dictionary specified by *name* is accessed by typing SPELL2 when in *Go mode. See the contents of files $TDO:SPELL.PROG and $TDO:SPELL as examples.

Chapter 13. IIAS/IIPS

IIPS/IIAS Codes

IIPS/IIAS uses special codes to identify authors, students and administrators. These codes have no connection with the MUSIC sign-on userids. The student codes start with the letter S, the author codes with the letter A and the administrator has the code of XUPER.

When the author signs on to a course, the course can be altered or author-as-student mode can be entered to check out the course. The administrator code is used to allocate course areas, register students and perform dump and restore functions over any of the course files.

Course File Organization

A single course disk file can contain many individual courses. Each course is allocated a specified number of blocks within this disk file. Within each course area is space to hold information about each student that is registered for that course. The IIPS student code is used to point to the user's specific status area within the course space. These status records allow continuation of the course through many terminal sessions.

The course author must have write access to the course disk file. The student needs write access to the student's own student status area. In most IIPS implementations, this means that the student also needs write access to the course disk. Control of the student write access to file can be performed in one of two ways on MUSIC: *Multiple Write Access* and *No Write Access*.

Multiple Write Access

This method allows the student to write directly to the student's own status area within the course disk file. Each student must be individually authorized so that a specific area can be reserved for the student within the course. The MUSIC disk file that contains the course must be defined so that the student users can have write access to it. The OPTIONS file (described later) should be used to prevent two users from attempting to use the same IIPS sign-on code simultaneously or from attempting to sign on as the author or administrator.

No Write Access

The MUSIC implementation of IIPS allows the student status area to be located on a separate file. This allows the course file to be defined with only read access from students. This provides additional protection against accidental or deliberate alteration of the course material by unauthorized users. Furthermore, it allows any user to take the course without being individually preregistered. In this case, all students must sign on to the course with the special code of STUDENT.

File Usage

A number of disk files can be defined for use with IIPS/IIAS. Most are created and initialized by the utility program IIS.INIT. A file with the necessary control statements is set up to point to these files. This file should be saved with the execute-only option. After this is done, simply type the name of the file to invoke IIPS/IIAS with your files.

It is wise to set up a file for student users and a separate one for course authors and administrators. This separation can provide for better control of write access to the files.

The file will be of the form:

```
/SYS NOPRINT,TIME=MAX
/FILE ..... (as required to point to the IIS files)
      ..... additional /FILE statements as required
/INCLUDE IIS
```

The /FILE statements are described in the following topics. Reference to CODE should be taken as referring to the first 4 characters of the MUSIC sign-on code of the file's owner.

Special note on UDS files

When UDS files or tapes are used with IIPS/IIAS the parameter BUFNO(2) must be specified on the /FILE statement. Failure to do so will result in errors during course execution.

Course Files

Course files are defined via /FILE statements that have ddnames of DISK20 through DISK59. (DISK60 through DISK69 are reserved for possible system use. Currently DISK69 is used for the IBM supplied course and macro library and DISK68 is used for the MUSIC LEARN courses.) First estimate the number of blocks needed by consulting the writeup under the REGISTER COURSE command in the *IBM IIPS/IIAS Administrator's Guide* (SH20-2449).

IIPS/IIAS should always access a course file through the same ddname. In other words do not define a course file as DISK30 one time and DISK31 a later time.

Course files up to 3900 blocks can be stored in the Save Library. To create and initialize one simply type IIS.INIT when your terminal is in *Go mode and respond to the questions it will ask.

Large course files can be set up in UDS files. The maximum number of blocks per disk file is limited by IIPS/IIAS to be 32700 blocks. To allocate and initialize the UDS file, execute a file containing the following:

```
/FILE 1 UDS(dsname) LRECL(512) NREC(n) VOL(MUSICx) NEW
/INCLUDE IIS.INIT
```

In the above, *n* is the number of course records. The 5th character of the dsname field must be a "#" character if the file is to be writable from any MUSIC user code.

Sample /FILE statements for course files on files are:

```
/FILE DISK30 NAME(MYCOURSE) SHR   (no write access)
/FILE DISK30 NAME(MYCOURSE) WSHR  (multiple write access)
```

Sample /FILE statements for course files on UDS files are:

```
/FILE DISK30 UDS(CODEII30) SHR VOL(MUSIC1) BUFNO(2) (no write access)
/FILE DISK30 UDS(CODE#I30) WSHR VOL(MUSIC1) BUFNO(2) (write access)
```

Student Recording Files

Records can be kept of student responses. (The utility programs used to analyse this data are described later.) This optional file is initialized via the IIS.INIT program.

A sample /FILE statement for a message file CODE:MYSREC.IISR follows. Note that the letters IISR are to be replaced by * on the /FILE statement. The /FILE statement for TEMPSREC must also be given as shown though it never needs initialization.

```
/FILE MAINSREC PDS(CODE:MYSREC.*)
/FILE TEMPSREC NAME(&&TEMP) SPACE(2) MAXSP(2) RECFM(VC) NEW
```

Student File

This file is used when the student status information is not to be written to the course file. (Do not confuse this file with the student recording file previously described.) A student code of STUDENT must be registered for the course and the student must sign on with STUDENT before this file will be used. The name of the file can be given as &&TEMP if the student is to start at the beginning of the course each time it is used.

Use the name @WARM (for example), to keep the student's status for the student's next use of the course. Thus, the point where the student left off in the course can be continued. If the student tries a different course next time, the warm start information for the first course is automatically erased. Each student should have a different MUSIC 4-character code so that each can get a unique warm start area.

The warm start information is dependent on the disk location of the student records. Thus if the course is moved from one disk to another, the warm start information will not be used causing the student to start at the beginning of the course.

A sample /FILE statement for a student file called @WARM is:

```
/FILE STUDSAVE NAME(*USR:@WARM) RECFM(U) SPACE(4) OLD(CREATE)
```

Options File

This optional file contains special run options. Create the options file by using the Editor and save it public. Each option is entered on a different line in the file.

NXTPGM=code:name This option specifies that file code:xxxx will be run after IIPS/IIAS stops for any reason. The program will be invoked with the non-cancelable option. The given name may even be the file that invokes IIPS/IIAS. If this file is also the name of the MUSIC code's AUTOPROG then the user will always be in the IIPS/IIAS environment when the user uses this code. The user may type /OFF at IIP/IIAS sign-on time to disconnect from MUSIC.

ENQPRE=xx — This option will perform a system enqueue operation using the IIPS/IIAS sign-on code prefixed with the two letters xx. If this name is already used, the sign-on will be rejected with the message `STD# IN USE..TRY AGAIN LATER.` This ENQPRE option has no effect when the code STUDENT is used and a /FILE statement for STUDSAVE is present.

A=code — This option checks to ensure that authors are signed on with the specified MUSIC code. Its use is mainly to protect files that are public writable. Several A=code statements can be used if more than one MUSIC code is to be allowed. If no A=code statements are used no MUSIC code checking is done. The length of the code given on these statements is the actual length compared. The message `UNAUTHORIZED TO USE THIS IIS CODE` will be issued if the code check fails.

X=code — Same as A=code except it gives the valid MUSIC codes for IIPS/IIAS administrators. The IIPS/IIAS code of XUPER is the only accepted administrator code.

A sample /FILE statement for an options file CODE:MYOPTS is:

```
/FILE OPTIONS NAME(CODE:MYOPTS) SHR
```

Location Table

This is used to store communications between the various users of the instructional system. These communications include course comments from students as well as messages from authors or administrators using the LOG command. This file also contains the location table used to verify student location during registration. By default, MUSIC provides a dummy file, resulting in a warning message during the *register student* process, and the loss of messages and comments. The IIS.INIT program can be used to create a location table file if one is required.

Sample /FILE statement for the location table file MYLOC is:

```
/FILE LOCTAB NAME(MYLOC) WSHR
```

Preparing the Course Material

Refer to the IBM publication *IIAS Course Authoring Guide* (SH20-2450) for the details of how to prepare and modify courses.

Course materials may be prepared outside of the IIPS/IIAS environment. This is done by using TEDIT to invoke the MUSIC's Editor with the option to preserve lower case letters. Do not enter TAB characters in course materials. Use files with names of 8 characters or less. Input these files into IIPS/IIAS by using the GETX command.

Use the sequence of two at signs (@@) to signify the entry of a carriage return when entering Coursewriter text.

The use of the Course Structuring Facility (CSF) is recommended to greatly ease the production of courses. A good overview of CSF is presented in the IBM publication *IIPS/IIAS General Information Manual* (GH20-2446). The user should note that the powerful CSF facility does take a fair amount of computer time to generate complete courses. Minor modifications of generated material presents no hardship. Execution speed of these generated courses is not slower by any noticeable amount.

Author Commands

The following are some notes about some IIPS/IIAS author commands. IIPS/IIAS limits external file names to 8 characters or less. These commands can be entered only in response to the TYPE COMMAND message.

getx aaaaaaaa	This command is used to read the file *aaaaaaaa* into IIPS/IIAS just as if you have typed its contents live at this point. When the end of the file is read, you may continue entering commands. If the file is not accessible, no action is performed and no message is given.
get aaaaaaaa	This command is the same as getx with the exception that the terminal output will be sent to the file IISOUT instead of to the terminal.
put bbbbbbbb	This command directs that the terminal output will be sent to the file *bbbbbbbb* instead of to the terminal.
putx bbbbbbbb	This command directs that the terminal output associated with TYPE commands will be sent to the file *bbbbbbbb*. All other terminal output will be ignored. The problem and sequence numbers will be stripped from the output. This allows the author to create a file that can be used with the Editor to make material changes. Reinsertion can then be done by the get or getx commands.
poff	This command stops the output to disk operation started by a put or putx command.
incore store nostore	These commands are not used because MUSIC automatically buffers disk I/O.
comments log	These commands are not used in the MUSIC environments.
regs	As this file requires the HISTORY file, it will not work on MUSIC.

Administrator Commands

The following are some notes about some administrator commands. The IBM publication *IIPS/IIAS Administrator's Guide* (SH20-2449) documents these commands. No secondary authority is required for any command. The get, getx, put and putx work in administrator mode as they do in author mode.

Some commands refer to work files. These normally refer to files of RECFM V or VC. Thus work11 is the file WORK11. If the file does not exist, one will be created. The file may refer to a tape file during a course on or off process. In this case, run the job from batch and use RECFM=U on the TAPE /FILE statement.

A number of commands have meaning only when HISTORY files are used or when IIPS/IIAS is directly managing the terminal network. The meaningful commands in the MUSIC environment are:

COPY
COURSE OFF
COURSE ON
COURSES

DELETE COURSE
LIST
LOC (useful only when a LOCTAB file used)
MODIFY STUDENT
PRINT
REGISTER COURSE
REGISTER STUDENT
REMOVE COURSE
REMOVE STUDENT
REORG
SEE DISK
SEE STUDENT
STUDENT STATUS
TRACE
TYPE
UPD
WRITE DISK
WRITE STUDENT

Notes

IIPS/IIAS automatically puts a read up on the terminal if an attempt is made to print more than about 100 lines without an intervening read. The entry of a blank line at this time will cause the output to continue. The entry of QUIT in author mode will stop the command. The entry of SIGN OFF will also be accepted at this time.

IIPS/IIAS accepts a line delete signal of 3 number signs (###) when entered at the end of a line.

Utilities

A number of utility programs are provided with IIPS/IIAS. Some examples of their output are given in the two IBM publications: *IIPS/IIAS Operations Guide* (SH20-2459) and *IIPS/IIAS Course Authoring Guide* (SH20-2450).

Analyzing the Student Records

The student record analysis utilities produce reports on student responses and sign-on/sign-off activity. The analysis is performed by running a set of jobs. Normally these jobs are run at batch due to the amount of output generated.

Input to these jobs is the student recording file. The recording option specified on the REGISTER STUDENT administrator command affects what records will be written to this file.

If the special code of STUDENT was used, the student code reported in the reports will be of the form "S*code" where *code* is the user's MUSIC sign-on code.

```
Job 1:      /FILE FILE10 NAME(CODE:MYSREC.IISR) SHR
            /INCLUDE IIS.SRAJ1
```

Job 2: `/INCLUDE IIS.SRAJ2`

Job 3:
```
/INCLUDE IIS.SRAJ3
...control statement
```

The format of the control statement is:

Cols 1-8	defines course selection information.
Col 1	controls the translation to lower case. If 1, then a translation will be made to lower case.
Cols 2-7	contains the course name. If blank all courses are selected.
Col 8	used when a course name is specified, a 1 causes all courses whose names precede the selected course in sorted sequence to be selected.
Cols 9-19	defines student selection information.
Cols 9-18	contain the student number. If blank all students will be selected. The student number must be prefixed with the letter s.
Col 19	used when a student number is specified, a 1 causes all students whose student number precede the selected student number in sorted sequence to be selected. A blank will cause only the specified student to be selected.
Cols 20-25	selects the type of response to report.
Col 20	set to 1 to select UN response records
Col 21	set to 1 to select HELP response records
Col 22	set to 1 to select GO TO response records
Col 23	set to 1 to select CA and CB response records
Col 24	set to 1 to select WA and WB response records
Col 25	set to 1 to select AA and AB response records
Cols 27-30	defines which course parameters to include in the report
Col 27	set to 1 to cause parameters and switches to print
Col 28	set to 1 to cause counters 1-10 to print
Col 29	set to 1 to cause counters 11-20 to print
Col 30	set to 1 to cause counters 21-30 to print

Job 4: `/INCLUDE IIS.SRAJ4`

Job 5: `/INCLUDE IIS.SRAJ5`

Job 6:
```
/INCLUDE IIS.SRAJ6
...location statement(s)
...date statement
```

Up to 24 location cards may be given. The format of each is:

Cols 1-2	area number
Col 3-19	location name (any printable characters will do)

The format of the date card is:

Col 1	must be a * to identify it as a date card
Col 2-26	date heading (e.g., JUNE 23, 1978)

Getting Course Batch Listings

To obtain a course listing at batch first use the administrator PRINT command documented in the *IIPS/IIAS Administrator's Guide* (SH20-2449).

For example, the administrator can issue the following commands:

```
print 11    (to output the file to the file WORK11)
cname
sign off
```

Once this has been done run the following jobs from batch:

```
Job 1:      /FILE TINN  NAME(WORK11) SHR
            /INCLUDE IIS.CBLJ1

Job 2:      /INCLUDE IIS.CBLJ2
```

Converting ITS and Coursewriter Course Material

The course conversion program converts ITS and Coursewriter course materials into a format acceptable to IIPS/IIAS. The utility accepts one course file per tape. If the tape is blocked, it should be first copied to a file.

```
/PARM cname 00
/FILE PRINTIN TAPE ....
/INCLUDE IIS.CCONV
```

The /PARM statement contains the 6 character course name in lower case letters followed by the segment number (normally 00). If the course name is less than 6 characters, put in blanks after such that the segment number always starts 6 characters after the course name.

After the course has been converted, register the course, sign on as the course author and fetch the file by using the get or getx author commands. Specify a file name of *convcrse*. (Purge this file after you have finished with it.)

Installing New Courses From Tape

The main problem in installing new courses is that they can be stored on tape in such a variety of ways. Also, some courses require that new subroutines or screen formats be installed along with the course material. These may be in source or object deck form. The key to the problem is selecting a method for getting the material off the distribution tape. The following MUSIC utilities can be useful to this end.

CMSTAPE	- cms tape dump format
GENSAV	- iebupdte format
LOADPDS	- iebcopy format
MOVEPDS	- iehmove format
TAPUTIL	- straight block by block copy
DBLOCK	- deblock tape file to 512 byte record on disk
UTIL	- record by record copy

The three topics which follow illustrate the steps involved in installing an actual course, installing some new functions, and installing new screen formats.

Installing New Course Material

Courses obtained externally (from IBM or other users) generally come on magnetic tape in one of three forms:

Course off
Print
Insert

The course tape package should include the following information required for installation:

Course name
Number of 512-byte course material blocks required
Number of auxiliary blocks required
Number of blocks required for each additional student

The procedure for *Course off* tape format will be shown.

Copying the Material to Disk

1. Run the following BATCH job to deblock/copy the course material from the tape to a disk file.

```
/FILE 1 TAPE VOL(COURSE) RECFM(U) SHR
/FILE 2 UDS(CODEICSR) NEW LRECL(512) VOL(MUSIC1) NREC(n)
/INC DBLOCK
SELECT=x
```

   where *n* is the number of 512-byte records to be moved from the tape to disk. *x* is the file number (on the tape) where the course is located.

Allocating the Course File

If the number of course blocks, (student and auxiliary if using *Multiple Write Access*) is less than 3900, then a file can be utilized. Otherwise, a UDS file will be needed. UDS files can handle the largest course file usable with IIPS. MUSIC requires that course files be pre-allocated and initialized. The following examples, run from a MUSIC terminal, allocate and format a course file and a UDS course file. Note that when UDS files are used the parameter BUFNO(2) is always specified on the /FILE statements.

Course File

```
---> /EXEC IIS.INIT
     PROGRAM TO INITIALIZE IIS COURSE AND UTILITY FILES
     ENTER  S  IF STUDENT RECORDING FILE
            M  IF MESSAGE FILE
            C  IF COURSE FILE
            L  IF LOCATION TABLE FILE
---> C
     ENTER NUMBER OF COURSE BLOCKS NEEDED
---> 2500
     ENTER FILE NAME
---> CODE:TEST.CBT
     ENTER THE NUMBER 1 IF THE FILE IS TO BE PUBLIC WRITABLE
     ENTER THE NUMBER 0 IF NOT
**** enter in 1 if you will be using "Multiple Write Access" ****
**** otherwise enter 0 ****
---> 0
     IIS COURSE FILE CREATED
```

UDS Course File

Execute a file containing the following:

```
/FILE 1 UDS(dsname) LRECL(512) NREC(n) VOL(MUSIC1) NEW
/INC IIS.INIT
```

In the above, *n* is the number of course blocks to be allocated. The *dsname* will be the file name. The 5th character MUST be a "#" character if the file is to be writable from any user code (Multiple Write Access).

```
      PROGRAM TO INITIALIZE IIS COURSE AND UTILITY FILES
      ENTER  S  IF STUDENT RECORDING FILE
             M  IF MESSAGE FILE
             C  IF COURSE FILE
             L  IF LOCATION TABLE FILE
----> C
      ENTER THE NUMBER 1 TO USE THE /FILE 1 FOR COURSE FILE
----> 1
      FILE CONTAINS xxxxx COURSE BLOCKS
      IIS COURSE FILE CREATED
```

Course files are defined to IIPS using the MUSIC /FILE statement. The DDNAMES of DISK20 through DISK59 MUST be used for IIPS courseware. For example, the /FILE statements for the above courses (using DISK30 as the DDNAME) would be:

```
/FILE DISK30 NAME(CODE:TEST.CBT) SHR   (no write access)
/FILE DISK30 NAME(CODE:TEST.CBT) WSHR  (multiple write access)
```

For the UDS course file, assume a *dsname* of I30.

```
/FILE DISK30 UDS(codeII30) SHR VOL(MUSIC1) BUFNO(2) (no write access)
/FILE DISK30 UDS(code#I30) WSHR VOL(MUSIC1) BUFNO(2) (write access)
```

In the above, *code* refers to the MUSIC four character code under which the course file was created and saved.

Performing the COURSE ON Operation

1. Depending on how the course file was allocated above, add the following statements to the file $IIS:LEARN and save it as a new file called MYLEARN.

```
/FILE WORK11 UDS(CODEICSR) VOL(MUSIC1) BUFNO(2) SHR
/FILE DISK30 NAME(CODE:TEST.CBT) OLD
      or
/FILE DISK30 UDS(CODEII30) VOL(MUSIC1) BUFNO(2) OLD
      or
/FILE DISK30 UDS(CODE#I30) VOL(MUSIC1) BUFNO(2) OLD
```

2. Execute the file MYLEARN to and sign to the instructional system as the system administrator and perform the *COURSE ON* operation. Most courses will have instructions regarding the parameters to be specified on the COURSE ON commands. The following asks for 10 blank course material blocks and space for 3 students.

```
---> /EXEC MYLEARN
     ENTER STD#/CNAME OR LIST OR HELP
---> xuper
     TYPE COMMAND
---> course on 11/30
     ENTER AUTHORITY
---> iis
     TYPE CNAME/CSMATL/STUDRCDS/AUXBLKS/LAB-RATIO/HF/NEWNAME...
---> sample/10/3
     COURSE CHANGES
     CSMATL   STUDRECS     AUXBLKS    LAB-RATIO
       10         3
      TYPE OK IF CORRECT
---> ok
     TYPE CNAME/CSMATL/STUDRCDS/AUXBLKS/LAB-RATIO/HF/NEWNAME...
---> end
     END OF COURSE ON PROCESS
     TYPE COMMAND
---> register student
     TYPE CNAME/STD#/TR/P
---> sample/student
     TYPE STD NAME/LOC#/AREA#
---> student/000/00
     TYPE STD NAME/LOC#/AREA#
---> end
     TYPE COMMAND
---> sign off
```

   The last step registers the special code of STUDENT. This will allow users of IIPS under MUSIC to take the course "SAMPLE" without being individually registered. If additional students are to be registered for the course, then add them at this time.

   The dataset created in step 1 can now be deleted. The following job will remove the dataset from the system:

```
/FILE 1 UDS(CODEICSR) VOL(MUSIC1) DELETE
/LOAD IEFBR
```

   Make sure that the /FILE statement pointing to this file is also removed from your MYLEARN file. If

you will be running the course "SAMPLE" in *No Write Access*, change the /FILE statement for DISK30 to SHR from OLD.

Refer to the *IIPS/IIAS Administrator's Guide* for further details on the *COURSE ON* procedure.

Adding New Functions

User courseware is sometimes provided with a set of user subroutines that must be installed along with the actual course. The following example will show a method of adding those *User Functions* to an IIPS system running with MUSIC. The courseware used in this example is COMSKL (IUP 5796-BBT).

1. You must restore some IIPS/IIAS macros and CMS macros. The macros will be used when assembling the *User Functions* and IIPS modules.

   Restore IIPS/AS macros from the IIPS/AS tapes.

   ```
   /INC IIS.GETMAC
   ```

   Restore CMS macros from the MUSIC source tape.

   ```
   /FILE 1 TAPE VOL(SOURCE) LR(80) BLK(13440) SHR
   /INC MFREST
   NAME='$MCC:+',REPL=TRUE,TAPFIL=2
   ```

2. On the COMSKL distribution tape, the *User Functions* are located in file three. File three is in CMS tape format and can be retrieved using the following job.

   ```
   /FILE 1 TAPE VOL(COMSKL) RECFM(U) SHR
   /INC CMSTAPE
   TAPFIL=3
   ```

   The *user Functions* will be restored to the system with the names "code:xxxxx.ASSEMBLE" where *code* is the sign-on code used for the job and *xxxxx* is the name of the function.

3. Each of the *User Functions* will need to be assembled. Execute a file containing the following job for that purpose. Remember that each of the *User Functions* MUST be assembled. In the case of COMSKL, there are seven.

   ```
   /FILE SYSPUNCH NAME($IIS:xxxxx.OBJ) NEW(REPL)
   /INC $IIS:IIS.ASM
   /OPT DECK
   /INC code:xxxxx.ASSEMBLE
   ```

   Where "code:xxxxx.ASSEMBLE" is the name of the file containing the function to be assembled.

4. Restore from the IIPS installation tape the module FUNCTION.

   ```
   /FILE 1 TAPE VOL(IIPS) BLK(8000) LR(80) SHR
   /INC GENSAV
   FILE=10,PREFIX='$IIS:',SUFFIX='.S',SELECT=TRUE
   FUNCTION
   ```

5. The *User Functions* must be known by IIPS. The module FUNCTION is used for this purpose. Refer

to the *IIPS/IIAS Operations Guide* (SH20-2459) for details. You will need to edit the file $IIS:FUNCTION.S and add the required statements.

```
Edit  $IIS:FUNCTION.S
L *PLEASE INSERT

   FUNID  NUMBER=3B0,NAME=STDATA
   FUNID  NUMBER=3C4,NAME=FIND
   FUNID  NUMBER=3C5,NAME=CHANGE
   FUNID  NUMBER=3CA,NAME=CLEAR
   FUNID  NUMBER=3CB,NAME=PLOT
   FUNID  NUMBER=3D8,NAME=CLEARF
   FUNID  NUMBER=3F8,NAME=MODIFY
```

Save the modified file for the next step.

6. Now re-assemble the FUNCTION file by executing a file containg the following:

```
/FILE SYSPUNCH NAME($IIS:FUNCTION#A.OBJ) NEW(REPL)
/INC $IIS:IIS.ASM
/OPT DECK
=FUNCTION
```

The object deck will be saved under the file name pointed to by the SYSPUNCH statement.

7. A table in the module IISSUB must be modified to reference the new functions. This format of the table is described in the module. The following shows the editor being used to apply the changes for the COMSKL functions.

```
EDIT $IIS:IISSUB.S
L SUBTAB
I        DC    X'03B0',X'0000',V(DITCW3B0),V(DITCW3B0)
I        DC    X'03CA',X'0000',V(DITCW3CA),V(DITCW3CA)
I        DC    X'03CB',X'0000',V(DITCW3CB),V(DITCW3CB)
I        DC    X'03C4',X'0000',V(DITCW3C4),V(DITCW3C4)
I        DC    X'03C5',X'0000',V(DITCW3C5),V(DITCW3C5)
I        DC    X'03D8',X'0000',V(DITCW3D8),V(DITCW3D8)
I        DC    X'03F8',X'0000',V(DITCW3F8),V(DITCW3F8)
SAVE $IIS:IISSUB#A.S
```

8. Assemble the updated IISSUB module by executing a file containing the following:

```
/FILE SYSPUNCH NAME($IIS:IISSUB#A.OBJ) NEW(REPL)
/INC $IIS:IIS.ASM
/OPT DECK
=IISSUB#A
```

9. To incorporate all of the *User Functions*, updates to the FUNCTION module and IISMUS module, a linkedit is needed. To create the linkedit job do the following.

```
EDIT $IIS:GEN.LKED
L FUNCTION
C/FUNCTION/FUNCTION#A/
L IISSUB
C/IISSUB/IISSUB#A/

      *  at this point you would include the names of the
      *  "user functions" that were assembled in step 8.
      *  In the case of COMSKL the statements would be

I /INC $IIS:DITCW3B0.OBJ
I /INC $IIS:DITCW3CA.OBJ
I /INC $IIS:DITCW3CB.OBJ
I /INC $IIS:DITCW3C4.OBJ
I /INC $IIS:DITCW3C5.OBJ
I /INC $IIS:DITCW3D8.OBJ
I /INC $IIS:DITCW3F8.OBJ
SAVE $IIS:MYLKED
```

10. Execute the job created in step 9. This will build a new IIS load module which includes the *User Functions* and updated modules.

Adding New Screen Formats

1. The first step is to restore the assembler source or object decks for the new formats from the distribution tape. One of the MUSIC utilities mentioned earlier can be used for this purpose, depending on the method used to store the files on tape.

2. If the formats are distributed in source form each one must be assembled as follows.

```
/FILE SYSPUNCH N(format.OBJ) NEW(REPL)
/INC $IIS:IIS.ASM
/OPT DECK
/INC format
```

   where *format* is the name of the save library containing the source for the format.

3. The formats must be placed in the format library using the FORMLIB utility.

```
/INC FORMLIB
-
- the object decks for any new screen formats.
-
```

4. Once the formats have been placed in the format library the object decks or source can be removed from the system.

Chapter 14. FSI Configuration

Tailoring FSI Compilers/Processors Menu

The "Compilers/Processors Menu" of FSI gives users access to languages and programming systems that are installed on your MUSIC system. When you install a new product from the Optional Products tape, this menu file is automatically updated to make the new product available to your users through FSI. The menu can also be invoked directly using the XCOMPILE exec. ($INT:XCOMPILE).

You can also add items to this menu by adding items to the file $INT:COMPILERS using the editor. The following shows an example of the contents of the file $INT:COMPILERS.

```
*         JCL     Opt Description
*         Name
*
-         ASM      V  VS Assembler
-         VSF      V  VS Fortran Compiler
-         FID      N  VS Fortran Interactive Debug
-         VSC      V  VS/COBOL Compiler
-         GPS      V  GPSS V
-         VSP      V  VS PASCAL
-         PLI      V  PL/I Optimizer
-         PLT      N  PL/I Interactive Test
-         FG1      V  Fortran G1 Compiler
COBTEST   NONE     N  COBOL II Test Environment
VSAPL     NONE     N  VSAPL
SAS       NONE     N  SAS Version 5.18
PASCAL    NONE     N  Waterloo Pascal
-         RPG      V  RPG II
-         RPA      V  RPG II Autoreport
-         CB2      V  COBOL II Compiler
GDDM      NONE     N  GDDM Interface
IBMBASIC  NONE     N  IBM Basic
-         C37      V  IBM C/370 Compiler
-         R37      V  SAA RPG/370 Compiler & Lib
```

Each record contains four fields. If the first character is a "*", the entire record is treated as a comment. Records that are not comments are displayed on the menu in the order that they appear in the file. Up to 38 non-comment items are supported.

The first field can be a program name or a "-". If it is a program name, the specified program will be executed when the user selects the item. In the example above, if the user selects the GDDM Interface the program in the file GDDM is executed.

If a "-" is specified in this field it indicates that the XCOMPILE exec will handle the process. In this case, the second field contains a pointer to sample job control statements and help files, and the third field contains optional parameters.

The forth field is the description of the program or product that appears on the user's menu.

Changing the menu items

You can use the editor to modify the description field or change the order of existing menu items. Use the TEDIT command to preserve upper and lower case.

Adding a program to the menu

Imagine you have written a local information retrieval program that is executed by the command INFOX and you want to make it available to users through FSI. Insert the following entry to the $INT:COMPILERS file. Note that the menu items are displayed in the order they appear in the file.

```
INFOX    NONE  V  Local Information Retrieval Program
```

The XCOMPILE Language Processor Interface

The XCOMPILE exec has built in logic to manage the user interface to various language processors. This allows people to use these languages without knowledge of MUSIC control statements or language options and parameters. The user is automatically lead through a series of steps to edit the program, compile it, execute it and review the output. If the first field in the menu file is a "-", it indicates that the XCOMPILE exec should treat this item as a language processor and handle the user interface itself.

In this case the second field is used to locate a sample control statement file and help information on the language itself. This is, by convention, a three character identifier.

For example, with VS Fortran the identifier is VSF. This means that the file $INT:JCL.VSF contains the sample control statements for running VS Fortran, and that help on the language is under the help topic VSF. The sample control file is as follows:

```
File: $INT:JCL.VSF

/SYS REGION=1024
/LOAD VSFORT
/OPT FLAG(E),SOURCE
/INC ?.S
```

The XCOMPILE exec asks the user to enter a program name. This program name is substituted into the sample control statements anywhere a ? is found. XCOMPILE then creates the following files in the user's library.

name.VSF the control statements to compile and execute the users program

name.S the source program

name.LIST output from the compile and execution

The third field tells the XCOMPILE exec how to handle the display of output from program execution. If V is specified, all the output is saved and viewed at the end of the job. When N (none) is specified the program output is displayed as it is produced. This second case is usually desirable for interactive applications.

Modifying Existing Interfaces

You can edit the menu file ($INT:COMPILERS) to change the item description and output viewing option. Language options and file definitions can be changed by modifying the sample control statement file.

Remember that this file contains the defaults seen by all users.

Adding a New Language Processor

Do the following to allow the XCOMPILE exec to handle the user interface for a language.

1. Invent a 3 character identifier. For example ABC.

2. Create the help text in a file ($HLP:@GO.abc). Add an entry to the HELP topics file ($HLP:@GO.TOPICS) so that "HELP ABC" will give information on the processor.

3. Create the sample control statement file ($INT:JCL.ABC). This should have the SHR or PUBL attribute and contain question marks (?) wherever you want the user's program name substituted.

4. Add the entry to the menu file ($INT:COMPILERS).

```
-       ABC  V  The ABC Language Compiler
```

Chapter 15. ACCESS Facility

Overview of ACCESS

The ACCESS facility allows users to connect to applications on CMS or other MUSIC systems using the MUSIC Passthru interface. Also, users can access systems outside of MUSIC such as CMS, VSE, MVS, and other MUSIC systems. The MUSIC user is automatically connected and signed on, required files are transferred, the application is run and results are transferred back to MUSIC.

How ACCESS works

When a user issues the ACCESS command the following occurs

1. The user's MUSIC ID is checked in the AUTHORIZATION table.
2. If UNIQUE access is specified, the user is prompted for a remote userid and a password. If SHARED access is specified, a userid and password from the SHARED IDs table is used.
3. A connection is made with the remote system and the user is signed on.
4. If automatic file transfer is selected, files are moved from MUSIC to the remote system at this point.
5. If user controlled file transfer is selected, the user is presented with the file transfer menu at this point.
6. The application is started by executing the specified EXEC. A parameter can be passed to the EXEC by specifying a parameter on the ACCESS command (ie "ACCESS accessname parameter").
7. When the application terminates it should write the termination sequence to the screen. When ACCESS recognizes this, it procedes to steps 8, 9 and 10.
8. If automatic file transfer is selected, files are moved back to MUSIC from the remote system.
9. If user controlled file transfer is selected, the user is presented with the file transfer menu at this point.
10. ACCESS terminate and returns control to the MUSIC session.

Component Description

There are four basic components to the ACCESS facility.

ACCESS SETUP Facility

The ACCESS SETUP facility allows the administrator to define the external systems that are available and manage which users can access these systems. This facility lets you define the system and assign an ACCESS NAME. This name is used by the users on the ACCESS command to specify which application they want to use. See below for a detailed description of this facility. The ACCESS SETUP facility is invoked from ADMIN option 4 10 12, and is also documented in the *MUSIC/SP Administrator's Guide*,

under the topic "Updating External Access Tables".

SHARED IDS Table

The SHARED IDs table is a file that contains a list of CMS userids, a password for the userid, and an optional application name for the userid. The CMS userids are used to allow MUSIC users to log on to CMS from a MUSIC application program. Thus, all MUSIC users share the CMS userids by running a MUSIC program requiring the SHARED IDs table.

An example of a MUSIC application that uses this facility is ACCESS. ACCESS calls the SIGNON subroutine to complete the signon procedure. (See below for further details on SIGNON.) If you specified SHARED IDs when you were setting up the ACCESS control file using the ACCESS SETUP facility, ACCESS calls the SIGNON subroutine with SHARED specified as a parameter in the calling sequence. (If you don't specify SHARED, UNIQUE is used which means that the user is prompted for a CMS userid and password when the time comes to log on to CMS.) This tells SIGNON to use the SHARED IDs table and find a userid to use.

The SHARED IDs table can be used to exclusively reserve a CMS userid for an application by specifying an application name for the CMS userid, and using the same application name for the application. (The ACCESS SETUP facility is used to set an application name for the application. Also, the SIGNON subroutine takes an application name as one of its parameters.) Thus, many userids can exist in the SHARED IDs table, and different userids can be reserved for different applications. This allows you to set up the CMS accounts differently based on the application.

The SHARED IDs table is displayed/updated from ADMIN option 4 10 10, and is also documented in the *MUSIC/SP Administrator's Guide*, under the topic "Updating Shared ID Table for CMS IDs".

Example of a SHARED IDs table

```
VMUSER01 VMUSERX TCPIP
VMUSER02 VMUSERX TCPIP
VMUSER03 VMUSERX SQL
VMUSER04 VMUSERX SQL
VMUSER05 VMUSERX
```

The first two CMS userids are reserved for a MUSIC program using an application name TCPIP. The second two CMS userids are reserved for a MUSIC program using an application name SQL. The last CMS userid can be used for any MUSIC program not requiring an application name.

ACCESS Facility

The ACCESS Facility is the user program that allows users to connect to applications on CMS or other MUSIC systems. The administrator has set up these external applications via the ACCESS SETUP facility. (See below for further details on the ACCESS SETUP facility.) The command syntax is

```
ACCESS accessname [parameter]
ACCESS ?
ACCESS
```

where

accessname is the name of the external facility that you wish to access.

parameter is an optional value that is passed to the external facility once it is started.

? displays the list of external facilities available at your site.

Brief help is displayed when ACCESS is entered without a parameter.

Examples:

1. "access telnet nic.ddn.mil" logs in to the foreign host nic.ddn.mil using the VM/SP telnet program, a TCP/IP protocol.

2. "access sql" connects the user to sql on the local VM/SP system.

SIGNON Subroutine (REXX)

This subroutine is used by the ACCESS facility to complete the signon procedure to the application. See *Chapter 22 - System Programming* for further details on the SIGNON subroutine.

ACCESS Setup Facility

This setup program lets you, the MUSIC administrator, define the systems that are accessible to users.

The first screen lets you define the system and assigns an ACCESS NAME. This name is used by the users on the ACCESS command to specify which application they want to use. Other items on this screen include

- type of system (MUSIC or CMS)
- name of the EXEC that executes the application
- string MUSIC uses to detect application termination
- VM Passthru node of the remote system
- whether user file transfer is allowed
- CMS T-disk space
- set an application name to match with the Shared IDs file

When you start up the ACCESS Setup facility, you should fill in the ACCESS NAME that you want to create or modify. For new facilities, the name chosen should generally be descriptive of the actual function that is using invoked. For example, if the external application being accessed is the ADA programming language, an appropriate ACCESS NAME might be "ADA".

Enter the name you wish to use in the ACCESS NAME field and press ENTER. If you had previously setup the facility with the selected name, the fields on the screen will be filled in and you may make changes to the existing information. If the name entered was not previously used, the fields will be blank, waiting for your input.

The message line (at the top of the screen), provides information on whether this is an existing or new facility name.

The following describe the options available for setting up the facility.

Command to issue

This command will be issued after the Remote system has been accessed. When this option is specified and you are accessing a VM/CMS system, the following occurs:

- CMS is IPLed but NO profile exec is started
- If requested, a temporary disk is allocated
- The command is issued, and if a parameter was specified on the ACCESS command (ie "ACCESS accessname parameter"), the command is issued with the parameter

When this option is NOT specified and you are accessing a VM/CMS system, the following occurs:

- CMS is IPLed AND the profile exec (if any) is invoked

When accessing a MUSIC/SP system, and you wish to use this option, make sure that the userid that is being used DOES NOT have an AUTOPROG specified.

This field is optional.

Return to MUSIC/SP message

This field defines the message the Access Facility searches for to know when the application has completed processing and is returning to the MUSIC/SP environment.

The information entered MUST match exactly the message produced on the remote system. For example, the message "This is a TEST" is NOT the same as the message "THIS IS A TEST". The case (upper/lower letters) does matter when the Access Facility checks the message.

This field is optional, but MUST be specified if you wish to transfer files from the Remote system back to MUSIC/SP.

One method of using the above on VM/CMS is the following.

1. Create a REXX exec on CMS that has the same name as that specified in the INITIAL COMMAND.
2. In the exec perform any command to setup and invoke the desired feature/function.
3. The last line of the exec would be: say " return to MUSIC/SP message "

Example: Initial command: MYSAMPLE
Return message: Completed Processing, Return to MUSIC/SP

```
MYSAMPLE exec
      /* Sample exec */
      ACCESS 319 P
      EXEC LOCAPPL  /* start an application */
      say ' Completed Processing, Return to MUSIC/SP'
      exit
```

System Type Defines the type of the REMOTE system. This field is required. The valid entries are:

CMS - remote system is a VM/CMS environment
MUSIC(vmid) - remote system is another MUSIC/SP system. vmid is the virtual machine name in which the remote MUSIC/SP system is running.

PVM Node Defines the VM Pass-Through node if when the Remote system is accessed via VM Pass-Through. This field is optional.

Allow File Transfer TO remote system

If this option is "YES" then the user will be prompted with a menu. The user enters the MUSIC/SP filename to be transferred, the name to be used on the Remote system and any options to be used when transferring the file. If this option is set to "NO', the user will NOT receive the menu.

Allow File Transfer from remote system

If this option is "YES" then the user will be prompted with a menu. The user enters the Remote system filename to be transferred, the name to be used on MUSIC/SP and any

options to be used when transferring the file. If this option is set to "NO', the user will NOT receive the menu.

TDISK address

When accessing a VM/CMS system, the facility can auto- matically allocated temporary disk space. This space is available during the time the user is on the system. Specify the disk address to be used when allocating the temporary space. This field is optional and may only be specified when SYSTEM TYPE is CMS.

Number of blocks

Specifies the number of blocks that should be allocated when using temporary disk space under CMS. The number of blocks will be automatically allocated on the supported disk device types. This field is required if the TDISK ADDRESS field is specified.

Access mode Specifies the CMS access mode to be associated with the temporary disk space that is allocated under CMS. The mode is a letter (A-Z).

The devices supported for allocation of temporary disk space are:
FB-512, 3380, 3375, 3350

Application name

Specifies a name that is used to match against an application name for a userid in the Shared IDs table. If the names match, the userid is used for this application. Using an application name allows you to exclusively reserve a userid or a group of userids for a given application. See ADMIN 4 10 12 to add an application name to the Shared IDs table.

Example: Assume the Shared IDs table is the following

```
VMUSER01 VMUSERXX TCPIP
VMUSER02 VMUSERXX SQL
VMUSER03 VMUSERXX
```

Also, assume we have set up our application, called TELNET, using the method to update the external access table (ie ACCESS Setup facility) and assigned it an application name of TCPIP. Then, when we run "ACCESS TELNET", only the CMS userid VMUSER01 is used for this application.

Once the first screen is filled in press PF5 to create the authorization table. By default all users have access, but you can limit access to a particular user or group of users if required. This table also specifies whether SHARED or UNIQUE access is to be used for a particular user. When SHARED is specified, one of the shared CMS userids is used. These are defined using item 10 of the ADMIN04A menu. When UNIQUE is used, each user is prompted for a remote userid and password. See the "Application name" field description above on how the name and the Shared IDs work together.

The ACCESS facility can also automatically transfer files to the remote system before the application starts and transfer any result files back to MUSIC once the application has terminated. You can set this up by pressing PF7 from the main screen. This facility should not be confused with the user controlled file transfer that can be selected from the main screen. With this user controlled option, a menu is presented before and after the execution of the application, allowing the user to transfer files.

In either case, files are sent to the remote system before the application is run and back to MUSIC after its termination. To do this ACCESS must be able to detect when the application has finished.

How to set up an Application

Assume that you have the VM/SP TCP/IP IBM program product installed on your local VM/SP system. You can set up an application for a group of users to have access to TELNET on VM/SP using this facility. There are a number of steps involved in setting up this application.

1. Allocate the CMS userids

   Allocate the CMS userids on VM/SP. Decide whether you want the application to use TDISK space, a 191 minidisk, or shared file system pool space and set up the appropriate directory for these userids. The following is a sample VM directory for the application TELNET

```
USER VMUSER01 VMUSERXX 2M 2M G 50 ON OFF OFF ON
 ACCOUNT CFVMTST MPG
 OPTION REALTIMER ECMODE
 IPL CMS
 CONSOLE 009 3215 B
 SPOOL 00C 2540 READER B
 SPOOL 00D 2540 PUNCH B
 SPOOL 00E 1403 B
 MDISK 191 3380 641 001 VMSYS2 MW
 LINK TCPMAINT 592 192 RR
```

2. Put the CMS userids in the Shared IDs table

   Use ADMIN 4 10 10 to add the CMS userids, passwords, and application names to the Shared IDs table. In this application, TELNET, add the application name 'TCPIP' to each CMS userid and password entry that you add. This reserves these CMS userids for use only with applications that have an application name of TCPIP. You can add other applications with different application names and their own CMS userids to use for that application. You can set up a second application, say FTP, and use the same application name which would give you use of the same CMS userids. This gives you the advantage of setting up a group of CMS userids that can be shared with a group of applications.

3. Set up the connect script

   Use ADMIN 4 10 12 to set up a connect script for the application. Set "Access name" to TELNET, "Command to issue" to TELNET, "Return to MUSIC message" to LOGOFF, "System type" to CMS, and "Application name" to TCPIP. Set Access Control (F5) "ID" to '*', and "Type" to SHARED. This allows all users access to TELNET, and they will be using the CMS userids in the SHARED IDs table designated for TCPIP. On a side note, if you decide to set Access Control "Type" to UNIQUE, the user is prompted to enter a userid and password for the signon when the ACCESS application is run.

4. Test it out

   Test out the TELNET application by typing "ACCESS TELNET" at *Go.

5. Tell your users about it

   Tell your users about the ACCESS facility and what applications are available to them. Note that there is online help from *Go for the ACCESS facility, and users can find out what applications are available by typing "ACCESS ?".

Chapter 16. Configuring MUSIC for TCP/IP

Overview of TCP/IP

MUSIC can communicate directly with VM TCP/IP Version 2 (and higher) through the socket interface. This allows you to run internet applications such as TELNET and FTP on MUSIC.

See *Appendix D - MUSIC TCP/IP for VM TCP/IP Version 1* if you are using TCP/IP Version 1. If you intend to use TCP/IP from MUSIC please read this section carefully.

The VM Configuration

The connection with VM's TCP/IP virtual machine is done through IUCV. An IUCV path is established for each application. Set the MAXCONN parameter on the OPTION statement in MUSIC's directory entry to allow enough connections for your MUSIC users. There is no overhead or penalty if you over estimate this number.

If you want MUSIC to be able to use the raw IP interface it must be authorized in the VM TCP/IP configuration. You can do this by adding the name of the MUSIC virtual machine to the OBEY statement in the VM TCP/IP configuration file. Note that the PING program requires this.

The $TCP:TCPIP.CONFIG File

This file contains information specific to your site, such as your MUSIC system's internet and e-mail name, the name of the TCP/IP virtual machine, the local domain name and the IP addresses of any name resolvers. The file has sample entries and comments indicating what the various parameters are. Edit the file and make any changes that are appropriate. Each entry consists of the parameter name (starting in column one) followed by the value of that parameter. The following parameters can be set.

TCPIP — The name of the TCPIP virtual machine.

HOSTNAME — The internet host name of your computer.

EMAIL — The E-mail address of your MUSIC system.

NAMESERV — The dotted-decimal internet address of a domain name server. (A number of these may be specified)

NAMEWAIT — This is the global (system-wide) name resolver timeout period, in in seconds. The resolver will read this at name resolution time and use this value for timeouts. Increase this number if your nameservers are not responding quickly enough to your requests - decrease it if response time is good and you wish to reduce the wait for bad name timeouts.

RESTRICT_TELNET_PORTS — This is a list of ports to which telnet users are to be denied access. This prevents MUSIC's status as a trusted machine from being abused by someone spoofing an internet client.

DOMAIN The domain portion of your host name.

NEWSFEED The internet address or name of a host providing an NNTP news feed.

NEWSGROUPFILE The name of a file to contain the list of NNTP news groups.

NEWSDROP_0_POSTS
Specifies whether news groups with no postings are dropped from the news groups list. (yes/no)

NEWSGROUPNAMESIZE
Maximum length of news group name before truncation.

ORGANIZATION The name of your organization.

TIMEZONE Your time zone. (GMT, EST, MST, etc.)

TIMEZONE_GMT Your time zone given as displacement from GMT (e.g. -5)

WHOISSSERVER This entry defines the whois server that the WHOIS command will connect to by default. If you have a preference, enter the server address here; otherwise, the default will be as coded in the $TCP:WHOIS program. Note that a list of whois servers can be obtained by running $TCP:GET.WHOIS.SERVERS - it ftps the file $TCP:WHOIS_SERVERS from:

```
sipb.mit.edu:/pub/whois/whois-servers.list
```

If the address 0.0.0.0 is found here, then it is not used and the the program default is used.

GOPHERD_ALLOWED_FILES
This entry defines the MASK for non BBS files allowed to be accessed via GOPHERD. For example to allow the subdirectory GOPHERD on userids that begin with dollar to be accessible you specify $*:gopher*

The default is $pub:*

GOPHERSERVER This is the name,port of the default gopher server that the MUSIC gopher client will connect to. This is only used when no other pointer (startup option, -d parm, @GOPHER.MENU) is available. If this is not present and no other option is available, then the MUSIC gopher client will connect to 'gopher.micro.umn.edu' at port 70.

GOPHERTIMEOUT This is the global gopher timeout value. It is overridden by the -tm parameter in gopher.

GOPHERD_DEFAULT_MENU
This entry defines the default FIRST MENU to be transmitted when no menu is specifically requested of the GOPHERD server. The default is $tcp:gopherd.menu

GOPHERD_BBS_LOG This entry defines a BBS log file for logging BBS topic file access. No menu is specifically requested of the GOPHERD server. This must be in the form "userid:@xxx.". Example cw99:@inf. is used for INFOMcGill. The default is none.

HTTPD_PERSONAL_PAGE
: This entry defines the default PERSONAL page to be transmitted when no page is specifically requested in the URL. for example http://musicm.mcgill.ca//abcd/ will get the default personal page text tacked on. The default is http\home.HTML

HTTPD_ALLOWED_FILES
: This entry defines the MASK for any file allowed to be accessed via HTTPD. For example, to allow the subdirectory HTTPD on userids that begin with dollar to be accessible, you specify $*:HTTPD* The default is *:*HTTP*

HTTPD_ALLOW_PUBLIC
: This entry specifies whether public files not fitting the HTTP_ALLOWED_FILES mask, should be accessible. The default is no.

HTTPD_ALLOWED_EXECS
: This entry defines the MASK for any file allowed to be executed via HTTPD. This is specifically meant for forms support in html files. The default is *:*HTTPEXEC*

HTTPD_DEFAULT_PAGE
: This entry defines the default page to be transmitted when no page is specifically requested in the HTTPD connection. The default is $000:http\home.HTML

Domain Name Servers

MUSIC will take advantage of a Domain Name Server if one is available. A Domain Name Server is usually a small computer that is dedicated to providing name lookup services. It keeps in contact with other name servers around the network to keep its information up to date. Given a host name in the format:

```
vm.mpg.mcgill.ca
```

It can quickly return the numeric internet address that is used by the TCP/IP applications to establish connections with remote hosts. When a user specifies a host name on a TELNET of FTP command, MUSIC asks the Domain Name Server to resolve the name and return the numeric internet address. The advantage of using names instead of numbers far outweighs the small overhead in doing the name lookup.

If a name server is not available, your users will have to specify all host addresses in the "dotted" decimal format (e.g. 132.206.120.2).

Domain Name Servers are defined in the $TCP:TCPIP.CONFIG file on the NAMESERV statements. The first one should be regarded as the primary name server. It is tried first. If it fails to respond within a 15 second time out period the system will try the second name server if one is defined. Note that if the primary server is down, users will see a 15 second delay in resolving host names. Extra name servers are only defined for emergency backup. If the primary name server is going to be down for any length of time it should be removed from the top of the list so that a backup can take over without the 15 second delay.

The $TCP:NET.LIST File

This provides your users with a list of network hosts and the services they offer. It is used by TELNET, FTP and NET. The list does not restrict or grant access to the host computers, it simple lets the user know that they exist. The sample that is distributed with the system should be modified to suite your local needs.

The format of the file is relatively simple. Each network host is represented by a record in the file. The first field of each record is the network address of the host. This can be specified in the dotted decimal format (e.g. 132.206.120.2) or in host name format (vm.mpg.mcgill.ca). The rest of the record should contain a description of the host or information about what services are available on that host.

The contents of the file are up to you. The file may simply contain a list of the computers on your campus, or could contain thousands of entries describing computers throughout the internet. At McGill for example we provide a list of archive sites that allow anonymous FTP access.

TCP/IP Log Files and Console Messages

TCP/IP applications use the log server to generate log files. Log file names are in the format:

```
$TCP:TCPIP.yyymm
```

where *yyyy* is the year and *mm* is the month. These log files contain messages from both outbound and inbound applications. Outbound applications that log actions to these files are TELNET and FTP; inbound applications are FTPD and FINGERD (FTPD and FINGERD are described later).

Also, FTPD, FINGERD and INETD (described below) issue messages to the console. This can be useful in checking to see if there is a problem with excessive access to your system. As well, in the case of an actual attempt at system penetration, these trails are useful in tracking remote connections, and in identifying TCBs where remote users are logged on, etc.

The Internet Super Server (INETD)

The INETD program (internet daemon) acts as an entry point for incoming connection requests to TCP/IP servers running on MUSIC. Its basic job is to wait for a call to come in on a port and then dispatch an appropriate server for that port.

In $PGM:SYSLG.DEFINE, TCP/IP applications are defined as monthly log files. INETD runs in the background as a BTRM. It creates sockets for each of the ports defined in the file $TCP:INETD.PORTS file and listens for incoming calls. When a call comes in, it accepts the connection and executes the appropriate server for that port. The server is started as another BTRM and is passed the connection. In other words, INETD runs one copy of the server for each incoming call. As the server program performs its task, INETD goes back to listening for more incoming calls.

Before INETD accepts a connection, it verifies that the incoming ip address and port are not in the killfile $TCP:IP.AUTHORIZE. If they are in the killfile, the connection is refused and a shutdown on the connection is done. If they are not in the killfile, INETD accepts the connection and it executes the appropriate server for that port.

Note that each copy of a server requires a MUSIC session. If you anticipate high usage, make sure you have sufficient extra sessions defined in the NUCGEN (XSES parameter) to handle the load.

The file $TCP:INETD.PORTS contains the list of ports and servers that are handled by INETD. The following is a sample.

```
;
;  The following are default ports
;
21   stream 10 $tcp:ftpd
```

```
70   stream  0 $tcp:gopherd
79   stream  0 $tcp:fingerd
80   stream  0 $tcp:httpd
106  stream  0 $tcp:poppassd
110  stream 10 $tcp:popd
370  stream 300 $mcs:mcsh
;
;  The following define ports for demonstration servers.
;
2000 stream  0 $tcp:demo.serv.fort
2001 stream  0 $tcp:demo.serv.rex
3000 stream  0 $tcp:demo.telnet.rex
```

The first column contains the port number of the service, the second the type of port, and the third contains the maximum number of simultaneous sessions allowed on that port (zero indicates no limit).

The remainder of the record contains the name of the execution file for the server, followed by any parameters that are to be passed. For example, if a connection comes in on stream (TCP) port 21, INETD will execute the program "$tcp:ftpd".

The port number should be between 1 and 3999. Certain ports are know as well known ports because they run well known services. For example:

```
  21 - File Transfer Protocol (FTP)
  23 - TELNET
  25 - Simple Mail Transfer Protocol (SMTP)
  53 - Domain Name Server
  70 - Gopher Server
  79 - Finger Server
  80 - Web (HTTP) Server
 110 - POPD Server
```

There are two values for port type: "dgram" and "stream". "dgram" ports are not really handled by INETD, it simply starts the server as a BTRM. In the above example INETD will start the program "$000:msgd" as a BTRM, but take no part in handling calls or data on the port.

The following files are distributed executor files for servers available with MUSIC, eligible for inclusion in the file $TCP:INETD.PORTS.

$TCP:FTPD	FTP (File Transfer Protocol) server
$TCP:FINGERD	Finger RUIP (Remote User Information Protocol) server
$TCP:GOPHERD	Gopher Protocol server
$TCP:HTTPD	Web server
$MCS:MCSH	MUSIC/SP Client/Server
$TCP:POPD	POP (Post Office Protocol) server
$TCP:POPPASSD	POP Password server
$TCP:DEMO.SERV.FORT	demonstration server writtem in VS/FORTRAN
$TCP:DEMO.SERV.REX	demonstration server written in REXX
$TCP:DEMO.TELNET.REX	demonstration TELNET server (line mode)

The file $TCP:IP.AUTHORIZE contains a list of ip address and port combinations which are not allowed access to the defined service. Before INETD accepts a connection, it verifies that the incoming ip address and port are not in the killfile $TCP:IP.AUTHORIZE. If they are in the killfile, the connection is refused and a shutdown on the socket is done. If they are not in the killfile, INETD accepts the connection and it executes the appropriate server for that port. The following is a sample.

```
*
*---Sample authorization file
*
*   Remove the leading  *' to activate the following statements
*
*Address          Port   Optional Comment
*-------          ----   ----------------
*vm.mpg.mcgill.ca  79    testing access via port 79
*123.456.789.012   21    numeric IP addresses also acceptable
*   *              70    any site; port 70; i.e. no gopher
*123.456.789.012   *     a particular site, any port
```

The first column contains an ip or symbolic address for which you want to block access. ";" or "*" in this column signifies a comment line. Enter a blank followed by "*" in column one if you want to restrict everyone.

The second column contains the port number to restrict or "*" to restrict all ports.

Anything after the second column is comments and can be used to document the entry.

The following options can be set as namelist parameters. Edit and modify the file $TCP:INETD to change these values.

TRACE=T/F specifies whether tracing is enabled or disabled. When tracing is enabled, tracing information from the socket calls is recorded in the file @TCPIP.LOG, and tracing information from the socket I/O is recorded in the file @TCPIP.BUFFERS. The default is TRACE=F.

RESET=T/F specifies whether INETD restarts (true) or halts (false) after ERRLIM errors are encountered. The default is RESET=T.

ERRLIM=nnnn specifies the number of consecutive SELECT or ACCEPT errors are accepted before INETD is stopped. INETD can be automatically restarted when it stops by setting RESET=T. The default is ERRLIM=20.

RETRYD=nnnn specifies the seconds to delay when a SELECT error occurs. The default is RETRYD=10.

RETRYB=T/F specifies whether BIND failures are retried every BINDTM seconds for BINDCT retries or whether BIND failures are not retried. The default is RETRYB=T.

BINDTM=nnnn specifies the seconds to delay between BINDCT retries when a BIND failure occurs. The default is BINDTM=31.

BINDCT=nnnn specifies the retries to attempt when a BIND failure occurs. The default is BINDCT=20.

TSKTIM=nnnn specifies the seconds waited for a server task to be started. If the server has not started, INETD does a shutdown on the socket. The default is TSKTIM=30.

NOAREP=T/F specifies whether the attempted access violations are reported (false) or not reported (true). Access is limited by use of the file $TCP:IP.AUTHORIZE. The default is NOAREP=F.

To use INETD:

1. Add a BTRM definition to the NUCGEN to run the program. Also make sure you allocate enough extra sessions (XSES parameter) to allow for the additional server sessions.

2. Create a $MONxxx userid with INETD as the autoprogram. This userid must be given the privileges

required to run any of the servers. The FTP server requires SYSCOM, MAINT, FILES, CODES, LSCAN, CREAD, VIP, and JOBLIM. Other servers may require other privileges.

```
(xxx-the device address of the BTRM)
```

Example of CODUPD command:

```
ADD $MON SC(xxx) AUTOPROG($TCP:INETD) NAME(BTRM FOR INETD) -
PW(*NOLOGON) BAT(0) PRIME(NL) NONPRIME(NL) DEF(NL) XSESS(200) -
SYSCOM MAINT FILES CODES LSCAN CREAD VIP JOBLIM
```

3. Add entries to $TCP:INETD.PORTS for any servers you wish to run.

FTP Client (FTP)

The FTP client is used to conduct FTP sessions with remote servers. There are a number of parameters available to configure or tailor your FTP client on a system-wide basis. (Individual users can of course override these defaults.)

The following options can be set as name-list parameters. To change these, edit and modify the file $TCP:FTP.

BUFSIZ=n	specifies the amount of buffer space used in TCP/IP requests - this number has an impact on the system buffer pool (default 8k, min. 50 bytes, max 31744 bytes).
WAIT=n	specifies the timeout period
ATODIR=T/F	specifies whether automatic directory display is enabled or not (default: autodir is enabled)
DIRIN=T/F	specifies whether or not to use LIST command by default (default: LIST is used instead of NLST)
DEFSP=n	specifies the default primary space allocation for created files (default: 100k)
TRA2E='name'	specifies the replacement ASCII-EBCDIC translate table (default: no replacement)
TRE2A='name'	specifies the replacement EBCDIC-ASCII translate table (default: no replacement)
TRE2E='name'	specifies the replacement EBCDIC translate table for screen display
FS=T/F	controls whether fullscreen startup is enabled or note (default: fullscreen startup is enabled)
SUNIQ=T/F	specifies whether or not to use STOU for FTP PUT requests instead of STOR (default: STOU is used)
LOG=T/F	specifies whether a log file entry is written (default: log file entries are written)
DEBUG=T/F	same as -t at startup - turns tracing on. This is useful for a standard debugging executor. (Default: this is not enabled.)
BINXFR=T/F	specifies whether the MUSIC FTP client is to attempt BINARY BLOCK transfers when connected to a MUSIC server (default: this is enabled).

FTP Server (FTPD)

The FTPD program works with INETD to provide the FTP server function. If you wish to provide this service you must activate INETD (see the previous section). It is also possible to start FTPD on its own. This will support only one connection at a time and is intended for debugging purposes.

The FTPD program has the following options that can be set as name-list parameters. To change these edit the file $TCP:FTPD.

ANONID='anon1 ,'anon2',... Userids that can be used to gain anonymous access to FTP. The default is ANONYMOUS. These users will be allowed to access files belonging to the MUSIC userids specified in the positionally corresponding PUBCOD parameter. Up to 20 anonymous userids can be specified.

ANONOK=T/F This parameter determines whether anonymous access is allowed. The default is T (true). If you wish to disable anonymous access set this to F (false).

DROP=n The number of minutes to wait before dropping an idle connection. The default is 10 minutes.

PORT=n The port number that FTPD should listen on for incoming connections. This is NOT used if FTPD is started by INETD. It is used when you start FTPD on its own for debugging purposes. Note that this SHOULD be different from the port that INETD uses - TCP/IP will accept 2 or more servers running on the same port number, and pass calls to them each in turn (in a round robin fashion).

PUBCOD='userid'...'userid'
A list of MUSIC userids associated with the anonymous userids as determined by the ANONID parameter above. These need not be different, but you should note that these are effectively public, and the information on them should be treated as such. The default is $PUB. The maximum is 20 userids.

UPANON=T/F,T/F,T/F
This determines whether anonymous users are allowed to upload files to your public library. The default is F (false). The maximum is 20 values (corresponding to the maximum 20 anonymous userids).

TRFLAG=T/F This is used to activate tracing. The trace information is written to unit 6 (the screen) but can be mapped to a file if FTPD is run in the background. This is intended only for debugging and the default is F (false).

FTP Ports

According to TCP/IP standards, port 21 is reserved for FTP services. By default VM TCP/IP uses port 21 for the CMS FTP server. If you wish to continue to provide this access, MUSIC's FTP will have to use another port, perhaps 1021. In this case the entry in the $TCP:INETD.PORTS file would read:

```
1021 stream 10 $tcp:ftpd
```

Port 21 would give FTP access to the CMS file system. Port 1021 would give FTP access to the MUSIC file system. Users would have to specify port 1021 on the FTP command to access MUSIC. If most of the files of interest are on MUSIC you may wish to give MUSIC the default FTP port. In this case the entry in $TCP:INETD.PORTS would be:

```
  21 stream 10 $tcp:ftpd
```

In addition to this you would have to modify the PORT section in "TCPIP CONFIG" on CMS and change PARM option in the FTPDEXIT EXEC to prevent the CMS FTP server from taking port 21 for its own use. (See the section on "Configuring the FTP Server" in the Planning and Customization manual for VM TCP/IP.)

Multiple FTP Servers

You may also wish to run a number of different FTP servers on MUSIC. Perhaps one for general use and another to give anonymous access to a specific group of files. In this case the $TCP:INETD.PORTS would contain an entry for each server:

```
  21 stream 10 $tcp:ftpd
2021 stream  5 $000:ftpd2
```

Calls coming in on port 21 would get the standard server, those on port 2021 would get a modified server executed from the file $000:ftpd2.

FTPD Security Exits

These exits allow a site to tailor the behaviour of the FTPD server with respect to allowing and disallowing incoming calls. INETD already allows a site to control the number of concurrent sessions being run by a server process; the FTPD server is not even invoked at that point. These exits provide a means of fine-tuning access based on time-of-day, remote site id, etc. For example, during the daytime hours, only local FTP connections might be allowed, or only certain FTP commands might be allowed.

There are four exits defined. They are:

EXCODE	provides for validation of the userid/password entered via the USER and PASS FTP commands
EXCONN	provides for validation of the remote IP address
EXCOMM	provides for validation of any command entered by the remote user
EXQUIT	called when the connection is shutting down, for logging purposes

In order to install the exit procedures, you must do the following:

1. Replace the module $TCP:FTPD.$EXITS.S with your routines. You can follow the model there (it uses ENTRY statements within one routine) or you can provide separate routines. You should take care not to use COMMON blocks from the main program, as in future these exits will be installed in the load library, and such information will not be available. The names EXCONN, EXCODE, EXCOMM, and EXQUIT must be resolved at link-edit time. Keep in mind while coding your routines that the module FTPD is not run as a privileged program (FTPD has no privileges) but rather is run on a privileged userid.

2. Compile your routines into the file $TCP:FTPD.$EXITS.OBJ.

3. Review the file $TCP:FTPD.LKED. Make sure that any routines (.OBJ files) exist on disk. (Note that .OBJ files are commonly not resident - you might have to use ADMIN to restore these files.) Once satisfied, re-link FTPD by running $TCP:FTPD.LKED.

4. Review the linkage editor output. If there are no errors, then your new version of FTPD is in production. (You can test it by running it from a terminal session - you'll need SYSCOM on the test userid.)

In the future, these exit routines will be installed in the system load library, and a much system installation process will be available. In the meantime, do not make use of any common block information within the FTPD source module. Later versions of the FTPD server with linkage to the exit routines performed from the system load library will make access to this information impossible.

Descriptions of the User Exits

The EXCODE Exit

The EXCODE user exit is invoked from FTPD as follows:

```
CALL EXCODE(RC, USERID, ULEN, UPASS, PWLEN, PUBCOD, ANONYM)
```

Arguments:

USERID	16 character userid
ULEN	raw length of the userid (note that this is NOT the file ownership id) (4)
UPASS	the password (8 char.)
PWLEN	the password length (4)
PUBCOD	the 16 char. userid of the public directory to use.
ANONYM	pointer to the anonymous userid common block. This can be moved to your own common block for processing. This should not be altered. The common block looks as follows:

```
CHARACTER*20 $PUB(20)
CHARACTER*20 ANONID(20)
LOGICAL      ANONOK
INTEGER      NANON
COMMON/ANONYM/ANONOK,NANON,ANONID,$PUB
```

Note: Do NOT use this common block directly. Future versions of the FTPD user exits will be loaded from the load libarary and this will not be available.

This exit provides the site with greater control over access to the FTP resource. Access can be denied based on time of day, etc. This exit is called in two places:

a. when processing the USER command, if the user enters an id that matches an anonymous userid (for which by default there is no password. Note that normal processing in this case means that the login is accepted, and that EXCODE will NOT subsequently be called with a password. If you want to block the login, this is it. RC=1 on entry here.
b. when processing the PASS command. The password that the user entered is provided. Note that FTPD runs with the CODES priv, so this exit can take advantage of that and query the code table itself. RC=0 on entry here.

When EXCODE returns, processing action taken will use the new values, if any, in USERID, ULEN, PASSWD, PLEN. These may be overriden by the processing action, in the case where you request that a password be required and it is an anonymous userid.

Meaning of the Return Codes that your routine sets:

0 - permit login (normal processing)
1 - reject login (socket is not closed)

2 - reject login (shutdown requested)
3 - allow login, but require password, even
if anonymous
4 - allow login, no password required.

Notes on Individual Cases (by return code)

rc=0 requests normal processing, as if no exit routine were called. This is the default action without an exit routine supplied.

rc=1 login is rejected, but the remote user remains connected. The user is free to try again, subject to the limit of the LOGATM value specified in the EXCONN exit. If the user has no hope of logging in due to the local time, etc. this is faster than rejecting at the PASS command.

rc=2 the userid is rejected, and the socket is shutdown. This is perhaps appropriate when login is disabled for any reason, and you wish to discourage attempts to login.

rc=3 normal processing can continue, but the user must supply a password. This applies even to anonymous login, so that this exit WILL be called again when the PASS command is processed. At this point, rc=4 can be specified to allow any password entered. This option allows the site to check for userid@site-address password forms for anonymous logins.

rc=4 Login is allowed, and in fact this return code validates the user. No subsequent access to the code table is required. This can be used in conjunction with rc=3 to enhance anonymous access by requiring a user@site format password string, which is not passed to the code code table.

Note that if login is denied, then message 531 is returned to the remote client.

The EXCONN Exit

The EXCODE user exit is invoked from FTPD as follows:

```
CALL EXCONN(RC, RMTIP, RMTPRT, LOCIP, LOCPRT, DROPTM, LOGATM)
```

Arguments:

RMTIP	Remote IP Address
RMTPRT	Remote Port
LOCIP	Local IP Address
LOCPRT	Local Port
DROPTM	Drop time in seconds (modifiable)
LOGATM	Number of login attempts (modifiable)

These values are passed by value only except as noted. All of these are of type INTEGER*4.

Meaning of the Return Codes that your routine sets:

0	- normal processing (call is accepted)
nonzero	- a message of 430 is issued and the socket is shutdown.

This exit allows you to ascertain if you wish the remote connection to be continued. The information provided is read-only - you cannot change it. It is provided only so that your user exit can decide whether to shutdown the connection. Note that you do not have access to the socket - if you wish to issue a message, a non-zero return code must be returned. Halting the program via CALL EXIT will also have the same effect,

except that VM's TCP/IP will issue a message that the connection was broken. With the 430 message, the user knows why the shutdown occurred.

The EXCOMM Exit

The EXCOMM exit is called from the FTPD server as follows:

```
CALL EXCOMM(RC, CMD, INPUT,RLEN, CMD,CLEN, CMDOPS,OPLEN, NXTCMD)
```

Arguments:

INPUT	raw message from the remote user
RLEN	length of this message - do not exceed this
CMD	Command as parsed by FTPD
CLEN	length of the command
CMDOPS	command operand(s) as parsed by FTPD
OPLEN	length of the operand field
NXTCMD	the next command that FTPD is looking for. (eg. user/PASS, rnfr/RNTO, etc.) Set this to blank to cause FTPD to ignore this test.

Meaning of the Return Codes that your routine sets:

0 - continue normal processing
1 - go directly to command interpretation
2 - Abort command: put out a message and get next user command
<0 - FTP server message code - see below

This allows your user exit to determine what will happen as a response to the user command just issued. If RC is negative (RC < 0), then the positive value of RC is interpreted as an FTP message code which is issued to the user. In this case, CMD is the (optional) text used to tailor the message. If no tailoring text is desired, set this field CMD to blanks. If R=2, then message 505 is issued, aborting the command. RC=1 means bypass any further command validation and execute the command.

The EXQUIT Exit

No optional actions are provided for. EXQUIT gains control only for local purposes (logging, etc.). Thus, no return code is required and no parameters are passed.

POP3 and POPPASS Servers on MUSIC

Introduction to POP

Post Office Protocol, POP, is a client/server model for mail designed to work with TCP/IP. The user can run POP client software on any platform, connect to a POP server, and download their mail to the client. The POP client software acts as the Mail User Agent, MUA. All clients can display the mail and perform various mail handling tasks such as answering it or copying it to a file. Clients also allow you to create mail to be sent without being connected to the server (ie offline), and then connecting to the server immediately or later and sending the mail. The POP server software acts as the mail Message Transport System, MTS. The POP server platform has sufficient resources for this service, whereas the POP client platform typically does not have the resources for this task. The POP client software allows the user to work offline and connect to the POP server only when required.

The POP specifications are developed by the Internet community and its governing bodies. The latest version of POP is POP version 3, POP3, and it has been widely implemented on most computer platforms.

MUSIC users have a choice of how they view their mail. Users can use the traditional way of connecting to MUSIC via a tn3270 client or a direct connection and view their mail interactively with the MUSIC MAIL program. Alternatively, users can use a TCP/IP POP3 client on their own computer, download their mail from MUSIC via the MUSIC POP3 server and view their mail on their computer.

POP3 Server

The MUSIC POP3 (Post Office Protocol version 3) server is a TCP/IP server that allows a user using a POP3 client to download their mail from MUSIC. Your users can connect to MUSIC with a POP3 mail client such as WinPMail or Eudora on a PC, or Eudora on a MAC, and fetch their mail from MUSIC. Once they have fetched their mail, they can read it offline. The POP3 clients allow the creation of mail to be sent offline and they have the ability to send it immediately or at a later time. Use of a POP3 client reduces the connect time to MUSIC when you compare it to the connect time for a user who signs on to MUSIC via tn3270 and reads their mail interactively.

Mail attachments and MIME support are available to your users via the MUSIC POP3 server as these features are a function of the POP3 client. Most POP3 clients include support to allow the attachment of any file(s) from the local environment to the message being sent. So, a user can send their WordPerfect, Microsoft Word, or spreadsheet file via mail to another person. Also, most POP3 clients support MIME, Multipurpose Internet Mail Extensions. MIME provides facilities to include multiple objects of any type in a single message. For example, a mail message can contain a text object, an image object, and an audio object. When the POP3 client receives a MIME message, usually a user can choose which object to "view" in the message.

POPPASS Server

The POPPASS server is a TCP/IP server that allows a user to change the password for their POP3 userid. On MUSIC/SP, the user changes their signon password as the POP userid is their signon password. This server supports the Eudora POP client "Change Password" and the WinPMail POP client "Change POP3 Password" requests. The Eudora products for various platforms are available from QUALCOMM Incorporated. The WinPMail product is available free on the internet.

POP3 Commands and RFCs

The MUSIC/SP POP3 server complies with RFC1939 and supports the optional POP3 commands:

TOP msg n	returns the mail headers and "n" lines of the mail body in a multi-line response.
UIDL [msg]	returns the unique-id of the message.
APOP name digest	is an authentication mechanism that does not send a server/userid specific password in the clear on the network.
USER name PASS string	when USER and PASS commands are used together is an authentication mechanism that sends a server/userid specific password in the clear on the network.

RFC1939 is the latest revision (May 1996) of Post Office Protocol Version 3 and it is a Standard Protocol.

The MUSIC/SP POP3 server also supports RFC1725 and RFC1460 which RFC1939 obsoletes, allowing

POP3 clients that are only RFC1725 and/or RFC1460 compliant to work with the server.

XTND XMIT Support

The MUSIC POP3 server also supports the Berkeley extension XTND XMIT which allows the POP3 client to send mail to the POP3 server for delivery. If the server does not support this feature, the POP3 client must send the mail to a SMTP server. The MUSIC XTND XMIT support includes sender verification. This prevents a user from masquerading as another user by setting a "sender" header in the mail to someone else's email address. The MUSIC POP3 server verified that the POP userid's email address is in one of the From/ Sender/Reply-To "sender" fields. If it is not, the POP3 server refuses to send the mail on the user's behalf and signals an error to the client.

Mail Management

Mail management within the POP3 server mail store is accomplished by the same features that are included in the standard MUSIC system. Sites can tailor the retention time of mail in the store for each user, group of users, or class of users. An automatic mail cleanup job can be set up to delete the expired mail from the system on a regular basis.

POP3 Clients

The POP3 server has been tested with WinPMAIL 2.01 (freeware), Eudora 2.0.3 for Windows (commercial software), Eudora 1.4 (freeware), and Spry's Air for Windows AirMail v3.0 (commercial software) and higher level versions working with the Novell LanWorkplace Winsock and Microsoft's Windows 95 TCPIP support. It has also been tested with Microsoft/Exchange Windows 95 Version 4 Internet mail option. The POPPASS server has been tested with Eudora 2.0.3 for Windows (commercial software), and Eudora 1.4 (freeware) and higher level versions, and WinPMail v2.54 (freeware) working with the Novell LanWorkplace Winsock and Microsoft's Windows 95 TCP/IP support.

The MUSIC/SP Ph server can be used for those mail clients such as Eudora 3.0 (32) for Windows (commercial software) and WinPMail v2.54 (freeware) that support Ph database queries. This version of Eudora supports this feature via its Directory Services option. WinPMail supports this feature via its Extentions option. See the file $TCP:PH.DOC for further details on the MUSIC/SP Ph server.

Latest POP3 Documents

To see the POP3 server documents coming from MUSIC or for details of how to obtain the free POP3 clients, just point your Internet Web browser to:

```
http://MUSICM.McGill.CA/msi/http/pop.html
```

Trademarks

POP3 Server Installation

As the POP3 specifications evolve, updates to the POP3 server are made available by anonymous FTP to MUSICM.MCGILL.CA. The following notes outline install procedures:

- If you have not installed the MUSIC TCP/IP support yet then you must do so first. Refer to the section "The Internet Super Server (INETD)" in this manual.

- If you are going to run the POPPASS server, you should add all of the code privileges to the BTRM running $TCP:INETD. These privileges are documented in this manual under the CODUPD program. These privileges are:

  ```
  SUPV LSCAN SYSCOM CREAD INFO FILES SYSMNT CODES
  DREAD MAINT VIP JOBLIM PRIV12 PRIV13 PRIV14 PRIV15
  ```

  Since a CODXXX job is run to change the password, if the BTRM userid does not have the same userid privileges as the userid whose password the BTRM is changing, the password change will fail. This is a standard action for all CODUPD/CODXXX programs. Giving the BTRM all the privileges allows the BTRM to change the password for any userid.

- Run MAIL.CONFIG to configure $TCP:POPD and $TCP:POPPASSD, and set the program parameters to agree with your current MAIL.CONFIG settings. See the Configuration Notes below for further details on the namelist parameters.

- If you have logging on for your MAIL facility in MAIL.CONFIG and for TCPIP, you should add the following entries to the system log server BTRM define file, $PGM:SYSLG.DEFINE:

  ```
  MAIL  TO MAIL1
  TCPIP TO TCPIP1
  BADPW TO BADPW
  MAIL1:  FILE($EMD:@MAILLOG) SP(100) SECSP(100%) DAILY SHARE
  TCPIP1: FILE($TCP:TCPIP.) SP(100) SECSP(100%) MONTHLY SHARE
  BADPW: FILE($000:BADPWLOG) SP(50) SECSP(100%) MONTHLY SHARE
  ```

  The POP3 and POPPASS servers both log failed signon attempts because of a bad password or userid to the application name BADPW. This is provided for the site to monitor attempts to break into the system. You can turn off logging (LOG=F) if this is not important for your site.

- Edit $TCP:INETD.PORTS and add lines at the bottom that reads:

  ```
  106  stream  10 $tcp:poppassd
  110  stream  10 $tcp:popd
  ```

  These definitions use the standard ports 106 and 110 for the POP3 and POPPASS servers respectively. These definitions allow 10 clients at one time for each server.

- Either re-ipl your MUSIC system or issue a /RESET on the BTRM that runs INETD. (You will see this number on the console messages at startup time and on the console messages associated with each GOPHER or FTP connection.)

Usage Notes

- The POP concept is limited in that you download your mail to another machine where it may be stored and delete the mail from the server. If you access a number of machines to read your mail, for example at the office and at home, the mail is not available in both environments. POP3 clients work around this

functionality issue with a feature that allows you to read your mail from any number of machines, but then the mail must not be deleted from the server. This issue puts the reliance on the system to manage the mail on the server.

Most POP3 clients allow the user to set an option that will leave the mail on the server, and not delete it once it is downloaded. For example, Eudora's option "Leave mail on server" is found in the Special item under Switches. Note that once mail is read on the server, the client usually decides not to download "old" mail. It only downloads new (ie unread) mail.

MUSIC/SP offers an ideal solution to the "mail must be deleted from the server" issue. With the combination of default expiry dates for user mail, the POP3 server's expired mail removal feature, and the mail cleanup program, the mail is managed on the server, MUSIC/SP.

- The MUSIC/SP POP3 server supports the RFC1460 POP3 LAST command between sessions. This is a more efficient way for the client to locate the first unread message in the mailbox on the server. This means that any mail previously read will not be downloaded to the client when the client has been told to use the POP LAST command. For example, Eudora supports the LAST command with the UsePOPLAST=1 entry in the [Switches] section of the file EUDORA.INI.

- The MUSIC/SP POP3 server uses the non-standard RFC822 header Status: R to flag old mail. Some POP3 clients use this information and do not download the "old" mail, only new mail. The POP3 server works hand in hand with these clients.

- The MUSIC/SP POP3 server creates RFC822 headers for SENDFILE items before they are downloaded to the client. By putting RFC822 headers on the SENDFILE mail item and thus "converting" it to a regular mail item, clients can now display the item's subject, sender, and send date.

- Your client software may support the send operation by connecting either to the defined SMTP server and using the SMTP protocol, or by using the POP3 server using the POP3 XTND XMIT feature. The commercial version of the Eudora mail client supports the POP3 XTND XMIT feature. For Eudora, under the tools options menu, sending mail category, the field "SMTP server" determines whether mail is sent via SMTP or POP3 XTND XMIT. Defining the SMTP server address sends mail via SMTP. When this address is left blank, Eudora sends the mail via POP3 XTND XMIT. Earlier versions of the commercial version of Eudora supported XTND XMIT when the line UsePOPSend=1 was found in the file EUDORA.INI in the [Switches] section. UsePOPSend=0 sent mail via SMTP. XTND XMIT mail sending is the faster method.

- Your site may have configured the POP3 server to restrict your access to your mailbox via your pop mail client. If this is the case, you may see the message "MAILBOX access denied, retry in x minutes" if your pop mail client reflects messages back to you. When you get this message, wait the indicated minutes before attempting to reconnect to the POP3 server.

Configuration Notes

The Mail facility configuration program $EML:MAIL.CONFIG can be used to set the common namelist parameters for the POPD and POPPASS servers, and generate the execs for the servers. Descriptions of the various namelist parameters are given below.

$TCP:POPD namelist parameters

The namelist parameters for $TCP:POPD are MUSNOD, ALIASn, N$xx, ZONE, LOG, CONSOL, ACCESS, ERRFIL, PCODE, PSTMST, DEADX, SNOOP, CODTBL, DBCS, EXPIRE, PORT, BUFSIZ, DROP, TRACE, APOP, POPFRQ, and REG. All of these parameters except TRACE are the same parameters as defined in $EML:MAIL.CONFIG for the Mail facility. The definitions for the important namelist

parameters PORT, BUFSIZ, DROP, TRACE, APOP, POPFRQ, and REG are given in $EML:MAIL.CONFIG and below.

PORT=nnn Where nnn is the port number to run this server on. PORT=110 is the default. 110 is the standard port for POP3. If run interactively, the default is PORT=3110.

BUFSIZ=n Where n is the buffer space (bytes) used in TCP/IP requests. This program reserves this many bytes in the system buffer pool for itself until it terminates. Thus, if you make this buffer size too large, it may impact the system by reducing the amount of space available in the system buffer pool for the rest of the system. The minimum size for n is 50, and its maximum size is 31744. The default is BUFSIZ=31744.

DROP=n Where n is the timeout period on the socket in minutes for reads and writes. DROP=10 (ten minutes) is the default and minimum value.

TRACE=t or f TRACE=t can be used to debug a problem. It traces all I/O for the connection. All socket reads and writes are written using the socket routine TRSOCK which writes to files @TCPIP.LOG and @TCPIP.BUFFERS, as well as writing the data to unit 6. Also, all program errors are written to unit 6. The default is TRACE=f.

APOP=t or f The APOP=t parameter can be used to force the POP3 server to accept only APOP authentication from any client. APOP=f allows any authentication method (ie USER/PASS and APOP). The default is APOP=f.

POPFRQ=n Where n defines the frequency in minutes a user can access the POPD server within a one day period. POPFRQ=10 (once every 10 minutes) is the default. POPFRQ must be between 0 (no limit) and 1440 (1440 minutes). POPFRQ=0 when it is set a value outside of this range. This parameter sets the system default value for all users. It can be overridden by a setting for POPFREQ in the Mail Authorization Table type 2 record for a user or group of users.

REG=n Where n defines the region size to use for the POPD server. The default is REG=600 (i.e. 600K) which allows for about 790 mail items to be stored in core. Each additional 100K allows for about 750 additional mail items to be stored in core. For example, REG=1024 allows for about 3190 mail items to be stored in core.

$TCP:POPPASSD namelist parameters

The namelist parameters for $TCP:POPPASSD are MUSNOD, ALIASn, N$xx, LOG, CONSOL, ACCESS, PORT, BUFSIZ, DROP,and TRACE. All of these parameters except PORT, BUFSIZ, DROP, and TRACE are the same parameters as define in $EML:MAIL.CONFIG for the Mail facility. The definitions for the exceptions are given below.

PORT=nnn Where nnn is the port number to run this server on. PORT=106 is the default. 106 is the standard port for EPASS.

BUFSIZ=n Where n is the buffer space (bytes) used in TCP/IP requests. This program reserves this many bytes in the system buffer pool for itself until it terminates. Thus, if you make this buffer size too large, it may impact the system by reducing the amount of space available in the system buffer pool for the rest of the system. The minimum size for n is 50, and its maximum size is 31744. The default is BUFSIZ=80.

DROP=n Where n is the timeout period on the socket in minutes for reads and writes. DROP=10 (ten minutes) is the default and minimum value.

TRACE=t or f TRACE=t can be used to debug a problem. It traces all I/O for the connection. All socket reads and writes are written using the socket routine TRSOCK which writes to files @TCPIP.LOG and @TCPIP.BUFFERS, as well as writing the data to unit 6. Also, all program errors are written to unit 6. The default is TRACE=f.

Deleting mail from the server

Some POP3 clients can be set up to leave mail on the server after it has been downloaded instead of deleting it from the server. The site may have a Mail policy that does not allow users to leave their mail on the server. The POP Mail Delete option in the Mail Profile facility, under General Mail Options can be set to Y to force mail to be deleted from the server after it is downloaded no matter what the client has requested. ADMIN 4 5 3 "Update site Mail Profile" can be used to set this option, and new mailboxes automatically get the set option. See the Mail Profile facility for further details on the POP Mail Delete option.

Logging Features

The namelist parameter LOG controls the logging features within the POP3 and POPPASS servers. If LOG=T, logging is done via the system log server BTRM. This logging information can be used to determine who is using these facilities, how much data is being transferred, and bad connection attempts. This data can help a site with mail management and capacity planning.

Both POP3 and POPPASS clients provide a userid and password to the respective server for the connection process. If the client enters an incorrect password for the userid, a log entry is written to the log server application name BADPW. This information can also show you who is trying to use the connection process to guess userids and passwords.

The POP3 server logs connection established and closed messages, sender verification failures for XTND XMIT support, and all other errors such as socket errors to the log server application name TCPIP. The connection closed entry also contains a byte count for all the mail downloaded to the client. A separate log message is written to the log server application names TCPIP and MAIL for each successful upload of mail for delivery as done by the XTND XMIT support.

POP3 server XTND XMIT

The POP3 server performs the XTND XMIT support by placing the incoming mail item into the reader queue, and letting RDMAILER deliver the item to the intended recipients. The reader queue information that RDMAILER uses to deliver the mail is hardcoded in the POP3 server. When RDMAILER reads the reader queue entry, it will produce a log server/console message like the following:

```
22:25 M405   7 *22.25.05 Mail received from POP3 at MUSIC to MUSIC
```

The POP3 server XTND XMIT support puts a Received: header in the uploaded mail item for traceability purposes. Also, if the POP3 client did not put a Date: header in the uploaded mail item, the MUSIC POP3 server will add a RFC822 Date: header.

$TCP:POPPASSD and passwords in the clear

The POPPASS server listens for incoming requests to change the password for a POP user. The client passes the required information (user name, old password, new password) to the server. After verifying the username and old password, the MUSIC POPPASS server runs a CODXXX5 job, the POPPASS helper program $TCP:POPPASSDH, to change the password.

This feature transmits unencrypted passwords over the network. The use of a dedicated port makes it easier for a network snooper to trap passwords off the wire. You may want to consider whether to implement the POPPASS server.

The commercial version of Eudora supports the POP3 XTND XMIT feature. For Eudora, defining the SMTP server address sends mail via SMTP. When this address is left blank, Eudora sends the mail via POP3 XTND XMIT.

Delays and Timeouts

The POP3 and POPPASS servers have a hardcoded delay of one second to establish a connection. This is done so that people using the connection process to guess userids and passwords will not think a delay is a bad password feature. This slowdown in the connection attempt is a first level deterrent for breakins.

The POP3 and POPPASS servers have a hardcoded delay of three seconds when either server receives a bad password for the user connection. This slowdown in the connection attempt is a second level deterrent for breakins.

The default timeout value on socket operations both the POP3 and POPPASS servers see is controlled by the namelist parameter DROP. The default DROP=10 allows a 10 minute timeout before the connection is closed. DROP=10 is also the minimum timeout value allowed by RFC1939.

Limiting How Often Users Can Connect to the POP3 Server

Often students configure their POP3 mail client to check their mail every minute. This is generally unnecessary and puts an added load on the POP3 server. Usually, mali can be checked every 10 minutes. The MUSIC/SP POP3 server POPFRQ namelist parameter can be used to define the frequency in minutes a user can access the POP3 server within a one day period. For example, if POPFRQ=10, a user is allowed access to their mailbox once every 10 minutes. If POPFRQ=0, users can access their mailboxes as often as they desire.

Tracing Features

There are a number of features which can be used to help determine problems within the POP3 and POPPASS servers. Each server supports TRACE=T creating unit 6 output. The POP3 server also supports an alternate port assignment and it does MUSIC file I/O which can be trapped. All of these may be useful.

The namelist parameter TRACE when set to true will dump all read/write information from the socket and to any file to unit 6. Normally, unit 6 is not defined, thus output to unit 6 goes to the screen. The MUSIC /REC command may be useful in capturing this data. However, if output is lengthy, you may have to define a file to capture unit 6 output. This can be done by adding a /FILE 6 statement to the version of POPD you run. Example:

```
/FILE 6 N(*USR:POPD.TRACE) NEW(REPL) RECFM(VC) SP(500)
```

The namelist parameter PORT=110 sets the assigned port (110) for the POP3 server. While debugging the server, you could run it interactively. But you should set the PORT to a number different from 110, say 3110, so as not to interfere with the production version of your POP3 server.

The POP3 server also does MUSIC file system I/O in the XTND XMIT support. This type of I/O can be trapped by adding a /FILE statement to the version of POPD you run. Example:

```
/FILE MFTRACE N(POPD.MFTRACE) NEW(REPL) RECFM(VC) SP(500)
```

The Phone Book Server (Ph)

The CCSO Nameserver (Ph) architecture is a client/server information model. The client sends commands to the server and the server sends responses to the client.

The Ph database is a telephone book containing local information about people or things. A client query would have the server return information stored in the database matching the query.

The Ph database is used to store a relatively small amount of information about a large number of people or things. Named fields are used in the database to associate a particular piece of information with an entry in the database.

The Ph specification was originally developed at the Computing and Communications Services Office (CCSO) at the University of Illinois at Urbana-Champaign. The Internet community has decided to publish a definition of the Ph specification to benefit the community. A number of servers and clients are available on various computing platforms.

Ph clients or a mail client containing Ph capabilities like the Eudora mail client provide easy access to local information such as email addresses or telephone numbers. Ph clients are available for a variety of platforms.

The Ph server is also be referred to as the Qi (Query Information) server.

Ph Server

The MUSIC/SP Ph server is a TCP/IP server that allows a user using a Ph client or mail client containing Ph client capabilities to query the Ph database for local information. After fetching local email addresses from the database, these addresses can be used when sending mail. The Ph database offers a centralized place to store the local information for people and things.

Ph RFCs

The MUSIC/SP Ph server complies with the Internet Draft draft-ietf-ids-ph-03.txt which can be found at ftp://ds.internic.net/internet-drafts/draft-ietf-ids-ph-03.txt

Ph Database and its Management

The MUSIC/SP Ph database cannot be updated using Ph server commands. The MUSIC/SP program $EML:DIRECT.PUBLIC is used for this purpose.

The public nicknames file on MUSIC/SP, $EML:@NAMES, is used as the default Ph Database.

Ph Clients

The Ph server has been tested with Eudora v3.0.1 for Windows (commercial software) and Microsoft's Windows 95 TCP/IP support.

Latest Ph Documents

To see the Ph server documents coming from MUSIC/SP or for details of how to obtain the free Ph clients, just point your Internet web browser to our site by using the pointer:

```
http://MUSICM.McGill.CA/msi/http/ph.html
```

Trademarks

Eudora is a registered trademark of the University of Illinois Board of Trustees, licensed to QUALCOMM Incorporated. QUALCOMM is a registered trademark and registered service mark of QUALCOMM Incorporated. All other trademarks and service marks are the property of their respective owners.

Ph Server Installation

As the Ph specification evolves, updates to the Ph server are made available by anonymous FTP to MUSICM.McGill.CA. The following notes outline install procedures:

- If you have not installed the MUSIC TCP/IP support yet then you must do so first. Refer to the section "The Internet Super Server (INETD)" in this manual.

- Run MAIL.CONFIG to configure the Ph server program, $TCP:PH. See the Configuration Notes below for further details on the namelist parameters used by the Ph server.

- If you have logging on for TCPIP, you should add the following entries to the system log server BTRM define file, $PGM:SYSLG.DEFINE:

  ```
  TCPIP TO TCPIP1
  TCPIP1: FILE($TCP:TCPIP.) SP(100) SECSP(100%) MONTHLY SHARE
  ```

  You can turn off logging (LOG=F) if this is not important for your site. See the logging features in the Configuration Notes below for further details.

- If you want to limit full printable information to only the local community, create the file $TCP:@PH.LCLIP to define the local community. Copy the model file $TCP:PH.LCLIP.DUM to the file $TCP:@PH.LCLIP. Then edit the file $TCP:@PH.LCLIP and follow the instructions at the top of the file to define your local community. If this file does not exist, any query from any user not specifying return data will get all printable information for the matches to the query. If a local community is defined, any non-local queries not specifying return data will get just the alias, name, and email address fields returned.

- * Edit $TCP:INETD.PORTS and add a line at the bottom that reads:

  ```
  105  stream  10 $tcp:ph
  ```

  This definition uses the standard port 105 for the Ph server. This definition allows 10 clients at one time for the server.

- Edit $TCP:TCPIP.CONFIG and add port 105 to the end of the line starting with "RESTRICT_TELNET_PORTS". Ports specified on this line are separated by one blank. This stops MUSIC/SP users from telnetting directly to port 105.

- A change to the VM/SP TCP/IP configuration file PROFILE TCPIP may be required to allow access to port 105. Details of this change are beyond the scope of this document.

- Either re-ipl your MUSIC/SP system or issue a /RESET on the BTRM that runs INETD. (You will see this number on the console messages at startup time and on the console messages associated with each GOPHER or FTP connection.)

- If you want to make available to your users an updated list of the Ph servers around the world, add an AUTOSUB job to run $TCP:PH.SERVERS on a regular basis to build the information required for this query.

Usage Notes

- The Ph phone book database is a centralized database for the site to store information about the local environment such as email addresses.

- The Ph database can be queried by users and the Ph server returns information matching the query to the user. For example, you are sending mail from your mail client on your workstation and you cannot remember the address of the intended recipient. You simply query the Ph server and have it return the address to you.

  Matching is not sensitive to upper or lower case letters and is normally done on a word by word basis. So, both the client query expression and the Ph database entry information are broken up into words, and the individual words are compared using exact matching. If the order of the words is important in a query, then the query string can be surrounded by "", whereby the complete query string is matched against information in the Ph database.

  The wild card character "*" within the query string can be used to represent zero or more characters. The wild card character "?" within the query string can be used to represent one or more characters.

  Examples of query expressions:

```
query name=Smith*  ;returns all matches with Smith* somewhere
                   ;in the name field
query alias=help   ;returns all matches with help somewhere in the
                   ;alias (i.e. nickname) field
Smith*             ;returns all matches with Smith* somewhere in
                   ;either the alias or name fields
"Donald Duck"      ;returns all matches to Donald Duck somewhere in
                   ;either the alias or name fields, where Donald
                   ;must immediately precede Duck in the field
```

- The MUSIC/SP Ph server does not allow the wild card character "*" in any query field. When the server is queried in this way, it returns the error message "Did not understand your query" to the user. The user must be more specific on the query. The server does not allow the entire Ph database to be downloaded in this fashion.

- The MUSIC/SP Ph server limits the number of matches returned to a query. If this limit is exceeded by a query, the server returns the error message "Too many entries to print" and does not return any other data. The user must be more specific on the query.

- The MUSIC/SP Ph server can be defined to qualify who is a local user. The Ph server returns only the alias, name, and email address fields for each entry in the Ph database matching the non-local user's query if the query does not specify any return fields. The Ph server returns all information pertaining to the matches when a local user's query does not specify any return fields.

- The MUSIC/SP Ph server does not support the commands constituting the update features of the server. The following commands are supported:

```
status   - returns status of the database
siteinfo - returns info about the server site
fields   - returns the fields available in the database
id       - enters the given info in the log
set      - sets an option for this session
query    - requests a query with the given info
ph       - equivalent to the query command
quit     - disconnect from the Ph server
exit     - equivalent to the quit command
stop     - equivalent to the quit command
help     - returns help on the Ph server
```

- The MUSIC/SP Ph server "help" command has the following syntax:

  ```
  help [native] topic
  ```

  The word native is optional, and if it is not specified, the Ph topic is displayed. If that topic is not available, the "native" topic is displayed. If a topic is not specified, the Ph or native general help is displayed depending on whether native was given. The help text is stored on MUSIC/SP in files $TCP:PH.P* and $TCP:N*, where P and N represent Ph and native respectively, and "*" represents the topic name.

- Each field in the Ph database has an associated field descriptor defined in the file defined by the Ph server PHFLDS namelist parameter. A field descriptor includes the fieldname, the maximum length of the field, keywords describing the properties of the field, and a description of the field. The MUSIC/SP Ph server supports a subset of the keyword properties. See the topic "Adding Fields to the Ph Database" below for a list of these properties.

Configuration Notes

$TCP:PH Namelist Parameters

The namelist parameters for $TCP:PH are MUSNOD, ALIASn, N$xx, LOG, CONSOL, ACCESS, ERRFIL, PORT, BUFSIZ, DROP, TRACE, PHADMN, PHPSWD, PHFILE, HLIMIT, LCLIP, PHFLDS, and PHSRVS. The parameters MUSNOD, ALIASn, N$xx, LOG, CONSOL, ACCESS, and ERRFIL are the same parameters as defined in $EML:MAIL.CONFIG for the Mail facility. The definitions for the exceptions are given below.

PORT=nnn	Where *nnn* is the port number to run this server on. PORT=105 is the default. 105 is the standard port for Ph. If run interactively, the default is PORT=3105.
BUFSIZ=n	Where *n* is the buffer space (bytes) used in TCP/IP requests. This program reserves this many bytes in the system buffer pool for itself until it terminates. Thus, if you make this buffer size too large, it may impact the system by reducing the amount of space available in the system buffer pool for the rest of the system. The minimum size for n is 50, and its maximum size is 31744. The default is BUFSIZ=4096.
DROP=n	Where *n* is the timeout period on the socket in minutes for reads and writes. DROP=10 (ten minutes) is the default and minimum value.
TRACE=T or F	TRACE=T can be used to debug a problem. It traces all I/O for the connection. All socket reads and writes are written using the socket routine TRSOCK which writes to files @TCPIP.LOG and @TCPIP.BUFFERS, as well as writing the data to unit 6. Also, all program errors are written to unit 6. The default is TRACE=F.

PHADMN=administrator_email_address

Where *administrator_email_address* is the email address of the system administrator. The Ph server command "siteinfo" returns information to the client about the server site. One of the fields returned is the system administrator's email address. The default is PHADMN='postmaster@MUSNOD', where MUSNOD is the MUSNOD namelist parameter.

PHPSWD=password_administrator_email_address
: Where *password_administrator_email_address* is the email address of the system password administrator. The Ph server command "siteinfo" returns information to the client about the server site. One of the fields returned is the system password administrator's email address. The default is PHPSWD='postmaster@MUSNOD', where MUSNOD is the MUSNOD namelist parameter.

PHFILE=phone_book_filename
: Where *phone_book_filename* is the filename of the database used for the phone book. The default is PHFILE='$EML:@NAMES'.

HLIMIT=n
: Where *n* is the maximum number of hits returned for each query. HLIMIT=25 is the default. HLIMIT=1 is the minimum value. HLIMIT=2147483647 is the maximum value. HLIMIT is set to 25 when HLIMIT is set to anything outside of this range. HLIMIT is meant as a partial deterrent from users trying to create a mail list by collecting all of the entries in the Ph database. HLIMIT=25 is the accepted value within the Internet community. See the topic Security Considerations below before changing this value.

LCLIP='filename'
: Where *filename* is the name of a list of IP addresses that represent the local domain. The default is LCLIP='$TCP:@PH.LCLIP'. The file contains one entry per line, and each entry can be a component of the full IP number (ie a network number). Blank lines or lines starting with a semicolon are considered comment lines. Any information on a line after a semicolon is considered a comment.

  Example:

```
132.206.120    ;defines all stations on 132.206.120 net
                as local
132.206.130.1  ;defines only 1 ip address on 130 net as
local
```

PHFLDS='filename'
: Where *filename* is a file whose contents defines the fields of the PH database. The default is PHFLDS='$TCP:PH.FIELDS'. Comments at the start of the file give instructions to add to file. Comment lines within the file start with a semi-colon in column 1. Do not remove the fields alias,email,name,phone,address,list. Add new fields following the list. When adding a new field, add 2 lines for each new field. First line format: field:max maximum_length keyword_properties Second line format: field:description_of_field There is no format checking so if the information is not typed in correctly, the Ph database may not be accessed correctly. The format can be verified with the PH client FIELDS command. The default type=person is used when the Ph database entry has not type. A new field name can be a maximum of 40 characters long. Each line can be up to 80 characters long.

PHSRVS='filename'
: Where filename is a file whose contents contains the Ph server response to a "query ns-servers" command as it would be responded to by the Ph server at ns.uiuc.edu (i.e. the mothership). The default is PHSRVS='$TCP:@PH.SERVERS'. Since the information contained in this file is not stored locally in the Ph database, a REXX exec, $TCP:PH.SERVERS, gets the data from the mothership and massages it as is needed so that it can be directly read and written to the client via the socket. This

exec could be scheduled as an AUTOSUB job to be run regularly.

Note: If you change this filename, you must edit the file $TCP:PH.SERVERS and change the variable file1 to agree with the filename entered here.

Logging Features

The namelist parameter LOG controls the logging features within the Ph server. If LOG=T, logging is done via the system log server BTRM. This logging information can be used to determine who is using these facilities, how much data is being transferred, and bad connection attempts. Since the phone book is typically used by mail clients to get local email addresses, the logging data can help a site with mail management and capacity planning. If LOG=F, logging is not done.

The Ph server logs established and disconnected connection messages, and all other errors such as socket errors to the log server application name TCPIP. The disconnected connection message also contains a byte count for all the bytes downloaded to the client.

Mail Client Usage

The Eudora mail client supports the use of the Ph server. Typically, the function is found in the Eudora client under the menus following Tools, Options, Directory Services or something similar. This feature brings up a window into which an input field for the Ph server and Ph request can be entered. Note that the Ph request field does not need to be prefixed the query or ph keyword, or with the field search name if the field is either "alias" or "name". For example:

```
Command => Smith
```

This query returns all entries in the phone book matching Smith either in the alias (i.e. nickname field) or anywhere in the names field.

The Eudora mail client supports the querying of the Ph server for a list of Ph servers. Typically, the "Server" button is found in the Eudora client under the menus following Tools, Options, Directory Services or something similar. The MUSIC/SP Ph server supports this feature.

MUSIC/SP Ph Server Implementation Specifics

The MUSIC/SP Ph server does not support the commands to update the Ph database. The MUSIC/SP program $EML:DIRECT.PUBLIC is used for this purpose.

Similarly, when the Ph server receives the "siteinfo" command from the client, the server returns the PHADMN and PHPSWD fields as set up in the MUSIC/SP configuration of the Ph server. These fields give the user contact information for the system administrator and system password administrator.

Fields within the Ph database are defined by the PH namelist parameter PHFLDS. The default is PHFLDS='$TCP:PH.FIELDS'. It is up to the site to get the extra fields into the Ph database and maintain these fields. See the description for the PHFLDS namelist parameter above under the topic "$TCP:PH namelist parameter" for further information. Also, the file $TCP:PH.FIELDS contains instructions on how to add fields to the Ph database. The topic "Adding Fields to the Ph Database" below provides further information. The keyword properties associated with each field are defined in the Ph specification as drafted by the IETF which can be found in file $TCP:PH.SPEC.

Only one field in the MUSIC/SP Ph database can possess the INDEXED keyword property. By default, the alias (i.e. nickname) field is given this property. It is up to the site administrator to keep the Ph database in INDEXED order (i.e. ascending, alphanumeric order). By default the Ph database is arranged such that the alias field is the first field in the database, so the command "SORT *" in $EML:DIRECT.PUBLIC can be used for this purpose. It is the site administrator's responsibility to maintain the Ph database in INDEXED

order for whatever field is given the INDEXED property.

The MUSIC/SP Ph server does not allow the wild card character "*" in any query field. When the server is queried in this way, it returns the error message "Did not understand your query" to the user. The user must be more specific on the query. The server does not allow the entire Ph database to be downloaded in this fashion. See the topic Security Considerations below for information on this feature.

The MUSIC/SP Ph server limits the number of matches returned to a query. If this limit is exceeded by a query, the server returns the error message "Too many entries to print" and does not return any other data. The user must be more specific on the query. See the topic Security Considerations below for information on this feature.

The MUSIC/SP Ph server can be defined to qualify who is a local user. The Ph server returns only the alias, name, and email address fields for each entry in the Ph database matching the non-local user's query if the query does not specify any return fields. The Ph server returns all information pertaining to the matches when a local user's query does not specify any return fields. See the topic Ph Server Installation above for details on how to define the file $TCP:@PH.LCLIP. This file uses IP numbers to define local users. If this file exists, it is assumed that a local environment has been defined and any user not listed here is deemed a non-local user. If the file does not exist, all users are deemed local.

The MUSIC/SP Ph server supports a subset of the keyword properties. See the topic "Adding Fields to the Ph Database" below for a list of these properties.

The MUSIC/SP Ph server supports the querying for a list of Ph servers. This function is not supported through the Ph database as is done traditionally. Instead, the REXX exec $TCP:PH.SERVERS is used to get the data for the query from the main Ph server, ns.uiuc.edu, massage it, and store it in the file $TCP:@PH.SERVERS. When a client queries the MUSIC/SP Ph server for a list of Ph servers, the file $TCP:@PH.SERVERS is given to the client as the response. You may want to set up an AUTOSUB job to run $TCP:PH.SERVERS on a regular basis to keep the Ph servers list up to date. This feature is supported through the PHSRVS namelist parameter.

Delays and Timeouts

The default timeout value on socket operations for the Ph server is controlled by the namelist parameter DROP. The default DROP=10 allows a 10 minute timeout before the connection is closed.

Security Considerations

All data within the Ph database as defined on MUSIC/SP is public with respect to a Ph client. Privacy issues may need to be addressed from a local standpoint.

The MUSIC/SP Ph server does not allow the wild card character "*" in any query field. When the server is queried in this way, it returns the error message "Did not understand your query" to the user. The user must be more specific on the query. The server does not allow the entire Ph database to be downloaded in this fashion. Using "*" in any query field is a convenient way for someone to build a mail list of the entire Ph database for the purposes of sending unsolicited mail. This feature makes it more work for a user to create a mail list from the Ph database.

The Ph database can be queried by any Internet user. Use of the file $TCP:@PH.LCLIP can define what information is returned on matches for non-local users. This file uses IP numbers to define local users. The topic Ph Server Installation above gives the details of how to install this feature. Essentially, with a local user, all data on a query match is returned to the user. With a non-local user, only the alias, name, and email address fields are returned to the query when the query does not specify any return fields. This feature provides limited information to non-local queries.

The MUSIC/SP Ph server namelist parameter HLIMIT is used to limit the number of hits returned to a query.

The default is HLIMIT=25. Although HLIMIT can be set to a value between 1 and 2147483647, HLIMIT=25 is the accepted value by other Ph servers on the Internet. This value is a balance between returning too few matches and returning the entire Ph database to a query. Setting HLIMIT to a high value means that any user creating a mail list from the Ph database would need fewer queries to create the list.

When the number of matches to a query exceeds HLIMIT, the MUSIC/SP Ph server returns the error message "Too many entries to print" and does not return any other data. The user must be more specific on the query.

Adding Fields to the Ph Database

Fields within the Ph database are defined by the PH namelist parameter PHFLDS. The default is PHFLDS='$TCP:PH.FIELDS'. It is up to the site to get the extra fields into the Ph database and maintain these fields. See the description for the PHFLDS namelist parameter above under the topic "$TCP:PH namelist parameter" for further information. Also, the file $TCP:PH.FIELDS contains instructions on how to add fields to the Ph database. The keyword properties associated with each field are defined in the Ph specification as drafted by the IETF which can be found in file $TCP:PH.SPEC.

Each field in the Ph database has an associated field descriptor defined in the file defined by the Ph server PHFLDS namelist parameter. A field descriptor includes the fieldname, the maximum length of the field, keywords describing the properties of the field, and a description of the field. The MUSIC/SP Ph server supports the following subset of the keyword properties:

Always: Always printed in addition to whatever fields specified by the query

Any: Always searched by queries

Default: Printed if no return clause is given in the query

Indexed: Indexed field to allow searches on these fields especially efficient

LocalPub: May be viewed by anyone in the "local" domain space

Lookup: May be used in the selection part of a query

NoMeta: Wild card searches are disallowed

NoPeople: No entry of type "person" may include this field

Public: May be viewed by anyone

The Ph database is defined by the $TCP:PH namelist parameter PHFILE. The default is PHFILE='$EML:@NAMES'. The Ph database contents are tag based. The following is a sample Ph database entry:

```
:nick.ABC :emailid.prez@abc.com :name.ABC Company Inc
```

The fields are nick (i.e. alias), emailid, and name. These are default tags and must not be removed from the system. If a new field "fax" was desired, our sample entry might look like the following:

```
:nick.ABC :emailid.prez@abc.com :name.ABC Company Inc :fax.123-456-7890
```

After adding the "fax" field to those entries in the Ph database that required this field, the next step would be to define the "fax" field in the PHFLDS file. The following two lines would be added to the end of the PHFLDS file:

```
fax:max 40 Lookup Public Default
fax:Defines a fax number
```

Verify the changes by using the "fields" command of a Ph client to make sure the new "fax" field appears in the list of fields available. Now, the "fax" field is available for display or queries from Ph clients.

You can also test the changes by telnetting to Ph port, the default is 105, and typing "fields" to see the fields of the Ph database. Type "quit" to disconnect from the Ph server. You may be required to do the telnet from a system different from MUSIC if you configured the Ph server to not allow MUSIC users to telnet to port 105 as the installation notes suggest.

Tracing Features

There are a number of features which can be used to help determine problems within the Ph server. The server supports TRACE=T creating unit 6 output. The Ph server also supports an alternate port assignment. All of these may be useful.

The namelist parameter TRACE when set to true will dump all read/write information from the socket to unit 6. Normally, unit 6 is not defined, thus unit 6 output goes to the screen. The MUSIC/SP /REC command may be useful in capturing this data. However, if output is lengthy, you may have to define a file to capture unit 6 output. This can be done by adding a /FILE 6 statement to the Ph server executor you run, normally $TCP:PH. Example:

```
/FILE 6 N(*USR:PH.TRACE) NEW(REPL) RECFM(VC) SP(500)
```

The namelist parameter PORT=105 sets the assigned port (105) for the Ph server. While debugging the server, you could run it interactively. But you should set the PORT to a number different from 105, say 3105, so as not to interfere with the production version of your Ph server.

The Web Server (HTTPD)

This server handles HTML documents accessible through your Web browser. MUSIC's line-mode Web browser and information about how to create Web documents can be found in the *MUSIC/SP Internet Guide*.

Tailoring and Changing Defaults

To change the default file matching or home page locations alter the following lines in $TCP:TCPIP.CONFIG:

```
HTTPD_ALLOWED_FILES *:*HTTP\*
HTTPD_ALLOW_PUBLIC no
HTTPD_ALLOWED_EXECS *:*HTTPEXEC\*
HTTPD_DEFAULT_PAGE $000:HTTP\HOME.HTML
```

HTTPD_ALLOWED_FILES — This entry defines the MASK for any file allowed to be accessed via HTTPD. For example, to allow the subdirectory HTTPD on userids that begin with dollar to be accessible, you specify $*:HTTPD* The default is *:*HTTP*

HTTPD_ALLOW_PUBLIC — This entry specifies whether public files not fitting the HTTP_ALLOWED_FILES mask, should be accessible. The default is no.

HTTPD_ALLOWED_EXECS This entry defines the MASK for any file allowed to be executed via HTTPD. This is specifically meant for forms support in html files. The default is *:*HTTPEXEC*

HTTPD_DEFAULT_PAGE This entry defines the default page to be transmitted when no page is specifically requested in the HTTPD connection. The default is $000:http\home.HTML

Creating Alternate HTTP Servers

You can define alternate http servers on MUSIC. Typically the port number used for http service is 80. Alternate servers usually use numbers in the same range as the standard. For example a second http server may use the port number 81 etc.

The file $TCP:INETD.PORTS is where all servers are declared. The format is as follows:

```
80 stream 5 $tcp:httpd  <-- standard http daemon server
81 stream 5 your_own_file <-- your own http server
```

Refer to the section "The Internet Super Server (INETD)" in this manual for more details.

Alternate HTTP Server Exec File

The following is description of how to set up an alternate http server exec file and how to specify values to tailor it.

Use $TCP:HTTPD as a template to create a file like the following:

```
/SYS NOPRINT,REGION=600,TIME=MAX
/LOAD XMON
httpD N($TCP:httpD.LMOD)
TRACE=F,PORT=N, .....
```

Namelist Parameters

PORT=n Used to override the port number specified by httpSERVER in $tcp:tcpip.config.

TRACE=F/T Used to turn on tracing. This is used for debugging only. The output is directed to @tcpip.log and to @tcpip.buffers

MYSITE='site_name'
Used to override the httpSERVER in $tcp:tcpip.config.

DPAGE='default_page_name'
Used to override the http default page in $tcp:tcpip.config.
default='$000:HTTP\HOME.HTML'

LOG=T/F logs transactions.

HTFILE='mask' Mask is 1-64 character mask for allowed files. Used to override the default mask in $tcp:tcpip.config. Default *:*http*

BUFSIZ=N bufsiz is used to control the size of the buffering done to the socket. See comments in

$tcp:tcwrit.s for more details. Default is 2000.

TIMOUT=n timout value on reads and writes to socket, a value of 0 defaults to 30secs

PUBFIL=t/f allow access to files not matching the pattern (mask), but are otherwise public files or have the SHR file attribute.

File Naming Conventions

Only files that match the mask for allowed files will be transmitted.

Data (file) types currently supported are defined in the file $TCP:MIME.SUPPORT.

Notes:

1. EBCDIC refers to files that are created and maintained on MUSIC or are in an ebcdic format.
2. ASCII refers to files that have been created in ascii and imported to MUSIC. These files can not be modified using the MUSIC EDITOR. They can be read using VIEW and the command "ascii" to provide a translated view of the file.
3. binary files are similarly imported to MUSIC but are not in a readable format.

Convention for URL's:

Rule 1

```
http://musicdns/userid|file/file/file
```

will be treated as

```
http://musicdns/userid:file/file/file
```

Rule 2

```
http://musicdns/~userid/file/file/file
```

will be treated as

```
http://musicdns/userid:file/file/file
```

Changing, Adding and Deleting MIME Types

The Software is sent out with as many known MIME and file types at packaging time. It may, however, become necessary to make changes to the list of file types. The list used by the HTTPD server is stored in the file $tcp:mime.support, and looks as follows.

```
;for type enter either bin or text
;
;type  MIME TYPES                SUFFIXES
;----  --------------------      -----------------
text   text/plain                .txt,.text,.etext
bin    text/plain                .atext, .atxt
```

```
text   text/html                 .html, .htm ITS
bin    text/html                 .ahtml, .ahtm
bin    image/gif                 .gif
bin    image/jpeg                .jpg,.jpe,.jpeg
;
bin    audio/wav                 .wav,.wave
bin    audio/au                  .au
bin    audio/x-midi              .mid
bin    image/x-tiff              .tif,.tiff
bin    video/mpeg                .mpg,.mpe,.mpeg
bin    video/quicktime           .mov
bin    video/msvideo             .avi
bin    application/postscript    .ps,.eps,.ai
bin    application/x-rtf         .wri
bin    application/octet-stream  EXE bin class
```

To make changes to the list, edit $tcp:mime.support and make the required change, deletion or addition. The syntax is

```
type  mime_type  suffixes
```

Notes:

1. For "type" only bin and text are allowed. This simply flags that the file is either EBCDIC (readable on MUSIC) or binary (not readable on MUSIC). When "bin" is chosen the file contents are sent to the client with no filtering or translations. While "text" indicates that the file contents are to be both translated to ascii and that a CRLF (carriage return/line feed) are to be appended to each line of the file as it is transmitted.

2. "mime_type" should be in lower case.

3. "suffixes" do not need commas or dots. Care should be taken that no standard suffixes are removed or corrupted. This could result in incorrect transmission to the client. Of special note, is the "ITS" suffix. This suffix is internal to MUSIC and is used in naming Indexed-Text-Searchable files.

The General Flow of HTML Forms Document Programs

1. Call to GTFORM to fetch the response string from the client. The string will have a series of name/value pairs seperated by ampersands (&). The following is a sample from the example presented later in this section:

   ```
   last name=&first name=&department=&email=<phone>=&
   month=April%2095&day=Monday&time=10%3A00-12%3A00%20am
   ```

   Note that special characters are transmitted as %xx, where xx is a hexidecimal number representing an ASCII character. The RDFOM and RDNUM routines automatically convert these for you.

2. Calls to RDINFO and either RDFORM or RDNUM will retrieve the name/value pairs. Please see the REXX sample program to help you understand the technique.

3. The program performs what ever process is necessary.

4. Calls to ADLINE are used either during or after processing to construct a response to the user.

5. Call to RSPOND is used to signal to the HTTPD server that processing is complete and that a response

is available for transmittal to the client.

Subroutines for Processing HTML Forms

Programs written to handle html forms will use some or all of the following routines (details of each routine can be found below). These routines can be used in any programming language including REXX.

GTFORM	get the client's returned string of variable names/ values from what the user filled in.
RDFORM	given a variable name return its value.
RDNUM	get the name/value strings of the ith pair.
RDINFO	tells you how many name/value pairs there are, as well as the maximum lengths of the name strings and value strings.
RDPOS	returns the positions and lengths of the name/value pairs.
ADLINE	add a line to the response to the client
RSPOND	used to tell HTTPD server that program has completed. The text passed back by adline will be transmitted to the client as a response to the user.
RDRSET	resets all pointers, variables etc.

Descriptions of HTML Subroutines

Parameter Descriptions

The following parameters are used in the subroutines below.

RC	returns the length of prmstr, if KWORD is not found, -1 is returned as the value.
KWORD	A 1 to 32 character keyword to be searched for
KLEN	length of KWORD
STRING	The string from the client with the forms response.
STRLEN	The length of string.
PRMSTR	The character string associated with KWORD.
PRMLEN	The maximum length of PRMSTR.
VALUE	Integer value of PRMSTR (where applicable)
VARNUM	ITH variable in the form
TOTVAR	total number of variable/value pairs in the form.
MXVARL	the length of the longest variable name.

MXVALL the length of the longest value string.

VARPOS position of the ith variable.

VARLN length of the ith variable

VALPOS position of the ith value string

VALLN length of the ith value string

GTFORM

This subroutine reads from the file a string returned from the client with the variable/value pairs. Calling sequence:

```
CALL GTFORM(RC,STR1,STRL1,STR2,STRL2)
```

Arguments:

RC mfio return code in case of error.

STR1 1-32760 text string to be retrieved from the file (line 1). this is in the form keyword1=text1&keyword2=text2

STRL1 MAXIMUM length of STR1

STR2 1-256 text string to be retrieved from the file (line 2) this is in no particular form.

STRL2 MAXIMUM length of STR2

RDFORM

This subroutine is used to get the value of a known variable name. Given KWORD, PRMSTR is returned as the value of KWORD. Calling Sequence:

```
CALL RDFORM(RC,STR1,STRL1,KWORD,KLEN,PRMSTR,PRMLEN,VALUE)
```

Some special characters are encoded with a preceding % (per cent). For example an astersik (*) will come to us as an encoded %2A, where 2A is the ascii hex value. We will convert "%2A" to "*".

Example:

STRING contains the following, and is of length 64
password=secret&fromemail=me@here&title=Duh..%20title&message=Text

```
CALL RDFORM(RC,'password',8,STRING,80,PRMSTR,80,VALUE)

returns: RC     6       (length of prmstr)
         PRMSTR secret  (character string associated with password)
         VALUE  0       ('secret' is not an integer)
```

Note that the keyword must be in the same case as was used in the HTML document form.

RDNUM

This subroutine is used to get the variable name and its value from the string. VARNUM is used to point to the ith pair. Calling sequence:

```
CALL RDNUM(RC,STR1,STRL1,KWORD,KLEN,PRMSTR,PRMLEN,VALUE,VARNUM)
```

RDINFO

This subroutine is used to return the number of variable/value pairs and the length of the longest variable name and the longest value string. Calling sequence:

```
CALL RDINFO(RC,STR1,STRL1,TOTVAR,MXVARL,MXVALL)
```

You must call one of RDINIT, RDFORM, or RDNUM before you call RDINFO.

RDPOS

This subroutine is used to return the character positions of the variable/ value pair pointed to by VARNUM. Calling sequence:

```
CALL RDPOS(RC,STR1,STRL1,VARNUM,VARPOS,VARLN,VALPOS,VALLN)
```

ADLINE

Adds line of text to a file. If the file is not open the file will be opened fro the caller. Calling sequence:

```
CALL ADLINE(RC,TEXT,TXTLEN)
```

Arguments:

RC mfio return code in case of error.

TEXT 1-256 CHARACTER text string to be written to the file.

TXTLEN length of TEXT.

RSPOND

This routine is used to signal the HTTPD server that processing is complete and take the text lines provided by ADLINE, and respond to the user. Calling sequence:

```
CALL RSPOND(RC,GEN)
```

Arguments:

GEN =0 means that we pass the contents added via ADLINE routine back to HTTPD
=999 means that we will use a generic "Everything is ok" file.
=888 means that we will use a generic "Something is wrong" file.

RDRSET

This subroutine is used to zero all counters and pointers into string. Calling sequence:

```
CALL RDRSET
```

The Gopher Server (GOPHERD)

GOPHERD is the name of MUSIC/SP's Gopher server. A gopher server provides access to documents residing on a host or server to a gopher client running on another computer. The gopher client may be running on a wide variety of platforms and operating systems. This section discusses how to set up documents on MUSIC/SP for access by gopher clients. (MUSIC's Gopher client is documented in the *MUSIC/SP Internet Guide*.) To know more about gopher in general please refer to the document, "The Internet Gopher Protocol" (RFC 1436).

Data and Documents on Gopher

Gopher supports several types of data or documents. These item-types are defined by the following characters:

0 Item is a file
1 Item is a directory
2 Item is a CSO phone-book server
3 Error
4 Item is a BinHexed Macintosh file.
5 Item is DOS binary archive of some sort. Client must read until the TCP connection closes. Beware.
6 Item is a UNIX uuencoded file.
7 Item is an Index-Search server.
8 Item points to a text-based telnet session.
9 Item is a binary file! Client must read until the TCP connection closes. Beware.
+ Item is a redundant server
T Item points to a text-based tn3270 session.
g Item is a GIF format graphics file.
I Item is some kind of image file. Client decides how to display.

The item-types currently supported on MUSIC are 0, 1, 7, 8, 9, T, g, and I.

Gopher Directories (menus)

A gopher server tells the a gopher client what it has available through directories transmitted to the client. Directory items transmitted to the client look as follows:

```
item-type caption|selector|site_name|port_number
```

where:

item-type defines the item (directory, document, etc.

caption is the descriptive text that the client displays to assist the user in making his selection.

selector this is a string that the client sends to the server in order to select the item.

site_name	internet address name (can be an IP address) This is the site to connect to to get this item.
port_number	This is the port number to be used to establish the connection with site_name.
|	This character represents the tab character hexadecimal '05'. MUSIC's GOPHERD automatically converts the '|' (vertical bar) to hex 05. '|' is used as a delimiter between elements of the the directory definitions. You can substitute tilda (/ xA1), exclamation mark (! x5A), not sign (• x 5F), or cent sign (x4A) for '|'.

The following is a sample gopher directory (menu) from RFC1436.

```
0About internet Gopher|Stuff:About us|rawBits.micro.umn.edu|70
1Around University of Minnesota|Z,5692,AUM|underdog.micro.umn.edu|70
1Microcomputer News & Prices|Prices/|pserver.bookstore.umn.edu|70
1Courses, Schedules, Calendars||events.ais.umn.edu|9120
1Student-Staff Directories||uinfo.ais.umn.edu|70
1Departmental Publications|Stuff:DP:|rawBits.micro.umn.edu|70
```

The client will display this as follows:

```
About Internet Gopher
Around the University of Minnesota...
Microcomputer News & Prices...
Courses, Schedules, Calendars...
Student-Staff Directories...
Departmental Publications...
```

The menu above will be displayed differently by different clients. For example some clients may number the items, while others may use icons and allow mouse support etc. In our display above '...' represents those items that are directories. Note that when the item refers to the home directory of a gopher server or gateway, there is no selector string, hence the adjacent '|' in the examples above.

Home Gopher Menu for GOPHERD

To provide a home directory (menu) for your MUSIC/SP gopher server, create or update the file "$tcp:gopherd.menu". What you place here is what gopher clients will see when they gopher to your site. Just follow the example above or consult RFC 1436 to make your own home directory.

MUSIC/SP Selector String Conventions

Gopher server/clients operate in a mode, that is sometimes referred to as "stateless and connectionless". This simply means that the client (user) is not connected to the gopher host as one might be connect via FTP or Telnet. Rather, after each request the connection between the server and client is severed. For every request a new connection is made to the site named on the item. After connecting to the gopher server the client passes it either a blank line to indicate that it wants its home directory or the selector string extracted from the directory item.

The selector string can be used by the gopher server to establish what the data is and where it is located. Although the client has been told what item-type to expect, the item- type is not returned to the server by the client. This means that the server has to know from the selector string or via some other method what the data type is.

MUSIC/SP uses several flags in the selector string for identifying the kind of request being made.

-BBS indicates that the item is a BBS text menu file
-BB1 indicates that the item is a BBS itemized menu file
-BIN item is a binary file
-GOP indicates that the item is a standard gopher directory
-GIF item is a GIF file
-IMG item is an image file
-ITS item is an INDEX TEXT SEARCH facility
-IT1 item is the pointer to the result of a text search
-PGM item is an executable MUSIC file (not supported)
-TXT item is a straight text file (can be a BBS file)

Examples:

```
7Telephone Book|-ITS/tb/|site_name|70
```

defines a gopher Indexed Text Search item called "Telephone Book" and the executor file name is TB. This is all that is required to define an ITS application for the gopher server.

```
1infoMcGill|-BBS/CW99:@inf.main.menu|site_name|70
```

defines a gopher directory item called "infoMcGill", which is a MUSIC CWIS application. Once the item is selected by the client, all other gopher menus required to display the text and CWIS menus in infoMcGill will be automatically generated by MUSIC's gopher server.

```
gCampus Map|-GIF/$ABC:MAP1|site_name|70
```

defines a gopher item-type of GIF in the music file "$ABC:MAP1".

```
1More documents|-GOP/$ABC:GOPHER.MENU2|site_name|70
```

defines a gopher item-type of directory (gopher menu) in the music file "$ABC:GOPHER.MENU2".

Note: '-BB1', '-IT1', and '-PGM' are used internally only.

Gopher Support for MUSIC/SP BBS Facilities

As was stated earlier, to point to a BBS for gopher clients you simply point to the main menu of the BBS. For example McGill's BBS (Campus-Wide Information System) called infoMcGill can be defined as follows:

```
1infoMcGill|-BBS/CW99:@inf.main.menu|musica.mcgill.ca|70
```

where:

1	item-type directory (menu)
infoMcGill	is the caption seen by the user of the gopher client
-BBS/	tells MUSIC's gopher server to treat the item as a BBS. Remember that this string is sent to the server by the client in order to request the item.
cw99:@inf.main.menu	is the name of the MUSIC file where the main menu for infoMcGill resides.
musica.mcgill.ca	address of McGill Universities MUSIC system.

70 port number

This is all that is required to establish access to the entire BBS. From this point on the directories and item-types will be automatically generated by the MUSIC gopher server. This includes automatically linking to other BBS and ITS applications if they occur as topics in the BBS.

Making Files Accessible via Gopher

The Systems Administrator can define a MASK for non BBS files allowed to be accessed via GOPHERD. For example to allow the subdirectory GOPHERD on userids that begin with dollar to be accessable you specify $*:gopher*

The default is "$pub:*". The following is an example of the parameter that is placed in the $TCP:TCPIP.CONFIG file.

```
GOPHERD_ALLOWED_FILES $PUB:*
```

Special File Flags for BBS Support

The gopher server uses several flags and clues in the content of the BBS files. Most of these flags are already used in the normal definition of BBS files. These are:

)menu the file is an itemized menu file

)Gmenu the file is a text menu file. These are menus that have highlighted text pointing to BBS topic files (see notes below).

)Gopher the file is a gopher directory (not often used) If "-GOP" flag is not used in the item definition example:

```
1More documents|-GOP/$ABC:GOPHER.MENU2|site_name|70
```

then you must place ")gopher" as the first line of the file.

)TEXTFILE the file is a straight text file. As with)gopher you can either specify the item as a text file via the "-TXT" flag or you must place ")textfile" in the document to correctly identify the item-type.

)INETACC on/off
This statement enables or disables access to particular topic files when GOPHER is being used. The default is "on".

Notes:

Most BBS facilities do not have)gmenu specified in the text menu files as this is not required to run BBS as an executable program on MUSIC. However, it is required when the BBS is accessed via gopher. You should place a)gmenu line in all text menu files of a BBS before making it available on gopher.

A text menu file is a BBS topic file with one or more columns of topic names and a companion descriptive text.

Example:

```
Place the cursor on the topic of your choice and press enter

+?library hours -? schedule for campus libraries
+?library loans -? how to take out a library book
+?library tours -? tours of the library facilities
```

should be modified as follows:

```
)gmenu
Place the cursor on the topic of your choice and press enter

+?library hours -? schedule for campus libraries
+?library loans -? how to take out a library book
+?library tours -? tours of the library facilities
```

)gmenu does not affect the BBS's operation as an executable program on MUSIC but is used as an aid in auto detecting the file types.

A simple method to retro-fitting)gmenu into the files that are text menus is to use FLIB to bring up a list of topic files for a given BBS. For example, infoMcGill's topic files (see above) can be accessed by typing the following "flib cw99:@inf.*".

Tailoring and Creating Alternate Gopher Servers

You can define alternate gopher servers on MUSIC. Typically the port number used for gopher service is 70. Alternate servers usually use numbers in the same range as the standard. For example, a second gopher server may use the port number 71 etc.

The file $TCP:INETD.PORTS is where all servers are declared. The format is as follows:

```
70 stream 5 $tcp:gopherd       <-- standard gopher daemon server
71 stream 5 your_own_file      <-- your own gopher server
```

Refer to the under the topic "The Internet Super Server (INETD)" in this manual for more details.

The following is a description of how to set up an alternate gopher server exec file and how to specify values to tailor it.

Usage

Create a file like the following, you can use $TCP:GOPHERD as a template.

```
/SYS NOPRINT,REGION=600,TIME=MAX
/LOAD XMON
GOPHERD N($TCP:GOPHERD.LMOD)
TRACE=F,FMENU='FIRST_MENU',PORT=N, .....
```

Namelist Parameters

PORT=n Used to override the port number specified by GOPHERSERVER in $TCP:TCPIP.CONFIG.

TRACE=F/T Used to turn on tracing. This is used for debugging only. The output is directed to @TCPIP.LOG and to @TCPIP.BUFFERS

MYSITE='site_name'
Used to override the GOPHERSERVER in $TCP:TCPIP.CONFIG.

LOG=T/F logs transactions to $TCP:TCPIP.LOG.

FMENU='menu_name'
Used to set a first menu other than $TCP:GOPHERD.MENU. It can be the main menu of a BBS or the standard gopher syntax. For example, to make the help facility the first menu, you would specify FMENU='$hlp:@go.main.menu'.

ALLOWF='file_pattern'
is used to override GOPHERD_ALLOWED_FILES from $TCP:TCPIP.CONFIG.

This entry defines the MASK for non BBS files allowed to be accessed via GOPHERD. For example to allow the subdirectory GOPHERD on userids that begin with dollar to be accessible you specify "$*:gopher*".

The default is "$pub:*".

BBSLOG='userid/prefix'
this parameter can be used to produce a bbstat like log of usage. The string must be in the form 'userid:@xxx.' This makes GOPHERD logging compatible with BBS logging.

The Finger Server (FINGERD)

This handles the "finger" requests that enable users on remote hosts to find out who is signed on to your MUSIC system. It basically sends them the output of the WHOSON utility. If you wish this service to be available, configure INETD.PORTS to run $TCP:FINGERD on stream port 79.

```
79    stream   2  $tcp:fingerd
```

IRC Client

The IRC client must be configured before using it. To configure the IRC client, simply edit the file $TCP:IRC and insert the names of your servers as: server='server-name' on multiple lines.

Each server will be tried in succession until a server accepts a link with your client. Thus, you should place your servers in order of preference. Note that the IRC server site must usually agree to accept communications from your site's clients.

Writing Your Own Servers

It is possible to write your own servers using the socket interface. For more information about the socket interface and writing socket applications see the file $TCP:SOCKETS.DOC. There are also two scaled down samples distributed with the system that can be used as models. See the files $TCP:DEMO.SERV.FORT and $TCP:DEMO.SERV.REX for details.

Production servers usually run as BTRMS scheduled by INETD and it is very difficult to debug them in this mode. For this reason the GETCON routine is provided to allow you to run the server independently of INETD on a regular terminal session so you have all the system debugging tools available. During testing use the GETCON routine to define the port:

```
S=GETCON(port)
```

This will initialize the specified port and wait for a connection to be made. The socket number (S) is set. This routine allows only one connection at a time on a given port, but this is usually sufficient for testing purposes.

When you want to put the server into production replace GETCON by GETSOC:

```
S=GETSOC()
```

This routine gets the socket number from INETD. Define the port that the server will use in the $TCP:INETD.PORTS file and restart INETD to activate the new server.

If you wish to code socket application in C, you must include a call to CARGENV (no parameters required) in order to cause the socket routines to automatically convert the arguments passed to assembler calling conventions. In this case, use the normal C calling conventions as defined by the SOCKET.HDR header file.

News Reader

The news reader (RN) allows users to access information on a remote network news feed. The program must be configured prior to use. RN establishes a TCP connection with the news feed machine and uses the NNTP protocol to access the information. News items are fetched over the network as required. News submissions can be made by direct posting to the server, if that is allowed, or through mail.

The RN program is configured by editing the file $TCP:TCPIP.CONFIG and setting the following parameters.

NEWSFEED The internet address or name of a host providing an NNTP news feed.

NEWSGROUPFILE The name of a file to contain the list of NNTP news groups.

NEWSDROP_0_POSTS
Specifies whether news groups with no postings are dropped from the news groups list. (yes/no)

NEWSGROUPNAMESIZE
Maximum length of news group name before truncation.

ORGANIZATION The name of your organization.

TIMEZONE Your time zone. (GMT, EST, MST, etc.)

News Groups File

This file is simply a sequential list of the news groups that are available. The $TCP:NEWSGROUP program contacts the NNTP server and builds this file based on information from the server. This process should be automated to update the list periodically (once a day perhaps). The AUTOSUB utility can be used to provide this automation by submitting $TCP:NEWSGROUP to batch when appropriate. The SYSCOM and FILES privileges are required.

Note: ADMIN (4 9) can be used to add $TCP:NEWSGROUP to the automatic maintenance jobs that AUTOSUB processes.

$TCP:NNTP.SERVICE.S

These routines provide the TCP/IP transport services for RN and the NEWSLIST program. There are several return codes that it generates. These are:

-101 - bad server name
-102 - could not determine local site IP address (problem with TCP/IP)
-103 - could not determine remote site IP address
-999 - connection closed by remote host

TCPSTAT - TCP Applications Analysis Facility

TCPSTAT is used to perform basic usage analysis of TCP/IP applications and facilities.

TCPSTAT can be used to either provide a display of a daily TCP/IP statistics log or to produce a statistical report on usage.

Select one of the following by typing its number in the command area.

```
1. Browse a TCP daily statistic file.
2. Produce a usage and activity report for TCP applications you want
```

After you have made your selection, press ENTER. The following sections describes the screens that assist you in customizing the browse or the report you want to generate on the usage patterns for a particular TCP/IP application.

Browsing TCP Statistics Logs

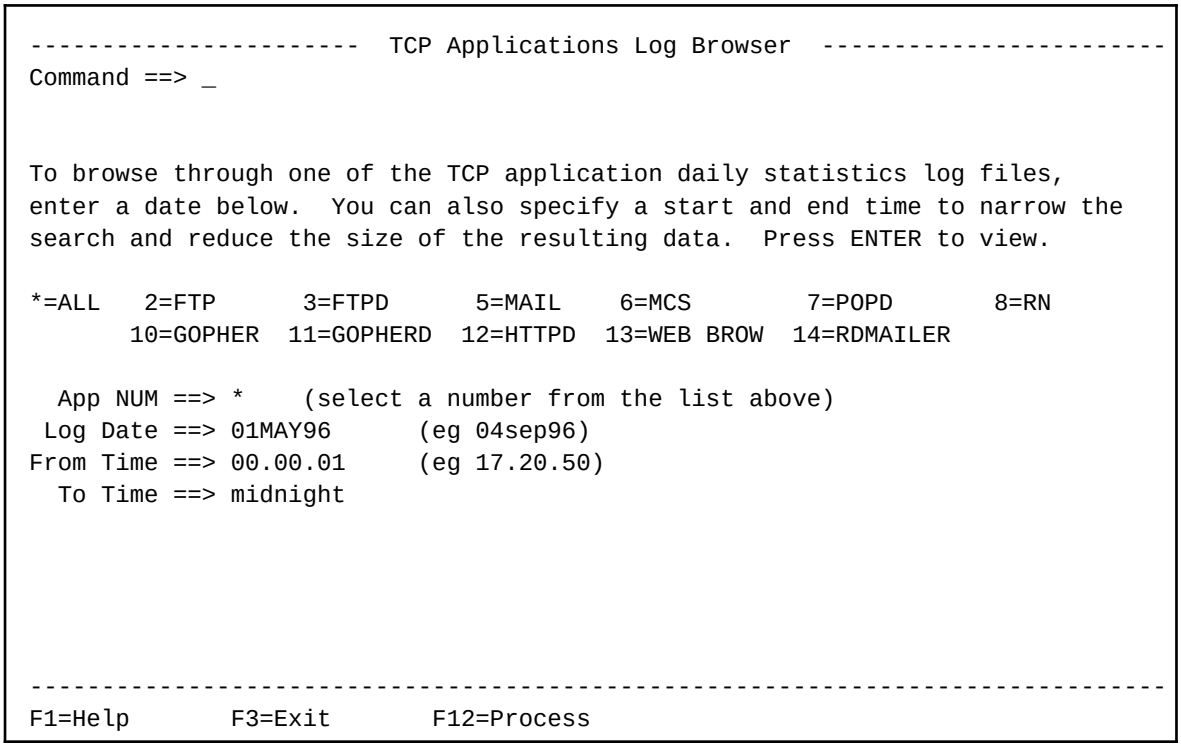


Figure 16.1 - Browsing TCP Statistics Logs

App NUM — You can choose to browse through one of the TCP application daily statistics logs by selecting one of the numbers below that correspond to TCP applications. "*" can be used to provide a report for all applications.

```
   *=ALL   2=FTP      3=FTPD      5=MAIL    6=MCS        7=POPD
  8=RN
            10=GOPHER  11=GOPHERD  12=HTTPD  13=WEB BROW  14=RDMAILER
```

Log Date — The daily log to be browsed is selected by entering a date on the "Log Date" field in the form ddmmmyy (28APR96).

From Time
To Time — You can also specify a start and end time to narrow the search and reduce the size of the resulting data.

When you have filled in the fields with the appropriate values press ENTER to browse the log file.

TCP Applications Statistics Report Generator

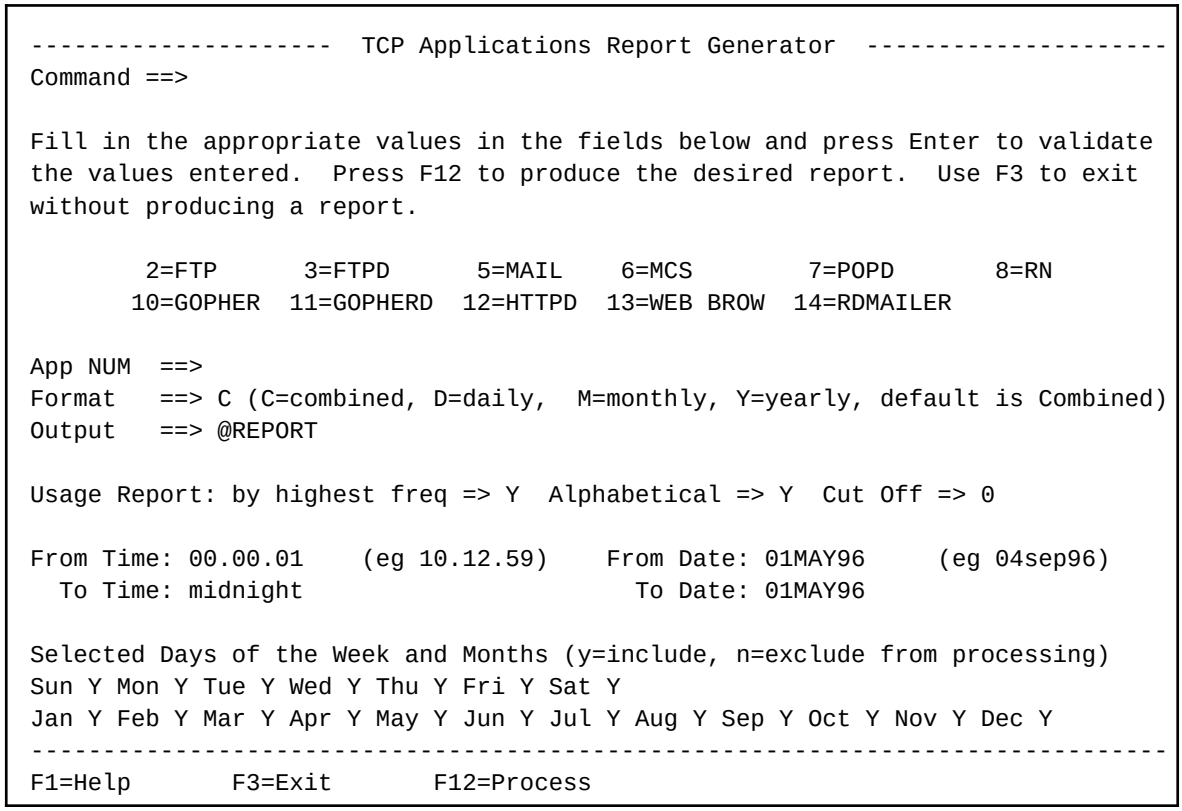


Figure 16.2 - TCP Applications Statistics Report Generator

You can obtain a statistical report on one of the TCP/IP applications. The report can be based on daily, monthly, yearly or a combined format. The report is divided into four parts. The first part reports to you all the values and parameters you entered to generate the report. The second part provides a frequency and usage chart based on the hour of the day. Column summaries are provided at the bottom of the chart. The third part provides summaries and averages for the entire report regardless of the report type. The fourth part provides a usage report of resources particular to the application.

App NUM — To select a statistical report for one of the TCP/IP applications enter its number.

Format — You can choose on of four formats for the report: daily, monthly, yearly and combined. The frequency charts based on the hour of the day will be produced based on the report format. The other parts of the report will be cumulative values based on all the data analyzed.

Output — You can specify a MUSIC file name to which the report is to be written. The default is "@report".

Usage Report — Each TCP/IP application has some resource or usage item that can be reported. For example HTTPD (the WEB server) downloads files. A frequency report on files downloaded is provided in order of high frequency of usage and by alphabetical order. Most applications will also report the IP addresses in the same fashion. This will allow you to determine what resources your system serves, what your users are fetching from the network and what IP addresses are involved.

You can also cancel or limit this feature of the report by specifying; "Y" yes produce usage report or "N" no do not produce the usage report. You may also want to reduce the output generated by limiting reporting to only those items that occur at least x number of times.

Use the "Cut OFF" field for this.

From TIme
To time — You can limit the report to a specific date and time range.

Selected Days... — It may be desirable to include or exclude specific months or week days. This section allows you to do this.

Making MUSIC look like an Internet Host

Introduction

It is possible to make the MUSIC virtual machine look like a physical internet host that functions independently from the VM system on which it is running. Currently the main advantage of this is that both MUSIC and VM can run servers on the same ports. For example, VM could run an FTP server on port 21 giving users access to the CMS file system and at the same time MUSIC could run an FTP server on port 21 giving access to the MUSIC file system.

To do this you must run two copies of TCP/IP under VM connected via an IUCV link. One copy of TCP/IP is used by the CMS users and the other is for use by the MUSIC virtual machine. This is similar in concept to using two RSCS virtual machines for BITNET connectivity, but has the significant advantage that you do not have to modify TCP/IP. The following diagram, figure 16.3, illustrates the situation.

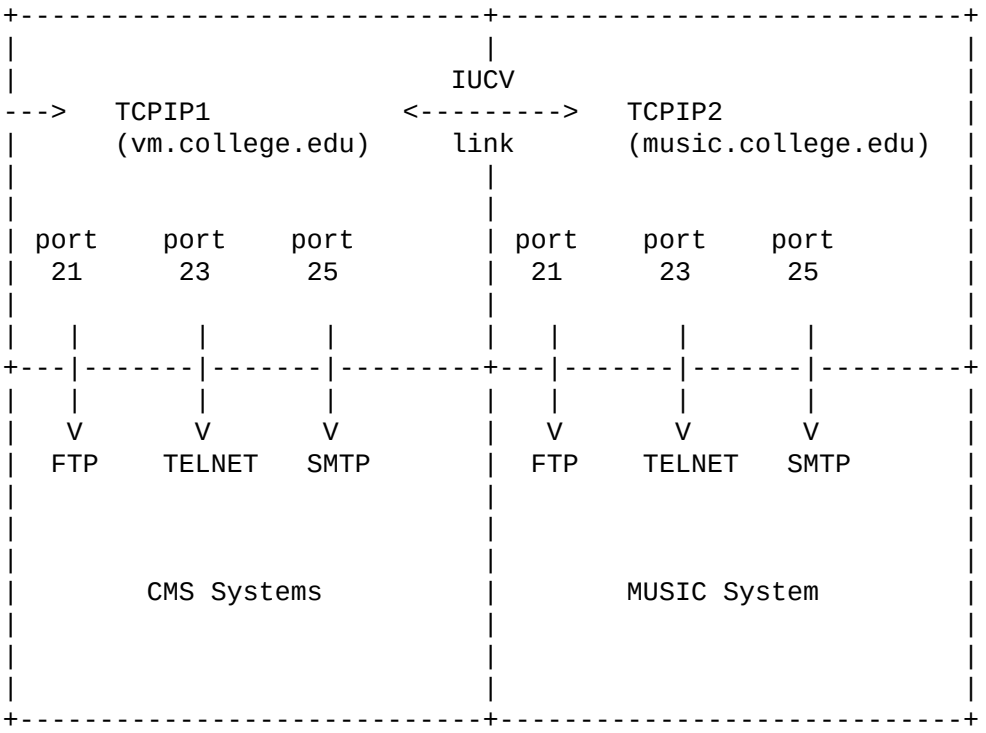


Figure 16.3 - Setting up TCP/IP for MUSIC

Setting Up an IUCV Link

The following assumes that you have already installed a TCP/IP system under VM for your CMS users.

If your site must provide TCP/IP services to your CMS users and your MUSIC users from their respective systems, one of the choices in the TCP/IP configuration for this setup is to run two TCP/IP systems. These two systems would run on separate virtual machines under the same VM, using an IUCV (point to point) link. One system is made available for access to and from CMS and the other system for access to and from the MUSIC system.

This section describes how to set up an IUCV link between two virtual machines running TCP/IP under the same VM.

Requirements

This can all be done using the IUCV driver included in the VM TCP/IP (alias FAL) software and does not require a hardware interface. You will need to get a new, unused subnet or network number (if your site isn't subnetted) assigned to the system you are going to create. You cannot use the same subnet(network) as the TCP/IP system you have already installed since the Internet Protocol specifies that different "physical" network types must have different subnet(network) numbers.

This support requires at least the 9201 service for the MUSIC/SP 2.3 system and the IBM VM TCP/IP V2, program number 5735-FAL. It is suggested that you have an installed TCP/IP system running under VM before proceeding with this procedure.

Along with the directions described below, you may also wish to consult the IBM TCP/IP Version 2 for VM: Planning and Customization manual, SC31-6082, for further details.

Routing Notes

If your campus is using STATIC ROUTING (ie a gateway/router has a file containing a list of the networks available for your site that it uses when it boots), then ignore the comments about the ROUTED virtual machine. You do not require this userid. However, you will need to have a gateway/router on your campus announce a static route to the new subnet(network) via the installed TCP/IP system's physical interface. Alternatively, you could add a static route in all of the other hosts that require access to the new host. In the configuration step below, this setup is referred to static routing.

If your campus is using DYNAMIC ROUTING (ie a router(s) discovers the networks available for your site via a routing protocol) with the Routing Information Protocol (RIP), then you will have to run ROUTED on both ends of the IUCV link. ROUTED can be used to manage the IUCV link only if both ends of the link run ROUTED. If only the installed TCP/IP system end runs ROUTED, it will not receive routing updates from the other end and it will assume the link is down. ROUTED will then delete all of the routes in its tables for the IUCV link. This problem is inherent to the RIP protocol and cannot be prevented. Therefore, you must run ROUTED on both ends of the IUCV link. In the configuration step below, this setup is referred to as dynamic routing.

Please consult the chapter on "Configuring the ROUTED virtual machine" in the TCP/IP VM Planning and Customization manual for further details.

TCP/IP V2R0 support notes

These notes reflect the fact that we have TCP/IP V2R2 installed. TCP/IP V2R2 has some notable changes over V2R0, one of which is important with respect to these notes on the implentation of an IUCV link.

V2R2 has both a 591 minidisk which contains the execs and modules required by TCP/IP servers, and a 592 minidisk which contains the files required by TCP/IP clients. V2R0 has only a 592 minidisk which contains all of the files required by TCP/IP servers and clients.

If you have V2R0 installed and are planning to implement the IUCV link as documented here, just ignore the references to the 591 minidisk. It is strongly advised that your site upgrade to V2R2, which contains the numerous bug fixes to V2R0.

A Sample Setup to use as a Model

We use our setup at the MUSIC Product Group as a model. The following diagram describes our setup. The addresses and domain names are used throughout this document in examples.

McGill University was assigned the domain name McGill.Ca. The MUSIC Product Group was given the subdomain MPG.McGill.Ca. The 132.206.120 network is our ethernet link to the campus router which connects us to the Internet. The 132.206.131 network is our point-to-point (IUCV) link connecting our two TCP/IP systems running under the same VM system.

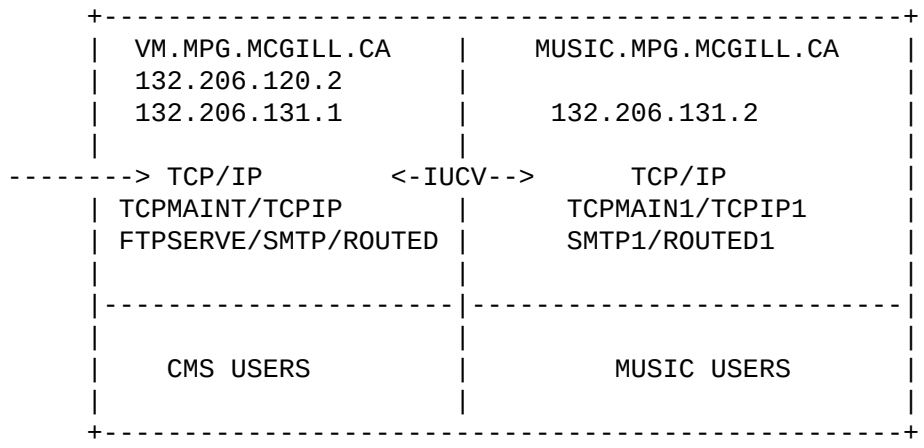


Figure 16.4 - Sample Setup for TCP/IP

IUCV Steps

There are five steps for this support:

1. Add the new system's domain name into the campus name server.
2. Define the set of required TCP/IP userids for the new TCP/IP system, format the minidisks, and copy the required files from the installed TCP/IP userids to the new TCP/IP userids
3. Configure the new TCP/IP userids
4. Test out the setup
5. Add the new TCP/IP userid to AUTOLOG1

1. Add the new system's domain name into the campus name server

Add the new system's domain name to your campus name server. You'll need A, HINFO, and PTR resource records for the new system.

Our installed TCP/IP system is known as VM.MPG.MCGILL.CA. Our new TCP/IP system is known as MUSIC.MPG.MCGILL.CA. The following example shows the resource records for a new system:

```
MUSIC.MPG.MCGILL.CA              A          132.206.131.2
MUSIC.MPG.MCGILL.CA              HINFO      IBM 9370 VM/SP
2.131.206.132.IN-ADDR.ARPA       PTR        MUSIC.MPG.MCGILL.CA
```

You will also need to add new A and PTR records for the installed TCP/IP system which represents the host's IP address for its end of the IUCV link. This is an example of the records required:

```
VM.MPG.MCGILL.CA                 A          132.206.131.1
1.131.206.132.IN-ADDR.ARPA       PTR        VM.MPG.MCGILL.CA
```

2. Define new TCP/IP userids/Format minidisks/Copy Required Files

First create the additional userids for the new TCP/IP system. Only the userids for TCPMAINT, TCPIP, SMTP need to be created. MUSIC does not have a TELNET server. MUSIC has its own FTP server so the FTPSERVE virtual machine is not needed. You must also create a ROUTED userid if you are using "dynamic routing" as described in "Routing Notes" above.

Of course, the new userids must have different names from the ones being used in the installed TCP/IP system. Let's use the userids TCPMAIN1, TCPIP1, SMTP1, and ROUTED1 for the new userids. These userids should be defined identical to their counterparts on the installed TCP/IP system. Note that TCPMAIN1 only requires the 191, 591, and 592 minidisks. The 591 minidisk is new for TCP/IP V2R2.

After the userids are created, format all of the new minidisks identical to the equivalent minidisk on the installed TCP/IP system.

Instead of reinstalling TCP/IP on the new system, we'll copy the files we need from the installed TCP/IP system to the new system. We only need the TCPMAINT files, so copy the TCPMAINT 191 minidisk files to TCPMAIN1 191, the TCPMAINT 591 files to TCPMAIN1 591, and the TCPMAINT 592 files to TCPMAIN1 592.

3. Configure the new TCP/IP userids

You'll need to make some changes to configure the new TCP/IP system. Both the TCPIP DATA and PROFILE TCPIP files on TCPMAIN1 need to be changed, as well as the PROFILE TCPIP file on TCPMAINT. During the initialization of the TCPIP virtual machine, it first searches for the file "nodename TCPIP", where "nodename" is the node name of the system, as specified by the CMS IDENTIFY command. If this file is not found, the initialization process uses the PROFILE TCPIP file. So wherever you find the PROFILE TCPIP file mentioned in these notes, you should use your "nodename TCPIP" file.

You will also create two files on TCPMAINT that can be used to START and STOP the IUCV link via the OBEYFILE command while you are doing your testing.

a) **Changes to TCPMAIN1's TCPIP DATA file**
For the TCPIP DATA file on TCPMAIN1, change TCPIPUSERID to TCPIP1, and change the HOSTNAME to the name you've decided for this new system. We have chosen the HOSTNAME MUSIC for our new system. We had previously chosen our DOMAINORIGIN to be MPG.MCGILL.CA, so the domain name for our new system is MUSIC.MPG.MCGILL.CA.

b) **Changes to TCPMAIN1's PROFILE TCPIP file**
A number of changes have to be made to TCPMAIN1's PROFILE TCPIP file. First, change the server userids in the AUTOLOG and PORT sections to the new server userids. Comment out those userids in the AUTOLOG section not being used on the new system, and comment out the ports required for those userids not being used in the PORTS section. You will need to change the DEVICE, LINK, HOME and START statements. The following are sample statements:

```
DEVICE IUCVDEV2 IUCV XYZZY XYZZY TCPIP B
LINK IUCVLNK2 IUCV 1 IUCVDEV2
....
HOME
    132.206.131.2 IUCVLNK2
....
START IUCVDEV2
```

If you are using "static routing" as described in "Routing Notes" above, then add the appropriate direct route and DEFAULTNET route in the GATEWAY section to get back to the installed TCP/IP system via the IUCV link using the same subnet mask (if subnetted). The following are sample statements:

```
HOME
; Local host's Internet address
132.206.131.2   IUCVLNK2

GATEWAY
; Network     First hop   Driver  Packet size  Subn mask  Subn value
; Direct routes
132.206.0.0   =          IUCVLNK2 DEFAULTSIZE  0.0.255.0  0.0.131.0
; Default for everything else
DEFAULTNET 132.206.131.1 IUCVLNK2 DEFAULTSIZE  0
```

If you are using "dynamic routing" as described in "Routing Notes" above, then add the appropriate link statement to the BSDROUTINGPARMS statement so that ROUTED can use this information to advertise the reachability of the IUCV link from this end using the same subnet mask (if subnetted). You should only use the statements in the BSDROUTINGPARMS section. All of the statments in the GATEWAY section should be commented out. The following are sample statements:

```
HOME
; Local host's Internet address
132.206.131.2   IUCVLNK2

; ; RouteD Routing information (if you are using the ROUTED server)
; ; If you are using RouteD, uncomment all the lines below for
; ; 'BSDROUTINGPARMS', and comment out all the lines for the 'GATEWAY'
; ; statement.
;
; ;  link     maxmtu   metric   subnet mask     dest addr
BSDROUTINGPARMS false
     IUCVLNK2 1500        0     255.255.255.0   132.206.131.1
ENDBSDROUTINGPARMS
```

c) **Changes to TCPMAINT's PROFILE TCPIP file**

A number of changes have to be made to TCPMAINT's PROFILE TCPIP file for the IUCV link. You will need to add additional DEVICE, LINK, HOME and START statements. The following are prototype statements:

```
DEVICE IUCVDEV1 IUCV XYZZY XYZZY TCPIP1 A
LINK IUCVLNK1 IUCV 1 IUCVDEV1
....
; add the new home definition to the end of the list of the
; other home definitions
HOME
    132.206.131.1 IUCVLNK1
....
```

```
; note this is commented out-we'll start it using obeyfile later
; START IUCVDEV1
```

If you are using "static routing" as described in "Routing Notes" above, then add the appropriate direct route in the GATEWAY section to get to the new subnet(network) via the IUCV link using the same subnet mask (if subnetted). The following are sample statements:

```
HOME
; Local host's Internet addresses
132.206.120.2  ETH1
132.206.131.1  IUCVLNK1

GATEWAY
; Network     First hop   Driver  Packet size  Subn mask  Subn value
; Direct routes
  132.206.0.0 =          IUCVLNK1 DEFAULTSIZE  0.0.255.0   0.0.131.0
  132.206.0.0 =              ETH1 DEFAULTSIZE  0.0.255.0   0.0.120.0
; Default for everything else
  DEFAULTNET 132.206.120.1   ETH1 DEFAULTSIZE  0
```

If you are using "dynamic routing" as described in "Routing Notes" above, then add the appropriate link statement to the BSDROUTINGPARMS statement so that ROUTED can use this information to advertise the reachability of the IUCV link from this end using the same subnet mask (if subnetted). You should only use the statements in the BSDROUTINGPARMS section. All of the statments in the GATEWAY section should be commented out. The following are sample statements:

```
HOME
; Local host's Internet addresses
132.206.120.2  ETH1
132.206.131.1  IUCVLNK1

; ; RouteD Routing information (if you are using the ROUTED server)
; ; If you are using RouteD, uncomment all the lines below for
; ; 'BSDROUTINGPARMS', and comment out all the lines for the 'GATEWAY'
; ; statement.
;
; ;  link     maxmtu   metric   subnet mask     dest addr
BSDROUTINGPARMS false
     ETH1      1500       0     255.255.255.0   0
     IUCVLNK1  1500       0     255.255.255.0   132.206.131.2
ENDBSDROUTINGPARMS
```

d) **Create files on TCPMAINT to START and STOP IUCV link**

During the testing step, you will use the TCPIP OBEYFILE command from TCPMAINT to start and stop the IUCV link. This is a convenient way to test a link without having it affect the operation of the other links. The OBEYFILE command takes a filename as a parameter. The file normally contains statements from the PROFILE TCPIP file. In our case, we will create two files, one to start the IUCV link and the other to stop the link. On TCPMAINT, create the file STARTIUC TCPIP on the 191 minidisk and the file will contain the line

```
START IUCVDEV1
```

Also on TCPMAINT, create the file STOPIUCV TCPIP on 191, and this file will contain the line

```
STOP IUCVDEV1
```

4. Test out the setup

When the above configuration step is complete, restart your installed TCP/IP system to get the new definitions in place. After it has been restarted, START the IUCV link on the installed TCP/IP system by logging onto TCPMAINT and issuing the command OBEYFILE STARTIUC. We are using the TCP/IP OBEYFILE command to run the configuration file STARTIUC TCPIP A on TCPMAINT's 191. This file we created in the previous step. Then logoff TCPMAINT.

Note that if any problems arise while you are testing the IUCV link, you can stop the IUCV link by logging onto TCPMAINT and issuing the command OBEYFILE STOPIUCV. The STOPIUCV TCPIP A file used in the OBEYFILE command was created in the previous step.

Next, start the newly installed TCP/IP system, by logging on TCPIP1 and running its profile exec. After it appears to have started okay, disconnect from TCPIP1.

Next logon to TCPMAINT and check the IUCV link using the NETSTAT DEVLINKS command. Also check the gateway configuration using the NETSTAT GATE command. Then logoff TCPMAINT.

Next run NETSTAT DEVLINKS and NETSTAT GATE from TCPMAIN1. Also from TCPMAIN1, try to PING the installed TCP/IP system (by IP number, not hostname, since the name service on the new system may not be working yet) to check that the new system can really talk to the production system. Then check-out whatever other clients you want to on the new system.

Then check out the servers on the new system by logging on to a VM userid that has access to the installed TCP/IP system, and then PING/TELNET/FTP/etc to the new system (by IP number, not hostname, unless you have enrolled the new system in the campus name server). Repeat on another host (PC, UNIX, etc) if desired.

When all of the testing is done and you are satisfied everything works, logon to TCPMAINT and edit PROFILE TCPIP. Remove the semicolon and all of the leading blanks from the start of the START IUCVDEV1 statement so that the START IUCVDEV1 starts in column 1. Now that the tests worked, you'll want this link started automatically. You can also purge the files STARTIUC TCPIP and STOPIUCV TCPIP on TCPMAINT's 191 as they are no longer needed.

5. Add new TCP/IP userid to AUTOLOG1

Now that everything is working, you'll want to add TCPIP1 to your AUTOLOG1 profile exec so that the new system is AUTOLOGged when VM is reIPL'd. If you have not added TCPIP to the profile exec prior to now, you should do it now. TCPIP should be autologged before TCPIP1.

MUSIC/SP Administrator's Reference

Part IV - Chapter 17 (AR_P4.PS)

Part IV. Utilities

Chapter 17. System Utility Programs

Overview of Utilities

This chapter provides information on the the MUSIC/SP system utilities. The descriptions are arranged in alphabetical order by the utility name. The "Summary by Function" topic below can be used to determine the name of the utility program required for a given task.

These utilities often require special privileges and therefore cannot be run by the general user. The special privilege names and meaning can be found in the description of the CODUPD utility. The required privileges are described with each utility. If the files of the programs are specified as PRIVATE (as they are on the distributed system), the user executing them must have sufficient privileges to access them (i.e. LSCAN).

All of the programs run under the control of MUSIC. Most of the MUSIC utilities can be run either from batch or a terminal. It is recommended that programs producing at lot of output be run from batch.

To execute a utility program from batch, the following commands are used:

```
/FILE                   <--/FILE statements (if any)
/INCLUDE name           <--where "name" is the name of the utility
....                    <--parameters (if any)
....                    <--data for the program (if any)
```

To execute a utility program from a terminal, the terminal must be signed on to the system with a sign-on code of sufficient privileges. Note: the VIP privilege is active only after the /VIP ON command is issued at *Go time by a sign-on code that has the VIP privilege.

Most utilities can be run by simply typing the name of the utility when the terminal is in *Go mode.

Namelist Input

Many utilities use the FORTRAN Namelist facility to enter the options. Namelist, as extended by MUSIC, offers a powerful and concise way of entering optional parameters. For example if a utility has three parameters called A, B and C, then they could be specified as follows:

```
a=10.5,c=2,b='musicx'
```

Notice that they do not have to be in any specific order. Omitted parameters will retain their value. This means that unspecified parameters will take on default values. Should the utility ask for the options again at a later time, then you need only specify the ones that you want changed.

Errors detected by the Namelist facility will be displayed and then you will have a chance to respecify them.

Namelist parameters can be extended over several lines. This is done by ending the previous line with a comma (,) as in the example:

```
a=10,
b=20
```

Be very careful to enter the comma. If you omit it then the utility would not know that it must read any additional lines and so will not work as intended.

Additional Namelist examples are shown below:

```
a=z130          <--hexadecimal number 130 specified
a='musicx'      <--characters string given
a=true          <--parameter given a "true" value
a=10,4,35       <--three numbers specified for an array
```

Summary by Function

The system utility names are listed below by function.

Accounting

ACCTDS.SCAN
: Display information from accounting data set.

ACTDMP
: Process the system accounting file to produce session accounting records.

DSACT1

DSACT2
: Produce UDS file accounting records.

MFACCT
: Produce file accounting records.

NOWDOL
: Scan the terminal accounting records and update the NOW$ field in the code table records for each user.

Backup

CODUMP
: Make a backup copy of the code table on tape and optionally print entire code table.

DSARCH
: Create backup copies of UDS or SDS files on tape.

MFARCH
: Used to create an incremental backup of the save library on tape.

MFARC2
: Used to backup selected save files to tape.

Change

CDUMP
: Display and change main storage.

CODUPD
: Add, delete, change, and display, code table records.

DSKDMP
: Display and change records on disk using absolute disk address. Display the VTOC, providing the names and extents of all UDS and SDS files, and free space on a disk. Display detailed VTOC entry for an individual UDS or SDS file. Reorganize the VTOC free space pointers (format-5 DSCBs).

EDTCAT
: Create or change the system catalogue. The catalogue contains entries to define SDS files, specify which modules go in the link pack area, and issue CP commands at IPL time.

FILECH
: Change the access controls of a save file.

FIXINDEX
: Change entries in the save library index to fix index/directory errors.

FIXINDEX.AUTO
: Change entries in the save library index to fix index/directory errors.

FORMAT
: Format entire disk packs, SDS files, or individual tracks on disk. Create or change disk volume labels. Write data to disk by absolute disk address.

LDLIBE
: Add, delete, or change members in the system or user load library.

NUCGEN
: Change system nucleus and/or I/O configuration.

SUBLIB.GEN	Used to make additions or changes to the subroutine library.
SYSDMP	Display or change records on disk based on relative record number within a specific SDS file or DEB.
SYSGEN1	Write system nucleus to disk.
SYSREP	Apply minor changes to load library members. Similar to OS superzap.
SYSUPDATE	Change Load Library Directory in main storage
UCR	Add, change, or delete save library user control records. These records contain the allocation limits for save files on any given user code.

Code

CODPRV	Lists all privileged userids (codes).
CODUMP	Make a backup copy of the code table on tape and optionally print entire code table.
CODUPD	Add, delete, change, and display, code table records.
GEN.CODES	Create input for CODUPD which adds a group of access codes.
NOWDOL	Scan the terminal accounting records and update the NOW$ field in the code table records for each user.
TRANS$	Allow supervisor code to transfer funds to codes under this supervision.
UCR	Add, change, or delete save library user control records. These records contain the allocation limits for save files on any given user code.
UCRFIX	Scan the code table and adjust or create a user control record for each code record.
WHOACT	Display a list of active terminal sessions indicating the TCB number, device address and user code.
WHOALL	Display a list of all terminal sessions indicating the TCB number, device address and user code if present.
WHOSON	Display TCB information about specific user codes that are currently signed on.

Disk

BUFLOG	List usage and error information for 3330 and 2305 disks.
CHKDISK	Verify disk formatting.
DSKDMP	Display and change records on disk using absolute disk address. Display the VTOC, providing the names and extents of all UDS and SDS files, and free space on a disk. Display detailed VTOC entry for an individual UDS or SDS file. Reorganize the VTOC free space pointers (format-5 DSCBs).
FMFREE	Create a job to format all the free space on a disk pack.
FORMAT	Format entire disk packs, SDS files, or individual tracks on disk. Create or change disk volume labels. Write data to disk by absolute disk address.
INITFBA	Create a VTOC on an FBA disk.
IOTIME	Display usage and timing information for SDS files, disk volumes, and channels.
SYSDMP	Display or change records on disk based on relative record number within a specific SDS file or DEB.
UCB	Display a list of disk and tape devices on the system.

Display

ATTRIB	Display save file attributes such as space allocation, record length, access control and date last used.
BPOOL	Display information about the terminal buffer pool.
BSTATUS	Display information about current batch job.
CDUMP	Display and change main storage.
CODUPD	Add, delete, change, and display, code table records.
CONLOG	Displays the current contents of the console log.

COUNTS	Display information accumulated by the system COUNTS macro.
DSKDMP	Display and change records on disk using absolute disk address. Display the VTOC, providing the names and extents of all UDS and SDS files, and free space on a disk. Display detailed VTOC entry for an individual UDS or SDS file. Reorganize the VTOC free space pointers (format-5 DSCBs).
ENQTAB	Display information about the system enqueue table.
IOTIME	Display usage and timing information for SDS files, disk volumes, and channels.
LDCNTS	Displays usage information for the system load library.
LDLIST	Display names, sizes, and attributes of the members of the system or user load library.
LIBSPACE	Display a list of available free extents in the save library.
LOOKUP	Display the table of privileged program names.
LPA	Display information on Link Pack Area (LPA).
MAPMEM	Display memory usage.
MFINDEX	Display information about the save library index.
RATE	Display system load.
ROUTETABLE	Display contents of the $ROUTING table.
SSTAT	Display the current system status.
SYSDMP	Display or change records on disk based on relative record number within a specific SDS file or DEB.
UCB	Display a list of disk and tape devices on the system.
WAITS	Display system wait time information.
WHOACT	Display a list of active terminal sessions indicating the TCB number, device address and user code.
WHOALL	Display a list of all terminal sessions indicating the TCB number, device address and user code if present.
WHOSON	Display TCB information about specific user codes that are currently signed on.
XTELL	Delivers message whether receiver has MESSAGES ON or OFF.

Information

BPOOL	Display information about the terminal buffer pool.
BSTATUS	Display information about current batch job.
BUFLOG	List usage and error information for 3330 and 2305 disks.
CONLOG	Displays the current contents of the console log.
COUNTS	Display information accumulated by the system COUNTS macro.
ENQTAB	Display information about the system enqueue table.
IOTIME	Display usage and timing information for SDS files, disk volumes, and channels.
LDCNTS	Displays usage information for the system load library.
LOOKUP	Display the table of privileged program names.
LPA	Display information on Link Pack Area (LPA).
MAPMEM	Display memory usage.
MFINDEX	Display information about the save library index.
SSTAT	Display the current system status.
SYSDATE	Displays nucleus level and date/time of IPL.
UCB	Display a list of disk and tape devices on the system.
WAITS	Display system wait time information.
WHOSON	Display TCB information about specific user codes that are currently signed on.

Load Library

LDCNTS	Displays usage information for the system load library.
LDLIBE	Add, delete, or change members in the system or user load library.
LDLIST	Display names, sizes, and attributes of the members of the system or user load library.

SYSREP	Apply minor changes to load library members. Similar to OS superzap.
SYSUPDATE	Change Load Library Directory in main storage

Restore

CMSTAPE	Restore files from tapes produced by the CMS TAPE DUMP command.
DSRST	Restore UDS and SDS files from tapes produced by DSARCH
LOADPDS	Restore files from a tape produced by the IEBCOPY utility.
MFREST	Restore save file from tapes produces by MFARCH and MFARC2.
MOVEPDS	Restore file from tapes produced by the IEHMOVE utility.
UDSRST	Restore UDS files from tapes produced by DSARCH.

Save Library Files

ATTRIB	Display save file attributes such as space allocation, record length, access control and date last used.
ELOG.CLEANUP	Free up save file space by deleting unused editor log files.
FILE.DELETE	Delete groups of files at a time.
FILECH	Change the access controls of a file.
FPRINT	Print the content of a group of files.
MFARC2	Used to backup selected files to tape.
MFHASH	Hash a save file name to give the index block number for that file.
MFREST	Restore save file from tapes produces by MFARCH and MFARC2.
UCR	Add, change, or delete save library user control records. These records contain the allocation limits for files on any given user code.
UCRFIX	Scan the code table and adjust or create a user control record for each code record.

Save Library

ADDPDS	Create a PDS from an IEBUPDTE tape file.
CHKFILES	Scan Save Library to find any files with errors 65 and 67.
FIXINDEX	Change entries in the save library index to fix index/directory errors.
FIXINDEX.AUTO	Change entries in the save library index to fix index/directory errors.
GENSAV	Create multiple files from a single file.
LIBINDEX.CLEAN1	Clean Save Library Index overflow area.
LIBINDEX.CLEAN2	Clean Save Library Index auxiliary area.
LIBINTEG	Compare the save library space allocation maps with the index and directory listing any anomalies.
LIBSPACE	Display a list of available free extents in the save library.
MFACCT	Produce file accounting records.
MFARCH	Used to create an incremental backup of the save library on tape.
MFARC2	Used to backup selected files to tape.
MFINDEX	Display information about the save library index.
MFMOVE	Reclaim fragmented save library space by moving the files to other library SDS files.
NEWINDEX	Create a new save library index data set of a different size.
SETFBN	Reset Save Library backup numbers.
UCR	Add, change, or delete save library user control records. These records contain the allocation limits for save files on any given user code.

SDS

DSARCH	Create backup copies of UDS or SDS files on tape.
DSCOPY	Copy UDS or SDS files.
DSKDMP	Display and change records on disk using absolute disk address. Display the VTOC, providing the names and extents of all UDS and SDS files, and free space on a disk. Display detailed VTOC entry for an individual UDS or SDS file. Reorganize the VTOC free space pointers (format-5 DSCBs).
DSREN	Rename UDS files (see MUSIC/SP User's Reference Guide).
DSRST	Restore UDS and SDS files from tapes produced by DSARCH
EDTCAT	Create or change the system catalogue. The catalogue contains entries to define SDS files, specify which modules go in the link pack area, and issue CP commands at IPL time.
FORMAT	Format entire disk packs, SDS files, or individual tracks on disk. Create or change disk volume labels. Write data to disk by absolute disk address.
IOTIME	Display usage and timing information for SDS files, disk volumes, and channels.
MFMOVE	Reclaim fragmented save library space by moving the files to other library SDS files.
SYSDMP	Display or change records on disk based on relative record number within a specific SDS file or DEB.

Storage

CDUMP	Display and change main storage.
PRDUMP	Run after a MUSIC storage dump is taken to print a formatted listing of system storage, registers, control blocks, and trace table.
SYSUPDATE	Change Load Library Directory in main storage

TCB

WHOACT	Display a list of active terminal sessions indicating the TCB number, device address and user code.
WHOALL	Display a list of all terminal sessions indicating the TCB number, device address and user code if present.
WHOSON	Display TCB information about specific user codes that are currently signed on.

UDS

DSACT1	
DSACT2	Produce UDS file accounting records.
DSARCH	Create backup copies of UDS or SDS files on tape.
DSCOPY	Copy UDS or SDS files.
DSKDMP	Display and change records on disk using absolute disk address. Display the VTOC, providing the names and extents of all UDS and SDS files, and free space on a disk. Display detailed VTOC entry for an individual UDS or SDS file. Reorganize the VTOC free space pointers (format-5 DSCBs).
DSREN	Rename UDS files. See *MUSIC/SP User's Reference Guide* for a description.
DSRST	Restore UDS and SDS files from tapes produced by DSARCH
TAPUTIL	Dump or summarize selected files from tape, and copy files from one tape to another (see See *MUSIC/SP User's Reference Guide* for a description.
UDSRST	Restore UDS files from tapes produced by DSARCH.
UTIL	List, punch, copy, and merge data on cards, card images, tape, or disk. See *MUSIC/SP User's Reference Guide* for a description.

VTOC

DSKDMP	Display and change records on disk using absolute disk address. Display the VTOC, providing the names and extents of all UDS and SDS files, and free space on a disk. Display detailed VTOC entry for an individual UDS or SDS file. Reorganize the VTOC free space pointers (format-5 DSCBs).
INITFBA	Create a VTOC on an FBA disk.

Utilities Listed Alphabetically

The descriptions that follow are arranged in alphabetical order by the names of the utilities.

ACCTDS.SCAN: Display Information from Accounting Data Set

This interactive utility displays information from the detailed records in the system accounting data set, SYS1.MUSIC.ACCT. You can specify the starting and ending block for the scan, and selection criteria such as a userid or userid prefix. To start the utility, enter the command ACCTDS.SCAN.

ACTDMP: Accounting File Dump

This program reads the MUSIC Accounting File normally called SYS1.MUSIC.ACCT. Three (3) types of records are produced by this program. One is a summary record of each terminal user's terminal session (or batch job), showing processing unit utilization, session time, charge for number of characters received or sent to the terminal. Another record is produced when UDS files are deleted by users. The third type provides information that can be used to maintain systems availability records.

Records produced by ACTDMP represent accounting information that has accumulated in the accounting data set (SYS1.MUSIC.ACCT) since the last time ACTDMP was run (or since the last time the =RESET option was used during MUSIC IPL).

ACTDMP accumulates the detailed information from the MUSIC accounting files for each session. If a session is still active when ACTDMP is run, the current totals accumulated for that session are written to the accumulation table file ($ACT:ACCTAB). When ACTDMP is next run the totals are re-loaded from the file and the accounting process proceeds from where it left off. The accounting record for the session is only generated when ACTDMP detects that the session is complete.

If the accumulation table file is invalid or out of date, ACTDMP will skip to the first IPL record that is found and begin the accounting process from that point. This is a normal situation after the system has been loaded with the =RESET option.

Usage:

The following JCL is used to run this ACTDMP program from batch. See *Chapter 5 - Routine Maintenance* for the recommended time to run this program.

```
/INCLUDE ACTDMP
```

Its output is directly written to the file $ACT:ACT.RCDS. If a file by that name already exists, it will rename the old one to "$ACT:ACT.RCDSnn" where *nn* is a number 01 to 09.

The installation can run its own billing program after ACTDMP has run. It can be setup to read the file $ACT:ACT.RCDS as input.

The file $ACT:ACTDMP contains a parameter statement that can be changed if the installation wants to use a different name for the output file or change the number of backup copies or alter the primary size of the output file. The following is the parameter statement format:

columns 1-4	primary space for output file in k bytes
columns 6-7	number of backup copies (must be 01 to 99)
columns 9-28	userid:filename for output

The format of the accounting records produced is as follows:

Session Accounting Record

Column	Contents
1-3	Identification 'RRR'.
4-16	Userid (first 13 characters). The remaining 3 characters are in columns 51-53.
17-24	Identification. terminal - third parameter from /ID command. batch - columns 5-12 of /ID statement (jobname).
25-29	Processing time (60/100 service units). A *service unit* (SU) will approximate 1 million machine instructions if the recommended value chosen for the CFACT parameter at the time the NUCGEN utility was run. For example, on a 3 MIP machine, 1 SU represents about 1/3 seconds of processing time.
30-34	Reserved.
35-39	Terminal I/O charge in cents. terminal: 5 cents per 1000 chars received and transmitted. batch: .065 cents per card read + 0.3 cents per card punched + 0.1 cents per line printed
40-44	Terminal time - elapsed time /ID to /OFF (1/100ths hours). For a batch job, it indicates the amount of time between reading the ID statement and the time the last output line was printed.
45-46	Internal terminal number (TCB number) - 2 hexadecimal characters, i.e. "1A". "00" if batch.
47-48	Reserved (terminal identification number). ("01" for normal signon, "00" for added session.)
49-50	Physical line address - hexadecimal. Notice that only the last two characters of the terminal address appear. Thus, terminals on address 120 would show as 20.
51-53	Last 3 characters of userid.
54	Accounting record format identifier (">").
55-60	processing unit charge in cents. A surcharge is added to this rate when the user uses a region size of greater than 108K. The surcharge is 1% per 4K requested above 108K.
61-65	Terminal: connect charge in cents Batch: 60 cent handling charge + 150 cents per tape mount + 300 cents per disk mount.
66-69	Time of day of /ID (hours and minutes). This field may show the user's sign-on time greater than 24 hours. For example, if the system were loaded yesterday and a user signs on at 2 AM, the accounting record will show 2600 in this field. The user will see the correct time at the terminal.

70-76	Date in the form of 20JUL74.
77-80	Card sequence number.

Notes:

1. Reserved fields are not necessarily blank.
2. Batch jobs can be identified by columns 45-46 equal to zero ("00").
3. Columns 40-44 and 47-50 should be disregarded for batch jobs.
4. Columns 55 through 65 contain a charge based on the following rate table. These rates can be changed by modifying the table in the module MACCNT which is part of the program 'ACTDMP'.

Time of day	processing unit charge per 60 service units	Connect charge per hour
8:15- 9:30	$10	$3
9:30-12:00	$12	$4
12:00-14:00	$10	$3
14:00-16:30	$12	$4
16:30-18:00	$10	$3
all other times and all day Sat and Sun	$8	$1

MD1, MD2 Records (produced by ACTDMP and DSACT2 utilities)

Column	Contents
1-3	Identification: 'MD1' (from DSACT2) or 'MD2' (from ACTDMP, for a deleted data set).
4-19	File ownership id (16 characters, padded with blanks).
20-29	Current data and time: yymmddhhmm
30-36	Number of minutes to be billed, since last accounting.
37-42	Number of tracks to be billed. For an FBA device, 1 track = 32 blocks.
43	Disk device type: 1 printable character. 7=3350, 8=FBA, 9=3375, A=3380, B=3390, C=9345.
44	'B' for data set with BACKUP attribute, 'N' otherwise.
45-50	Volume name.
51-80	First 30 characters of data set name.

MUSL Cards (MUSIC Availability Log Records)

Column	Contents
1-4	Identification 'MUSL'
5-6	processing unit identifier (the contents of the 1-byte area at location 3E5 in main storage, in hex)
12-15	Time in form HHMM
16-21	Date in form DDMMYY
22	Identifying letter M
23	System State Indicator: =1 MUSIC IPL done =2 MUSIC system shutdown =3 (Reserved) =4 (Reserved) =5 Unscheduled system down (This is indicated by no previous shutdown record. The time and date is that of the last recorded activity.)

Privileges Required: VIP and FILES.

ADDPDS: Create a PDS from an IEBUPDTE tape file.

This program creates a MUSIC PDS from a tape in IEBUPDTE format (./ add format).

Usage:

```
/FILE 1 TAPE VOL(xxxxxx) RECFM(U)
/INC ADDPDS
FILE=n,PREFIX='code:',SUFFIX='.s',SELECT=,REPL=,PUBLIC=,SL=
<-- member names to be selected -- one per line if required
FILE=n, etc.
FILE=n, etc.
```

Control statements must be in order of ascending file number. The list(s) of files to be selected can terminate with an end of file, or "-" in column 1 if subsequent control statements are present. Selected files pertain only to the preceding control statement. Values specified on previous control statements are carried over to subsequent ones unless specifically modified.

Parameters:

FILE=n	*n* is a file number on the input tape.
PREFIX=	prefix of the newly created file name. Default is PREFIX=' '.
SUFFIX=	suffix of the newly created file name. Default is SUFFIX=' '. The total length of the PREFIX and the SUFFIX should not exceed 9 characters (or 14 characters if 'code:' is specified in the PREFIX).

REPL= if REPL=.true. is specified, then an existing save file is replaced by the newly created save file of the same name. Default is REPL=.false.

PUBLIC= if PUBLIC=.true. is specified, the files are saved public.

SELECT= if SELECT=.true. is specified, then only those './ add' statements with names that are specified in a save file (unit 5) are processed (one name is specified on each line starting at column 1 in the save file). The default is SELECT=.false. which processes all './ add' statements in the tape.

SL= if true, skips the header and trailer files of a standard labelled tape.

Privileges Required: None.

ATTRIB: Display File's Attributes

This program is used to list information associated with a file. Information such as access control, record length, record format, space allocated, number of records used etc. will be displayed.

Usage:

The program can be invoked in 2 ways:

1. Enter "ATTRIB name" or "AT name", where *name* is the name of the file desired. The file name may or may not have the user code prefix (code:), and special codes *COM and *USR may be used if desired. This will list the information about the specified file.

2. Enter ATTRIB or AT. This will go into conversational mode and allow information on many files to be retrieved. Enter /CANCEL or a blank line to terminate the program.

The following is a typical listing of the information associated with a file:

```
NAME=$HLP:HELP                PUBL XO XO(OWN)
TAG=
LRECL =   80      RECFM = 0200 (FC)    ACCESS CONTROL = 80 20 60 60
SPACE (K):  PRIMARY =      2     SECONDARY =      0     MAXIMUM =   -1
LINES =       2    HIGH BLK =      1  (MAX =      3)   EOF DISPL = 46
BACKUP NUMBER = 121     USAGE COUNT =      0    EXTENTS =  1
CREATED 01APR89  LAST OPN FOR READ 17APR89  FOR WRITE 12APR89 10:05:09
CREATOR: ABCD000    LAST WRITER: ABCD000
```

Refer to *Chapter 20 - File System* for a description of the access control (1 byte for the general control flags and 3 bytes for the access control bits). HIGH BLK indicates the number of 512-byte blocks used, and MAX is the number of 512-byte blocks available. EOF DISP indicates the displacement, in bytes, of the last byte of the file from the beginning of the last 512-byte block used. USAGE COUNT is the number of times the file has been used, and is equals to zero if the count is not kept.

Privileges Required:

No privileges for user's files or common or public files. LSCAN for other private files.

AUTOSUB: Automatic Submitter

This program reads a list of file names containing data to be submitted at specified times during the day. The program goes to sleep (using CALL DELAY) until the next specified time, when it wakes up and does the required submit.

This program is normally set up to run automatically when MUSIC is initialized. Consult the documentation on BTRM in the *MUSIC/SP Administrator's Guide* under Installing MUSIC/SP for more information.

The executor file for this program, complete with data and any /FILE statements needed, should be set up as the AUTOPROG for userid $MONxxx, where the subcode xxx is the device address of the pseudo terminal, defined as BTRM in the MUSIC NUCGEN job, to be used. The code $MONxxx must be allocated with the privileges LSCAN, FILES, and SYSCOM, and with unlimited time: PRIME(NL), NONPRIME(NL), DEFTIME(NL).

Since the program must not try to read from the terminal or write to the terminal, Unit 6 output should be directed to a file (or be defined as /FILE 6 DUMMY). Each time this program begins (i.e., at each MUSIC IPL), any items not done yet today but whose time has passed, are done. Each time an item is done, a record is added to a control file. The record format is:

```
*ID=nnnn DONE DDMONYY HH:MM:SS
```

nnnn is the ID number associated with the item (specified in the ID= parameter).

Virtual punches on device addresses 011 and 012 are used for the submit. The program assumes that these devices are defined for the MUSIC virtual machine.

Parameters:

The following parameters are for unit 5.

SPECS='filename' The name of the file containing the items to be done (see below).

CTLFIL='filename' The name of the control file. The control file must be initially set up as an empty file, with LRECL 80.

The list of things to do is specified by namelists in the specifications file (SPECS=). The parameters are:

ID=nnnn A unique ID number for the item. (Required.)

TIME=hhmm Time of day (24-hour clock) at which the action is to be taken. (Required.) Must be from 0000 to 2400. Avoid specifying the value 2400 (exactly midnight), because of complications due to the date changing at midnight.

DAYS=n1,n2,... Days of the week (Sunday=1, Monday=2, etc.) on which the item is to be done. If the current day is not one of these, the item is ignored at namelist read. Default is DAYS=2,3,4,5,6 (i.e., weekdays, Mon-Fri).

FILE='filename' Name of the file containing the card images to be submitted. % in column 1 is changed to /. (Required.) Normally these records are submitted to VM. They are submitted to the internal reader instead, if MUSIC is not running with VM and the SPLID name (see below) is "*" or starts with the characters "MUS". a submit to the internal reader can be forced by specifying SPLID='*INTRDR'. Internal reader queue 1 is used. A file for holding the records for each job is created with file name uuuu:@AUTOJOBnnnn.m, where nnnn is the 4-digit id number (see ID parameter

above) and m is a 1- to 4-digit sequence number. uuuu is the userid running autosub. These files are used only when the internal reader is used. for batch passwords, see the GENBPW parameter below.

JCLCTL='filename' This optional parameter specifies a file which can be used to adjust text (such as volume and file names) in submitted records. Refer to the comments in the source file $PGM:AUTOSUB.S for details.

GENBPW=TRUE This causes batch password records to be generated in the submitted MUSIC jobs. A password record is generated after each /ID record. The CODES privilege is needed for this. The default is GENBPW=FALSE.

SPLID='name' 1 to 8 character name of the virtual machine to which the data is to be spooled. Default is SPLID='MUSIC'.

CLASS='x' VM Spool class. default is CLASS='J' (as needed for MUSIC batch jobs). A queue number for the internal reader can be specified as CLASS='n', where n is a digit 1 to 9 (anything else causes internal reader queue 1 to be used).

TAG='xxx' TAG information for RSCS, if the data is spooled to RSCS. Default is no tag. *xxx* is 1-48 characters of tag information to be used on the VM command "TAG DEV uuu xxx" for the submit. This would be used if SPLID='RSCS'. The tag command is issued only if a nonblank tag string is specified. The default is blanks.

CMT='xxxxx' is 1 to 80 characters of comment info (optional). The comment is ignored by the program but can indicate what the item is used for.

START=yyyymmdd optional starting date for the item. Before this date, the item is ignored. For example, START=19860525. The current year is assumed if a year is not specified (i.e. if the value is less than 10000).

EXPIRE=yyyymmdd optional expiry date for the item. After this date, the item is ignored. For example, EXPIRE=19860619. the current year is assumed if a year is not specified (i.e. if the value is less than 10000).

SKIP=yyyymmdd,yyyymmdd,...
optional dates on which this item should be skipped. Up to 100 dates may be specified. If a year is not specified in a date (i.e. the value is less than 10000), the previous year in the list is used. For example, SKIP=19860521,0523,0525 skips 3 dates in May 1986. If a year is not specified in the first date in the list, the current year is assumed.

Note: Dates for start, expire and skip parameters must use 4-digit year (optional), 2-digit month, and 2 digit day of month, in that order. The dates are not checked for validity.

Privileges Required:

The code $MONxxx must be allocated with the privileges LSCAN, FILES, and SYSCOM. Also, the CODES privilege is needed if the parameter GENBPW=TRUE is used.

BPOOL: Display Buffer Pool Buffer Usage

This program can be used to find out about the utilization of the terminal buffer pool. It is helpful in determining whether the correct number of buffers have been allocated at NUCGEN time.

Usage:

This program is invoked by entering BPOOL on the terminal.

Privileges Required: CREAD.

BSTATUS: Display Status of Current Batch Job

BSTATUS, when executed on a terminal, wakes up every 5 seconds (approximately) and checks the status of the job running on batch. If there is a change, a line of information is displayed. This information includes the batch job code and subcode, the number of service units used so far, and the time of last activity (TLAT). If the job has ended, this is indicated.

Since BSTATUS only checks batch every 5 seconds or so, it is possible for short batch jobs to execute without being displayed by BSTATUS.

If a number is specified on the command, for example "BSTATUS 30", it defines the approximate number of seconds between checks. The default is 5 seconds.

Privileges Required: LSCAN and CREAD.

BUFLOG: List 3330 & 2305 Usage/Error Statistics Log

This program empties and prints the current contents of the usage/error logs for all 3330 devices on the system. If any of these counters overflow, the device automatically sends the log information to the processing unit which will cause MUSIC to print the information on the console. (When run under VM/370, all the counts will normally be zero as VM/370 will dump these counters just prior to performing this operation for MUSIC.)

The log for each drive consists of 24 bytes of data. This gives such information as the number of arm motions performed, the number of seek errors, the number of bytes of data read, the number of data checks, etc. The exact format and meaning of this data can be found in the IBM publication *Reference Manual for IBM 3830 Storage Control Model 1* (GA26-1592) or equivalent.

This program also prints any 2305 buffered logs which may have been dumped since MUSIC was last loaded. Further information on the usage of BUFLOG with 2305s can be found in the file $PGM:BUFLOG.S. Under VM/370, these logs are meaningless.

BUFLOG does not support the following direct access devices: 3310, 3340, 3350, and 3370.

Usage:

This program may be run from a terminal by typing BUFLOG or from batch with the control statements:

```
/INCLUDE BUFLOG
```

Privileges Required: VIP.

CDUMP: Interactive Storage Dump Utility

The CDUMP program allows the programmer to inspect and alter areas of main storage. If MUSIC is being run under VM/370, this program also allows an authorized user to dump VM's real storage.

Usage:

It is executed at a terminal by means of the command:

```
CDUMP
```

Parameters are entered separated by commas, with no extra blanks. This corresponds to standard MUSIC FORTRAN NAMELIST input.

A sample run of CDUMP is shown in Figure 17.1. Note that for lines in which all bytes are the same, only one byte is printed.

This utility can also be executed from batch. The control statement format follows.

```
/INCLUDE CDUMP
 control lines
```

Parameters:

START=	Specifies the first main storage location to be dumped. The default is zero.
LEN=	Specifies the number of bytes to be dumped. The default is 16.
M='name'	Specifies a system module name or entry point. Refer to nucleus map produced at NUCGEN time for valid names.
DISPL=	Specifies a value to be added to the one given on the START parameter. Default is 0.
REP=	Specifies the first main storage location to to be altered. The system will then prompt for the hexadecimal data.
CP=	Specifying CP=T will dump VM/370 real storage. Default is CP=F.
RET=	Specifying RET=T will stop the program. The program may also be terminated by the /cancel command in the usual way. Default is RET=F.
HELP=	Specifying HELP=T will print the names and default values of the various parameters.
LINE=	Number of bytes per line of output. The default value is 16. The only allowed values are 16 and 32.

SKIP=x,y After the regular dump is completed, x will be added to the START value and the dump will be performed again. In all, y segments are dumped. This allows the programmer to inspect small sections of main storage at regular intervals.

Several abbreviations and alternate keywords are allowed:

Keyword	Alternatives
START	S, ADDR, AD, A, BEGIN, B, FIRST, F
LEN	LENGTH, L, COUNT, C
DISPL	D, DISP
LINE	None
SKIP	None

Privileges Required:

LSCAN is needed to access the CDUMP file. If memory is to be dumped above the user region, the user must have the INFO or CREAD privilege. The CREAD privilege is required if VM Storage is to be dumped. The user must have VIP code privilege active in order to alter system storage.

```
*Go
cdump
*In Progress
CDUMP. ENTER S=,L=,M= OR HELP=T
?
start=z238
000238  00000398 FF953844 80000000 0000F0F6  *...q.n.. .....06*
?
m='tcs',len=128
833908  00010010 050F0003 00000000 00000065  *................*
833918  00000000 0000001D 00000001 00000000  *................*
833928  050784C7 00004650 00000317 01000040  *..dG...&.......  *
833938  0F400000 00000000 00000000 00000000  *. ..............*
833948  00  *.*
833958  0083FFE8 00000200 00000000 00000000  *.c.Y............*
833968  00000000 C3C6D4D5 F0F1F0F0 F1000000  *....CFMN01001...*
833978  81000000 00000000 0083D328 00000000  *a........cL.....*
?
first=z2c8,l=16
0002C8  700352D8 0F0398D8 0401D790 0101DA40  *...Q..qQ..P.... *
?
f=z352d8,length=32,skip=160,5
0352D8  00010020 00041001 00000000 00000000  *................*
0352E8  2FBE31E4 2FBD0000 C9C8CA00 006D006D  *...U....IH..._._*

035378  00020021 0404940B 00010001 00010006  *......m.........*
035388  2F503176 2F4F0003 C9C8CA00 006D006D  *.&..._..IH..._._*

035418  00030022 01041103 00000000 00010012  *................*
035428  2EE23108 2EE10000 C9C8CA00 006D006D  *.S......IH..._._*

0354B8  00040023 0104140B 00140014 00010002  *................*
035558  2E742E74 30990B01 C9CAC800 006D006D  *.....r..I.H._._*

035558  00050024 0204150B 00070007 00000000  *................*
035568  2E06302C 2E050000 C9C8CA00 006D006D  *........IH..._._*
?
s=z1000
001000  47F0F15A 47F0F162 47F0F54C 47F0F588  *.01!.01..05<.05h*
?
rep=z1000
ENTER REP
?
00000000
ORIGINAL STORAGE:
001000  47F0F15A 47F0F162 47F0F54C 47F0F588  *.01!.01..05<.05h*
NEW STORAGE:
001000  00000000 47F0F162 47F0F54C 47F0F588  *.....01..05<.05h*
?
/cancel
*Terminated
*Go
```

Figure 17.1 - Sample run of CDUMP

CHKDISK: Check System Data Sets for Correct Format

This program is used to detect areas on disk which have not been formatted correctly. It is also a convenient way of reading a data set to test for read errors.

CHKDISK processes only count-key-data (CKD) disk devices. It does not process FBA devices such as 3310 or 3370. Only the first extent of each data set is checked. All tracks of a data set are assumed to have the same number of records. The program checks the count area of each record to verify that the key length and data length match the values from the format-1 DSCB in the VTOC.

Usage:

To run the program, type CHKDISK, then enter:

```
VOL='volume',DSN='data set name'
or
VOL='volume',DSN='ALL'
```

DSN='ALL' processes all data sets on the volume. Initial defaults are the IPL volume (UCB 0) and DSN='ALL'.

Notes:

1. This program does not process FBA volumes (3310, 3370, 933x).

2. Only the first extent of a data set is checked.

3. All tracks of the data set are assumed to have the same number of records. The program checks the count area of each record to see that the key length and data length of the record match the values from the format-1 DSCB (key length and blocksize - key length). The block size in the DSCB is assumed to be *key len + data len.*

Privileges Required: LSCAN, CREAD, DREAD.

CHKFILES: Check All Files

This utility provides a way of finding all files which have errors such as index/directory mismatch (error 67) or directory in error (error 65). The bad files could then be processed by a program such as FIXINDEX or FIXINDEX.AUTO.

CHKFILES does an EXTRACT request for each file in the index. Only private index entries are used; common entries are skipped. For each non-zero MFIO return code from EXTRACT, the program displays a message and writes a record to the file defined on unit 1 (logical record length must be at least 100). By default, unit 1 is defined as new file FILE.ERRORS. You can edit the file $PGM:CHKFILES and change this if desired.

Usage:

Enter CHKFILES to run the program.

Privileges Required: LSCAN.

CMSTAPE: Retrieving CMS Tape Files

This utility program retrieves CMS files which were dumped to a tape by using the CMS TAPE DUMP command. The retrieved CMS files will be saved under separate MUSIC files. It can also be used to scan through the CMS tape to give information of the CMS files contained in it. The information given for each CMS file includes file name, file type, file mode, record format, logical record length, and number of records. Both block sizes of 805 and 4101 bytes on the CMS tape are supported.

The name of the file created for a CMS file is in the format of *fn*.*ft*, where *fn* is the file name, and *ft* is the file type of the CMS file. Sometimes, a fixup character will also be used in the name to make it a valid MUSIC file name, or to make the name unique in the user's library. For more details, see the description of the FIXUP parameter below.

The record format of the file created is similar to that of the CMS file. If the CMS file has a record format of F (fixed), the record format FC (fixed compressed) will be used for the file created. Likewise, a V (variable) record format CMS file will result a VC (variable compressed) record format MUSIC file.

Usage:

The following control statements are used to execute the program. Notice that the input CMS tape is defined with a record format of U (undefined).

```
/FILE 1 TAPE VOL(...) RECFM(U) SHR     (input CMS tape)
/FILE m ...     (if needed by the NAMES=m parameter)
/INCLUDE CMSTAPE
parameters separated by commas (see below)
filename filetype     )
filename filetype     )   if SELECT=TRUE is specified
...                   )
```

Parameters:

INPUT=n	Input unit number of the CMS tape. Unit number 15 cannot be specified as it is used for a work file. Default is INPUT=1.
TAPFIL=n1,n2,...	Specifies the physical tape files in the CMS tape to be processed. A maximum of 99 tape files can be specified. The numbers specified must be in ascending order. If TAPFIL=0 is specified, the entire CMS tape will be processed (the tape will be read until two consecutive EOF marks are encountered). Default is TAPFIL=1, meaning only the first tape file is to be processed.
NAMES=m	If specified, the names of all the created files will be written to unit *m*, one file name per line, starting on column 1. As above, unit number 15 cannot be used. Default is NAMES=0, meaning no names are to be written.
LIST=FALSE	Specifies that the names and the corresponding information of the created files are not to be printed. Default is LIST=TRUE.

REPL=TRUE — Specifies that if a file under the name of *fn.ft*, where *fn* and *ft* are the file name and file type of the CMS file retrieved respectively, already exists, then it will be replaced by the new contents. Default is REPL=FALSE (see below for more details).

FIXUP='x' — Specifies a fixup character 'x' to be used for generating an alternate name for the file to be created if the file under the name of *fn.ft* (see above) already exists and is not to be replaced. The fixup character will be added to the end of *fn.ft* and the resulting name is tried again. At most three such retries are done per file. The fixup character will also be used in the case where *fn* starts with a digit, which, of course, does not conform to the MUSIC file naming convention. In this case, the fixup character will replace the digit. The valid fixup characters are '$', '#', '@', and the letters A to Z. If an invalid fixup character is specified, the default character, which is '$', will be used.

SCANTP=TRUE — Specifies that the input CMS tape will only be scanned for information of the CMS files contained in it. No files will be created. If TAPFIL=0 is specified, the entire tape will be scanned. Otherwise, only the tape files specified in the TAPFIL parameter will be scanned. Default is SCANTP=FALSE, meaning files will be created for the CMS files contained in the tape.

SELECT=TRUE — Specifies that only certain CMS files are to be retrieved from the CMS tape. The file name and the file type of the CMS files to be retrieved are specified after the parameter line. For each CMS file to be retrieved, the file name must be specified on column 1, and the file type on column 10 on a new line. A maximum of 99 CMS files could be specified. Notice the TAPFIL parameter also applies for the tape files to be searched for the desired CMS files. The default is SELECT=FALSE, meaning that all CMS files in the specified tape files are to be retrieved. This parameter will be ignored if the parameter SCANTP=TRUE is specified.

Note: There is a temporary work file defined on unit 15 in the file CMSTAPE. The primary space defined for the temporary work file is 200K bytes. If this work file does not have enough space to process a particular CMS file, just run the program again to retrieve the failing CMS file, by using the SELECT=TRUE parameter, with a bigger work file defined before the line /INCLUDE CMSTAPE in the job set up mentioned above. The work file must have a record format of V (variable).

Privileges Required: None.

CODUMP: User Code Table Dump/Restore

The MUSIC user code table dump/restore program has several functions. It can copy the code table to magnetic tape or a sequential file, as 512-byte records. It can restore tapes created by the dump function. It can produce printed lists of authorized users. It is recommended that the user code table be regularly backed up to tape with this program. This ensures that if any hardware or software problems cause the code table to become unusable, a recent good copy is available. It should be noted that users can frequently change the code table by means of the user profile program (PROFILE). If a backup copy of the code table were to be restored and it was not quite recent, many user's passwords, autoprogs, operating defaults, etc., would be incorrect.

Another use of this program is to condense the code table. As many additions and deletions are made, space in the code table index can become unavailable. This space can be reclaimed by dumping and immediately restoring the code table. This should be done whenever the number of index entries used starts approaching the maximum number of index records. Both these numbers are printed during every dump or restore of the

user code table.

There should be no other users on MUSIC when a code table restore is done. Otherwise user updates to code records (from sign-ons or the PROFILE program) may produce code table errors.

The program creates either one or two magnetic tapes. If two are created, they are identical. The number of tapes to be created is specified by listing the unit numbers onto which the dumps are to be done (UNITS= parameter). To obtain a printout only, simply specify DUMP=T and the appropriate print option. A record length (LRECL) of 512 must be specified on the /FILE statement.

If an error is found while the code table is being dumped, a message is issued, along with a dump of the appropriate code and index records. Serious errors stop the program with the message `PROGRAM TERMINATED ABNORMALLY.` For other errors, the dump continues, but one or more code records may be missing from the output. Restoring the dump should correct the errors, but the missing codes will be lost and should be re-added manually, using CODUPD. If codes are found not to be in increasing alphabetical order, the dump file should be sorted before it is used in a restore job.

Usage:

```
/FILE x TAPE VOL(volnam) BLK(bbbbb) LRECL(512)
/INCLUDE CODUMP
parameters separated by commas (see below)
```

Namelist Input:

DUMP=T or F	F means do not dump. Default is T.
UNITS=x,y	Gives output units for dump. Default is 0.
PRINT=n	Print format for code listing. Default is 1. 0 no listing 1 listing of codes and subcodes
RESTOR=z	Gives input unit for restore. 0 means do not perform restore. Default is 0. Note: a restore cannot be done in the same job as a dump, so DUMP=F must be specified for a restore job.
RSTUCR=T or F	RSTUCR=T restores UCR limits as codes are restored. Default is RSTUCR=T. This option applies only to a restore operation. The UCR limits are stored in the Save Library index and control the allocation of file space. When codes are dumped, the UCR limits are obtained from the SL index and dumped with the code records. During a restore with RSTUCR=T in effect, the UCR limits are set to their values at the time of the dump. A new UCR is created if one does not already exist.

Privileges Required: LSCAN and CODES or MAINT.

CODPRV: List Privileged Userids

CODPRV lists all privilged userids (codes).

Privileges Required: LSCAN and CODES.

CODUPD: Userid Authorization

The Code Table Update Program (CODUPD) enables a privileged user, normally the system administrator, to modify the table of authorized MUSIC users. Each user is identified by a 1-16 character userid (user code), and is represented in the Code Table by a record which contains the user's userid, password, time limits, privileges, and other attributes. The userid (or sign-on code) is also the key used for accessing records in the Code Table.

The CODUPD program processes keyword-type commands, which are entered conversationally from a terminal or read from a file. Only one user may run CODUPD at a time (but see the CODXXX program below). The ADD command adds a new userid record to the table (enabling the user to sign on MUSIC). The DELETE command removes a record from the table. The CHANGE command modifies items in a record. The GET command displays information from a record. Changes made to the code table record of an active user take effect the next time the user signs on; they do not affect the user's current session.

The Code Table is made up of two system data sets: the code table index and the code table itself. The maximum number of userid records in the Code Table depends on the sizes of these two data sets, but is normally a number from 20000 to 50000. Refer to the descriptions of data sets SYS1.MUSIC.CODINDX and SYS1.MUSIC.CODTABL in the section "Direct Access Storage Usage" in this manual.

There is another record associated with each user: the User Control Record (UCR), which resides in the Save Library index. The UCR controls allocation of file space in the Save Library. It contains the limit for the userid's total file space and the limit for the size of each file. There is one UCR per userid; different subcodes of a userid share the same UCR. When a userid is added to the Code Table, CODUPD automatically creates a new UCR for the userid if a UCR does not already exist. When the last subcode of a userid is deleted, CODUPD automatically deletes the UCR. The UCR and UCRFIX utilities can also be used to alter these control records.

The User Profile Program (PROFILE) is a restricted version of CODUPD. It allows the user to modify some of the items in the user's own Code Table record, such as passwords and default time limit. PROFILE is described in the *MUSIC/SP User's Reference Guide*.

Further information on the structure of the Code Table and the internal workings of the update program may be found in *Chapter 18 - System Internals*.

Usage:

To run CODUPD at a terminal and enter commands conversationally, type the command

```
CODUPD
```

When the "?" prompt appears, you can enter any of the CODUPD commands (ADD, GET, etc.) For help enter the command HELP. Terminate the program by entering END.

To pass a single command to the program and then terminate, enter the following:

```
CODUPD command
```

where *command* is the CODUPD command to be executed. For example, entering CODUPD GET ABCD in *Go mode displays the code ABCD.

There is a version of CODUPD called CODXXX, which allows userids to be displayed and changed, but does not allow adds or deletes. The advantage of CODXXX is that multiple users can run it at the same time, and while a normal CODUPD is running. See the topic below.

To have CODUPD read commands from a file (for example CMDFILE), run the following job:

```
/INCLUDE CODUPD5
/INCLUDE CMDFILE
```

CODUPD5 is a version of CODUPD set up to read commands from unit 5 instead of from unit 9. This job can be run at a terminal or on batch.

Privileges Required:

Usually the system administrator userid ($000) is used to run CODUPD. Only one user may run CODUPD at a time. The user must have at least the CODES and LSCAN privileges. Also, as a security precaution, the user is not allowed to add, change, delete or get any userid with a privilege that the user does not have. An attempt to do so will result in an ACCESS DENIED message. An example of this is trying to get a userid that has the VIP privilege when the user has not used the /VIP ON command. Also, a userid with VIP must use the command /VIP ON before using the PROFILE program.

VIP Privilege Notes

The VIP privilege gives the user only the **ability** to request the VIP privilege. That is, when the user signs on, the VIP privilege is not automatic, but has to be requested by means of the command:

/VIP ON

This command may be entered in command (*Go) mode. The actual privilege may be turned off by the command /VIP OFF. This implementation is to ensure that the VIP privilege is not used accidentally (and destructively).

Note that to run the user profile (PROFILE) program, a user whose userid has the VIP privilege must first set /VIP ON. Also, before running CODUPD to access or change userids having the VIP privilege, VIP must be set on.

If VIP userids are run from batch, no special commands are needed to enable the privilege. For this reason, all VIP userids that are to be allowed from batch should be given the RESTR option (restricted access). This ensures that the userid cannot be used except on explicit operator authorization, by means of the console command /CTL CD-ON. It is also recommended that the RESTR option be given to all batch userids having special privileges.

Use of Subcodes

Use of a 1-8 character "subcode" is optional. A subcode allows users to share the same Save Library but have different userid attributes. File ownership is based on the userid excluding any subcode. This part of the userid is referred to as the "ownership id". The total length of the userid cannot exceed 16 characters.

It must be remembered that different subcodes of a userid share the same files and the same UCR (as discussed above), since file ownership is based on the userid without the subcode.

Because of the naming convention used for the /INPUT, Editor log, and Unit 10 Holding files, it is strongly recommended that different people using the same userid be assigned different subcodes. Otherwise, for example, they would share the same /INPUT file (@INPUT.sss, where *sss* is the subcode) and one user entering /INPUT would erase the other user's file or give a FILE IN USE message.

Typical Sign-on Userid Options

The CODUPD command options are described in detail in the topics which follow, but this topic will give you an idea of what is involved in creating a userid.

The following is a typical userid allocation command for a student or project member.

```
ADD ABCD PW(SAMPLE) PRIME(60) NONPRIME(180) DEFTIME(30) -
  BATCH(60) MAX$(300) TOTLIM(200) FILLIM(500) -
  SNGL NOCOM NAME(JOHN SMITH)
```

This creates the userid ABCD with the password SAMPLE. The time limit per terminal job is 60 SU (service units) during prime time (normally defined as 9AM-5PM weekdays), 180 SU during nonprime time, and 30 SU by default. The time limit for batch jobs is 60 SU. The userid has a fund limit of $300 and can store up to 200K on the Save Library (not counting &&TEMP temporary files) and can allocate individual files (including temporary files) as large as 500K. Only one terminal session is allowed at a time (SNGL). NOCOM prevents the user from storing files in the common index. The NAME parameter is optional.

A userid allocation for a research or commercial user or for a project or course supervisor would typically have larger numbers in the time limit, MAX$, TOTLIM, and FILLIM fields.

An instructor who will be using the TRANS$ program would have a userid that includes the SUPV privilege.

Userids for the MUSIC Support Staff would typically include the LSCAN privilege, and NOCOM would be omitted. Other options and privileges would be assigned as needed.

Format of CODUPD Commands

Commands may use columns 1 through 80 of each input line. The general format is:

```
command   keyword(value) keyword(value) ...
```

where:

command — is the command name (for example, ADD, CHANGE, DELETE or GET) or its abbreviation. It must start in column 1 and must be followed by at least one blank.

keyword — is a keyword (or its abbreviation) that identifies a userid option or parameter. The first keyword after the command name is special; it is the userid that the command is referring to. The keywords after the userid can be in any order.

(value) — is the value being assigned to the userid option. It is enclosed in parentheses and must immediately follow its keyword, with no blanks between. Some options require a value and some don't. For example, the password option has a value that is a character string from 1 to 16 characters long, e.g. PASSWORD(ABCDE). The SNGL option (concurrent sessions not allowed) is an example of a keyword not followed by a value.

The value may be a character string, or a number, or in some cases a list of items separated by blanks or commas. For a character string, the string may contain special characters such as blanks, commas, and parentheses (but left and right parentheses must be matched); trailing blanks are ignored but leading blanks are not.

The parameters (i.e. the *keyword*(*value*) items) are separated from each other by 1 or more blanks and/or commas.

If the command is too long to fit in one line, it can be continued on another line by ending the line with a hyphen (-). The "-" character is normally preceded by a blank, to provide a blank between the parameters. A parameter can be split between lines if necessary. Several continuation lines can be used.

Here are some examples of continued commands:

```
ADD ABCD001 PASSWORD(PWWWWWWD) PRIVONLY MAX$(750) -
  NAME(JOE PROGRAMMER) ID(123-4567) -
  TOTLIM(300),FILLIM(400)

CHANGE ABCD001 AUTOPROG(PROG1.SETUP) TAG(THIS USERID IS USED -
  FOR DEMONSTRATION PURPOSES) SNGL
```

A command that starts with an asterisk (*) is treated as a comment (it is not processed). However, comment lines may not be used between lines of a continued command. The COMMENT parameter can be used to add comments to a command.

Commands can be typed in mixed upper and lower case; they are automatically converted to upper case by the system before they are passed to CODUPD, except that values for some keywords (such as NAME and TAG) are left as entered.

Summary of CODUPD Commands and Keywords

The following table lists the command names, their abbreviations, and the keywords which may be used with them. Some of the keywords have abbreviations or aliases. The commands and keywords are described in detail later.

Note that the first parameter of an ADD, CHANGE, GET or DELETE command is always a userid specification. The remaining keyword-type parameters may appear in any order.

Command	Abbreviations	Keywords
ADD	A	userid specification SUBCODE(xxxxxxxx) TYPE(n) PASSWORD(string) BATCHPW(string) PWEXP(n) NEWPW sign-on options: FIXPW, SNGL, etc. operating defaults: NOCOM, etc. privileges: LSCAN, FILES, etc. PRIME(n) NONPRIME(n) DEFTIME(n) BATCH(n) AUTOPROG(filename) ALWAYSPROG(filename) FIRST(n) LANGUAGE(langname) ROUTE(string) TERMINAL(string) ITABS(n1 n2 n3 ...) OTABS(n1 n2 n3 ...) TAB(x)

		BACKSPACE(x) IDLE(x) HEX(x y) MAX$(n) ADD$(n) NOW$(n) TRK/UDS(n) TOTLIM(n) FILLIM(n) XSESSIONS(n) START(yyyy/mm/dd) EXPIRY(yyyy/mm/dd) NAME(string) ID(string) TAG(string) LIKE(userid) COMMENT(string)
CHANGE	C, CH	Same keywords as for the ADD command, except that TOTLIM, FILLIM, and LIKE are not allowed.
GET	G	userid specification SUBCODE(xxxxxxxx) SHOW$ SHOWSPACE BRIEF DUMP COMMENT(string)
DELETE	DEL	userid specification DELUCR DELMAILBOX COMMENT(string)
END	(none)	
HELP	(none)	
.LIST	(none)	
.NOLIST	(none)	
.PRINT	(none)	
.NOPRINT	(none)	

General Information on CODUPD Commands

ADD Creates a new record in the Code Table, for the specified userid. The userid must not already exist in the table. If a subcode is added, the total length of the new userid cannot exceed 16 characters. Default values are used for any fields which are not specified on the ADD command (see below).

CHANGE Changes the contents of an existing record in the Code Table. The userid and subcode in the

record cannot be changed, since they make up the 16-character key by which the record is accessed, via the Code Table Index. Only the specified fields are changed.

GET — Displays information from a Code Table record. Options (such as SHOW$, SHOWSPACE) can be used to select specific information. If no options are used, the entire userid record is displayed in a readable format. The display includes some fields from the userid record that do not correspond to command parameters, such as the date of creation of the record, date and time of last sign-on and last batch job, date of last sign-on password change, and date of last batch password change. See the sample output below.

DELETE — Removes a record from the Code Table. The record is usually displayed as it is deleted.

END — Terminates the CODUPD program.

HELP — Displays information on how to use CODUPD. The text is taken from the file $COD:@HLP.CODUPD. (The corresponding file for the PROFILE program is $HLP:@GO.PROFILE.)

.LIST — Causes subsequent command input lines to be displayed on unit 6 as they are read. The default is .LIST for batch jobs and CODUPD5, and .NOLIST for CODUPD terminal jobs.

.NOLIST — Suppresses the display of command input lines.

.PRINT — Causes the userid record to be displayed after any successful ADD, CHANGE or DELETE commands that follow the .PRINT command. This display is always omitted when the command specifies a range of userids. Default is .PRINT for CODUPD and .NOPRINT for CODUPD5.

.NOPRINT — Suppresses the automatic userid record display.

Return Codes for CODUPD and PROFILE

0 — No errors or unusual conditions.
4 — One or more commands were invalid or unsuccessful.
8 — CODUPD is already in use; try again later.
16 > — A fatal error occurred. Note that some commands preceding the error may have been successful. Also, a large return code results if the program is ended by /CANCEL instead of by the END command.

Description of CODUPD Keywords

In the descriptions that follow, any abbreviations or aliases for the keyword are shown at the right. Usually the shortest abbreviation is shown; in most cases, intermediate forms are also allowed.

Many option-setting keywords have a negative form obtained by putting NO in front of the keyword. For example, SNGL, NOSNGL; FILES, NOFILES. One of them turns the option on and the other turns it off.

Some keywords define numeric limits, such as limits on job time or file space. For these, a value NOLIMIT can be specified instead of a number. Abbreviations NOLIM and NL can be used. For example, PRIME(NOLIMIT), MAX$(NL), TOTLIM(NOLIM). In the userid record, NOLIMIT is represented by the value -1.

Many options can be removed or reset to their normal values by specifying a null value: keyword(). For example, to remove a NAME field, specify NAME(), which sets the field to all blanks. OTABS() removes

all output tabs. EXPIRY() removes the expiry date. TAB() undefines the input tab character, etc.

userid specification

This must be the first keyword on an ADD, CHANGE, GET or DELETE command. It is a 1-16 character userid.

Wild characters * (matches any group of 0 or more characters) and ? (matches any single character) can be used in the userid, except on the ADD command, to refer to several userids.

A range of userids, e.g. XX01-15 or XXAA-BC, can also be specified. See the topic below. To avoid confusion with a userid range, do not define a subcode that starts with "-".

The userid must start with a letter (A to Z), or digit (0 to 9), or one of the four special characters, $, @, _, or #. Avoid assigning userids starting with the $ sign as they may conflict with the userids used to store system files. (The topic "Usage of $ Userids on MUSIC" in Chapter 22 - System Programming contains a list of these $ userids.)

SUBCODE(xxxxxxxx) SC SUBC

Specifies a 1 - 8 character subcode. Letters, digits, and special characters can be used, although special characters other than $ # @ / _ and . should be avoided. If no subcode is wanted, omit the SUBCODE parameter on the ADD command.

SUBCODE(*) can be used on a GET or CHANGE command. It means that the command will be applied to each subcode of the userid. In general, wild characters * and ? can be used in the subcode, except on the ADD command.

A range of numeric subcodes, e.g. SUBCODE(001-025), can also be specified. See the topic below.

TYPE(n)

A type number (0 to 255) can be assigned to each userid in the Code Table. It is copied to the XTCB at sign-on. The intention is to use the type number in various authorization files to specify e-mail access, e-mail retention period, access to ADMIN, etc. It should indicate the general category of user, but NOT detailed information such as course number or department number.

Each site can choose their own numbering scheme, but a suggested scheme is given below.

If no type is specified when the userid is created, and for userids that exist from before, the type number is 0 (general user).

CODUPD Command Examples:

```
ADD ABCDE TYPE(30)
CHANGE XYZ TYPE(40)
```

A TSUSER call is provided to get the type number for the current userid:

```
CALL TSUSER(15,NUM)
```

Field in Code Record:	$CDTYPE	(1 byte at record + x'18')
Field in XTCB:	XTBITYP	(1 byte at XTCB + x'3F')

(Note that the
above field lengths and locations may change in the future.)

Suggested Userid Types:

0- 9	Student; general user
10-19	Anonymous access userid (e.g. CWIS)
20-29	Demo or free userid; external or guest user
30-39	Course instructor; group supervisor; academic
40-49	Special user; university officer; company executive
50-59	Research user
60-69	Commercial user
70-79	University or company administrator; non-academic
80-89	Computing Center administrator/staff
90-99	Computing Center operations; system maintenance
200	MUSIC system administrator (userid $000 or equivalent)
210	MUSIC system userid ($xxx)

PASSWORD(string) PW PASSW

The terminal (sign-on) password, from 1 to 16 characters. Letters, digits and special characters can be used. It must not contain imbedded blanks.

PASSWORD() results in no password being required at sign-on time. This should be used with caution.

PASSWORD(*NOLOGON) defines a special password that prevents sign-on except as a BTRM.

It is strongly recommended that all users be assigned passwords, and that users be encouraged to change their own passwords frequently (using the PROFILE program). The date that each of the passwords was last changed is remembered in the userid record and is displayed by the GET command.

PWCASENS

The Profile option PWCASENS can be used to make the user's sign-on password case sensitive. When this option is set, the user's sign-on password is considered to be mixed case, and the exact upper or lower case letters must be entered at sign-on time. WARNING: When you set PWCASENS, you should also set a new password, so that you know the exact case of the letters in the password.

BATCHPW(string) BPW

The password for batch jobs, from 1 to 8 characters. Letters, digits, and special characters can be used. A batch password may or may not be required, depending on the installation. BATCHPW() results in no batch password. A batch password should be specified if the user will be submitting jobs to MUSIC batch. If no batch password keyword is specified on an ADD command, the batch password is set to the same value as the terminal password.

PWEXP(n)

A password lifetime can be specified for each userid, if desired. *n* is a number of days. After *n* days have passed, the user must choose a new password, which is valid for another *n* days. This feature improves security by requiring users to change their passwords regularly. Different userids can have different password lifetimes. Unlimited lifetime can be specified by PWEXP(0). The auto-program that enforces password changes is $COD:PWEXP.

NEWPW

This is used on an ADD or CHANGE command for setting a new terminal password. After the command is entered, you will be prompted to enter the new password in a non-display field.

Sign-on Options

Various keyword options can be set to control the sign-on procedure. They are:

SNGL — Only one terminal at a time can be signed on to this id. This prevents the userid from having multiple concurrent sessions on different terminals. If the user tries to sign on and the userid is already signed on, the user is given the option of cancelling the previous session.

FIXPW — The user is not allowed to change the terminal or batch password.

NONCAN — (or NOCAN or NOCANA) The user may not cancel the auto-program (AUTOPROG) that runs after sign-on. Also, the user may not change the auto-program name.

UNAME — A user name parameter must be entered at sign-on time. (This is not the CODUPD NAME parameter.)

DISABLE — Marks the userid record as not authorized, but does not delete it from the Code Table. The user will not be allowed to sign on or run batch jobs. The userid can later be re-authorized by the ENABLE option.

RESTR — The userid cannot be used from batch unless the operator specifically allows it by entering the console command "/CTL CD-ON". This can be used as a security precaution for privileged userids.

ANYSC — Allows use of any subcode. Applies to 4-character userids without a subcode.

The opposites of the above options are: NOSNGL, NOFIXPW, CAN (or CANA), NOUNAME, ENABLE, NORESTR.

Operating Defaults

These keyword options control various things during the user's session or job.

NOINPROG — This option suppresses the `*In progress` message that is displayed at the beginning of terminal jobs.

NOINVIS — Makes userids visible to programs like FINGER. Use the INVIS option if you do not wish other users to know that this userid is signed on.

SLASH — The slash (/) character is required at the beginning of each command in *G0 mode. E.g. /LIST must be typed instead of LIST.

EXEC — The implied /EXEC command is not allowed. I.e. /EXEC filename must be typed instead of *filename*.

PRIVONLY — (Private only.) The user is not allowed to create files which are accessible by other userids. The options PUBL and SHR cannot be used when saving files. Also, new UDS files must be private.

NOCOM — The user is not allowed to create files in the common index. This is less restrictive than PRIVONLY. SHR can be used when saving files, but not PUBL. If another user wants to read a file created as SHR, this user must specify the owner's userid on the front of the file name, as in /LIST ABCD:FILEXYZ.

FIXALPROG — The user is not allowed to change the always program name. Also, the command /CANCEL ALL cannot be used to cancel an always program. Abbreviation FIXALP may be used.

NOMULTI Access to multi-session is restricted to that initiated from programs. Multi-session commands and functions keys are disabled.

SIGNOFF User is automatically signed off when auto program finishes.

JA1 to JA8 These 8 options refer to the Submit Facility. If the model JCL for a submit processor has a nonzero access restriction number n (1 to 8), then option JAn must be set in the user's userid in order for the user to submit jobs to that processor. See *Chapter 8 - Job Submission and Retrieval Programs* in this manual for a discussion of this topic.

The opposites of the above options are: INPROG, INVIS, NOSLASH, NOEXEC, NOEDMSG, NOPRIVONLY, COM, NOFIXALPROG, MULTI, NOSIGNOFF, NOJA1, ..., NOJA8.

Privileges

These keywords grant special privileges to the user, allowing this person to do things that nonprivileged users are not allowed to do. Many of the system maintenance programs require one or more of these privileges. The privileges are listed in approximate order of increasing power.

SUPV Save Library supervisor privilege. The user can create, read, modify and delete any file or UDS file, provided the first 2 characters of the owner's userid match the first 2 characters of the user's userid. Also, the user can run the TRANS$ program to transfer funds and assign new passwords. For example, userid XY00 with the SUPV privilege can supervise any userid starting with XY. This is useful for courses and project groups.

LSCAN The user can inspect private files, and use the CODE parameter on the /LIBRARY command. Note that most system utility programs have private executor files, and therefore at least the LSCAN privilege is required in order to run them.

SYSCOM Allows the user to send data directly to a VM virtual punch or printer, which may be spooled to MUSIC, some other virtual system, or RSCS. Without this privilege, only the standard programs (SUBMIT, VMSUBM, etc.) can be used to spool data.

CREAD Lets the user inspect any location in main storage, even if the storage is fetch protected. VM's real main storage can also be read (via the CDUMP program) if the MUSIC virtual machine is authorized to do so.

INFO The user can run general system information programs.

FILES The user can create, read, modify and delete any file or UDS file.

SYSMNT The user can run some system programmer maintenance programs. For example, the Load Library and System Subroutine Library can be updated.

CODES Access to the Code Table is allowed. The user can run the CODUPD program.

DREAD The user can read from any location on disk.

MAINT The userid can run computer room maintenance programs, such as accounting and file archive utilities.

VIP The ability to modify any location on disk, to modify any part of MUSIC's main storage, and to run any utility program not provided for by the above privileges. For each terminal session, this privilege is active only after it has been specifically

requested by the user typing /VIP ON.

JOBLIM This option allows a program to increase its execution time limit.

PRIV12 to PRIV15
These keywords correspond to privilege bits which are not currently used. Only the userid $000 has these privileges. (There should be at least one userid in the system, namely $000, that has all possible privileges, otherwise additional future privileges could not be assigned.)

To remove a privilege, simply type NO before the corresponding keyword, as in NOSUPV, NOLSCAN, etc.

It is recommended that privileges be granted and used with great care. The security of the entire system disappears if privileges are granted without discretion. No privilege (except possibly SUPV), should ever be assigned to personnel not directly responsible for the maintenance of MUSIC.

The FILES, LSCAN and SUPV privileges allow a user to have access to files belonging to other users. Refer to the topic "Working with Files" in *Chapter 20 - File System* in this manual for techniques of how to access and work with files belonging to other userids.

PRIME(n) PRI
Maximum number of service units (SU) per terminal job during prime time. Prime time normally means 9:00 AM to 5:00 PM on weekdays (Monday through Friday). PRIME(0) means that the user cannot sign on during prime time. PRIME(NOLIMIT) can be specified. NOLIMIT can be abbreviated NOLIM or NL, as in PRI(NL). The number *n* in this and the other time limit parameters cannot exceed 32000.

The definition of prime time can be changed by changing the system module URMON, near labels TPRON and TPROF.

A service unit is a measure of work done by a job, and is defined elsewhere in this manual. On some systems 1 SU is equal to 1 second of processing unit processing.

NONPRIME(n) NONPR
Maximum number of service units (SU) per terminal job during nonprime time (see the description of PRIME above). NONPRIME(0) means no access, and NONPRIME(NOLIMIT) means no limit on job service units.

DEFTIME(n) DT DEF
Default time limit, in service units, for terminal jobs that do not specify a time limit via the "/SYS TIME=m" control statement. The number *n* must be from 1 to 32000, or NOLIMIT.

BATCH(n) BAT
Maximum number of service units (SU) that any one batch job is allowed to use. BATCH(0) means that the user may not run batch jobs. BATCH(NOLIMIT) may be specified.

AUTOPROG(filename) AUTO PROG
Specifies the name (1 to 22 characters) of a file that is to be executed automatically each time the user signs on. The auto program can be made non-cancellable by the NONCAN option (described above under "sign-on options"), if desired. To remove the auto program option, specify AUTOPROG().

A sample autoprogram is provided for doing extra user authentication at sign-on time. See $pgm:guard1.sample.s . It uses pre-established key values known only to the user's workstation and to the autoprogram. The autoprog sends a random, time-of-day dependent number (a "nonce") to the

workstation. The user runs a program on the workstation to combine the nonce with the fixed keys, giving a one-time password, which he or she enters and which the autoprog verifies. The only values transmitted over the communications channel are the nonce and the one-time password, both of which are different for each sign-on. This make connection over the Internet more secure.

ALWAYSPROG(filename) ALWAYS ALPROG
The name (1 to 22 characters) of a file that is to be executed automatically every time the terminal would otherwise be in *Go mode. This means that whenever a terminal job ends and no additional job is scheduled, the always program is run. For example, the always program could display a menu of available functions and let the user choose which one to do next. To remove this option, use ALWAYSPROG(). Note that the PROFILE program allows the user to change (or remove) the always program name, unless the FIXALPROG option is specified for the userid (see "operating defaults").

FIRST(n)
The number n is from 1 to 255 and identifies a one-time program that is to be run the first time the user signs on. For example, this could be a system program that prompts for information about the new user and stores it in a log file. The program id numbers are defined in the system SIGNON module. If an auto program also exists, the one-time program is run first. To remove this option, specify FIRST(0). See "Defining a First-Time Program" in *Chapter 22 - System Programming* for additional information.

LANGUAGE(langname)
This specifies the national language for messages, etc. Not all applications support all languages. If an application does not support the language you request, it uses English. National language names are: English, French, Kanji (Japanese), Portuguese, Spanish, German. Other language names MAY also be accepted. Most language names have a 2 or 3-character abbreviation, for example LANG(ENG). To remove the LANGUAGE option, specify LANGUAGE() or LANGUAGE(DEFAULT), both of which use the site-dependent default language.

ROUTE(name)
A default route name (1 to 8 characters) to be used by programs such as SUBMIT. To set no default route name in the userid record, use ROUTE(), which sets the name to all blanks.

TERMINAL(name) TERM
Defines a default terminal type, which is a name from 1 to 8 characters long. Terminal type names are defined in the system module TRMCTL, and identify various specific terminal models or classes of terminals. Examples are TERM(3270), TERM(TTY) and TERM(DECII).

Some userid options, such as BACKSPACE and OTABS, are defined for a specific terminal type. At sign-on time, MUSIC will check the type of terminal used against the one defined with this option. Only if they match will the options be set as defined in the userid record. This prevents, for example, the tab settings for a 2741 terminal from being used when the user signs on using an ASCII (TTY) type terminal.

The TERMINAL parameter can also used to define a terminal subclass. to enable the system to distinguish between a 72-character TTY terminal and a 132-character TTY terminal.

Refer to the *Chapter 7 - Terminal Configuration and Tailoring* and *MUSIC/SP User's Reference Guide* for more information.

The terminal type is removed from the userid record by TERMINAL().

ITABS(n1 n2 n3 ...)
Defines the default input tab positions. From 1 to 80 column numbers (separated by blanks or commas) can be specified. Each is a number from 1 to 80. Input tabs make it easier to enter data in

specific columns, by pressing a tab key or a logical tab character to space to the next tab position in the input line. The TAB parameter can be used to define a logical tab character. To remove all input tabs, use ITABS().

OTABS(n1 n2 n3 ...)
Defines the default output tab positions. From 1 to 11 column numbers (separated by blanks or commas) can be specified. Each is a number from 1 to 254. MUSIC uses these tab settings to speed up output to the terminal, provided the user's terminal matches the terminal type specified in the TERMINAL parameter. It is up to the user to set the physical tabs correctly. To remove all output tabs, use OTABS().

TAB(x)
Defines an input character that MUSIC is to interpret as a tab operation (i.e. space to the next tab position in the input line). *x* is a single character, or the 2-digit hexadecimal value of a character. Any character other than a letter, digit, slash, blank, equal sign, or comma can be used. For example: TAB(&), TAB(35), TAB(E0). To undefine the tab character, use TAB(), which is equivalent to TAB(00).

BACKSPACE(x) BS BACKSP
Defines an input character that MUSIC is to interpret as a backspace operation (i.e. delete the preceding character). *x* is specified as in the TAB parameter (see above).

IDLE(x)
Defines a character that MUSIC is to use as the idle character for output to the terminal. *x* is specified as in the TAB parameter (see above).

HEX(x y)
Defines an input replacement character. When MUSIC sees the character *x* in a terminal input line, it will replace it with *y*. *x* and *y* are specified as in the TAB parameter (see above) and are separated by a blank or commas. *y* may be any of the 256 possible characters. Example: HEX(&,02) means that "&" will be interpreted as 02 (hex). To remove this option, enter HEX().

MAX$(n)
This sets the limit for the number of dollars of computing funds that the userid is allowed to use. The number may be any integer 0 or more, or NOLIMIT.

When a new userid record is created, its funds usage number (the NOW$ field) is set to 0. Each time the NOWDOL accounting program is run, the userid's usage charges since the previous NOWDOL run are added to the NOW$ field. When the NOW$ field reaches or exceeds the MAX$ limit, the user is not allowed to sign on or run batch jobs, until a higher MAX$ limit is set or the ADD$ parameter is used to increase the limit. A supervisor userid can also run the TRANS$ program to change the MAX$ limit.

The MAX$ and NOW$ fields can be displayed by the SHOW$ option of the GET command. NOW$ is shown as the amount of funds *used* and MAX$ as the *limit*. Note that, internally, the MAX$ field of the userid record is in dollars, while the NOW$ field is in cents. The values are always displayed as $n.nn amounts by SHOW$.

ADD$(n)
Adds *n* dollars to the userid's MAX$ limit. The number *n* may not exceed 1 million. To set unlimited funds, specify MAX$(NOLIMIT).

NOW$(n)
(This parameter is normally not used, since only the NOWDOL and TRANS$ programs are supposed to update the NOW$ field.) This sets the userid's funds usage number (the accumulated usage charge) to a specific value. The value can be specified as a number of dollars, or as a

dollar.cents amount. For example: NOW$(55), NOW$(129.17).

TRK/UDS(n)

Gives the maximum size, in tracks, for new UDS files created by the user. This limit does not apply to temporary (&&TEMP) UDS files. The number may be from 0 to 32000, or NOLIMIT. A limit of zero prevents the user from creating any permanent UDS files.

TOTLIM(n)

Specifies the total space limit in the user's UCR (User Control Record). This is the limit, in units of 1K = 1024 bytes, on the total space of all the user's files, not counting temporary files. TOTLIM(NOLIMIT) gives unlimited space.

This parameter can be used only on an ADD command. If a UCR already exists for the userid, the UCR value is set to the larger of the existing value and *n*. To change the limit after the userid is created, use the UCR program.

FILLIM(n)

Specifies the individual file limit in the user's UCR (User Control Record). This is the maximum amount of space, in units of 1K = 1024 bytes, that can be used for any one file in the user's Save Library, including temporary files. FILLIM(NOLIMIT) allows unlimited size files, up to the system-wide limit per file (normally 4000K).

It is recommended that FILLIM never be set less than 300K, since some of the compilers require temporary files up to this size.

This parameter can be used only on an ADD command. If a UCR already exists for the userid, the UCR value is set to the larger of the existing value and *n*. To change the limit after the userid is created, use the UCR program.

XSESSIONS(n) XSES

Defines the maximum number of extra sessions per terminal. The value *n* must be 0 to 250, or NOLIM. The default on an ADD command is 3.

START(yyyy/mm/dd)

Gives the starting date (year/month/day) for the userid. The user will not be able to sign on or run batch jobs before this date, even though the userid record exists. *yyyy*, *mm*, and *dd* must be 4, 2, and 2 digits respectively. For example, START(1985/07/31). To remove a starting date, specify START().

EXPIRY(yyyy/mm/dd) EXP

Gives the expiry date (year/month/day) for the userid. The user will not be able to sign on or run batch jobs after this date, even though the userid record still exists. *yyyy*, *mm*, and *dd* must be 4, 2, and 2 digits respectively. For example, EXPIRY(1985/12/25). To remove an expiry date, specify EXPIRY().

NAME(string)

The user's name, up to 48 characters. The character string is stored as is, without any processing or checking, and is optional. It may contain blanks and special characters. To set the field to all blanks, use NAME().

ID(string)

A user identification or account number, up to 16 characters. It could be used to store the user's telephone number, for example. The character string is stored as is, without any processing or checking, and is optional. It may contain blanks and special characters. To set the field to all blanks, use ID().

TAG(string)

A descriptive field, up to 64 characters, that could be used indicate the purpose or origin of the userid. The character string is stored as is, without any processing or checking, and is optional. It may contain blanks and special characters. To set the field to all blanks, use TAG().

LIKE(userid)

This parameter is valid on an ADD command only. It specifies an existing userid whose Code Table record is to be used as a model for the userid being created. All fields (except for those listed below) are copied from the existing record, then any overriding values specified on the ADD command are filled in. UCR limits are also copied from the model userid's UCR.

The following userid record fields are not copied from the model record: userid, date of creation (set to the current date), NOW$ (set to zero), date and time of last sign-on and last batch job (set to zero), and dates of last password changes (set to zero).

Examples:

```
ADD ABCD LIKE(EFGH) PW(ANEWPW)
ADD XXAA002 LIKE(XXAA001) TAG(MY 2ND SUBCODE)
```

COMMENT(string) CMT

The character string is a comment field on the command, and is not stored in the userid record. The string may be of any length.

SHOW$ PRINT$ $

This option on the GET command displays the userid's dollar amounts (the MAX$ and NOW$ fields). NOW$ is shown as the amount of funds *used* and MAX$ as the *limit*.

SHOWSPACE SHOWSP PRINTSPACE PRINTSP

This option on the GET command displays the values from the userid's UCR: the current total Save Library space, the limit on the total space (TOTLIM), and the limit per file (FILLIM).

BRIEF BR

This option on the GET command displays the user's ID and NAME fields.

DUMP

This option on the GET command displays the userid record in the form of a hexadecimal dump. The block numbers of the code and index records are also displayed. These are the block numbers that could be used in the SYSDMP utility program to access these records in the code and index data sets.

DELUCR

This option on the DELETE command deletes the UCR (User Control Record), even if other subcodes still exist. This option should not be used in most cases.

DELMAILBOX

This option on the DELETE command also deletes the user's mailbox file.

Default Settings for New Userid Records

Figure 17.2 shows the default options for an ADD command (when the LIKE parameter is not used). Any options specified on the command override these.

```
PASSWORD(MUSIC) BATCHPW(same as terminal password)
sign-on options: NOSNGL NOFIXPW CANA NOUNAME
    ENABLE NORESTR
operating defaults: INPROG NOSLASH NOEXEC NOEDMSG
    NOPRIVONLY COM NOFIXALPROG NOJA1 NOJA2 ... NOJA8
no privileges
PRIME(180) NONPRIME(180) DEFTIME(60) BATCH(180)
AUTOPROG() ALWAYSPROG() FIRST(0) ROUTE()
TERMINAL() ITABS(7 73) OTABS() TAB() BACKSPACE()
IDLE() HEX() MAX$(NOLIMIT) NOW$(0) TRKS/UDS(NOLIMIT)
TOTLIM(10000) FILLIM(NOLIMIT) START() EXPIRY()
NAME() ID() TAG() XSESSIONS(5) TYPE(0) PWEXP(0)
```

Figure 17.2 - New Userid Record Default Settings

An installation may change the above defaults by modifying the subroutine DEFLT in the CODUPD program. For example, coding could be added to set different defaults for different classes of userids.

Code and Subcode Ranges

A group of several code records can be referred to in a single CODUPD command by specifying a range of codes and/or subcodes. The ADD, CHANGE, GET or DELETE operation is applied to each code record in the range.

A range of codes is specified as AABB-CC, where all the codes start with AA (1 or more characters), the first code is AABB, and the last code is AACC. The sequence of codes can be numeric (each code ending in a 2-digit or 3-digit number) or alphabetic (each code ending in a 2-digit base-36 *number* whose *digits* are chosen from A, B, ..., Z, 0, 1, ...,9).

An example of a numeric code range is XY05-20, which refers to codes XY05, XY06, ..., XY20. With this method, up to 100 codes can be referred to per command, or up to 1000 codes if 3 ending digits are used.

An example of an alphabetic code range is XYAA-CF, which specifies the 78 (=2*36+6) codes XYAA, XYAB, ..., XYAZ, XYA0, XYA1, ..., XYA9, XYBA, XYBB, ..., XYBZ, XYB0, ..., XYB9, XYCA, XYCB, ..., XYCF. The lowest starting suffix is AA and the highest ending suffix allowed is 99, which gives 1296 (=36*36) combinations.

A subcode range must be numeric, using subcodes 000 to 999, and is specified as SUBCODE(nnn-mmm) For example, SUBCODE(010-135) refers to the subcodes 010, 011, 012, ..., 135.

The automatic code record display for an ADD, CHANGE or DELETE (see the .PRINT command above) is always omitted when a range of codes or subcodes is specified.

Examples:

```
ADD TS00-25 PASSWORD(GREEN)

CHANGE RHMM SC(000-015) PRIME(600)

GET XXBA-D9 SHOW$

DELETE PQMA-M9 SUBCODE(MUS)
```

Limited Access to the Code Table by CODXXX

Only one user at a time is allowed to run CODUPD to update the Code Table. This can be a nuisance if the code administrator is in the middle of a long CODUPD run and some other system user wants to display a code record and, for example, wants to make a minor change to it.

For this reason, a special version of CODUPD is available, called CODXXX, which allows concurrent access to the code table but does not allow ADD or DELETE commands. GET and CHANGE commands are allowed in CODXXX, but records cannot be added to or removed from the Code Table. CODXXX reads commands conversationally from the terminal.

Sample CODUPD Session

The following sample session shows a code being created, changed, displayed, and deleted. The lines entered by the user are preceded by "?" (MUSIC's conversational read prompt).

```
codupd
*In progress

CODE TABLE UPDATE  --  TUE DEC 18, 1984  16:19
?
add abcd sc(001) pw(mypassw)
  USERID=ABCD001         FILE OWNERSHIP ID=ABCD        TYPE=0
  ID=                   NAME=
  PASSWORD=MYPASSW    BATCH PASSWORD=MYPASSW
  NO PRIVILEGES
  TIME LIMITS (IN SERVICE UNITS):
    PRIME=180  NONPRIME=180  BATCH=180  DEFAULT=60
  MAX NUMBER OF EXTRA SESSIONS PER TERMINAL:    5
  PASSWORD CAN BE CHANGED BY USER
  AUTOPROG:  (NONE)
  INPUT TABS:     7  73
  NO OUTPUT TABS
  FUNDS ($):        0.00 USED,  NO LIMIT
  SAVE LIBRARY:  TOTAL =     0K   LIMIT = 10000K   MAX/FILE =  4000K
  MAX TRACKS PER DATA SET (UDS) AT ALLOCATION:  NOLIMIT
  CREATED 1984/12/18  (YEAR/MONTH/DAY)
  LAST SIGN-ON: 0                  LAST BATCH JOB: 0
  LAST PASSWORD CHANGE:  TERMINAL PW 0            BATCH PW 0
  PASSWORD LIFETIME:  NO LIMIT
  USERID OF CREATOR:  $000000

?
.noprint
?
c abcd001 autoprog(check.test) itabs(10 20 30) -
ENTER CONTINUATION LINE
?
name(J. Smith) id(123-456-789)
?
g abcd001
  USERID=ABCD001         FILE OWNERSHIP ID=ABCD        TYPE(0)
  ID=123-456-789       NAME=J. Smith
  PASSWORD=MYPASSW    BATCH PASSWORD=MYPASSW
  NO PRIVILEGES
```

```
  TIME LIMITS (IN SERVICE UNITS):
    PRIME=180  NONPRIME=180  BATCH=180  DEFAULT=60
  MAX NUMBER OF EXTRA SESSIONS PER TERMINAL:    5
  PASSWORD CAN BE CHANGED BY USER
  AUTOPROG:  CHECK.TEST               (CANCELLABLE)
  INPUT TABS:    10  20  30
  NO OUTPUT TABS
  FUNDS ($):        0.00 USED,  NO LIMIT
  SAVE LIBRARY:  TOTAL =     0K   LIMIT = 10000K   MAX/FILE =  4000K
  MAX TRACKS PER DATA SET (UDS) AT ALLOCATION:  NOLIMIT
  CREATED 1984/12/18  (YEAR/MONTH/DAY)
  LAST SIGN-ON: 0                  LAST BATCH JOB: 0
  LAST PASSWORD CHANGE:  TERMINAL PW 0           BATCH PW 0
  PASSWORD LIFETIME:  NO LIMIT
  USERID OF CREATOR:  $000000

?
g abcd001 brief $ showsp
  ABCD001   123-456-789       J. SMITH
  FUNDS ($):        0.00 USED,  NO LIMIT
  SAVE LIBRARY:  TOTAL =     0K   LIMIT = 10000K   MAX/FILE =  4000K
?
del abcd001
?
get abcd001
USERID ABCD001 NOT FOUND
?
end
*End
*Go
```

CONLOG: Incore Console Log Utility

This utility displays the current contents of the console log that is retained in memory. If you ar on a 3270-compatible terminal, you are placed into BROWSE to view the CONLOG output. An optional parameter can be specified to limit the list to a fixed number of entries. Only the most recent entries are listed. The utility always lists console entries in chronological order.

Several lines of addresses are printed before the console log listing. These addresses occupy the 20 bytes prior to the start of the console log. On each log entry, a sequence number, the start address of the entry, its length, and the message itself are printed. (See also CONSOLE program in *Chapter 4 - The System Console*.)

Usage:

From command mode:

```
CONLOG  nn  NOFS
```

Where *nn* is the number of lines to display and *NOFS* forces non-full-screen mode of operation.

Privileges Required: LSCAN and CREAD.

COUNTS: List System Counters

The program COUNTS dumps the system statistics counters. These are counts maintained by various system routines. The meanings of the individual counters can be found by listing the file XCOUNTS.

Usage:

From a terminal this program is run by means of the command COUNTS.

Note: These counters are reset at MUSIC IPL time (and only then).

Privileges Required: CREAD.

DSACT1, DSACT2: User Data Set Accounting

This utility is used in conjunction with the ACTDMP utility to provide billing information for User Data Set (UDS) files. Typically this utility is run weekly to produce input to the installation's billing program.

The records produced by this utility give the size and length of time each data set has been allocated. If this data set was allocated before the utility was last run, then the time reflects the amount of elapsed time since the last run of this utility. The ACTDMP utility produces records in this same format when a UDS file is deleted. Together these utilities maintain records of the usage of UDS files. Consult the description of the ACTDMP Utility for the output record format.

(Temporary data sets allocated without a data set name are not charged. These temporary data sets come out of a pool of system space in the data set SYS1.MUSIC.SCRATCH. There is a limit to the amount of space any user can get from this pool. These temporary data sets are only used during the duration of a single job. See the *MUSIC/SP User's Reference Guide* for information regarding the use and size limitations of these data sets.)

The DSACT utility is run in two parts. The first part (DSACT1) scans the UDS volumes and writes the data set names to a file. The second part (DSACT2) uses the information produced by the first step to update the date last billed on the VTOCs of the UDS packs and also produces the accounting records. The format of these accounting records (MD1) is given in the description of the ACTDMP utility.

This utility is separated into two parts for the sole purpose of allowing either part to be rerun if necessary. The second part DSACT2 must not be run until the first part has completed successfully. Although it is unlikely that either step will fail, this procedure does handle the case.

When these utilities are run for the first time, the installation must allocate a file with a LRECL of 65, RECFM(F), and number of records (NREC) greater than the number of data sets on all the UDS volumes. Make this data set private to protect systems security. When DSACT1 is run for this first time, you must specify the NOCHEK=T option.

When DSACT2 is run for the first time, the message `INVALID NUMBER OF MINUTES` may be printed.

To run the DSACT1 section use the following control statements. The /FILE statement must point to the file you allocated to contain the data set names. The logical record length of this file is 65. The record format must be F. The volume names of the UDS packs are given on the option statement. (DSACT1 will check to ensure that at least 12 hours have elapsed since it was last successfully run. This checking can be suppressed

by using the option NOCHEK=T. Normally you will not use this NOCHEK option.)

```
/FILE 1 NAME(namesfile) OLD NORLSE
/INCLUDE DSACT1
VOL='udsnm1','udsnm2',....
```

To run DSACT2 use the following:

```
/FILE 1 NAME(namesfile) OLD NORLSE
/FILE 2 NAME(output) NEW(REPL) SP(100)
/INCLUDE DSACT2
OUT=2
```

The OUT=n parameter specifies the I/O unit number where the accounting records are to be placed. Do not use n=1. n=7 can be used to punch the records. An option of PRINT=F can be given to suppress the printing of the data set information.

VERTAP=T can be used to check that the correct output tape is mounted.

If DSACT2 must be rerun, then the option of RERUN=T may be required. Thus the option statement in this case will read OUT=2,RERUN=T.

Privileges Required: VIP.

DSARCH: Data Set Archive

This program will copy to tape (or to sequential disk files), a data set (or sets). It will produce two identical copies simultaneously if desired. The dump is in the format of 80-byte card images. The dump of each data set is preceded by two header statements (starting with '*DS1' and '*DS2') and is followed by a trailer statement ('*DSEND'). For the contents of the *DS1 and *DS2 records, see the comments in file $PGM:DSDMP.S or refer to the description of the UDSARC utility in the *MUSIC/SP User's Reference Guide*. Check sums are maintained to guard against undetected I/O errors.

Usage:

```
/FILE n TAPE VOL(xxxxxx) OLD LR(80) BLK(nnnnn) RECFM(U)
/INCLUDE DSARCH
UNITS=n,m
dump parameters
```

Either one or two /FILE statements should be supplied depending on whether two copies of the dump are required. The output files should have a record size of 80 bytes.

The UNITS= keyword is followed by either one or two unit numbers corresponding to the /FILE statements. If two are used, the numbers are separated by a comma.

Each dump command following the UNITS= will dump one or more data sets. Each must contain a VOL= parameter and one of either DSN= or CODES=

Parameters:

VOL='volume','volume',...or VOL='/ALL/'
where 'volume' is a UDS volume name and '/ALL/' means all UDS volumes. (Abbreviation

V=)

DSN='name' (abbreviation D=) where 'name' is one of the following types -

'dsname'	actual 1 to 44 character data set name eg. use 'sys1.music.acct' not '%acct'
'pref..'	refers to all data set names starting with the given 1 to 42 character prefix
'/ALL/'	all data sets on the volume(s)
'/BACKUP/'	all data sets for which BACKUP was requested on the /FILE statement at allocation
'/NOBACK/'	all data sets for which BACKUP was not requested on the /FILE statement at allocation

Note: Only one name may be specified per NAMELIST. The total number of data set names and codes specified must not exceed 200/n, where *n* is the number of volumes.

CODES='code','code'...
(abbreviation C=) Where 'code' is a 4 character user code. All data sets belonging to the code(s) will be dumped from the specified volume(s). Up to 20 codes may be specified. However, if several volume names are given, then *n***m* must not exceed 200, where *n* is the number of volumes and *m* is the number of codes.

VERTAP=T Verifies that first record of existing tape file begins *DSARCHIVE to ensure that a data set archive tape is mounted. The default is VERTAP=F.

Examples:

Dump nucleus data set on MUSICX:

```
VOL='MUSICX',DSN='SYS1.MUSIC.NUCLEUS'
```

Dump all UDS belonging to $L06 on all mounted volumes:

```
VOL='/ALL/',CODES='$L06'
```

Dump all data sets from MUSICZ if name starts with SYS:

```
V='MUSICZ',D='SYS..'
```

Privileges Required: LSCAN; DREAD or MAINT.

DSCHK: Dump Tape Verify

The DSCHK program scans a dump tape or tapes and verifies that the format is correct, check sums tally and if two tapes are input, that they are identical.

```
/FILE    one or two file statements, RECFM(u) for tape only
/INCLUDE DSCHK
UNITS=n,m,NAMES=TRUE,INFO=TRUE
                 FALSE     FALSE
```

In the parameter list, *n* and *m* are the unit numbers of the /FILE statements pointing to the dumps. If NAMES=TRUE is specified, the names and volumes of all data sets on the tape(s) are listed. If INFO=TRUE is used all data set header statements are printed.

Privileges Required: None.

DSCOPY: Data Set Copy Utility

The DSCOPY utility program described in the *MUSIC/SP User's Reference Guide* may also be used to copy system data sets. The following is a sample of how it can be used. First enter /VIP ON, then execute a file containing the following:

```
/FILE 1 UDS(%name1) VOL(volume1) SHR
/FILE 2 UDS(%name2) VOL(volume2) OLD
/INCLUDE DSCOPY
```

The above example will copy from SYS1.MUSIC.name1 to SYS1.MUSIC.name2. Both data sets must have the same blocksize and the receiving data set must have been allocated and formatted before the copy is done.

The DSCOPY program cannot be used for copying files. Use the /COPY command for this. Another technique for creating a copy of a file is to archive it to a file or UDS file (MFARC2 program), then restore to the target file (MFREST program). The editor may also be used for copying most files, but not for files with record format U or load modules with record format FC or VC.

Privileges Required:

The user must have the VIP privilege active at the time the job is run.

DSKDMP: General Disk Utility

This program enables the system programmer to inspect and change records on direct access devices by absolute disk locations. It includes functions for inspecting Volume Table of Contents (VTOC) entries of disk packs. (Refer to the SYSDMP utility if you want to inspect and alter contents of a SDS or UDS data set by relative block number.)

The program is normally run from a terminal, but can be used at batch. The command to execute it from a remote terminal is:

```
DSKDMP
```

The primary keywords accepted by DSKDMP are:

CYL=x,y	Specifies the range of cylinders to be dumped. If the second one is omitted, it is assumed to be equal to the first. The default values are both zero. This parameter is not valid on FBA volumes.
HEAD=x,y	Range of heads (tracks) to be dumped. If the second one is omitted, it is assumed to be equal to the first. The default head values are zero. This parameter is not valid on FBA volumes.
RECORD=x,y	Range of record numbers to be dumped. If the second is omitted, it is assumed to be equal to the first. The default is one. On FBA volumes, this parameter gives the absolute block numbers For type 3 and 5 requests, the

record number must be one less than the read record number.

LEN=	Specifies the number of bytes to be read or written. The default is 16. Note, if the length written is less than the physical record size on disk, then the remainder of the record will be filled with zeros. This zero fill is automatically done by the hardware.
UNIT=	Specifies the internal MUSIC unit control block or Data Extent Block on which the I/O is to be done.
ADDR='cuu'	Specifies the physical device address on which the I/O is to be performed.
VOL='xxxxxx'	Specifies the logical volume name onto which the I/O is to be done. The names which can be used are detailed below.
TYPE=	Specifies the type of operation to be performed. The values of TYPE are given below. On FBA volumes, only TYPE=1 is valid.
ECHO=TRUE or FALSE	Specifies whether the above parameters are to be printed prior to printing the dump information.
REP=	Gives the displacement into the record at which some data is to be changed. The changed bytes are entered next. The record must have been previously read by means of a CYL, HEAD, RECORD read request.
WRITE=TRUE or FALSE	Set to TRUE when it is desired to rewrite the current buffer. TYPE must be set to either 1 or 2. The buffer must have been filled by a previous read request. All other parameters are ignored if WRITE=TRUE is given. Refer to the NORD parameter for usage notes.
NORD=TRUE or FALSE	Set to TRUE if no actual disk reading is to take place. This option can be used to alter the disk location for a subsequent write operation.
VTOC=TRUE or FALSE	Record is assumed to be part of a disk pack VTOC (Volume Table Of Contents) will be displayed appropriately (and if it 'looks' like a VTOC record). This command forces LEN=140,TYPE=7. VTOC=TRUE stays in effect until it is set FALSE. This parameter is not valid on FBA volumes.
LISTV=TRUE or FALSE	This command causes a VTOC listing (sorted by track address) to be produced. It checks for overlapping data sets and gaps (lost space). A GAP message is normal for a volume that is used by DOS or DOS/VS. A UNIT, ADDR, or VOL specification should accompany this parameter.
LISTDS=TRUE or FALSE	Similar to LISTV but extents are sorted by data set name. Any gaps or overlaps are ignored.
FREE=TRUE or FALSE	Similar to LISTV but only free space is listed.
ZAPF5=TRUE or FALSE	All free space information records (ie. format 5 DSCBs) are cleared from the VTOC. The VTOC will indicate that no free space is left on the pack at the end of this operation. This command is usually used before a FORMF5 command. A UNIT, ADDR, or VOL specification should accompany this parameter. This parameter is not valid on FBA volumes.
FORMF5=TRUE or FALSE	New format 5 DSCBs (free space information) are created. The ZAPF5 operation must have been performed previously to clear the old format 5

	entries. A UNIT, ADDR, or VOL specifications should accompany this parameter. You should perform a LISTV function to verify that the number of data sets on the pack will not exceed the table size in this program before doing the FORMF5 operation. This parameter is not valid on FBA volumes.
DSN='...'	This command causes the VTOC entry for the specified data set name to be printed.
END	This command terminates DSKDMP.
HELP	This command displays information on how to use the program.

Several abbreviations and alternative keywords are allowed.

Keyword	Alternatives
CYL	CC, C
HEAD	HD, HH, H
RECORD	RCD, REC, RR, R
LEN	L, COUNT, CNT
UNIT	U, UCB
ADDR	UAD, ADR, A
VOL	VOLUME, V
TYPE	none
ECHO	none
REP	none
WRITE	WRT
VTOC	none
LISTV	none
LISTDS	none
FREE	none
ZAPF5	none
FORMF5	none
DSN	none
NORD	none
END	none
HELP	none

Device Specification

The disk drive to be used by this utility is specified by one of three parameters: UNIT, ADDR, VOLUME. If more than one of these is given, a check is made to ensure that the parameters all indicate the same device. An error message is issued if they conflict.

VOLUME Names

The names to be specified for MUSIC system volumes are as follows:

SYSRES	System residence device
zzzzzz	Disk volume zzzzzz.

TYPE Specification

Value	Meaning
1	Read/write data
2	Read/write key, data
3	Read count, key, data
4	Read home address
5	Read count
6	Read record R0
7	Read VTOC record (See VTOC parameter)

Notes:

1. Only TYPE 1 or 2 may be used if the record is to be rewritten. Counts, home addresses, and record R0 cannot be rewritten with this program.

2. If TYPE 3 or 5 is used, the record number requested must be one less than the real record number.

3. The default parameters at the start of program execution are:

```
CYL=0,HEAD=0,RECORD=0,LEN=0,VOL='SYSRES',TYPE=1,ECHO=FALSE
```

4. Parameters are entered in standard MUSIC FORTRAN NAMELIST form. No blanks should be used and parameters are separated by commas.

5. With the exception of WRITE and REP, all parameters retain their values unless explicitly changed. WRITE and REP must be explicitly stated each time they are needed.

6. If the UNIT= parameter is used to input a DEB number, or if VOL= LIBEnn is used, the cylinder, head and record specified will be translated to a real disk address according to the position of the data set on the real drive and the pseudo device format of the data set. If the DEB number specified has no pseudo device format, the request will be rejected.

7. To copy data from one record to another, first read the record with a VOL, CYL, HEAD, RCD, LEN (or equivalent) request. Then issue a similar request for the address to which the data is to be written, not specifying LEN, but specifying NORD=TRUE. Then use the WRT=TRUE command to write the buffer to the new location.

REP Data Format

Up to one card image of data can be specified for each REP command. More than one REP command may be given before the record is rewritten. The data must start in column one of the line. It is entered in hexadecimal. Commas may be inserted at will between bytes, but not between hexadecimal digits of a byte. A blank (or end of card) terminates the data.

Sample Execution

A sample run of this program illustrating most of its features is given in Figure 17.3. The first operation requested is to print the data portion of the record located at cylinder 99, head 0, record 1. Several bytes starting at byte 14 of the record are replaced. The record is then written back to the device.

If a line is to be printed in which all bytes are the same, only one copy of the byte is shown.

Note that changes to disk records should be made with extreme care.

The next request prints records 1, 2 and 3 of the track located at cylinder 0, head 0 of the system residence volume. These records are the IPL record, the IPL program, and the volume label. The program is requested to read and print 32 bytes of each record including the count and key fields.

Note that the actual length of the disk blocks can be determined by this technique. It is shown in the second word of each record. In the above example they were x'18', x'AA8' and x'50'.

Privileges Required:

The DREAD and CREAD privileges are needed to run this program. The VIP privilege must be active in order to use the WRT, ZAPF5, or FORMF5 commands.

```
*Go
dskdmp
*In Progress

DEFAULTS
VOLUME = SYSRES
TYPE = 1   READ DATA
DISK UCB = 0
CC =   0,   0     HH =  0,  0     R =  1,  1     LEN =   16
?
vol='disk01',cc=99,len=262
CCHHR= 99  0  1
0000  D3C2F0F1 F0F9F1F5 F6F7FFB5 BE191201  *LB01091567......*
0010  00  *.*
/skip 20
?
rep=14
ENTER REP
?
16,00,AB89
0000  D3C2F0F1 F0F9F1F5 F6F7FFB5 BE191600  *LB01091567......*
0010  AB890000 00000000 00000000 00000000  *................*
0020  00  *.*
0030  00  *.*
0040  00
/sk 20
?
write=true
RE-WRITTEN   99   0   1  DATA
?
type=3,vol='sysres',r=0,2,cyl=0,len=32,echo=true
VOLUME = SYSRES
TYPE = 3   READ C K D
DISK UCB =  0     HH =  0,  0     R =  0,  2     LEN =   32
CCHHR=  0  0  0
0000  00000000 01000018 00000000 00001000  *................*
0010  06001000 20000AA8 00000000 00000000  *.......y........*

CCHHR=  0  0  1
0000  00000000 02000AA8 05F0D207 0070F396  *.......y.0K...3o*
0010  8200F39E 48100002 4130F114 D204F140  *b.3.......1.K.1 *

CCHHR=  0  0  2
0000  00000000 03000050 E5D6D3F1 D4E4E2C9  *.......&VOL1MUSI*
0010  C3E7F000 00000101 40404040 40404040  *CX0.....        *

?
end
*Terminated
*Go
```

Figure 17.3 - Sample Run of DSKDMP

DSRST: Data Set Restore

This restore program will restore the contents of a data set from a dump tape created by the data set archive program.

Usage:

```
/FILE n ... (dump tape, lrecl 80, RECFM(u) for tape)
/INCLUDE DSRST
IN=n,TAPFIL=m,VOL='vol',DSN='dsn',
TOVOL='tovol',TODSN='todsn'
```

Where *n* is the unit number of the /FILE statement, *m* is the tape file number (default m=1), *vol* is the volume label where the data set originally resided and *dsn* is its original name. The data set on which the data is to be restored is pointed to by TOVOL and TODSN. The TODSN must already exist though no /FILE statement is required to point to it. All parameters must be supplied with the exception of the input unit number which defaults to 1.

More than one data set can be restored in the same job by supplying additional sets of NAMELIST parameters. If IN=5, only one data set can be restored. Only the DSN and TODSN parameters must be entered for each subsequent request. The VOL, DSN, TOVOL and TODSN may be abbreviated as V, D, TOV, TOD respectively.

Note: Archive and restore jobs are normally run from batch, although if a disk data set is used instead of tape, they may be run from a terminal.

Contents of Information Records

In a dump produced by UDSARC, each data set is preceded by a *DS1 record and a *DS2 record. The *DS1 record has the volume name, the data set name, and the date and time of the dump. The *DS2 record contains the following information:

Column 8	B for backup, N for no backup.
9-12	Device type.
13-15	Data set organization.
16-19	Record format.
20-25	Block size.
26-31	Logical record length.
32-37	No. of tracks (no. of physical blocks if FBA device).
38-40	No. of extents.
41-44	No. of blocks per track (32 if FBA device).
45-48	Key length.
49-53	FBA physical block length (0 if not FBA device).

Privileges Required: LSCAN, VIP.

EDTCAT: System Catalog Creation Utility

This utility program creates a system catalog on disk. It also produces a listing of the new catalog. If run from batch the job format is:

```
/INCLUDE *COM:EDTCAT
/INCLUDE xxxxxx
```

where *xxxxxx* is the name of a file containing the catalog statements. The file $GEN:SYSCAT contains the statements used to create the catalog on the production system created at the time MUSIC was installed. If run from a terminal, a /INPUT and /ENDRUN replace the /ID and /END respectively. When being run from a terminal, the program pauses after the displaying information concerning the catalog location, for permission to proceed. If it is desired **not** to continue, a /CANCEL command must be entered. Any other response will allow catalog to be re-written.

In all cases, the first two lines of the catalog input must be:

```
'CATALOG' 'volume'
 data set name
```

The word 'CATALOG' (in quotations) must be the first parameter of the first line. The next parameter is the volume label of the pack on which the catalog is to reside (also in quotations). Columns 1-44 of the next line contain name of the catalog. This is normally SYS1.MUSIC.CATALOG.

The data set must have previously been allocated and formatted.

System Catalog

The system catalog contains the following types of records:

- System Data Set statements
- Pseudo Device statements
- Operator Command statements
- Volume Mount Request statements
- Link Pack Routine Names

System Data Set Statements

Column	Description
1-8	Unique 1-8 byte identification used during catalog console editing
10-13	Data set identification U000 for the Save Library Index Unnn for Save Library spaces Bnnn for batch spooling Fnnn for others (here nnn will be the DEB number, for U and B SDS, the DEB number is assigned at initialization time).
15-16	Pseudo-device type (same as column 7-8 of pseudo cards). If set to 00, no pseudo-device requests may be made.
18	=R if data set must be found for proper system operation. =M if system is to attempt to run without this data set, but a warning message should be issued =I same as M but no message is issued.
19	=L when data set is located, issue an informative message =N no message

20	=S keep separate statistics for this SDS =N keep statistics under the physical device.
22-24	Number of records per track for this data set. If this is used, it overrides the normal value calculated from the device characteristics and the block size. Zero indicates to use the normal value.
26-31	Volume serial number on which the SDS is to be found. If blanks are specified, a search for the SDS will made on every ready device.
33-76	Data set name

Pseudo Device Statements

Column	Description
1-6	PSEUDO
7-8	Pseudo device type referenced by column 15-16 of data set statements.
10-12	Flag bytes - must be zero
14-18	Blocks per track on pseudo device
20-24	Tracks per cylinder on pseudo device.

Operator Command Statement

Column	Description
1-5	OPCMD
6-8	Unique three digit number to distinguish one OPCMD statement from another.
10-12	/CP
14-80	VM command to be issued. Sample command might be: OPCMD001 /CP SET RUN ON

Volume Identification Statements

Column	Description
1-6	VOLUME
7-8	Unique two digit number to distinguish one VOLUME statement from another.
10-15	Volume label of the disk pack. A message may be issued if the volume is not mounted at IPL time, depending on the contents of column 18. This statement can be used to ensure that the proper UDS packs are present. Note that it is **required** to identify the UDS packs in this manner in order that users may create new files on the pack.
17	The letter A in this column means that users can allocate new files on this volume. The

letter P means that non-privileged users may not refer to any UDS file on this volume. The letter N means that users can refer to UDS files on this volume, but they cannot allocate new files on it.

18 This column contains a letter code specifying whether a message is to be sent to the operator at IPL time if it is not mounted. The allowed values are R, M and I. See the description of column 18 for System Data Set statements for meaning of the letters.

19 The letter R in this column means that MUSIC will not attempt to write to this disk during normal processing of user jobs. Privileged users could deliberately write to this disk via utility programs such as DSKDMP.

Link Pack Area Names

Column	Description
1-5	RESPG
6-8	Unique 3-character id to distinguish one RESPG statement from another.
10-17	Eight character module name to place in the Link Pack Area. Refer to *Chapter 21 - Load Library and Link Pack Area* for details of usage of this area.
19-22	The letters FLPA or PLPA. FLPA means that main storage will be allocated to hold the specified module. PLPA means that the re-entrant module named will be written on the first page data set in page format. FLPA is assumed if this field is blank.
24-30	Virtual storage location (in hex) of where the PLPA member is to be loaded. This field is ignored for FLPA items. The storage location must be at least 300000. The location must be chosen so that the module ends before location 700000. Members that can be simultaneously used by the same program must not overlap their virtual addresses.

In the catalog statements you can use special characters in the volume name fields to allow for a more flexible catalog. The character ? in any position of the volume name will be taken to mean that it will match any character. The character $ in any position in the volume name will match the corresponding character in the volume name of the IPL volume.

Privileges Required:

The VIP privilege is needed to run EDTCAT. Refer to the topic "VIP Privilege Notes" in the CODUPD writeup for details of how to activate this privilege.

ELOG.CLEANUP: Delete Empty Editor Log Files

This utility deletes any files whose names are of the form @ELOG.xxx.n and which are empty (i.e. have a line count of 0). These are log files corresponding to edit sessions which terminated normally. It is run by entering ELOG.CLEANUP (on a terminal) or /INCLUDE ELOG.CLEANUP (on batch).

This program should be run regularly, at a time when there is little activity on the system. A recommended time is each Friday night.

Privileges Required: FILES and LSCAN.

ENQTAB: Display Enqueue Table

A privileged system programmer can use the ENQTAB utility to display the contents of the system's enqueue table. This program is run by entering ENQTAB at a terminal. Optionally, you can enter "ENQTAB string" where *string* is a character string to be searched for in enqueue names and file names. Only entries containing the *string* (as a substring) are displayed. If no string is specified, all entries are displayed.

The ENQTAB utility is useful for determining the number of entries in the enqueue table, and for finding which user (identified by TCB number) has control of a particular resource, such as a file or a UDS. Once the TCB number is known, the user code can be found by running the WHOALL utility. The size of the enqueue table may be increased if necessary by changing the system module ENQRTN. There should be approximately 800 entries for each 100 users signed on.

Privileges Required: LSCAN, CREAD, and DREAD.

FILE.DELETE: Delete a Group of Files

This program deletes (purges) all files whose code and name match specified prefixes.

Note: A similar function can be done by using a file name pattern on the /PURGE command.

Usage:

```
/INCLUDE FILE.DELETE
parameters separated by commas (see below)
```

Parameters:

CDPREF='xxx' Prefix for user code part of file name (0 to 4 characters). This parameter must be specified.

NMPREF='xxx' Prefix for name part of file name (0 to 17 characters). Either NMPREF or NAME (but not both) must be specified.

NAME='xxx' An actual file name (1 to 17 characters). All files with this name whose code matches the specified code prefix will be deleted. NMPREF and NAME must not both be specified.

DELETE=T This parameter must be specified to actually delete the files. If this parameter is omitted, the file names will be displayed but not deleted.

The entire Save Library index is scanned for file names which match both the specified code and name prefixes, and all such files are deleted.

If CDPREF is all blanks, it is considered to match all codes, and similarly for NMPREF. However, CDPREF and NMPREF must not both be blank.

Examples:

```
1.    CDPREF='AB',NMPREF='XY.Z',DELETE=T

2.    CDPREF=' ',NAME='MYPROG',DELETE=T
```

Example 1. deletes all files which have a code starting with AB and a name part starting with XY.Z. Example 2. deletes all files which have MYPROG as their name part.

Privileges Required: LSCAN, FILES.

FILECH: Alter File Names or Attributes

This program changes the name and/or access control of a file or a list of files.

Input control statements are read from unit 5. Each control statement has the following format:

1. Old name in columns 1 to 22 (with or without project code).

2. New name in columns 24 to 45 (or blank if no change in name).

3. 4-byte access control (in hex) in columns 47 to 54 (or blank if no change in access control). Refer to *Chapter 20 - File System* for a description of the access control bits. Access control may also be specified by keywords as follows:

   PRIV
   PUBL (public, but entry not in common index)
   COM (private, but entry in common index)
   SHR
   PRIV,XO
   PUBL,XO
   COM,XO
   SHR,XO

The context editor COPYCOL command can be used to help create the second column of file names. Simply edit a file that contains the list of files you want FILECH to work on. Issuing the editor "COPYCOL 1 22 24 999" command will then create an identical list starting in column 24. This two column list can then be further edited as required.

Privileges Required: LSCAN and FILES.

FIXINDEX: Remove Save Library Index Entry

This program can be used to remove an entry from the Save Library index if something is wrong with it.

This could be necessary if the file cannot be deleted by a /PURGE because the index entry does not point to a valid directory block (file system error 65 or 67).

Note: FIXINDEX should be used when the number of files is small, or when it is desirable to work conversationally. For a larger number of files, utility program FIXINDEX.AUTO should be used rather than FIXINDEX. Even for a small number of files, you will find that FIXINDEX.AUTO is easier and safer to use.

If the file to be removed has the PUBL or COM attribute, both the private and common index entries must be removed. To locate the entry in the common index, specify NAME='*COM:filename' to MFHASH. (FIXINDEX.AUTO automatically removes both.)

First, use MFHASH to get the DEB number for the Save Library index and the index block number that contains the incorrect entry. Next use SYSDMP to dump the index block to get the displacement of the entry and its length. Note the first 4 bytes of the entry. Finally, run the FIXINDEX program specifying the appropriate values.

The following is an example of the procedure described above.

```
*Go
mfhash

-- MFHASH --

DEB NUMBER OF SAVE LIBRARY INDEX =   9
NUMBER OF BLOCKS IN INDEX PRIMARY AREA =   6041
START BLOCK NUMBER =   9
NUMBER OF SEGMENTS IN INDEX =   7

ENTER: NAME='CODE:NAME' OR NAME='*COM:NAME' OR CODE='CODE'
?
name='ih39:.0000476'
IH39:.0000476           INDEX BLOCK NUMBER =   4778
/ca
*Terminated

*Go
sysdmp

SYSDMP -- ENTER PARAMETERS (DEB,DSN,VOL,BLK,...) OR HELP
?
deb=9,blk=4778,blksiz=512
DEB   9 FOUND, BLK SIZE =   512
BLK    4778 READ, LENGTH =     512
0000  13C9C8F3 F94BF0F0 F0F0F4F7 F6000000  *.IH39.0000476...*
0010  010CC300 00000000 00000000 00000000  *..C............*
/can
*Terminated
```

```
*Go
fixindex

-- SAVE LIBRARY INDEX ZAP PROGRAM --

ENTER DEBNUM=,BLKNUM=,DISPL=ZXX,ZAPLEN=ZXX,VER=ZXXXXXXXX
(TO REMOVE BYTES FROM AN INDEX BLOCK)
?
debnum=9,blknum=4778,displ=z00,zaplen=z13,ver=z13c9c8f3

DEBNUM =   9   BLKNUM =  4778
DISPL =   0 (Z0000)   ZAPLEN =  19   VER = Z13C9C8F3

0000  13C9C8F3 F94BF0F0 F0F0F4F7 F6000000  *.IH39.0000476...*
0010  010CC300 00000000 00000000 00000000  *..C............*
-------//--------
-------//--------
01F0  00  *.*

TYPE OK TO REREAD, CHANGE AND WRITE THE BLOCK
?
ok

CHANGED BLOCK:
0000  00  *.*
0010  00  *.*
-------//--------
-------//--------
01F0  00  *.*

ENTER DEBNUM=,BLKNUM=,DISPL=,ZAPLEN=,VER=ZXXXXXXXX
(TO REMOVE BYTES FROM AN INDEX BLOCK)
/can
*Terminated
*Go
```

Privileges Required: LSCAN, DREAD and either MAINT or VIP.

FIXINDEX.AUTO: Automatically Remove Save Library Index Entry

This program is an automated version of FIXINDEX. It removes entries from the Save Library index for files which have error 65 or 67 (directory in error or index/directory mismatch). It is recommended that this program be used rather than FIXINDEX, in most cases.

WARNING: There must be no other users or jobs on the MUSIC system when this job is run. Otherwise the Save Library index may be severely damaged.

Usage:

```
/INCLUDE FIXINDEX.AUTO
code:name1
code:name2
   ...
```

The full file names to be removed from the index are read from unit 5, one name per line, starting in column 1. These names could be found by running the CHKFILES program.

Privileges Required: LSCAN, FILES, DREAD, VIP.

FMFREE: Formatting Free Space

The FMFREE program reads the format-5 DSCB's of a specified disk volume and outputs to unit 10 (or to unit 7 if batch) a job to format all the free extents of the volume. A blocksize of 512 is usually used. The LISTV (list VTOC) function of the disk dump utility (DSKDMP) should be used before running FMFREE, to verify that the free extents do not overlap any allocated space on the volume.

Usage:

```
FMFREE
or
/INCLUDE FMFREE
VOL='volume',BLKSIZ=n,OUT=unit
```

The program may be run from batch or terminal. The default blocksize is 512. The default output unit number is 10 (terminals) or 7 (batch). FMFREE contains default /FILE statements, defining unit 10 as file @FMFREE.OUTPUT and unit 7 as file @FMFREE.OUTPUT7. After running FMFREE, execute the resulting format job in order to actually format the free space.

Note: There is no need to format free space on FBA devices.

Privileges Required:

DREAD and CREAD privileges are needed to run FMFREE. The format job requires the VIP privilege.

FORMAT: Disk Format Utility

The disk format utility is available to MUSIC maintenance personnel for use in formatting disk packs. Along with the program, there are sets of control statements saved in the Save Library. These files have the appropriate control statements for the disk format utility for formatting the standard MUSIC data sets. The format program is normally run from batch. The MUSIC control statements for running a standard system data set format job are:

```
/INCLUDE FORMAT
TITLE='page heading'
VOLUME='xxxxxx'
DEVICE=nnnn   or   DEVICE='FBA'
VOLID='xxxxxx'
DSN='data set name'
/INCLUDE FMTxxx (if needed - see below)
```

The statements used to format a complete User Data Set pack are:

```
/INCLUDE FORMAT
TITLE='page heading'
VOLUME='xxxxxx'
DEVICE=nnnn,CYL=0,cccc   or   DEVICE='FBA'
VOLID='xxxxxx'
/INCLUDE FMTUDS
```

If the FORMAT program is used from a terminal, the control statements may be entered as SYSIN lines (as in the above jobs) or conversationally. The program first reads any input on SYSIN, then switches to read conversationally from the terminal. The normal way of terminating FORMAT is by entering /CANCEL in response to a read prompt. The conversational reads may be suppressed by using the statement /FILE 9 DUMMY before /INCLUDE FORMAT.

The text within quotations of the TITLE statement will be printed at the top of the printed output to identify the job. The value of cuu on the UNIT statement specifies the unit address of the pack to be formatted. The value within quotations on the VOLUME and VOLID statements is the volume label with which the disk pack was initialized. The DEVICE= parameter specifies the type of device being formatted. The DEVICE specification may be one of the following: 2314, 3340.35, 3340.70, 3330, 3330.11, 2305.1, 2305.2, 3350, 3380, 3380.2 (double capacity models), or 'FBA'. For a VM/370 minidisk on a non-FBA device, the CYL parameter must be used to specify the number of cylinders in the minidisk. For further information on these control lines (or those within the files), see "Format Utility Commands".

The /INCLUDE statement points to one of the files listed below. It is required to format entire User Data Set packs and for System Data Sets explicitly mentioned in the list.

FMTUDS to format an entire disk pack as a User Data Set (UDS) volume. (No DSN parameter is needed.) /INCLUDE FMTUDS must not be used if only part of the volume is being formatted.

FMTCOD to format a User Code Table data set

FMTCDX to format a User Code Index data set.

While formatting the disk, any bad tracks found are noted on the output listing along with their alternate tracks. They should be retained for future reference. Note that 2314 and 3340 packs should *not* have any bad tracks, if they do, care should be taken to allocate dummy data sets over the bad tracks so that they will never be used. Alternate tracks on 3330's and 3350's are allowed but may cause system performance degradation due to the automatic alternate seek done by the hardware.

The following is a detailed description of each of the format utility commands. Most commands are made up of FORTRAN NAMELIST-like input. Care should be taken to make sure character parameters are enclosed in quotes. Parameters are separated by commas. Commands too long for one input line can span lines by ending each line at a comma.

Format Utility Commands

Page title generation:

```
TITLE='text'
```

Prints 'text' (maximum 72 columns) at top of next page of output.

Comment generation:

```
COM='text'
```

Prints 'text' (maximum 72 columns) on output.

Volume specification:

```
VOLUME='vvvvvv'
```

Specifies the disk to be formatted was online at IPL time and had the volume label *vvvvvv*.

Unit address specification:

```
UNIT='cuu'
```

Specifies pack to be formatted is mounted on the disk drive with physical unit address *cuu*. This parameter is particularly useful when it is required to point to a pack that was not mounted at IPL time.

Note that either the VOLUME or the UNIT parameter must be specified before any of the following parameters can be used.

Device specification:

```
DEVICE=xxxx,HEAD=a,b,CYL=c,d  or  DEVICE='FBA',BLOCK=nnn
```

Specifies device type. Valid types are given in the table below. Use 3340 for 3344 devices and use 3350 only when operating in native mode. The HEAD and CYL parameters are optional. If given, they override the default head and cylinder ranges for the specified device type. The HEAD and CYL parameters should be used with great care. Figure 17.4 gives the defaults for these values.

Device	Head Range	Cylinder Range
2314	0,19	0,199
3340.35	0,11	0,347
3340.70	0,11	0,695
3330	0,18	0,403
3330.11	0,18	0,807
2305.1	0,7	0,47
2305.2	0,7	0,95
3350	0,29	0,554
3375	0,11	0,958
3380	0,14	0,884
3380.2	0,14	0,1769
3380.3	0,14	0,2654
3390	0,14	0,1112
3390.2	0,14	0,2225
3390.3	0,14	0,3338
9345	0,14	0,1439
9345.2	0,14	0,2155

Figure 17.4 - Default Head and Cylinder Ranges for Formatting Disks

When formatting a 'minidisk' under VM/370, the CYL parameter must be used to define the number of

cylinders on the minidisk.

If 'FBA' is specified CYL and HEAD parameters are ignored. The BLOCK parameter is optional, it can be used to specify the number of blocks on the device and is useful only in the event the device was not attached at IPL time.

Volume label checking:

```
VOLID='xxxxxx',NEWID='yyyyyy'
```

Label of pack on specified unit is read. If pack label is not equal to *xxxxxx*, an error message is issued. If 'SCRATCH' is specified, the label is read and printed but no compare is done.

If the NEWID parameter is specified, the pack is relabeled with the name *yyyyyy*. This parameter is optional.

Data Set Format Request:

```
DSN='data set name',DATA=Zgg
```

The specified data set is to be formatted. The block size specified when it was allocated will be used. The DATA parameter is optional and is described under the next command.

Absolute Format request: (CKD)

```
START=a,b,END=c,d,KL=e,DL=f,DATA=Zgg,NREC=h,LABEL=xxx.
```

Parameters a,b specify the starting cylinder and head (relative to the start of the logical volume as modified by the HEAD and CYL commands) of the area to be formatted; c,d are the ending cylinder and head locations. (Specifying END=9999,9999 will be taken as the end of the pack as determined from the DEVICE specification.) These parameters must be specified but all the following ones are optional.

KL specifies the key length of the records.

DL specifies the length of the data portion of the record.

DATA can be used to specify the character used to fill the formatted records. The default is DATA=ZFF. Use DATA=Z00 to format the Save Library Index data set (SYS1.MUSIC.UIDX).

NREC specifies the number of records per track. If omitted, a value calculated from DL, KL, and the device type is used.

LABEL is used to specify whether the label record (cyl=0, head=0,record=3) is to be preserved. The value of *xxx* should be either TRUE or FALSE. This parameter may be TRUE only if cyl=0,head=0 is included in the START-END range.

The default values for the optional parameters are:

```
KL=0,DL=512,DATA=ZFF,LABEL=FALSE
```

Once KL or DATA parameters are specified, the new value remains in effect until explicitly changed.

Absolute Format request: (FBA)

```
START=a,END=b,DL=t,DATA=Zgg,NREC=h,LABEL=xxx
```

The parameters have the same functions as the CKD case with the following exceptions.

START - starting block number (1st block is 0)

END - Ending block number

KL - is NOT valid and should not be used

Write Data Set Record Request:

```
WRDATA=n
```

A data record is written to the nth block of the data set last formatted by a DSN statement. The first block is numbered zero. The actual data to be written is specified on statements following the WRDATA command. Its format is the same as the data for the RCD command documented below.

Data record request:

```
RCD=c,h,r,REPT=n,RPT=r      or      RCD=b,REPT=n
```

This command allows special data to be written to records already formatted on the disk pack. The parameters *c,h,r* specify the absolute cylinder, head, and record number of the record to be written. The data may be written more than once on successive records. REPT defines the number of times the record is to be written. If REPT is given, RPT should also be given to specify how many records can fit on a track. If omitted, it is assumed that all records requested can be written on the same track as the first one. The second form is used for FBA devices *b* specifies the starting block for the write. The data to be written is specified in the card(s) following the RCD command.

The cards specify either EBCDIC character data or hexadecimal data. As many of these cards as necessary may be used, intermixed as needed. A blank card (blank at least in column 1) indicates end of data specification.

The card format is:

1 C for EBCDIC character data
X for hexadecimal data

2-3 Number of bytes (right-adjusted) of data in the data area of the card. This can be omitted if all the data is punched (that is, no trailing blanks) and no excess data is present on card. For hexadecimal data this is the number of bytes of data (normally one-half of the number of characters punched on data area of card).

4-6 Repetition factor. Data will be repeated this many times. If omitted, one is assumed.

7-9 Must be blank

10-80 Data in appropriate format

Privileges Required:

A code with the VIP privilege must be used. The command /VIP ON must be in effect if the program is run from a terminal.

FPRINT: File Print Routine

This utility prints the contents of the specified list of files. The TAG, attributes and dates last used are printed for each. Each page contains a title giving the file name. Sample JCL to run FPRINT follows:

```
/INCLUDE FPRINT
UNITS=6,ALLUP=T,MAXLIN=NN  (Sample line..see below for details)
FILENAME1
FILENAME2
....etc
```

Parameters:

UNITS=6 Output to unit 6

ALLUP=T Will translate lower case to upper case characters

MAXLIN=nn Will stop listing each file after the specified number of lines have been printed on unit 6

SKIPRD=T Stops the reading of a file after MAXLIN lines have been printed. This saves time for large files, but the file total line count may be omitted. The default is SKIPRD=F.

Privileges Required: None.

GEN.CODES: Generate Groups of Sign-on Codes

This program generates groups of MUSIC user codes, with 4-character random passwords. Output is in the form of ADD commands acceptable to the CODUPD utility. The commands are written to unit 1.

Most installations will want to modify this program to suit their particular needs. The coding here should provide a good starting point to work from.

Usage:

```
Step 1.        /FILE 1 NAME(CMDFILE) NEW
               /INC GEN.CODES
               GROUP='xx',NUM=n,BASE=b,OPT='xxx',OPT2='yyy'
               GROUP='xx',NUM=n,BASE=b,OPT='xxx',OPT2='yyy'
               etc.

Step 2.        /INC CODUPD5
               /INC CMDFILE
```

Step 1 generates the ADD commands, written to the file CMDFILE in the above example. Step 2 actually adds the codes to the Code Table. CODUPD5 is identical to the CODUPD utility, except that it reads commands from Unit 5 instead of 9 and does not print as many information messages.

Parameters:

GROUP='xx' This specifies the first two characters for each user code in the group.

NUM=n Specifies the number of codes to be generated in the group. The maximum is 99 for BASE=10, or 1296 for BASE=36.

BASE=10 OR BASE=36
Defines how the 3rd and 4th characters of the codes are to be generated. BASE=10 uses a 2-digit decimal number, starting at 01. BASE=36 uses the characters A to Z as well as 0 to 9: AA, AB, AC, ..., AZ, A0, A1, ..., A9, BA, BB, etc. BASE=10 is assumed if no base is specified. With BASE=10, the maximum number of codes per group is 99. With BASE=36, the maximum is 1296. The sub code is 000 in all cases.

OPT='xxx' Specifies optional additional CODUPD parameters to be placed on the ADD commands. The maximum length of the character string *xxx* is 59. If you need more than this, specify some of the parameters in OPT2='yyy'. These will be placed on a continuation line in the command file.

OPT2='yyy' More optional CODUPD parameters if the desired parameters do not fit in OPT='xxx'. The maximum length for *yyy* is 80 characters.

Example:

```
/FILE 1 NAME(ADD.FILE) NEW
/INC GEN.CODES
GROUP='XY',NUM=25,BASE=10,OPT='MAX$(100) TOTLIM(200)'
GROUP='AB',NUM=150,BASE=36
GROUP='T2',NUM=300,BASE=36,OPT='PRIME(32) NONPRIME(60)',
OPT2='BATCH(60) DEFAULT(16) AUTOPROG(HELLO)'
```

Privileges Required: LSCAN

GENSAV: Generate Multiple Files from One File

This utility can be used to create individual files from one original file. The utility can create a separate file for each object deck in the original file. Alternately it can produce separate files based on "./ ADD" control statements in the original file.

The original file is usually a tape but it could be a file or a UDS file.

Sample control statements:

```
/FILE N TAPE   ...SOURCE TAPE
/INCLUDE GENSAV
parameters (or blank line)
/INCLUDE FILNAM (IF REQUIRED FOR SELECTION OF './ ADD' CARDS)
```

Parameters:

INPUT input unit of the source tape. Default is INPUT=1.

FILE vector of length 10 containing file #s to be searched on input tape. Default is FILE=1,0,0,0... ie. only file 1 is used

PREFIX prefix of the newly created file name. Default is PREFIX=' '.

SUFFIX suffix of the newly created file name. Default is SUFFIX=' '. The total length of the prefix and the suffix should not exceed 9 characters (or 14 characters if 'code:' is specified in the prefix).

REPL if REPL=.true. is specified, then an existing save file will be replaced by the newly created save file of the same name. Default is REPL=.false.

PUBL if PUBL=.true. is specified, then the files are saved public.

SELECT if SELECT=.true. is specified, then only those './ add' cards with names that are specified in a save file (unit 5) will be processed (one name is specified on each line starting at column 1 in the save file). The default is SELECT=.false. which will process all './ add' cards in the tape.

OBJ OBJ=f causes the source to be scanned for "./ add" cards and moved to save files accordingly. If OBJ=t the source is scanned for *esd* cards. Default=.false.

Privileges Required: None.

INITFBA: Initialize FBA Packs

This program creates a volume label and VTOC on an FBA volume. Also, the first block of the volume is zeroed.

Usage:

```
/INCLUDE INITFBA
parameters (separated by commas) (see below)
additional parameter statements for other volumes (if desired)
```

Parameters:

ADDR='cuu' This parameter is the device address of the volume. It must be specified. This address must have been defined as an FBA device in the NUCGEN device statements when MUSIC was generated, or in the =CONFIG specifications when MUSIC was IPLed. The device may or may not have been ready when MUSIC was IPLed.

VERIFY='xxxxxx'

xxxxxx is the old volume name. This must be specified if a volume label (VOL1) already exists on the volume. Verification may be bypassed by specifying VERIFY='SCRATCH'.

NEWVOL='xxxxxx'

xxxxxx is the new volume name to be used. Must be specified.

OWNER='xxxxxxxxxx'

This is the owner identification to be placed into the new volume label (maximum of 10 characters). The default is OWNER='MUSIC'.

VTOC=n The starting 512-byte block number for the new VTOC. The first block of the volume is number 0. It must be 2 or more. The default is 32.

BLOCKS=n This is the number of 512-byte blocks in the new VTOC. It must be 2 or more, and should

be even. The default is 640 = 20 * 32. Each block can hold 3.5 DSCBs. At least one DSCB is required for each SDS and UDS allocated on the pack.

Example:

```
/INCLUDE INITFBA
ADDR='1A0',VERIFY='MUSXXX',NEWVOL='MUSIC2'
ADDR='1A1',VERIFY='MUSYYY',NEWVOL='MUSIC3'
```

Privileges Required: VIP.

IOTIME: List Input/Output Usage Statistics

This program prints a variety of values associated with DASD and magnetic tape data sets and devices. Cumulative values are logged for each device and channel on the system. In addition, individual logs may be maintained for specific system data sets. The system catalog contains parameters specifying which (if any) data sets are to be logged separately.

For each data set/device/channel listed, the following items are printed (if applicable).

1. Unit Control Block (UCB) or Data Extent Block (DEB) number.
2. Identification
3. Volume Label
4. Usage Count: Total number of I/O requests
5. Busy: Total time (in minutes, seconds and milliseconds) spent executing all I/O requests. Time is measured between the SIO and the first (usually only) interrupt for the device.
6. Busy+Wait: Time spent executing I/O requests while the processing unit was in wait state awaiting completion of the I/O.
7. Queue Count: Number of requests which could not be processed upon receipt but had to be queued due to a device or channel busy condition.
8. Queued: Total amount of time spent in queues waiting for a resource. Note that this value is cumulative. For example, if 3 requests were concurrently waiting for a device for one second, a total of 3 seconds would be added to this counter during this period.
9. Queue + Wait: Time spent in queues waiting for a resource while the processing unit was in wait state.

IOTIME is normally run from batch due to the large amount of output produced.

Usage:

```
/INCLUDE IOTIME
```

It may be run from a terminal by entering IOTIME.

The values printed by this program are those accumulated since the last IPL. Counters printed as '******'

are those in which I/O is in progress at the time the program was run.

Privileges Required: INFO.

LDCNTS: Load Library Usage Count Display

The LDCNTS program prints out load count statistics. Counts are automatically maintained for members loaded from the system load library.

To assign a count to a member loaded from a user load library, run the LDLIBE utility specifying an option such as:

```
NAME='xxx',CHANGE=T,NWCID=n
```

Privileges Required: LSCAN and CREAD.

LDLIBE: Load Library Update

The LDLIBE Utility converts link edited modules into the format used on the system SYS1.MUSIC.LOADLIB data set. Specifically it can perform the following functions:

- Create a load library
- Add, delete, replace and rename members
- Modify attributes of members
- Dump/restore members

This program copies load modules created by the MUSIC Linkage Editor to load libraries. These load modules should be created under the following rules:

- If the module will normally be loaded at a specific address, then specify this address on a .ORG statement. This will save the relocation step that must otherwise be performed.
- Place any .ADD statements before the .ORG statements. These ADD statements are used to define values for external symbols.
- The load module's name must be the same as the member name on the load library. The name is either given on the linkage editor NAME statement or with the NAME= parameter on the /JOB statement.
- The NOSEGTAB option must be used on the /JOB statement.
- Use the MODE=OS option on the /JOB statement if the conflicts with the predefined names of IBCOM, FIOCS#, etc., would otherwise exist.

The following shows the required JCL:

```
/FILE . . .      0 to 4 file statements
/INCLUDE LDLIBE
description statement
command statement
 . . .
 . . .
```

The file statements are used as required to define data sets for dump/restore, load modules and, in the case of UDS type load libraries, the actual load library.

The description statement (1 per job) describes the load library. It has the following parameters which are specified in namelist format.

LIBE=n *n* is the DEB number or unit number of the load library. If LIBE='SYST' is specified, the system load library will be assumed. LIBE may be abbreviated to LIB.

DSN='datasetname',VOL='volume'
The name and volume of the load library data set can be specified instead of the LIBE=n parameter. This allows the LDLIBE program to work with a non-standard or alternate system load library. Use of DSN and VOL requires the VIP privileges, and may result in slower execution of the program.

IN=m *m* is the unit number of the input load module data set.

TEST=T Will suppress all writes to the load library. TEST=F is the default.

DISP='NEW' Specifies that a new load library is to be created.

DIRSIZ=k *k* is the number of directory entries to be allocated. DISP='NEW' must be specified. This is the maximum number of members the load library will be able to contain.

NOTE='text' text to append to $PGM:LDLIBE.LOG when the LDLIBE job is attempted. The text can be used to document the change or why it was done. The file $PGM:LDLIBE.LOG must exist before the LDLIBE job is run for the text to be appended to the file. Otherwise, the text is ignored.

Command statements, if specified, tell the LDLIBE program what to do. The following is a list of parameters that you could specify on a command statement.

NAME='member'
Specify member to be processed. When adding a member, this is the name of the member in the load module file (IN=).

RENAME='name'
name to be assigned to the added member in the load library (if different from NAME=).

DEL=T To delete member.

REPL=T To replace member. This option should not be used unless the member will not to be used until the system is re-IPLed.

RENT=T Member is to be flagged as reentrant.

CNTID=n Usage count ID. Default is 0. The maximum is 50. (Additional information can be found under the LDCNTS writeup.)

DUMP=n *n* is the unit number of data set for dump. A logical record length of 80 is used. Any block-size that is a multiple of this can therefore be used.

RESTOR=n *n* is the unit number of data set for restore.

CHANGE=T To modify attributes of an existing member in the load library.

Note: If CHANGE=T is specified, the attributes which can be changed (RENT, CNTID, and NAME) should be specified as NWRENT=, NWCID=, and NWNAME= respectively.

Examples:

1. Creating a load library on a user data set.

```
/FILE 1 UDS(CODEXXXX) VOL(MUSIC1) OLD
/INC LDLIBE
LIBE=1,DISP='NEW',DIRSIZ=20
```

2. Adding the member XX1 to the system load library.

```
/FILE 1 UDS(CODELMOD) VOL(MUSIC1) OLD
/INC LDLIBE
LIBE='SYST',IN=1
NAME='XX1'
```

3. Changing some attributes and the name of the member XX1 in a UDS load library, adding the member XX2 and XX3, and replacing XX4.

```
/FILE 1 UDS(CODEXXXX) VOL(MUSIC1) OLD
/FILE 2 UDS(CODELMOD) VOL(MUSIC1) OLD
/INC LDLIBE
LIBE=1,IN=2
NAME='XX1',CHANGE=T,NWRENT=T,NWCID=1,NWNAME='XX5'
NAME='XX2'
NAME='XX3'
NAME='XX4',REPL=T
```

4. Dumping members of the system load library to a card image file.

```
/FILE 1 NAME(DUMP.FILE) OLD
/INC LDLIBE
LIBE='SYST'
DUMP=1
name specification     (see below)
     . . .
     . . .
```

name specification may be any of the following:

name	single name in col 1-8
name1-name2	all members from name1 to name2
prefix..	all members beginning with "prefix"
*ALL	all members

The RESTOR operation follows the same conventions.

Privileges Required: LSCAN, DREAD, SYSMNT or VIP.

LDLIST: Load Library Directory List

The LDLIST program displays a listing of a load library directory. The listing shows the name, length, starting block number, origin, entry point address, and attributes of each member.

Usage:

To list the system load library, enter LDLIST. For other load libraries, use the following job:

```
/FILE . . .    (if UDS load library)
/INCLUDE LDLIST
    parameters separated by commas
```

Parameters:

LIBE=n *n* is the unit number or DEB number of load library. If LIBE='SYST' is specified, the system load library is listed.

Privileges Required: LSCAN and DREAD.

LIBINDEX.CLEAN1: Clean Save Library Index Overflow Area

There are two programs for cleaning up the Save Library Index overflow area: $PGM:LIBINDEX.CLEAN1 and $PGM:LIBINDEX.CLEAN2.

Parts 1 and 2 are independent, but normally part 1 would be run before part 2.

Important: This program can be run only when there is no other activity on the Save Library. No other users should be on the system when this program is run. otherwise the Save Library Index may be damaged.

Usage:

```
/INC LIBINDEX.CLEAN1
NOWRT=T OR F
```

To run the program with disk writes bypassed, set NOWRT=T. To actually update the index, set NOWRT=F.

LIBINDEX.CLEAN1 reads Save Library Index blocks, looking for pointers to auxiliary blocks. It moves as many index entries as possible from the auxiliary block to the primary block, and, if all is moved, it sets the pointer in the original block to zero.

This cleanup makes the /LIBR command more efficient, and avoids running out of auxiliary blocks.

Messages are issued for the following cases:

1. all entries (if any) moved from auxiliary to primary and pointer in primary has been set to zero.

2. No entries were moved to primary, because there was not enough space in primary ("no change").

3. Some, but not all, entries were moved to primary.

4. The auxiliary block points to another auxiliary block. In this case, no change is made. (This should be very rare.)

Privileges Required: LSCAN, DREAD, CREAD, and VIP.

LIBINDEX.CLEAN2: Clean Save Library Index Auxiliary Area

The $PGM:LIBINDEX.CLEAN2 program (part 2) works in conjunction with $PGM:LIBINDEX.CLEAN1 (part 1 above). Part 2 cleans up the Save Library Index auxiliary area.

Important: This program can be run only when there is no other activity on the Save Library. No other users should be on the system when this program is run, otherwise the Save Library Index may be damaged.

Usage:

```
/INC LIBINDEX.CLEAN2
NOWRT=T OR F
```

To run the program with disk writes bypassed, set NOWRT=T. To actually update the index, set NOWRT=F.

This program moves used auxiliary blocks to the beginning of the overflow area. This is necessary from time to time because MFIO never frees auxiliary blocks, and would eventually run out of blocks in the overflow area. This causes severe errors in the library, because MFIO does not handle the error condition correctly. Errors 67, etc. occur.

Privileges Required: LSCAN, DREAD, CREAD, and VIP.

LIBINTEG: Check Save Library Integrity

The LIBINTEG utility, normally run as a batch job, reads the entire Save Library index and all file directory blocks, in order to check the correctness of the library space maps.

Note: This utility should be run only when there is no allocation or deallocation activity on the Save Library. To ensure this, no users should be allowed to sign on the system while LIBINTEG is being run.

Usage:

```
/FILE ... (as needed)
/INCLUDE LIBINTEG
parameters separated by commas (see below)
```

Parameters:

SPCINF=n — Unit number of an optional sequential output file (normally a UDS), LRECL=150. This records file name, directory pointer, and extent info for each private entry in the index. It can be scanned later to find which file or files are allocated to a particular space unit in the library. The default is SPCINF=0, meaning that this output file is not used.

COMINF=n — Unit number of an optional sequential output file (normally a UDS), LRECL=69. This records file name, directory pointer, and '0' (private) or '1' (common) for each entry in the index. If COMCHK=T, this file is later sorted by file name and read back to detect inconsistencies in the index (e.g. private and common have different index pointers, common has no private entry, more than 1 private or common entry for a file). The WORKU parameter must be specified if this parameter is used. The default is COMINF=0, meaning that this output file is not used.

COMCHK=T/F — Default is F if COMINF=0, otherwise T. Specifying COMCHK=F suppresses sorting and checking of COMINF data. The idea is that the sorting and checking could be done later by a separate program. This is desirable for very large Save Libraries.

WORKU=n — Unit number of a sort work file (a UDS). This is required when the COMINF parameter is used. The default is WORKU=0, i.e. no work file.

WRTBAC=T — This option causes reconstructed space maps to be written to disk, if they are different from the existing maps. The VIP privilege is required if this parameter is used. The default is WRTBAC=F (no writes to disk).

Note: Use the WRTBAC=T parameter with caution. Under no circumstances should any users be signed on to MUSIC when this option is used.

Privileges Required:

LSCAN, DREAD, and CREAD. The VIP privilege is also required if the parameter WRTBAC=T is used.

LIBSPACE: Show Library Space Status

Type LIBSPACE to display the space status of each Save Library data set. The system data sets making up the Save Library are numbered consecutively starting at 1. The data set names are normally SYS1.MUSIC.ULnn. LIBSPACE shows the amount of free (unused) space on each data set and the total free space available on the library as a whole. For each data set, the amount of space in the largest 5 free extents is also shown. Space values are in units of 1K = 1024 bytes.

As in the MVS system, MUSIC never uses more than 5 extents to satisfy a single request for primary or secondary space allocation. For example, it may be that a Save Library data set has 200K of free space but the largest 5 extents are only 2K each. That would mean that the largest file the system could allocate on that data set is 10K. The free space is there, but it is fragmented and therefore not useful for allocating larger files.

The LIBSPACE utility calculates a *fragmentation index* for each SL data set. It is a number between 0 and 1 which is a rough measure of how fragmented the space is. It is defined as (t-s)/8096, where t is the total free space in the data set and s is the sum of the 5 largest free extents. The higher the index, the more fragmented the space is. SL data sets with the highest fragmentation indices are the best candidates for the free space reorganization procedure described in *Chapter 20 - File System*. The MFMOVE utility is used to do the reorganization.

Privileges Required: LSCAN, DREAD and CREAD.

LOADPDS: Restore Members from OS IEBCOPY Tape

This program can restore, to the MUSIC Save Library, members of OS partitioned data sets which have been dumped onto tape using the IBM IEBCOPY utility.

Usage:

```
/FILE 1 TAPE VOL(xxxxxx) RECFM(U)
/INC LOADPDS
FILE=n, PREFIX='code:', SUFFIX='.s', SELECT=,REPL=, PUBLIC=
member names to be selected - one per line if required
FILE=n, etc.
FILE=n, etc
```

Parameters:

FILE=n *n* is file number on input tape.

PREFIX='code:' prefix for generated file names.

SUFFIX='.s' suffix for generated file names.

SELECT= can be TRUE or FALSE. If TRUE, a list of member names must follow.

REPL= can be TRUE or FALSE. if TRUE, existing files are replaced.

PUBLIC= if TRUE, files are made PUBLIC.

Control statements must be in order of ascending file number. The list(s) of files to be selected can terminate with an END OF FILE or "-" in column 1, if subsequent control statements are present. Selected files pertain only to the preceding control statements. Values specified on previous control statements are carried over to subsequent ones unless specifically modified.

Note: Record format "U" should be specified for the tape regardless of its actual record format.

Care should be taken when specifying the file number for standard labeled tapes, since MUSIC counts both header and trailer labels as files.

Privileges Required: None.

LOOKUP: Display the Table of Privileged Programs

This program displays the table of privileged programs. The table consists of file names and associated privileges, as defined in the system module LOOKUP. Each of the file names in the table should have the exec-only (XO) attribute, in order for the privileges to apply.

Usage:

To run the program, simply type LOOKUP when in *Go mode.

Privileges Required: LSCAN and CREAD.

LPA: Display the Contents of the Link Pack Area

This program displays the names of all the members in the Fixed Link Pack Area (FLPA) and the Pageable Link Pack Area (PLPA). Other information such as starting address, length, and entry point address is also displayed for each member.

Usage:

To run the program, enter LPA when in *Go mode.

Privileges Required: LSCAN and CREAD.

MAPMEM: Memory Map

This utility displays memory usage.

```
 MAPMEM: MEMORY REPORT    THU MAY 12, 1994  13:58:17

TOTAL MEMORY:  8192K,   PAGE POOL:  4740K
TCBS:  105, RCBS:   44, MAXRRS:   592K, MAXMPL:  8

                          V-START V-END   R-START R-END    LEN
LOW CORE                  000000  000FFF  000000  000FFF     4K
PAGE POOL 1                                000000  030000   192K
NUCLEUS                   800000  867FB0  030000  097FB0   415K
TCBS,UCBS,TERM BUFS,ETC.  867FB0  8850A4  097FB0  0B50A4   116K
BPOOL                     8850A4  8A5034  0B50A4  0D5034   127K
TCPR/TCPW BUFS FOR TCP/IP 8A5034  8A5034  0D5034  0D5034     0K
SYS DS BLKS, DEBS, ETC.   8A5034  8A6AF0  0D5034  0D6AF0     6K
LIB BITMAP CACHE (ULMAPS) 8A6AF0  8B7370  0D6AF0  0E7370    66K
FIXED LPA                 8B7370  A7F2E0  0E7370  2AF2E0  1823K
RAM DISK                  A7F2E0  AFF2E0  2AF2E0  32F2E0   512K
TRACE TABLE               AFF2E0  B032E0  32F2E0  3332E0    16K
NUCLEUS MAP TABLE         B032E0  B04000  3332E0  334000     3K
RCBS                      B04000  B5C000  334000  38C000   352K
PAGE POOL 2                               38C000  7FFFFF  4559K
```

Figure 17.5 - Sample run of the Map Memory program

Usage:

This program is available from the ADMIN facility option 1 14. You can also run this program by entering MAPMEM when in *Go mode.

Privileges Required: CREAD

MFACCT: Save Library Accounting Dump

This program scans the entire MUSIC Save Library and produces printed and/or card image output giving information on library utilization. A record is produced for each user code, giving the total number of files in the user's library, the total space occupied by the files, and the corresponding cost. The format of these records is given at the end of this writeup.

MFACCT is normally run when no users are signed on the system. If this is not the case, then the UPDUCR parameter described below must left as FALSE, since apparent mismatches would be likely.

To run this program from batch, the following deck set up is used:

```
/INCLUDE MFACCT
    input parameters
```

The input parameters are:

PRINT=TRUE or PRINT=FALSE

OUT=n (unit number for output)

CHARGE=X (charge rate in cents/K-bytes)

UPDUCR=TRUE or UPDUCR=FALSE

The PRINT parameter specifies whether printed output should be produced.
The OUT parameter specifies the unit on which the card images are to be produced. If 1, 2, 3, or 4 is specified, a /FILE statement for a tape or disk data set must be present (placed before the /INCLUDE statement). If OUT=0 is specified, no card image output is produced. The CHARGE parameter specifies the number of cents to be charged for each 1024 bytes of library space. The number of cents may be given as a floating point value (with a decimal point). The UPDUCR (for *Update User Control Record*) parameter indicates whether the User Control Records should be updated to match the actual library space if a discrepancy is found. If more than one parameter is used, they should be separated by commas. The default parameters are:

```
PRINT=TRUE,OUT=1,CHARGE=3.0,UPDUCR=FALSE
```

If the default parameters are not to be changed, the entire statement can be omitted.

If the program is run from a terminal, the input parameters are entered conversationally.

MFACCT Statements

Column	Contents
1- 4	Identification 'ARXD'
5- 8	First 4 characters of file ownership id; ends in a + if id is greater than 4 characters.
9-11	User subcode '000'
12-19	Blank
20-25	Time of MFACCT run, HHMMSS
26-31	Date of MFACCT run, DDMMYY
32-39	Total cost in cents
40	System number or blank (n from call SYSENV(2,n))
41-47	Total space of user's files, in units of 1024 bytes
48-52	Total number of files
53-55	Charging rate, in 1/10 cent per 1024 bytes
56	Blank
57-72	File ownership id (16 characters, trailing blanks)

Privileges Required: DREAD or MAINT.

MFARCH: Save Library New Program Archive

This program is designed to provide backup copies of files. It copies to tape all files which were created or modified since the previous run of the program. In addition, over a period of a specified number of program runs, it attempts to dump all files at least once, even those files which are not modified. Each file's attributes, tag field, etc. are dumped along with the file's data.

The program should be run on a regular schedule, such as once a night. It can be run while terminal users are active, but you may find it more convenient to shut down MUSIC and reload with NOTERM before running the archive.

If you have not run this utility before there is a chance that a large number of files have been changed on your system. This will cause this utility to backup an unusual number of files the first time it is run. This would probably exceed the capacity of a single tape reel. This could also happen if you changed an unusual number of files during a system upgrade. To avoid this you should run the SETFBN utility which will solve this problem and start you off in a way that a reasonable portion of your library is backed up each night over your cycle.

A companion program, MFCHEK, reads the output from MFARCH, ensures that it is readable, cross-checks several statement and file counts within the output, and updates the master backup number. The master backup number cycles between 1 and 255. It is kept in a system file, and is increased by 1 for each successful MFARCH/MFCHEK run. It is important to run MFCHEK after running MFARCH; otherwise the master backup number will not be updated, and the same files will be dumped again the next time MFARCH is run.

If desired, two identical dump tapes can be produced. These two tapes normally would be interchanged on their drives before being read by the checkout program. MFCHEK compares the two tapes to make sure they are identical. This procedure minimizes undetected tape hardware problems and checks that the data is valid.

The MFREST program is used to retrieve files from a dump tape and restore them to the library.

To run the archive program, the following statements are used:

```
/FILE 1 TAPE BLK(4000) LRECL(80) VOL(TAPE1) OLD RECFM(U)
/FILE 2 TAPE BLK(4000) LRECL(80) VOL(TAPE2) OLD RECFM(U)
/INCLUDE MFARCH
UNITS=1,2,PERIOD=n,MAXOLD=m
```

If only one copy of the output tape is desired, omit the file statement for unit 2 and use UNITS=1 instead of UNITS=1,2. A different tape blocksize may be used if desired. To fit as much data on a tape as possible, use a large block size, such as BLK(16000), and use the highest density, i.e. DEN(6250) if available.

UNITS=n,m specifies the output unit number(s) for the dump. If two units are specified, two identical copies of the dump are produced. specifying unit 0 suppresses dump output. Default is units=1. Units 3 and 4 are reserved for work files and hence cannot be used.

PERIOD=n specifies the number of MFARCH/MFCHEK runs over which the program attempts to dump every file at least once, even those files which are not modified. The default is PERIOD=12. For example, with PERIOD=12, a set of 15 tapes could be used. Each MFARCH run would use the next tape in the cycle, so that the set of tapes should contain at least one copy of every file.

A file is a candidate for archiving if (1) the file was created or modified since the last MFARCH/MFCHEK run, or (2) x is greater than or equal to PERIOD, where x=k-f (if k-f>0) or x=k-f+255 (if k-f<=0). Here, *k* is the master backup number and f is the file's backup number. The file's backup number is set to *k* once it has been archived by MFARCH.

MAXOLD=m is the maximum number of non-new, non-modified files to be dumped (case (2) above). The default is MAXOLD=2500. MAXOLD*PERIOD should exceed the total number of files in the library, since approximately (total files)/PERIOD *old* file can be expected as candidates for dumping on each run.

MAXREC=n If a non-zero number is specified, it is the approximate maximum number of 80-byte logical records that should be written to each output tape. After that many records, the program will stop, with a message that there are more files to do and that the operator should run another MFARCH job to dump the remaining files to the next tape. Also, a job return code 2 is set (if there are no serious errors). This is to handle the case where there are too many files to fit on one tape. MAXREC should be calculated as about 90% of the number of records on a full tape, based on tape density, blocksize, tape gap length, etc. For example, for a 2400-ft tape at 6250 bpi, blksize 32000, gap 0.3 in., a full tape holds about 2,120,000 80-byte records, so use about MAXREC=1900000. The MAXREC=n parameter is ignored if dumping files from specified library data sets (ARCLIB=). Default is MAXREC=0, i.e. the program will assume an infinite size tape.

Note: an MFCHEK job should be run before the second MFARCH job, so that the master backup number will be updated. Otherwise the same files will be dumped again.

MBNFIL='filename'
specifies the file which contains the master backup number. The default is MBNFIL='$PGM:MASTER.BACKUP.NUM'. and this file is automatically set up during MUSIC installation. The same file name should be specified to the MFCHEK program, via the MBNFIL parameter.

ARCLIB=n1,n2,...
causes all files in library numbers *n1, n2,* ... to be archived to tape. Library number *nn* is the system data set SYS1.MUSIC.ULnn. From 1 to 100 numbers may be specified. The regular dumping of new or modified files is not done if the ARCLIB parameter is used.

NAMES=n requests that the names of all archived files are to be written to unit number *n*, one file name per record, starting in column 1. The unit number *n* must be from 7 to 15. The file must have record format V or VC, or have a record length of 64 or more.

LISTNG=n specifies an output unit number for the sorted listings of the names and attributes of the archived files. The unit number *n* should be from 6 to 15. This provides a way of suppressing the listing or directing it to a file. The file should have record format V or VC, or have a record length of at least 133. LISTNG=0 suppresses the listing output. The default is LISTNG=6 (output to the printer).

VERTAP=T causes the output tape or tapes to be read to check that an archive tape is mounted. The first record of the first tape file is read and checked for *MFARCHIVE in columns 1-10. The default is VERTAP=F (no checking). VERTAP must be omitted or specified as F if a tape is being written by MFARCH for the first time.

VERVOL=T causes the volume name in the first record of the output tape or tapes to be compared with the volume name on the /FILE statement. If the names are different, MFARCH assumes that the operator mounted the wrong tape and does not do the archive. This parameter is ignored unless VERTAP=T. The *MFARCHIVE header record should contain VOL=vvvvvv in columns 51-60, where *vvvvvv* is the volume name. Starting with MUSIC/SP Release 1.1, MFARCH puts the current volume name on the header record when it writes to the tape. The default is VERVOL=F (no volume name checking).

MFCHEK=T Causes the MFCHEK function to be done at normal end of MFARCH, as part of the same job. The advantage of this is that a second tape mount is avoided. MFCHEK reads and checks the output tape(s) and adds 1 to the master backup number. Refer to the description of the MFCHEK utility. When MFCHEK=T, MFARCH rewinds the output unit(s) and calls MFCHEK as a subroutine. All of the required parameters for MFCHEK must follow the MFARCH parameters in the input stream. The default is MFCHEK=F.

UPDATE=F (Normally not used) causes the program not to change the file's backup number as the file is dumped. The default is UPDATE=T.

In the following example, only one output tape is produced, volume name checking is done, file names and output listings are written to files instead of being printed, and the MFCHEK function is done in the same job. The final parameter line is for MFCHEK. The tape number 03 in the /FILE statements could be generated automatically by the AUTOSUB utility when this job is submitted to MUSIC.

```
/FILE 1 TAPE VOL(ARCH03) LR(80) BLK(16000) DEN(6250) OLD
/FILE 11 NAME(ARCH03.LIST) RECFM(VC) NEW(REPL) SPACE(100)
/FILE 12 NAME(ARCH03.NAMES) LRECL(22) NEW(REPL)
/INCLUDE MFARCH
UNITS=1,PERIOD=8,MAXOLD=2000,VERTAP=T,VERVOL=T,
LISTNG=11,NAMES=12,MFCHEK=T
UNITS=1,UPDMBN=T
```

Note on work files:

Work files on units 3 and 4 are defined in the MFARCH executor file. /FILE 3 should have a record length of 80 and at least as many records as the number of files to be archived. /FILE 4 is a sort work file, and must be twice that size. The standard MFARCH file allows for archiving up to 7500 files. If you need more, provide larger work files. /FILE 4 must be a UDS file.

Information on 3480 and 3490 Cartridge Tapes:

You don't have to change anything in the MFARC2 jobs to use 3480 cartridge tapes. You can specify

DEN(6250) or DEN(HIGH), they give the same result. The actual density is 38000bpi, but MUSIC does not know about that.

For the MFARC2 tape size parameters, use TAPDEN=38000,TAPSIZ=500, TAPGAP=0.3. A high block-size like 32000 is recommended. These numbers are equivalent to 2.1 million 80-byte records. (See the TAPELEN program.)

The IBM manuals say that nominal capacity of a cartridge recorded in 18-track mode is 200MB, i.e. 2.5 million 80-byte records. Using density 38000BPI, block size 32000, and gap 0.12 inches, this gives 501 feet. It's not clear what the actual gap size is.

Some 3490 models can record in 36-track mode, resulting in a capacity of 400MB. Also, the 3490E can use the "enhanced capacity" cartridge (physically different), which doubles the capacity to 400MB (18-track mode) or 800MB (36-track mode).

All of the above capacity numbers are for uncompacted data, i.e. IDRC (Improved Data Recording Capability) off. Capacity with IDRC can be much higher.

Privileges Required: LSCAN, FILES, MAINT, DREAD, and CREAD.

MFARC2: Selective File Archive

This program archives to tape (or disk) a specified group of files. The dump format is the same as that produced by the MFARCH utility. The MFCHEK program (with UPDMBN=FALSE) can be used to check the dump, and MFREST can be used to retrieve files from the dump.

Usage:

The following control statements are used to execute the program:

```
/FILE 1 TAPE BLK(4000) LRECL(80) VOL(TAPE1) OLD RECFM(U)
/FILE 2 TAPE BLK(4000) LRECL(80) VOL(TAPE2) OLD RECFM(U)
/INCLUDE MFARC2
parameters separated by commas (see below)
code:name       )
code:name       )   optional additional file names
...             )
```

One or two output tapes may be used. For one tape, omit the second /FILE statement and specify UNITS=1. For two tapes (two copies of the dump), specify UNITS=1,2. A different tape blocksize may be used if desired. Units 3 and 4 may not be used.

Parameters:

The following may be specified on the parameter statement. If more the one statement is needed, terminate with a comma and continue parameters on the next statement.

UNITS=n,m (or TAPE=, OUTPUT=, OUT=)
: The output unit number or numbers. A zero unit number suppresses output. The default is UNITS=1.

TAPFIL=n
: When archiving to a multi-file tape, this parameter specifies the sequence number (1,2,...) of the tape file to be written. The default is TAPFIL=1. Note that writing to file *n* (TAPFIL=n)

does not disturb existing files 1 through n-1, but destroys any following files on the tape.

CODES='xxxx','xxxx',...
: A list of one or more userids, or userid patterns, whose files are to be archived. A maximum of 3000 userids may be specified. For each userid all files belonging to that userid are archived. These are actually file ownership ids, rather that userids. A pattern contains wild characters * and/or ?. If the UDATE parameter is given, only files satisfying the date condition are archived. The special userids *COM and *USR may not be specified in the CODES parameter.

ALL=TRUE
: This causes the entire library (all codes) to be archived, subject to the UDATE parameter if UDATE is used.

UDATE='ddmonyy'
: Specifies a 7-character date, for example, 01JAN78. The 3-letter month abbreviation consists of the first 3 letters of the month's name. Only files whose last-read and last-written dates are both less than or equal to UDATE are archived. This provides a means of archiving inactive files.

NAMES=n
: This parameter, if used, requests that the names of all archived files are to be written to unit number *n*, one file name per record, starting in column 1. Do not use unit 3 or 4, since the program uses these as work files. For example, the resulting file of names could be used to purge the archived files: change the names to purge commands by using the editor command C//PURGE /*, then feed the purge commands to the Editor. The file must have record format V or VC, or have record length 64 or more.

LIST=FALSE
: This suppresses the printed list of archived file names. The default is LIST=TRUE.

JOBFIL='filename'
: This parameter is used if a single tape is not large enough to hold the output. It allows continuation jobs to be scheduled, in order to complete the archive on additional tapes. *filename* is the name of a file containing sample control statements to be used for the continuation job. It should specify the same parameters as in the initial job. Do not place /ID, /PASSWORD, /PAUSE, or /END statements in this file. The file is not modified by the program. A new file is created for the submit. Characters "01" in any tape volume names in /FILE statements are changed to the run number, and appropriate run=n and indstb=n parameters are added. Characters "01" at the end of a file name, defined by a /FILE statement with the NEW or NEW(REPLACE) option, are automatically replaced by the 2-digit run number (this is useful for the output file for the NAMES=n option). The new file is called @ARCJOB.CONT, and the continuation job is submitted to the internal reader, class 1, with a /PAUSE and using the same code/subcode as the current job. Example of file contents:

```
/FILE 1 TAPE VOL(ARC01A) LR(80) BLK(16000) DEN(6250)
/INC MFARC2
ALL=T,LIST=F,JOBFIL='THIS FILE',
TAPSIZ=2400,TAPDEN=6250,TAPGAP=0.3,TAPBLK=16000
```

: (the 2nd tape volume will be ARC02A, the 3rd ARC03A, etc.)

TAPSIZ=n,TAPDEN=n,TAPGAP=x,TAPBLK=n
: Values used in estimating the length of tape used and available. These parameters are used only if the JOBFIL parameter is specified. TAPGAP is REAL*4. The others are INTEGER*4. TAPSIZ is in feet, TAPDEN in BPI, and tapgap (interblock gap size) in inches. TAPBLK is the tape blocksize. The program will reduce TAPSIZ by 50 feet as a safety margin.

Defaults are: TAPSIZ=2400,TAPDEN=1600,TAPBLK=16000 the default interblock gap is TAPGAP=0.3 if TAPDEN=6250, or TAPGAP=0.6 if tapden is other than 6250.

Note: The nominal tape gap for 9-track 1600 BPI is 0.6 inches; for 6250 BPI it is 0.3 inches.

A list of additional specific file names to be archived may be specified after the parameter statement, one file name per statement, starting in column 1. Each file name must include the 4-character userid. The special userids *COM and *USR may be used here, but they will automatically be changed to the actual userid of the owner of the file. The UDATE parameter does not apply to these files.

Wild characters * and ? (file name patterns) may be used in the file names, in order to archive groups of files. Files to be excluded from the archive can be specified by placing - (dash) in column 1 and a name or pattern starting in column 2. Exclusions do not apply to specific non-pattern file names or to the CODE parameter.

The optional parameter DUMPNAME=filename may appear on the file name statement. It specifies a different file name to appear in the archive output. For example:

```
/FILE 1 ...
/INCLUDE MFARC2
UNITS=1
ABCD:FILEX
ABCD:FILEY      DUMPNAME=ABCD:FILEPQ
ABCD:SAMPLE     DUMPNAME=EFGH:TEST
```

The files ABCD:FILEY and ABCD:SAMPLE will appear in the output with names ABCD:FILEPQ and EFGH:TEST.

The following example shows how all files belonging to several codes (ABCD, DEFG, and all codes starting with JK) can be dumped in one run:

```
/FILE 1 ...
/INCLUDE MFARC2
CODES='ABCD','DEFG','JK*'
```

Privileges Required:

LSCAN, FILES, MAINT, SYSCOM and CODES. SYSCOM and CODES are needed only if JOBFIL parameter is used.

MFCHEK: Save Library Archive Checkout

The statements used to run the Save Library archive tape checkout program are as follows:

```
/FILE 1 TAPE VOL(xxxxxx) SHR LR(80) BLK(nnnnn) RECFM(U)
/FILE 2 TAPE VOL(xxxxxx) SHR LR(80) BLK(nnnnn) RECFM(U)
/INCLUDE MFCHEK
parameters separated by commas (see below)
```

As with the archive program, omit the second /FILE statement and use UNITS=1 if only one tape is involved.

The following may be specified on the parameter statement.

UNITS=n,m The input unit number or numbers. For two tapes, specify UNITS=1,2. The default is UNITS=1.

UPDMBN=TRUE This causes the program to add 1 to the master backup number if the checkout is successful. This should always be specified when MFCHEK is run in combination with MFARCH, except that the UPDMBN=TRUE parameter must not be used if the ARCLIB parameter of MFARCH was used. MFCHEK can also be used to check dump tapes produced by the MFARC2 utility described in this chapter. In that case, omit the parameter UPDMBN=TRUE. The default is UPDMBN=FALSE.

NAMES=TRUE This causes a listing of the names of all the files contained on the archive tape to be printed. The default is NAMES=FALSE.

INFO=TRUE This causes the information line for each file contained on the archive tape to be printed. The information line has the file name, logical record length, record format, access control options, total space allocated, the number of 512-byte blocks dumped, and the time and date the file was archived. The default is INFO=FALSE. If the file name is larger than 22 characters, the complete file name appears on the additional second INFO line.

TAPFIL=n This specifies the tape file sequence number (1,2,...) containing the archive output to be checked. The default is TAPFIL=1 which means the first tape file is to be checked. This parameter is used only when reading from a multi-file tape.

NFILES=n Specifies the number of tape files to process. Default is 1. For example, TAPFIL=3,NFILES=2 would process tape files 3 and 4.

CHKMBN=T Specifies that the master backup number on the tape header record (*MFARCHIVE record) is to be compared with the current master backup number. If they are unequal, the master backup number is not updated. This parameter is used only if UPDMBN=T. the default is T if UPDMBN=T, F otherwise.

MBNFIL='filename'
Specifies the full file name of the file containing the master backup number (max 22 chars). This is used only if UPDMBN=t. Default is MBNFIL='$PGM:MASTER.BACKUP.NUM'.

SCAN='xxx' Causes info line (see the info parameter) to be printed for each file whose full name begins with the specified characters *xxx*. *xxx* can be 1 to 64 characters long. File names include the userid (code:...). For example, SCAN='ABCD:TEX' prints info on all files belonging to userid ABCD and starting with 'TEX'. The name part of the file name must not start with a backslash. Use SCAN='ABCD:XYZ" but not SCAN='ABCD:\XYZ'

If the archive or checkout job fails, or checkout does not print the message `CHECK-OUT OK,` either or both jobs should be rerun, as appropriate, until a successful checkout is obtained. A rerun of the archive job dumps the same files as the original run.

Privileges Required: LSCAN and FILES.

MFHASH: Save Library Index Hasher

MFHASH is used to locate the Save Library index block number, given the file name or UCR code. The first block of the index is numbered 1.

To locate the index block number for a given private file name specify:

```
NAME='code:filename'
```

To locate the index block number for a given file name in the common index specify:

```
NAME='*COM:filename'
```

To locate the index block number for a UCR entry specify:

```
CODE='code'
```

Privileges Required: None.

MFINDEX: Summarize Save Library Index Usage

This program reads the Save Library index and displays various values and tables for the number of files, the index space used, file name lengths, etc.

Privileges Required: LSCAN.

MFMOVE: Reorganize Save Library Free Space

The MFMOVE utility can be used to move files from fragmented library data sets, eliminating this fragmented condition. (The LIBSPACE utility can be used to locate these fragmented data sets.)

It can be run on batch or on a terminal, but must be run on a code with the VIP privilege. The program can process a number of Save Library data sets in a given run, and functions by moving files from those specified libraries to free space elsewhere. Normally only a small number of libraries (say 4 or 5) are specified. The program suppresses allocation on the libraries it is processing for the duration of its execution. For this reason, there should be sufficient space in the remaining library data sets to contain the files to be moved. If for some reason the program is cancelled or runs out of time, the specified library data sets will be unavailable for the allocation of new files until the next IPL.

Example:

```
/SYS TIME=MAX
/FILE 1 NAME(ERRS) NEW(REPL)
/INCLUDE MFMOVE
LIBS=5,6,8,9
```

This will move files from Save Library data sets 5, 6, 8 and 9 to elsewhere in the Save Library. If for some reason a file cannot be moved, it is left as it is and its name and the error condition are recorded on unit 1.

Privileges Required: LSCAN, CREAD and VIP.

MFREST: Restoring of Archived Files

The MFREST utility reads an archive tape produced by the MFARCH or MFARC2 programs, searches for specified files, and copies the files to disk, using either the original names or new names.

The files to be restored are specified by one or more parameter statements. Each parameter statement gives the name of a file (or group of files) to be searched for on the tape, using the NAME parameter, and optionally, gives the name of the file (or group of files) to which the files are to be restored, using the TO parameter. The first parameter statement may specify other options (INPUT, TAPFIL, FIXUP, ALL, RLSE, REPL, SETUI, FIXUCR, SETBUP, PERIOD, and CURMBN) which apply to the entire job.

The control statement set up is as follows:

```
/FILE 1 TAPE VOL(vvvvvv) BLK(nnnnn) LRECL(80) SHR RECFM(U)
/INCLUDE MFREST
first parameter statement
second parameter statement
...
```

Parameters:

NAME='filename' or NAME='prefix*'
: The full file name (including ownership userid) to be searched for in the dump. If the last character is an asterisk (*) or a plus sign (+), all files whose names start with the specified prefix will be restored. NAME= may be abbreviated to N=. Up to 3000 NAME parameters can be specified in a single run of this utility. If the NAME parameter is used, the ALL=TRUE parameter should not be used.

TO='filename' or TO='prefix*'
: The full file name (including ownership userid) to which the file is to be restored. Usually, this parameter is omitted, in which case, the original name is used if possible. TO= may be abbreviated to T=. If the TO parameter is used, the ALL=TRUE parameter should not be used. An ending * or + indicates a prefix.

INPUT=n or IN=n
: The input unit number for reading the dump. The default is INPUT=1.

TAPFIL=n
: Specifies the tape file sequence number to restore from. The default is TAPFIL=1.

EOFTXT='xxxxx'
: Specifies a character string (up to 80 chars) which indicates end of data on the input unit. When this parameter is specified (and the string is nonblank), the program keeps reading after EOF (used for restoring from MUSIC/SP service tape).

FIXUP='x'
: Specifies a fixup character, *x*, to be used for generating an alternate name if the receiving file already exists (and should not be replaced). The fixup character is added to the end of the TO name, and the resulting name is tried. At most, 3 such retries are done per file. The default is FIXUP='$'. To prevent fixups, specify FIXUP=' '.

ALL=TRUE
: Causes all files to be restored. Default is ALL=FALSE. Do not use the NAME or TO parameters if ALL=TRUE is used.

RLSE=TRUE
: Releases unused space as files are restored. Default is RLSE=FALSE.

REPL=TRUE
: Causes the TO file to be replaced if it already exists on the library. Default is

REPL=FALSE.

LIST=F — Suppresses messages for successful restores. Message is still issued if name FIXUP was done or check sum was incorrect. Default is LIST=T.

SETUI=TRUE — Causes the program to restore certain file attributes. The attributes affected by this option are: usage count, use dates, code of creator, code of last writer. The default is SETUI=FALSE (i.e., do not restore these attributes).

FIXUCR=TRUE — With this option, MFREST temporarily creates a UCR (User Control Record) for the target code if one does not exist, or temporarily increases the UCR limits if necessary. This enables the file to be restored. After the file is restored, the temporary UCR add or change is undone. The default is FIXUCR=FALSE, in which case a file error 40 or 41 would occur if there is no UCR or the UCR limit is reached. Note that the code running MFREST must still have an adequate UCR, even if FIXUCR=TRUE is used, in order to allow creation of a temporary file for the restore. The UCR fixup is done when the temporary file is closed and renamed to the target file.

SETBUP=TRUE,PERIOD=n,CURMBN=m

The parameter SETBUP=TRUE causes each restored file to be marked as backed up, and the file's backup number to be set to a nonzero value within the range specified by PERIOD=n and CURMBN=m.

The default is SETBUP=FALSE, in which case PERIOD and CURMBN are ignored, and each restored file appears as a new or modified file and its backup number is set to zero. This means that the incremental archive utility (MFARCH) will dump the file to tape on its next run.

SETBUP=TRUE is useful when many files are being restored, to prevent MFARCH from archiving the files all at once. PERIOD=n specifies the MFARCH period and CURMBN=m specifies the current master backup number. The restored files are assigned backup numbers evenly distributed among the n numbers preceding m (0 is skipped and 255 is considered to precede 1.) For example, PERIOD=4, CURMBN=1 spreads the backup numbers among 255, 254, 253, and 252. PERIOD=5, CURMBN=3 spreads them over 2, 1, 255, 254, and 253. The defaults are PERIOD=12, CURMBN=1.

PERIOD=n — Used with SETBUP=T, it specifies the MFARCH period number.

CURMBN=m — Used with SETBUP=T, it specifies the current master backup number.

MBNFIL='filename'

Specifies the full file name of the file containing the master backup number (max 22 chars). This is used only if UPDMBN=T. Default is MBNFIL='$PGM:MASTER.BACKUP.NUM'.

Important note: The parameters INPUT, TAPFIL, FIXUP, ALL, RLSE, REPL, SETUI, FIXUCR, SETBUP, PERIOD, and CURMBN may only be specified on the *first* parameter statement, but they automatically apply to all the other parameter statements also.

Examples of parameter statements:

```
NAME='ABCD:FILE1',REPL=TRUE
NAME='XY*',TO='PQ*'
NAME='CCDE:PROGA',TO='CCRM:COPY.PROGA'
NAME='ABCD:PAY*',TO='ABCD:OLDPAY*'
```

This restores ABCD:FILE1 to ABCD:FILE1, all codes XY... to PQ..., CCDE:PROGA to CCRM:COPY.PROGA, and all files ABCD:PAY... to ABCD:OLDPAY... Any existing files will be replaced, since REPL=TRUE is specified on the first parameter statement.

The following example restores three specific files, with replacement of existing files:

```
/FILE 1 TAPE VOL(ARCVOL) BLK(4000) LRECL(80) SHR
/INCLUDE MFREST
REPL=TRUE,NAME='ABCD:PROG1.S'
NAME='ABCD:PROG1.LKED'
NAME='XY25:SAMPLE'
```

The following example restores all files for code DEFG, but adds XX at the beginning of each name:

```
/FILE 1 TAPE VOL(TAP123) BLK(3200) LRECL(80) SHR
/INCLUDE MFREST
NAME='DEFG:*',TO='DEFG:XX*'
```

Privileges Required: LSCAN, FILES, and MAINT.

MOVEPDS: Restore Files from OS IEHMOVE Tape

This program reads a sequential card-image file (normally on tape) which is a dump, produced by the MVS utility 'IEHMOVE', of a partitioned data set (PDS). It outputs the PDS members either as a single sequential file with each member preceded by a ./ ADD statement, or as individual save files. The record format of the original PDS must be fixed (i.e., F or FB or FBA, etc.). It must not be V or U. The original PDS must not contain MVS load modules.

Control Statements:

```
/FILE 1 ... (The input tape or file, LRECL=80)
/FILE 2 ... (The output file, needed only if DOTSL=T)
/INCLUDE MOVEPDS
parameters (separated by commas)
member names (needed only if SELECT=T)
```

Parameters:

FILE=n	The file sequence number (1,2,...) of the input, if reading from a tape. (Files are counted as on a non-labeled tape.) Default is FILE=1.
DOTSL=T or F	If DOTSL=T, output is to a single sequential file on unit 2, with each member preceded by "./ ADD NAME=xxxxxxxx". Output LRECL=80. "./ ALIAS NAME=yyyyyyyy" statements may also be put out. If DOTSL=F (the default), each member is put out as a separate save file (alias names are ignored.)
SELECT=T or F	If SELECT=T, the member names to be restored follow the parameters, one statement starting in column 1. They may be in any order. The maximum number of names is 500. These names must be actual member names, not alias names. If SELECT=F (the default), all members are restored.

The following options PREFIX, SUFFIX, REPL, PUBL, and LRECL only apply if DOTSL=F.

PREFIX='xxx'	Prefix characters to be placed before the member name when forming the save file name. Default is no prefix characters.
SUFFIX='xxx'	Suffix characters to be placed after the member name when forming the save file name. Default is no suffix characters.
	Note: The total number of prefix and suffix characters must not exceed 9 (or 14 if PREFIX starts with "CODE:").
REPL=T or F	Replace an existing save file. Default is F.
PUBL=T or F	Create public save files. Default is F.
LRECL=n	Logical record length for the save files. Truncation or blank padding will be done. The default is the logical record length of the original PDS.

Privileges Required: None.

NEWINDEX: Create a New Save Library Index

This program creates a new save library index data set of a different size.

Warning: There must be no other users or BTRMS on the system (and no batch jobs) when this program is run.

This program copies all save library index entries from the production index to a new index, rehashing the names according to the size of the new index. This is needed in order to change the number of segments in the index, or the number of blocks per segment.

The new index is defined on unit 1 as:

```
/FILE 1 UDS(%xxxxxxx) VOL(xxxxxx) OLD
```

Unit 2 is a work file, logical record length is 80, with room for at least as many records as there are entries in the index. Units 3 and 4 are sort work files, each at least as big as unit 2.

The privileges that are required are: VIP, DREAD, and LSCAN.

Return codes:

```
  0 success
  1 failure
 >1 system error
```

Notes:

1. The new index must be formatted (by the format utility) with 0's, i.e. using the option data=z00 in format, before this program is run.

2. When you initialize the new index by the ULINIT utility, use the "NSEG=n" parameter to specify the desired number of segments. The number of segments must be a prime number, e.g. 7, 11, 13, 17, 19, 23 or 29. ULINIT must be run after formatting and before this program. When increasing the index

data set size, you should also increase the number of segments (the NSEG=n parameter for ULINIT), so that the size of each segment stays about the same. Otherwise /LIB and /DIR commands will be slower. The number of blocks per segment is approximately m/(n+1), where m is the number of blocks in the entire data set and n is the number of segments.

3. The MFINDX subroutine used must be the version that puts the index entry into common block /MFINEN/. (MFINDX distributed with music release 5.2 does not do this, but the one in music/sp 1.0 and later releases does.)

4. If this program must be rerun, the new index must be reformatted to zeros, and ULINIT run, before the rerun.

5. This program may take a long time to run.

6. After the entries have been successfully moved, the system catalog must be changed to point to the new index and music re-IPL'd before any changes are made to save library files, otherwise the new index will not reflect the changes.

7. See also file $PGM:NEWINDEX.DOC.

Important: Verify that all programs worked before changing the catalog to use the new index.

NOWDOL: User Time Accounting Update

This program reads the accounting statements produced by the accounting dump program (ACTDMP) and adds processing unit charges plus connect charge for each user code to the NOW$ field of the user's Code Table record. If the installation wishes to maintain the NOW$ fields, the NOWDOL program should be run daily using the output of the program ACTDMP.

The deck set up for executing NOWDOL is as follows:

```
/INCLUDE NOWDOL
```

A user will be prevented from signing on MUSIC if the user's NOW$ value exceeds the amount allocated to the user with the MAX$ parameter of the CODUPD program. The CODUPD program can be used to increase the user's MAX$ limit.

A user may use the PRINT$ command of the User Profile Program (PROFIL) to inspect the MAX$ and NOW$ fields. The system administrator may also inspect these fields by using the '$' option on the GET command of CODUPD. The NOW$ field is initialized to zero when a code is authorized.

Privileges Required: MAINT and CODES.

NUCGEN: Nucleus Generation Utility

This utility is run whenever changes are required to the MUSIC nucleus. This nucleus contains the main controlling program for the entire MUSIC system. This controlling program is made up of more that 50 modules. It also contains the I/O devices and installation-dependent parameters. When MUSIC is IPLed, the nucleus is read into main storage where it resides when MUSIC is operational. Changes to the nucleus

made by this utility take effect the next time MUSIC is IPLed.

The NUCGEN utility produces a sequential file that contains a job stream that will create a new nucleus when run. This sequential file can be written on tape. This tape can then be IPLed which will run the job stream to put on the new nucleus. If this nucleus does not work correctly then you can go back to using the tape that contained the previous copy of the nucleus by IPLing from that one.

The NUCGEN utility can also be run so that the new nucleus is written to disk without using a tape. This involves running the SYSGEN1 utility using as input the sequential file produced by this NUCGEN utility. While this process is faster than using a tape, it does not allow you to go back to a copy of the previous nucleus should the new one cause MUSIC not to operate correctly.

The ADMIN facility described in the *MUSIC Administrator's Guide* uses the method that bypasses the use of tape. A tape backup copy of the existing nucleus can be made first by selecting one of the menu options.

When using the ADMIN facility you only need be familiar with the device specification statements and options rather how to run the NUCGEN utility itself. Before we describe those parameters, we will describe how to run the NUCGEN utility directly without using the ADMIN facility.

The following is the way to run NUCGEN to produce a tape that contains a job stream that will form a new MUSIC nucleus The first time you run this utility you specify the file $GEN:NUCLEUS as input. It creates an output tape called OURGEN that will be used in a later step. This OURGEN tape contains all the object decks which form the MUSIC nucleus. (Subsequent runs of the NUCGEN utility can use the output of the last run as input.) For more information about modifying MUSIC's nucleus, see *Chapter 18 - System Internals*.

Prepare and run the following job using the options and device specifications as detailed below. Consult the printed output to verify that this program has run successfully. Omit the /ID and /END commands if the job is submitted from the terminal. The file NUCGEN.SAMPLE contains sample control statements for this step.

```
/ID                 $000 000 060 999 999
/FILE 1 NAME($GEN:NUCLEUS) SHR
/FILE 2 TAPE BLK(80) LRECL(80) VOL(OURGEN)
/FILE 4 NAME($GEN:NUCMAP) NEW(REPL)
/INCLUDE *COM:NUCGEN
/INCLUDE *COM:MDLSYG
.... options ....        (see below)
.. device statements ..  (see below)
DEVEND
... replacement nucleus object modules (optional)
/END
```

Notes:

1. The /FILE 2 statement can be replaced with the following if SYSGEN1 will be used.

   ```
   /FILE 2 NAME($GEN:NUC) NEW(REPL) SP(500)
   ```

2. If a replacement object module is specified more than once, the first one is used.

Options

Separate all options from each other by commas. The following gives a sample set of options. Each option is explained below.

```
IN=1,OUT=2,XMAP=4,
SYSRES='160',CONSOL='01F',PRINTR='000',
CFACT=100,3,
REGION=3000,0,
BPOOL=300,
MAXTRC=15,MAXCOR=65536,MAXRRS=392,RAMDSK=512,ULMAPS=1,
LEVEL='MUSIC',SIGNON='MUSIC/SP,'
```

IN — Specifies the unit number of the input file. Use IN=1 in this case.

OUT — Specifies the unit number of the output file. Use OUT=2 unless the nucleus is to be punched on cards in which case use OUT=7. The output tape will always have one file.

TAPFIL — Specifies which file of the input tape is to be used. If input is from a file, then use TAPFIL=1. If not given, then TAPFIL=1 is assumed.

XMAP — XMAP=4 specifies that a copy of the storage map will be produced on unit 4. This storage map shows the location of the modules which form the MUSIC nucleus. A /FILE statement for unit 4 must then be provided. The default of XMAP=0 will not produce a copy of the map on a data set.

SYSRES — Specifies the disk address where the nucleus is to be written. (This is not the address of the starter system pack.) Normally this is the address of volume MUSICX. Make sure that you enclose the address in single quotation marks as in the example above.

CONSOL — Specifies the device address of the console to be used when the nucleus is written on disk.

PRINTR — Specifies the device address of the printer to be used when the nucleus is written to disk. If specified as PRINTR='000', then no storage map is printed when the nucleus is written to disk.

CFACT — Specifies two charging factor numbers. Both are used to calculate the number of service units a job uses. The first number gives processor time component. It is recommended that a number be chosen that is approximately 100 times the MIP rate of your processing unit. MIP means million instructions per second. It is not a precise number but is useful to determine a reasonable number for this option. If you upgrade to a faster processor you then need only adjust this number and users will see their jobs taking approximately the same number of service units. Some examples might be: 20 for 4331M1, 42 for 4331M2, 70 for 4361M3, 100 for 4361M4, 160 for 4341M12, 230 for 4381M1, 300 for 4381M2, 63 for 9370M20, M40, 92 for 9370M60, 230 for 9370M90, 200 for 4381-11, 300 for 4381-12, 400 for 4381-13.

The second number is used to factor in the I/O charge. The I/O charge is done by multiplying the number of SIO instructions by this factor and then dividing by 300. Paging SIOs are not counted. A recommended number is 3 meaning that 100 SIOs will result in a charge of one service unit.

REGION — This option takes two numbers. The first specifies the maximum size of the user region. Valid values range from 256 to 3000 meaning 256K to 3000K. The number must be a multiple of 4. The default is REGION=1024,0. Note that if the value for REGION is too small, a large program like MAIL will not be able to run.

The second number specifies the region size value to be used when the user specifies /SYS REG=MAX. The use of REG=MAX should be discouraged. This parameter is provided for compatibility with the IUP version of MUSIC. For example, these sites can install MUSIC/SP and increase the user region maximum from 256K to 2000K without causing

large charges for users who used REG=MAX when they really just wanted 256K not the new larger 2000K. If the number is set to 0, then users will not be allowed to use MAX specification. The default for the second number is 0.

BPOOL Specifies the number of buffers to be allocated in the buffer pool. This pool is used to buffer terminal I/O between the user program and the terminal itself. Each buffer is 516 bytes long. Allocate at least 4 buffers for each terminal control block (TCB). The number of TCBs is the number of terminals (including BTRMs) plus the number of extra TCBs allocated for multi-session (see the XSES parameter). The utility program BPOOL can be used during production to monitor the usage of the buffer pool. If the number used consistently approaches the number allocated this parameter should be increased.

MAXTRC Specifies the maximum amount of main storage MUSIC will use for its trace table. This number is expressed in K (K=1024) bytes. Recommended size is in the range 10 to 20.

MAXCOR Specifies the maximum amount of main storage that MUSIC will use. This number is expressed in K (K=1024) bytes and must be a multiple of 2. If the specified size exceeds that of main storage then this option will be ignored. It is recommended that the maximum value of 65536 be used so that MUSIC will use all the storage available to it as defined in its VM directory.

MAXRRS=n Maximum real region size in K. This must be 72 to 952. The default is 272. (*n*+8 should be a multiple of 40 otherwise it is rounded down.) You may need to increase the size of the swap data sets if a value greater than 272 is used, and a lot of jobs run with large regions. In the messages from the NUCGEN program, a value of -1 for MAXRRS indicates the default value.

LEVEL Specifies an 8-character identification to be printed when this nucleus is used. The current date will automatically be added to this identification. The MUSIC dump print program (PRDUMP) will also print the nucleus identification in MUSIC main storage dumps.

SIGNON Specifies a sign-on message (max 50 chars) to be displayed when a user connects to your production MUSIC system. You may use this field to show your company name at sign-on time. The system will add the words " SIGN ON." to the end of the message specified by this option, and add "*" at the front.

XSES=n Specifies the number of extra terminal control blocks (TCBs) to be allocated for multi-session support. If this parameter is omitted, the number of extra TCBs is 50 per cent of the number of terminals defined. If multi-session is heavily used, a larger number of extra TCBs may be needed, however the total number of TCBs, including terminals, BTRMs, and extras, will not exceed 999 regardless of the number specified. If you do not wish to use the multi-session feature you should specify XSES=0 and set the profile limit for extra sessions to zero for all users. In the messages from the NUCGEN program, a value of -1 for XSES indicates the default value. (The XSES parameter must be omitted in MUSIC/SP Release 1.0. It is valid in Releases 1.1 and higher.)

RAMDSK=n Specifies the amount of memory to be reserved for a RAM disk. It is expressed in kilobytes (1K = 1024 bytes). During system startup, files are loaded from disk into this area. The file $PGM:RAMDISK.LIST contains a list of the files that should be loaded. Typically they should be high usage, read only files such as command files, load modules, REXX procedures, macros, and panel definitions. Considerable saving in I/O overhead can be obtained by having files in RAM, since subsequent read access requires no I/O operations at all. If a file is changed it is automatically removed from RAM and the system reverts to using the DISK version of the file until the next time that RAM disk is loaded.

The amount of space you specify depends on what files you want to load into RAM. It must

be at least as large as the files plus a few K for an index. The RAMREP utility can be used to monitor RAM disk usage. This will help you adjust the file selection and space requirments to your own specific needs.

See also the RAMDLD utility.

ULMAPS=n This field specifies the maximum size (in number of K bytes) of the memory area used for caching the space bit maps of the library data sets (SYS1.MUSIC.ULm). Specify ULMAPS=1 to cache all bitmaps in memory. Specify 0 to use no bitmap caching. For best performance, 1 is recommended - unless your library is unusually large AND memory is scarce. ULMAPS=1 eliminates almost all bitmap reads from disk. For more information, run the MAPMEM and COUNTS utilities. If ULMAPS is not specified, ULMAPS=0 is assumed.

MPLLIM=n This parameter sets the limiting value for the maximum multi-programming level (the MAXMPL number reported by the SSTAT program). The standard and default value is MPLLIM=10. On some MUSIC/SP systems with large amounts of memory, specifying a value for MPLLIM greater than 10 may give higher performance. You may try a value between 10 and 30, for example. Note that a larger MAXMPL also results in a smaller time slice length. You can set the MPLLIM value via ADMIN 4 10 5.

Setting Up Device Statements

These statements are used to specify the addresses to be used by the production MUSIC nucleus. Each statement may contain 2 or 3 fields called *Devtype*, *Address*, *Options*. Devtype starts in column 1 and each field is separated from each other by blanks. Commas MUST be used to separate items in the options field from each other. An address range can be specified as shown in the following example:

```
TAPE 180-183 9TRK
```

A sample set of device statements is shown below. Lines starting with an asterisk (*) are comments and are ignored. The device statements can appear in any order.

```
* UNIT RECORD DEVICES:
2540 00C READER
2540 00D PUNCH
1403 00E PRINTER
1052 01F CONSOLE
* DISK DEVICES:
3350 148-14B
F512 260-261
3380 270-272
* TAPE DRIVES:
TAPE 180-183 9TRK
* PSEUDO TERMINALS THAT HAVE BACKGROUND JOBS STARTED AT IPL TIME
BTRM 005
BTRM 010-019
* TERMINAL DEVICES:
TTY 020-03F DIALUP,300
TTY 040-047 DIRECT,1200
3270 0C0-0CF DIRECT
* AUTOMATICALLY SIGNED ON TO BE USED FOR AUX PRINTERS
TTY 060 DIRECT,1200,SIGNON
3270 0F0 DIRECT,SIGNON
DEVEND
```

The following describes each device type in detail. Some device type names also apply to other devices of the same nature. For example, a 3211 printer is generated using the device type of 1403.

If you do not have a card punch device, omit the record "2540 xxx PUNCH", and similarly for the card reader. Disk and tape devices can have channel addresses from 1 to F. **All other devices must be on channel 0.** BTRM devices must also be on channel 0. The only exception is that 3270 terminals can be on channels 1 through 15 if no tapes or disks are also on that channel. (The channel address is the first digit of the 3-digit device address.) The total number of disk and tape devices must not exceed 64.

Specifying Unit Record Devices

2540 This device type is used to specify any supported card reader or punch. You must specify READER or PUNCH in the options field as appropriate. Only one reader and one punch may be defined. Note: the extra virtual punches, readers, and printers, used with the SUBMIT, AUTOPR, VMREAD, and VMPRINT, must not be defined here. Use an address of 000 if you do not have a reader or punch.

1403 This device type is used to specify any supported batch line printer. You must specify PRINTER in the options field. Only one printer can be defined.

1052 This device type is used to specify any supported console. Only one console may be defined. (This address can be automatically changed at IPL time. See *Chapter 3 - Loading the System* for details.)

Specifying Disk and Tape

2314 This device type applies to any device in the 2314 class such as 2314 or 2319.

3330 This device type applies to any device in the 3330 class such as 3330 or 3333. No distinction is required between Model 1, 2 or 11.

3340 This device type is used to specify a 3340 or 3344 type of disk. No distinction is required between Model 35, 70 and 70F. Define each logical 3344 volume separately as a 3340 device.

2305 This device type is used to specify a 2305 fixed-head disk device. The options field must specify MOD1 or MOD2 as appropriate. Since multiple requesting is not supported on MUSIC, the device address should end with a 0 or 8.

3350 This device type is used only when the 3350 DASD device is working in native mode. (Use the 3330 specification for this device if it is in 3330 compatibility mode.)

F512 FBA (fixed block architecture) DASD device, such as 3310, 3370, 9332, or 9335.

3375 3375 disk device.

3380 3380 disk device. Refer to the topic "3380 Model AA4 Addressing Notes" in Chapter 1 for cautions about 3380 device addressing.

3390 3390 disk device.

9345 9345 disk device.

TAPE This device type is used to specify any supported magnetic tape device. The options field must be 7TRK, 9TRK, or 8809 (the IBM 8809 tape drive), as appropriate. Not more than 4 tape

devices can be used in any one user job.

See the topic "Tape Drives on IBM 9371 Processor" in Chapter 1 for more information.

Special Specifications

TERM This specification is used to give the preparatory I/O command to the telecommunications control unit before a line is enabled. This specification is required if using an IBM 2702 communications control unit. It may be optionally used with the Integrated Communications Adapter (ICA) on the S/370 Model 125 processing unit. In all other cases, this specification is meaningless.

For 2702 units, specify the 2 character SAD command followed by 2 zeros in the address portion of this card. For example, use 1700 for a SAD command of 1. (The SAD identifications for you particular 2702 unit can be obtained from your local hardware support personnel.)

For the ICA on the Model 125, specify the 2 characters 2B followed by the 2 character *set line mode* options in the address field of this card. Refer to the *S/370 Model 125 Functional Characteristics* manual (GA33-1506) for the meanings of these options. Examples are 2BDC, 2B9C. The line mode may also be set at IMPL time. Refer to Chapter 1, topic "Transmission Control Units" at the beginning of this publication for details.

The options field of this card must be of the form CTLn, where n is a number from 1 to 9. For example, you can use CTL3. This CTLn option is used to relate to the terminal specification cards. When the TERM specification is required, then all affected terminal device cards must have the corresponding CTLn specified in their options field. An example is shown below:

```
TERM 2BDC CTL1
2741 030-3F CTL1
```

Up to 9 TERM cards may be specified defining the 9 possible CTLn options. The device re-configuration of MUSIC at IPL time allows the specification of one 4-character value that applies to all terminal types defined at that time. This 4-character value is the same as that which can be specified in the address portion of the TERM card just described.

Specifying Terminals

IBM This specification is used to denote dialup IBM terminals such as 2741, 3767, 1050 and CMCST devices. MUSIC will automatically handle any of these terminals on the same line. (If 3767 terminals are used at 300 baud, then those 300 baud lines cannot be used for the 2741, 1050 and CMCST devices.) Specify DIALUP in the options field. The line speed should be specified as 134. Lines defined as 300 baud for 3767 terminals should use the line speed option of 300.

3767
1050 These specifications must be used with directly connected IBM-type terminals. The options field must specify DIRECT. A numeric line speed option should also be given in the options field.

3270 This specifies any terminal in the 3270 product line or a TTY terminal connected through a protocol convertor that emulates a 3270 terminal. They must be locally attached to the processing unit or be specified as SPECIAL in the VM directory for MUSIC.

Caution: Make sure that the tens digit of the 3270 address is not used by any other type of

terminal. For example, if a 3270 device has address 0E2, make sure that no other non-3270 device has an address in the range 0E0 to 0EF.

TTY This option specifies any of the asynchronous ASCII (TTY) class of terminals. It is also used for the IBM 3101 terminal and personal computers connected to MUSIC via asynchronous lines which do not come through a protocol converter. (Ones that do come in through a protocol converter look like 3270 terminals to MUSIC and must be specified as such.) Auxiliary ASCII printers should also be designated as TTYs.

The option field may specify either DIRECT or DIALUP depending on the type of connection. The speed option should be used in order to ensure correct operation of these devices.

Normally the RETURN key on ASCII terminals is used as the end of line character. You can use the option RNA to specify that this RETURN key is not an active end of line character. When RNA is used, RETURN characters are ignored. The transmission control unit and MUSIC must both be set whether to accept the RETURN as end of line or not accept it. This specification can be done on a line by line basis.

BTRM No actual terminal will be attached to this address. This specification causes the system to create the appropriate control blocks, as if a terminal were there, and automatically start a background program when the system is IPLed. The system will attempt to automatically sign on the code $MONsss where the subcode, *sss*, is the device address of the BTRM. The background program started is the AUTOPROG specified in the profile for the code. This facility is used by programs such as AUTOSUB, AUTOPR, and VMREAD, which run as background tasks without a terminal. For further details see Chapter 22, topic "Defining BTRMs and Auto-Sign-on". BTRM device addresses must be on channel 0.

Note: The distributed system assumes there are BTRMs defined on addresses 005 and 010 through 019 for use by the SYSLOG, AUTOPR, VMREADX, RDMAILER and the ADMIN facility. If you do not have them defined, the programs will not function.

Terminal Options

The following options can be specified on the terminal device statements.

DIALUP

For a TTY device, DIALUP should be specified if the terminal is connected to the communications controller via a switched line. This usually means a modem connection that can be made and broken by the user. Virtual TTY ports defined in MUSIC's VM directory as SPECIAL should also have the DIALUP option specified. TTY devices used as auxiliary printers should always be defined as DIALUP regardless of the connection.

If the DIALUP option is specified for a 3270, MUSIC will issue a host reset command at sign off time, if the terminal is really a TTY device connected through a protocol convertor. (See 7171 option).

DIRECT

The DIRECT option should be specified for TTY devices that are connected directly to the communications controller, or connected in such a way that they appear directly connected to the controller.

DIRECT should be specified for real 3270 terminals and for emulated 3270 terminals when you do not wish a host reset to automatically be issued at sign off.

Speed Option

This speed option should be given on all terminal specifications except the 3270. It is used by the terminal handler to determine the number of idles to send to each terminal as well as other speed-dependent functions.

nnnn The allowable options correspond to the line speed in bits per sec (baud). They are: 110, 134, 300, 600, 1200, 1800, 2000, 2400.

AUTOSPEED The line speed is determined dynamically by the system at time of the initial line connection. This option requires that the transmission control unit have the speed detection feature. See the topic "Transmission Control Units" in Chapter 1 of this publication. The DIALUP option must also be specified for these lines.

SIGNON

The SIGNON option indicates that once the line has been successfully enabled, the terminal is to be automatically signed on and a program started. This option is usually specified for terminals which are to function as auxiliary printers using the AUTOPR program. The system will attempt to automatically sign on the code $MONsss where the subcode, *sss*, is the device address of the terminal. The background program started is the AUTOPROG specified in the profile for the code. For further details see Chapter 22, topic "Defining BTRMs and Auto-Sign-on".

7171

The 7171 option indicates that the terminal is actually an ASCII terminal connected through a 7171 protocol convertor or through something that is compatible with the 7171 such as the 9370 ASCII Subsystem.

When specified on a TTY device definition, it indicates that 7171 ASCII transparent mode should be used. In this mode, protocol conversion is bypassed and the terminal functions as a native ASCII device.

When specified on a 3270 device device definition, normal protocol conversion is done and the terminal functions as a 3270 device. However, the TOTTY and TO3270 can be used to switch between 3270 emulation mode and ASCII transparent mode. This allows PCWS file transfer and number of other MUSIC to PC connectivity features to work.

If MUSIC is running under VM this option need not be specified. When a terminal connects from VM, MUSIC determines if it is connected through a protocol convertor by asking VM. The VM module DMKRIO should specify "FEATURE=EMUL3270" on the RDEVICE statements for these terminals, so VM will provide the correct information.

Privileges Required: LSCAN and FILES.

PRDUMP: Storage Dump Print

This program is designed to print storage dumps taken by the MUSIC stand-alone storage dump program. The dump program resides on the MUSIC1 disk pack. It is run by performing an IPL from this pack. The dump program dumps all of main storage onto a preallocated data set on the MUSIC1 pack. When MUSIC is running again, the storage dump print routine is run from batch to print the dump. This program, in addition to printing the contents of main storage in standard dump format, formats many of the MUSIC internal

tables and control blocks to enable quick analysis of the dump.

This program can also be run from a terminal. In this case it allows for the interactive inspection of the dump.

The storage dump print program is run as follows:

```
/FILE 1 UDS($PGM$DMP) VOL(MUSIC1) SHR
/INCLUDE PRDUMP
.....parameters   (if batch)
```

The file statement for unit 1 should point to the installation dump data set.

The parameters that may be specified on batch are:

MUSIC=TRUE/FALSE	If TRUE, format MUSIC status information
TRACET=TRUE/FALSE	If TRUE, format trace table
SELCHQ=TRUE/FALSE	If TRUE, format selector channel queues
NT=nnn	Number of trace table entries to format. A value of zero indicates that all entries are to be formatted.
TCB=TRUE/FALSE	If TRUE, format TCBs

If more than one parameter is to be specified, they should be separated by commas (MUSIC FORTRAN NAMELIST format).

The default parameters are:

```
MUSIC=TRUE,TRACET=TRUE,SELCHQ=TRUE,NT=500,TCB=TRUE
```

The code under which this job is run must be able to access the dump data set.

Privileges Required: LSCAN and FILES.

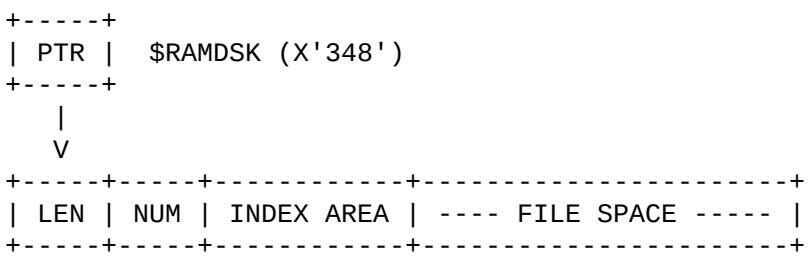
RAMDLD: Load MUSIC RAM Disk

To improve system performance it is possible to define an area of memory to be used as a RAM disk area. During system initialization files are brought from disk into this area. Subsequent access to these files requires no I/O operations. If the right files are chosen, a significant reduction can be made in the I/O overhead incurred in accessing high usage files. The RAM disk is read only. If a file in the RAM disk is changed the system automatically removes it from RAM and uses the modified version on disk until the next time RAM is loaded.

RAMDLD loads the RAM disk area from the Save Library. It is usually called by JOBONE to do this at system startup time, but it can also be run on its own to re-load the RAM disk while the system is running. VIP must be set ON to run this program. You can run the program by issuing the RAMDLD command. Note that reloading the RAM disk on a running system may cause a few users to experience temporary problems if they happen to be accessing an old RAM file at that instant.

The file $PGM:RAMDISK.LIST contains a list of files that are loaded by this program. If the program runs out of room in the RAM disk area it simply gives up. The size of this area is defined by the RAMDSK parameter in the NUCGEN. Note that once loaded this is a read-only RAM disk, so good candidates for this area are high access read only type files, like the execution files and load modules for common commands, programs, and utilities.

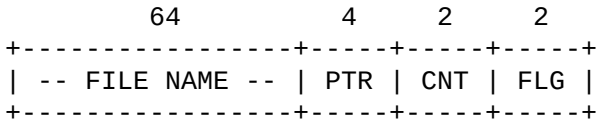
Location X'348' contains the pointer to the RAM disk area. If this is zero the program does nothing. The first word of the area contains the length of the entire area. This is used to calculate the number of index entries. The rest of the space is used for files. the format of the RAM disk area is as follows.



LEN Total length of area

NUM Number of index entries

Each index entry has the following format:



PTR address of first block of file

CNT usage count

FLG FLAGS X'80' - RAM copy is invalid (file changed)
X'40' - File is in common index

$PGM:RAMDISK.LIST

The format of the $PGM:RAMDISK.LIST file is simply a list of fully qualified file names (userid, subdirectory path, and name) that are to be loaded into the RAM DISK area in memory. There should be only one file name per line and it must start in column 1. Lines starting with a "*" are ignored and can be used as comments.

RAMREP: Report on RAM Disk Usage

This program displays a report on how RAM disk is being used. It is run by entering the RAMREP command. The name, location, flags, and usage count of each file in the RAM disk are displayed.

The usage count is reset to 0 when the RAM disk is loaded. This is usually done at IPL time. This is deffe-rent to usage count that can be maintained with the file and viewed by the ATTRIB command. The count seen by the ATTRIB command is updated by 1 when the file is loaded into the RAM disk and is not updated when it is used from the RAM disk area.

Example:

```
>ramrep
```

```
RAM Disk at     B359A0
Total RAM Space=   512K
Total Files    =   114


Location Flags   Count   Size     Name
-----------------------------------------------------
B391E8    4000    64     1.0 K    $REX:REXX
B395E8    4000     5     1.0 K    $REX:REX
B39DE8    4000     5     1.0 K    $PGM:ATTRIB
B3A1E8    4000     5    20.0 K    $PGM:ATTRIB.LMOD
B3F7E8    4000    11     1.0 K    $PGM:CD
B3FBE8    4000    11     2.5 K    $PGM:CD.LMOD
B469E8    4000    74     1.0 K    $PGM:EDITOR
B471E8    4000     8    12.5 K    $PGM:FT
B4C1E8    4000     5     1.0 K    $PGM:OUTPUT
B4C5E8    4000     5   155.0 K    $PGM:OUTPUT.LMOD
B731E8    4000     2     1.0 K    $PGM:PRINT
B735E8    4000     2     7.0 K    $PGM:PRINT.LMOD
B751E8    4000     1     1.0 K    $PGM:PRTSCR
B755E8    4000     5    11.0 K    $PGM:PURGE
B781E8    4000     0     1.0 K    $PGM:RD
B785E8    4000     0     4.5 K    $PGM:RD.LMOD
B797E8    4000     1     1.0 K    $PGM:SUBMIT
B79BE8    4000     2     1.0 K    $PGM:TELL
B79FE8    4000     2     4.0 K    $PGM:TELL.LMOD
B7AFE8    4000    81     1.0 K    $PGM:VIEW
B7B3E8    4000     0     1.0 K    $PGM:VM
B7B7E8    4000     0    31.0 K    $PGM:VM.LMOD
B8B9E8    4000     2     1.0 K    $EML:GETMAIL
B8C1E8    4000     8     1.0 K    $EML:MAIL
B9CDE8    4000    31     1.0 K    $EML:VIEW.MAIL.COMI
B9D5E8    4000    17     1.0 K    $EMD:MAIL.AUTHOR
BA37E8    4000    69     1.0 K    $EDT:EDITOR.SHOWTEXT
BADDE8    4000     1    14.0 K    $INT:FLIB
BB15E8    4000     3    22.0 K    $INT:FSI

Total Space=   330.0K ,Total Usage Count=     457   ( 10.90% of opens)
```

RATE: Check System Load

The RATE program samples I/O, SVC, Paging and Swapping rates every ten seconds and displays the results on the screen. These numbers can be used as a good indicator of the instantaneous system load.

The COUNT field is the count since the last time the system was loaded. The RATE/SEC field is the rate averaged over the past 10 seconds. The RESPONSE TIME is not an average. It is the time it took for the system to wakeup the RATE program itself. The counters maintained by RATE are as follows.

I/O	- I/O interrupts
SVC	- SVC interrupts
PAGE WR	- Pages written to disk by demand paging
PAGE RD	- Pages read by demand paging
PLPA RD	- Page read from the PLPA

SWAP RD - Swap read operations
SWAP WR - Swap write operations

```
>rate

      SYSTEM RATES

  MON JAN 22, 1990     12.24.01
  SYSTEM LOADED AT      7.59.38

               COUNT  RATE/SEC
I/O           465372     7.000
SVC          4404351    38.000
PAGE WR        15360     1.200
PAGE RD        13817     1.100
PLPA RD        31662     2.100
SWAP RD         1012     0.200
SWAP WR         1564     0.210

RESPONSE TIME:  0.13 SEC
```

ROUTETABLE: Display Routing Table

This program displays the contents of the routing table (member $ROUTING). See the topic "Setting up the Routing Table" in *Chapter 8 - Job Submission* for information about the $ROUTING load library module.

Return Codes:

0 Normal End
1 Routing table not accessible

Privileges Required: none.

RTM: Response Time Monitor

The response time monitor (RTM) program is designed to run continuously when MUSIC is operating. It periodically samples the response times for the various MUSIC queues, number of active users, and VM overhead numbers and writes its output to a Save Library file. One file is created for each day. If the system is reloaded on the same day, the information is appended to the file previously used with an information line noting that the system was reloaded. At midnight, a new file is created for the new day. The information in the output records is described below.

Automatic Program Initiation Facility (BTRM)

MUSIC contains a facility for automatically initiating programs when the system is loaded (IPLed). These programs run as user programs with the exception that no physical terminal is associated with them. This facility is used to run the RTM program.

The "phantom" terminal is defined in the MUSIC nucleus generation (NUCGEN) data as a BTRM. The

program to be run ($PGM:RTM) is defined as an AUTOPROG for a subcode of the userid $MON. The subcode is the 3-character hexadecimal device address of the BTRM terminal. $PGM:RTM is automatically executed, and remains running, whenever MUSIC is loaded (except when the NOTERM option is used at IPL).

To cancel the RTM program, enter the console command "/HALT n", where n is the TCB number of the BTRM. The TCB number can be found by entering "WHOSON $MON" or "ENQTAB RTM" at a terminal, or "/FIND $MON" as a MUSIC operator console command. To restart the program, enter the console command "/RESET n".

For a general discussion of BTRMs, refer to the section "Defining BTRMs and Auto-Sign-on" in Chapter 22 - System Programming.

The files for the RTM program are $PGM:RTM*.

You can adjust the recording period by changing the value of the PERIOD parameter in file $PGM:RTM. The distributed value is PERIOD=10, meaning that a line of data is written to the recording file every 10 minutes.

How to Install the Response Time Monitor

1. The default period for RTM is 10 minutes (PERIOD=10). This means that the program wakes up every 10 minutes and writes a statistics record to the output file. If you wish to use a different period, edit file $PGM:RTM and change the number of minutes in the last line.

   Other options in $PGM:RTM include NEWNAM=t and KEEP=n. NEWNAM=T uses file name $MON:RTM.yyyymmdd for the daily log file, instead of $MON:Ryyddd. The KEEP=n option automatically deletes old RTM log files which are more than n days old. The default is KEEP=0, which means that no log files are deleted. KEEP=n is effective only if NEWNAM=T is also used.

2. Choose a device address xxx for the BTRM's phantom terminal. Note that this address does **not** correspond to a real terminal or to a virtual device defined in the VM directory for MUSIC. xxx is a 3-digit hexadecimal value. The first digit should be 0. Choose a device address that is not already defined in the MUSIC NUCGEN as a terminal, unit record device, or another BTRM.

3. Create a user id (code) for $MONxxx by running the CODUPD utility. The subcode xxx is the BTRM device address chosen above. The special password *NOLOGON prevents sign-on except as a BTRM, for security reasons; you may assign a different password if you wish. Enter the following command to CODUPD to create the id:

   ```
   ADD $MON SC(xxx) AUTOPROG($PGM:RTM) PW(*NOLOGON) -
   FILES LSCAN CREAD BATCH(0) -
   PRIME(NOLIMIT) NONPRIME(NOLIMIT) DEFTIME(NOLIMIT)
   ```

4. Define the BTRM to MUSIC, by adding the record

   ```
   BTRM xxx
   ```

   to the device statements for the nucleus generation (NUCGEN) utility. Usually the control statements for NUCGEN are stored in the file $GEN:NUCGEN.JOB. Run the NUCGEN utility and IPL from the tape it creates, in order to apply the new nucleus. Refer to the description of NUCGEN earlier in this chapter. You can also apply the nucleus by ADMIN 4 10 5 ("NUCGEN: update I/O config. & nucleus"). The two methods are equivalent.

RTM should start running when you next IPL MUSIC, with data records accumulating in file

$MON:Ryyddd, where yy is the year and ddd is the day of the year. If the RTM option of NEWNAM=T is included in $PGM:RTM then $MON:RTM.yyyymmdd is used.

Response Time Monitor - Output Records

The Response Time Monitor program writes records to the file $MON:Ryyddd (where yy is the year and ddd is the day of the year or $MON:RTM.yyyymmdd if NEWNAM=T). Each record represents a sampling period, normally 10 minutes. The fields in the 80-byte record are as follows:

Cols. 1-8 Time of day (HH.MM.SS) at the end of the period sampled.

Cols. 9-12 Number of users signed on at the end of the period sampled.

Cols. 13-18 Number of requests of type 1 (see below) during the period.

Cols. 19-25 Average response time, in seconds, for type 1 requests during the period.

Cols. 26-31 Number of requests of type 2 during the period.

Cols. 32-38 Average response time, in seconds, for type 2 requests during the period.

Cols. 39-44 Number of requests of type 3 during the period.

Cols. 45-51 Average response time, in seconds, for type 3 requests during the period.

Cols. 52-57 Number of requests of type 4 during the period.

Cols. 58-64 Average response time, in seconds, for type 4 requests during the period.

Cols. 65-71 The virtual CPU time used by MUSIC during the period, expressed as a percent of the length of the period. For example, if the period is 5 minutes and the reported percent is 40.00, then the virtual CPU time used by MUSIC was 0.40 * 300 = 120 seconds. The percent is the portion of the entire real machine used by the MUSIC virtual machine (not counting VM overhead). The field is zero if MUSIC is not running under VM.

Cols. 72-78 The total CPU time (virtual plus VM overhead) used by MUSIC during the period, expressed as a percent of the length of the period. The percent is the portion of the entire real machine used by the MUSIC virtual machine, including CPU overhead for privileged operations, CCW translation, etc. done by VM for MUSIC. The field is zero if MUSIC is not running under VM.

The percent VM overhead for MUSIC can be found by subtracting cols.65-71 from cols.72-78.

Cols. 79-85 The amount of time that the MUSIC user region was busy during the period, expressed as a percent of the period. This value is obtained from the "idle time" reported by the WAITS utility.

The types of requests are as follows:

Type 1 Request for the first time slice of a job (/RUN, /EXEC).

Type 2 Request for /ID, /CANCEL, /DISPLAY, /SAVE, /PURGE, etc.

Type 3 Request for a time slice when the job's immediately preceding time slice ended because of a

conversational read (or a call to the subroutine DELAY). This includes FSIO reads, such as done by the Editor in full-screen mode. It does not include spooled conversational reads, such as done by the Editor in INPUT mode on an ASCII terminal.

Type 4 Request for a time slice when the job's immediately preceding time slice went to completion. This is the case for non-conversational jobs, CPU-bound jobs, and batch jobs.

SETBFN: Set File Backup Numbers

This program resets the master backup number to 1 and sets the backup numbers in all files to values in the range 255, 254, ..., 255-n+1 (where n is the period specified below). At the same time, all files are marked as being backed up.

This program would be run if the incremental archive utility (MFARCH) could not fit all the new or modified files onto a single tape, and so could not terminate normally. This would be the case that if you have run MUSIC for some time without running the incremental backup or when you have made major changes to your system that caused an unusually large number of files to appear changed.

To run the utility, use the following statements:

```
/INCLUDE SETFBN
PERIOD=n,MBNFIL='filename'
```

If no parameters are needed, a blank line must be used after the /include.

PERIOD=n specifies the archive period number that will be used with MFARCH. See the description of MFARCH. Default is 12. Users of the ADMIN automatic maintenance facility should specify n as 10 as that is the period used there.

MBNFIL='filename'
specifies the name of the file containing the master backup number. Default is MBNFIL='$PGM:MASTER.BACKUP.NUM'. It is most unlikely that you need to specify this option.

Privileges Required: LSCAN, FILES, MAINT

SETBUF: Load Printer Buffer

This program can be used to load the buffers on the (real) system printer, if this has not already been done by VM. Normally it is not necessary to run SETBUF, since VM will do this function. The UCS or the FCB buffer may be loaded. The program must be run from batch.

Only one buffer may be loaded with each running of this program. If both an FCB and a TRAIN name are specified an error message will be displayed for the user. In this case neither buffer will be affected.

The system will issue the M308 console message when the UCS and/or FCB are loaded. The operator must respond with a "/REPLY 0" before the output will continue.

Two FCB buffers are supplied with MUSIC:

```
FCB Name    Description

 MUS6       SIX LINE TO THE INCH
 MUS8       EIGHT LINES TO THE INCH
```

UCS buffers images available are:

```
3203/1403  PRINTERS

AN    NORMAL AN ARRANGEMENT
HN    NORMAL HN ARRANGEMENT
PCAN  PREFERRED SET, AN
PCHN  PREFERRED SET, HN
QN    PL/I - 60 GRAPHICS
QNC   PL/I - 60 GRAPHICS
RN    FORTRAN, COBOL COMMERCIAL
YN    HIGH SPEED ALPHAMERIC
TN    TEXT PRINTING - 120 GRAPHICS
PN    PL/I PRINTING 60 GRAPHICS
SN    TEXT PRINTING - 84 GRAPHICS

3211 PRINTERS

A11   STANDARD COMMERCIAL
H11   STANDARD SCIENTIFIC
G11   ASCII
P11   PL/I
T11   TEXT PRINTING

3289 PRINTERS

F48   48 CHARACTER GRAPHICS
F64   64 CHARACTER GRAPHICS
F96   96 CHARACTER GRAPHICS
F127  127 CHARACTER GRAPHICS

3262 PRINTER

P48   48 CHARACTER EBCDIC
P52   52 CHARACTER AUSTRIA/GERMANY
P64   64/72 CHARACTER EBCDIC
P63   PREFERRED 64 CHARACTER EBCDIC
P96   96 CHARACTER EBCDIC
P116  116 CHARACTER FRENCH CANADIAN
P128  128 CHARACTER KATAKANA
```

A parm statement is read from unit 5 specifying which UCS or FCB are to be loaded. Also the option of FOLD may be specified. When fold is specified, it applies to either the UCS or FCB load.

The format of the parm statement is:

```
TRAIN='XXXX',FOLD=Y    ...  or  ...
FCB='YYYY',FOLD=Y
```

Where XXXX is the name of the print train (UCS) to be loaded. YYYY is the name of the FCB image to be loaded.

FOLD is either Y or not specified, the other options have no defaults and must be specified.

Example:

```
/INC SETBUF
TRAIN='PN',FOLD=Y
```

Privileges Required: LSCAN and MAINT.

SSTAT: System Status

This utility displays a screen that shows the system status. The screen is updated approximately every 10 seconds, or when a function or action key is pressed. Press F3 to exit this utility. This utility can be helpful in determining if there is a performance problem with your system.

```
------------------------------ System Status --------------- 09.24.03

Storage Size: 8192K  TCBs:   80  RCBS:  82  MAXMPL:  8  MAXRRS:  592K

Users:  18 Sessions:  23  Response Time: 0.02 (sec) CPU Usage: 49.68%

Running:   17      In Core:   16   Swapped:    1

 Active:    2       Queued:    0      Idle:   15

Total Pages: 5248K   Free Pages: 1052K


Activity       Count  Rate/Sec
--------   --------- ---------
I/O           747360   22.64
SVC          6841682  178.00
PAGE WR          406    0.00
PAGE RD         2884    0.00
PLPA RD        24639    0.00
SWAP WR            2    0.00
SWAP RD            3    0.00
```

Figure 17.6 - Sample Screen of the System Status program

Usage:

This program is available from the ADMIN facility option 1 13. You can also run this program by entering SSTAT when in *Go mode.

Privileges Required: CREAD

SUBLIB.GEN: Subroutine Library Creation

The MUSIC/SP system subroutine library consists of subprograms that are automatically available to programs written in Fortran, Assembler, PL/I, and other languages. The subroutine library is used by the MUSIC/SP loaders and the Linkage Editor. It is not used by /LOAD EXEC or /LOAD XMON.

The system subroutine library is contained in Save Library files with names of the form $SUB:SUBLIB.xxx. These files are created by the SUBLIB.GEN utility, which reads object modules as input and builds members and an index in the $SUB:SUBLIB.xxx file. Refer to the description of SUBLIB.GEN in the utilities chapter of this publication.

The standard subroutine library files are:

1. $SUB:SUBLIB.FG1 contains the library routines for the Fortran G1 compiler. The subprograms in this file are not available to OS-mode programs.

2. $SUB:SUBLIB.MUS contains the MUSIC-supplied system subroutines, plus other miscellaneous routines. The subprograms in this file are available to both non-OS-mode and OS-mode programs.

3. $SUB:SUBLIB.OS contains the library routines for VS Assembler, VS Fortran, PL/I, Cobol, and other OS-mode languages.

4. $SUB:SUBLIB.EXT is an optional library that could be used to contain subroutine packages installed locally at your site. For example, the large IMSL package of mathematical subroutines could be stored in this library. (Local subroutines could be added to $SUB:SUBLIB.MUS instead.) In the standard MUSIC/SP system, this file does not exist.

Each of the files $SUB:CREATE.xxx contains the control statements for the SUBLIB.GEN utility to create the corresponding subroutine library file. SUBLIB.GEN copies object modules, defined by /INCLUDE statements in the input, into the subroutine library file and builds an index of subprogram and entry point names.

Output, including informative messages, a list of the input modules, and an index listing sorted by name, is normally written to the file $SUB:LISTING.xxx, which you can keep for future reference. To have SUBLIB.GEN write messages and listings to the terminal (or to the printer if run on batch), use /FILE 6 PRT.

To add subprograms to the library, add appropriate /INCLUDE statements for the object modules to either $SUB:CREATE.MUS or $SUB:CREATE.OS and execute the file. This completely recreates the corresponding subroutine library file ($SUB:SUBLIB.MUS or $SUB:SUBLIB.OS). The previous contents of the file are deleted. You should normally add /INCLUDE statements at the end of the file. Routines added to $SUB:CREATE.OS are available only to programs running in MUSIC's OS simulation mode.

In order to replace a subroutine library file ($SUB:SUBLIB.xxx), there must be no user jobs running that access it. This usually means that no user jobs or BTRM jobs (userid $MON) should be running. To cancel all $MON jobs, enter the command "/FIND $MON" on the operator console. This displays the TCB numbers. Then enter "/HALT n" for each TCB number.

The general control statements for SUBLIB.GEN are:

```
/FILE 1 N(sublibfile) NEW(REPL) SPACE(nnnn)
/INCLUDE *COM:SUBLIB.GEN
INDEX=mmm
/INCLUDE objectfile
/INCLUDE objectfile
etc.
```

INDEX=mmm defines the number of 512-byte blocks to be reserved for the index at the beginning of the subroutine library file. Each index block can hold up to 51 routine or entry point names.

The following optional control statements may appear in the input, after the INDEX=mmm statement:

Comments Records with * in column 1 are ignored.

./ ADD NAME=xxxxxxxx
: This indicates the start of a member which may consist of more than 1 object module. The end of the member is indicated by the next ./ ENDUP or ./ ADD statement. Only the specified member name *xxxxxxxx* and any names given on ./ ALIAS statements are added to the index. If a ./ ADD statement is not in effect for an input object module, the csect and entry point names in the object module are added to the index and the output member contains only that object module. Use of ./ ADD allows grouping of several object modules into one member.

./ ALIAS NAME=xxxxxxxx
: This specifies an additional entry point name (alias) to be added to the index for the current member. Any number of ./ ALIAS statements can be used. They can appear anywhere after the ./ ADD statement and before the next ./ ADD or ./ ENDUP statement.

./ ENDUP Indicates the end of a member begun by a ./ ADD statement. A ./ ENDUP is also implied by the next ./ ADD.

/NOEP This statement indicates that the entry point names in the immediately following object module only are not to be added to the index.

Subroutine Library Considerations

In each job that requires the subroutine library, the subroutine library files to be used are defined by a /FILE statement with ddname SUBLIB. The Linkage Editor with /JOB MODE=OS also requires a /FILE statement with ddname SUBLIBOS. Each *xxx in the PDS parameter of these /FILE statements refers to the subroutine library file $SUB:SUBLIB.xxx. The files are searched in the order given. If a file does not exist or is not accessible, it is skipped. The following /FILE statements are automatically supplied by the /LOAD statement processor (module CTL):

For /LOAD FORTG1, /LOAD LOADER, and /LOAD LKED:

```
/FILE SUBLIB PDS(*FG1,*MUS,*EXT) DEF
```

For /LOAD LKED for /JOB MODE=OS:

```
/FILE SUBLIBOS PDS(*MUS,*OS,*EXT) DEF
```

For /LOAD ASM, /LOAD ASMLG and other OS-mode processors that use the loader:

```
/FILE SUBLIB PDS(*MUS,*OS,*EXT) DEF
```

Since these are generated with the DEF parameter, a job can use an overriding /FILE statement that may specify different libraries or a different search order.

For special applications, it is also possible to create a private subroutine library, by means of the SUBLIB.GEN utility. Then use of

```
/FILE USERLIB NAME(filename)
```

would cause the private library to be searched ahead of the standard library, for the current job only. Note that for ddname USERLIB, the NAME parameter is used rather than PDS. Here, *filename* is the name of the private subroutine library file, created as /FILE 1 by the SUBLIB.GEN utility.

Privileges Required: LSCAN and FILES.

SYSDATE: Display Nucleus Level and IPL Date/Time

SYSDATE displays the current date and time, the date and time that MUSIC was loaded (IPL), and the nucleus level and date. The nucleus level is the value specified by the LEVEL parameter of the NUCGEN utility. The nucleus date is the date NUCGEN was run.

Privileges Required: None.

SYSDMP: Relative Block Disk Utility

The SYSDMP utility can be used to inspect or alter the contents of a SDS or UDS data set by relative block number. (Refer to the DSKDMP utility if you want to do similar operations by absolute disk location.)

The data set is pointed to by either (a) DEB number or (b) data set name and volume name.

The parameters are as follows. Separate them by commas.

DEB=n Specifies DEB number.

DSN='dsname' Specifies the data set name.

VOL='volume' Specifies the disk volume name.

BLK=n or BLK=n,m or BLK=n,m,i
n is the starting block number (the first block of a data set is numbered 1). Default is n=1. *m* is the ending block number to be dumped. Default is *m=n*. *i* is the block number increment. Default is i=1. For example, BLK=3,10,2 dumps blocks 3, 5, 7, and 9.

BLKSIZ=n Length of each block (used in read and write). Default is DEB or data set blksize.

ADDR=n or DISPL=n
Displacement for start of dump display. Default is 0.

LEN=n or DMPLEN=n
Number of bytes to be dumped. Default is 32.

REP=n	Displacement for start of REP (bytes to be modified).
WRITE	Causes block to be written back to disk.
COPYTO=n	*n* is the unit number of a sequential file (default 0, i.e. no output). Each requested block is written as a record to the sequential file. This option is turned off by specifying COPYTO=0. For example, BLK=1,999,COPYTO=1 can be used to copy the records of a MUSIC catalog data set to a file.
HELP	Displays information on how to use the program.
END	Terminates the program.

Notes:

1. All parameter values are remembered between parameter reads, except REP, WRITE, and HELP. Also, the following are reset when a new DEB or data set is started: BLK=1,ADDR=0,DMPLEN=32.

2. A block is read only when a new data set is started, or when BLK or BLKSIZ is specified, or if the previous request was for more than one block.

3. Each of the parameters REP, WRITE, HELP, END must be used alone. They may not be used in combination with other parameters.

4. Replacement text (in hex) is read immediately after the REP specification. Commas may be used to separate bytes or groups of bytes. The text ends at the first blank.

5. Abbreviations: VOL: V, ADDR: A, DISPL: D, LEN: L, DMPLEN: DL.

6. A system data set may be specified as DSN='%XXXXXXX'.

Example:

```
DEB=226,BLK=15,DISPL=Z2A0,DMPLEN=64

V='MUSICX',DSN='%SCRATCH',BLK=318,L=512
```

Privileges Required: LSCAN, CREAD, DREAD, VIP to change disk.

SYSGEN1: Write New System Nucleus to Disk

This utility program reads the nucleus data created by the NUCGEN utility and writes a new system nucleus to the current MUSIC/SP IPL volume. An IPL record is also written to the volume. The current nucleus on disk is replaced. The new nucleus will take effect at the next IPL of MUSIC/SP.

As a method of applying a new nucleus, SYSGEN1 is an alternative to IPLing from a tape or VM spool file created by the NUCGEN utility. SYSGEN1 is fast and easy, but has the disadvantage that no backup copy (on tape or as a VM spool file) is produced. The IPLable backup is useful for restoring a working nucleus if a future nucleus fails.

Usage:

Run the following job after typing the command "vip on":

```
/FILE 1 N(filename)
/INCLUDE *COM:SYSGEN1
```

where filename is the output file created by the NUCGEN utility, (normally defined as /FILE 2 in $GEN:NUCGEN.JOB).

Return codes:

0 System nucleus has been successfully written to disk.

1 An error occurred; the nucleus was not successfully written to disk.

Privileges Required: VIP, LSCAN, FILES, DREAD, CREAD

SYSREP: Load Library Patching

The SYSREP program applies *reps* (patches or fixes) to load library members. It replaces specified existing bytes by new bytes, at a specified displacement within a member. The VER (verify) statement gives the existing data and the REP (replace) statement gives the replacement data. Use the following control statements:

```
/FILE . . .    (if UDS load library)
/INCLUDE SYSREP
LIBE=n,INPUT=n,TEST=T/F
NAME membername
BASE baddr
VER addr hexdata
REP addr hexdata
 . . .
 . . .
```

Notes:

1. LIBE=n specifies the unit number or DEB number of load library. If LIBE='SYST' is specified, the system library will be assumed. LIBE can be abbreviated as LIB.

2. INPUT=n specifies the unit number for the input data. The default input unit number is 5.

3. TEST=T specifies that changes will not actually be done to the load library member. The default is TEST=F.

4. Reps may be applied to more than one member per job by using a NAME statement to specify the next member.

5. If a verify fails, no further changes will be made until the next NAME statement.

6. If the base address *baddr* is not specified, it defaults to zero. If specified, it causes the verify (VER) and/or replacement (REP) to apply at displacement *addr-baddr* from the start of the member. The base

address is reset to zero each time a NAME statement is processed.

7. *hexdata* may contain commas, but must not contain embedded blanks.

8. Statements with * in column 1 are considered to be comments and are ignored.

Privileges Required: LSCAN, DREAD, CREAD, VIP or SYSMAINT.

SYSUPDATE: Update Load Library Directory in Main Storage

This program performs one of two functions, which are referred to by the keywords LDDIR and NONRES.

The LDDIR function of SYSUPDATE compares the Load Library directory entries in main storage with those on disk and updates any entries in main storage which are different. Differences could be due to updates (by the LDLIBE utility) to the Load Library since MUSIC was last IPLed, which result in new starting block number, length, etc. for some members. SYSUPDATE causes the changes to take effect immediately, rather than after the next IPL as would otherwise be the case.

However, only existing entries in main storage are updated. No new entries are created. Also, the entry is not updated if the member is in the Link Pack Area (FLPA or PLPA).

The NONRES function of SYSUPDATE temporarily removes member names from the list of members in the Link Pack Area (FLPA and PLPA). This prevents the use of an LPA copy of a member. The change remains in effect only until the next IPL of MUSIC.

To run the program, type SYSUPDATE, then enter LDDIR or NONRES when prompted. The command /VIP ON must be entered before running SYSUPDATE.

Privileges Required: LSCAN, CREAD, DREAD, and VIP.

TRANS$: Transferring Funds Between User Codes on MUSIC

The TRANS$ program can be used by a course or project supervisor to transfer computing funds from one MUSIC user code to another. It can also be used to change their passwords. This allows authorized persons to distribute allocated funds among the users that they supervise. This authorization is given via the SUPV code privilege. The code that the funds are transferred from, and the code receiving the funds must both start with the same two characters as the code with the SUPV privilege. For example, the authorized code XY00 could use TRANS$ to move funds from XY15 to XY25.

A log file of all TRANS$ change requests can be kept. The installation can set up this log file by specifying its name in the file $PGM:TRANS$.

Each user code on the MUSIC system has two dollar values associated with it: a current amount and a limiting amount. The current amount starts at zero and is increased each night by the charges for that day by the NOWDOL system utility. When the current amount reaches the limiting amount, the code may no longer be used until the limit is increased by adding more funds. This can be done by the MUSIC System Administrator, or by using the TRANS$ program to transfer funds from another code.

A user can use the SHOW$ command of PROFILE to display the current and limiting amounts. (This program is described in the *MUSIC/SP User's Reference Guide*.) The money remaining is the limit minus

the current amount.

To run the TRANS$ program, type TRANS$ at the terminal. You will then be prompted to enter the request, which can be any of the following commands:

TRANSFER $nnn FROM xxxx TO yyyy

ALLOCATE $nnn FROM xxxx TO yyyy

GET xxxx

CHANGE cccc PW=pppppppp

HELP

END

On the TRANSFER command, *$nnn* is the number of dollars (without cents) to be transferred from code *xxxx* to code *yyyy*. The $ sign may be omitted. Each code may be specified as a 4-character code (e.g. xy15) or as a 7-character code and subcode (e.g. xy17001). The command name TRANSFER may be abbreviated TR, TRA, TRAN, etc. If "FROM xxxx" is omitted, the transfer is from the user code running the TRANS$ program.

The ALLOCATE command is handled like the TRANSFER command, except that only enough is transferred to make the unused amount in the receiving code the specified number of dollars (nnn). Normally a range or list of codes would be specified. This does a *top up* on each of the codes. Abbreviations AL, ALL, ALLOC may be used.

The GET command displays the dollar values for the specified code.

The CHANGE command is used to assign a new 1- to 8-character password *pppppppp* to the specified code *cccc*.

The HELP command displays information on how to use the TRANS$ program.

The END command stops the program.

A numeric or alphanumeric range of codes can be specified on the GET and for the *to* code on the TRANSFER and ALLOCATE commands. A list of codes, within parentheses, can also be specified. Type the HELP command for further information.

Examples of TRANS$ commands:

```
TRANSFER $100 FROM XY32 TO XY75

TR 150 FROM UV25001 TO UV17003

GET XY79

GET UV15999
```

Sample TRANS$ session (on user code XR00):

```
trans$
*In Progress

-- MUSIC FUNDS TRANSFER --

ENTER YOUR REQUEST, OR HELP
?
get xr51
  XR51      $LIMIT:   500     USED:   491     LEFT:     9

ENTER REQUEST, OR END
?
transfer $250 from xr73 to xr51
$250 TRANSFERRED FROM XR73 TO XR51

ENTER REQUEST, OR END
?
get xr51
  XR51      $LIMIT:   750     USED:   491     LEFT:   259

ENTER REQUEST, OR END
?
end
*End
*Go
```

Privileges Required: SUPV.

UCB: Display Unit Control Blocks

This program displays Unit Control Blocks (UCB). Each UCB represents a disk or tape device on the system. For each UCB the list includes: main storage address of the UCB, device address, type, volume name, and the number of errors that have occurred since the last IPL of MUSIC.

To run the program, type UCB in *Go mode.

Privileges Required: LSCAN and CREAD.

UCR: User Control Record Utility Program

In order to create a file, a user must have a User Control Record (UCR) associated with the user's sign-on code. This record contains five values:

1. 1. Limit of the total space for code (TOTLIM)

2. Maximum size of a file the code can allocate (FILLIM)

3. The current total space of the code

4. High water mark to date

5. Reserved

The above values are in units of K (1024) bytes. A value of -1 means *no limit*.

This program is used to maintain the UCR records. It can be invoked by entering UCR and will accept the following requests conversationally:

GET	Display the UCR values for a code
SET	Set values 1 and 2 for a code. A new UCR will be created if one does not exist. Optional parameters TOTLIM=n and FILLIM=n may be used to change the limits individually.
REP	Replace a UCR for a code
DEL	Delete a UCR for a code
HELP	Ask for information about the UCR program
END	Terminate the UCR program

UCR values are specified by the UCR parameter, as shown in the examples below. The CODE parameter specifies a single userid, or a numeric range of userids. Do not include the subcode (if any) when specifying a userid to UCR. For example, CODE='XX01-25' applies the same request to each of the userids XX01, XX02, ..., XX25.

The Code Update (CODUPD) utility automatically creates a UCR record, if necessary, when a userid is allocated, and deletes the UCR when a userid is deleted.

Examples:

```
GET,CODE='ABCD'
SET,CODE='XY11-20',TOTLIM=500
SET,CODE='X123',UCR=5000,300
REP,CODE='DC77',UCR=10000,-1,327,327,0
DEL,CODE='DC88'
```

Privileges Required: LSCAN, FILES, CODES, or MAINT.

UCRFIX: Fixup UCR Records Based on Code Table

This program scans the code table and for each user code encountered, it adjusts the UCR (User Control Record) limits for that code, or creates a UCR record if there is none.

This program would be run if the Save Library was destroyed and it was necessary to restore files from incremental MFARCH archive tapes. UCRFIX should be run before the restores, to ensure that restores will not fail because of missing UCRs. Specify FACTOR=0 to do this. If you also want to increase existing UCR limits, to avoid restores that fail because the UCR total limit is reached, specify a value such as FACTOR=1.5. An up-to-date code table dump should be restored, if available, before running UCRFIX and before restoring files.

There are two UCR limits: (a) the maximum size of each file, in units of 1K = 1024 bytes, and (b) the maximum total amount of space occupied by the code, also in units of 1K. Limit (a) is set to the value -1 (i.e. no limit) by the program when it creates a UCR. Limit (a) for a UCR that already exists is not changed. Limit (b) is set according to the program parameters, as described below.

Usage:

```
/INCLUDE UCRFIX
TOTAL=n,FACTOR=x,MARGIN=k
```

TOTAL=n specifies the value for limit (b) when a new UCR is created. Default is TOTAL=10000.

FACTOR=x controls how limit (b) is to be changed for existing UCRs. The number *x* can be a floating point value such as 1.5. If FACTOR=0, existing UCRs are not changed at all. Otherwise limit (b) is set to min(x*c,c+k), where *c* is the total space currently occupied by the code's files and *k* is the MARGIN parameter. However, limit (b) is never decreased. The defaults are FACTOR=0 and MARGIN=500.

Privileges Required: LSCAN, FILES, CODES or MAINT.

UDSRST: User Data Set Restore

An alternate version of the restore program exists which can restore only User Data Sets (not System Data Sets). It does not require any privileges to be run and may run under the user's code who owns the file restored.

```
/FILE n TAPE ...    dump tape
/FILE m UDS(receiving data set) VOL(volume) OLD
/INCLUDE UDSRST
IN=n,OUT=m,VOL='from vol',DSN='old dsn'
```

The file statements point to the dump tape and the receiving data set respectively. Normally *n* is set to 1 and *m* to 2.

The other input parameters give the data set name and its original volume at the time it was dumped. This is used to locate it on the tape.

Contents of Information Records

In a dump produced by UDSARC, each data set is preceded by a *DS1 record and a *DS2 record. The *DS1 record has the volume name, the data set name, and the date and time of the dump. The *DS2 record contains the following information:

Column 8	B for backup, N for no backup.
9-12	Device type.
13-15	Data set organization.
16-19	Record format.
20-25	Block size.
26-31	Logical record length.
32-37	No. of tracks (no. of physical blocks if FBA device).
38-40	No. of extents.
41-44	No. of blocks per track (32 if FBA device).
45-48	Key length.

49-53	FBA physical block length (0 if not FBA device).

Privileges Required: None.

ULINIT: Initialize Save Library Data Sets

This program initializes save library data sets. Either an index or data space can be initialized. See the topic "Adding Space to the Save Library" in *Chapter 6 - System Reconfiguration* for more information.

VMSUBM: Submission to Other Systems

VMSUBM allows a MUSIC system running under VM to transmit data (i.e. jobs) to any other system on that machine. This is accomplished by creating a spooled punch file and then spooling it, via the VM SPOOL command, to a reader of the appropriate class defined on the target system. Configuration parameters are read in at execution time and are in MUSIC NAMELIST format. For an example of how to specify these parameters see the file $PGM:VMSUBM.

Configuration Parameters

UNIT	Addresses of spooled punches defined for use by VMSUBM and SUBMIT program. A maximum of ten addresses may be specified.

The following parameters describe the target systems. Up to ten may be defined.

INSYS	A list of names (1-8 characters long) which the MUSIC user uses to identify the target systems.
OUTSYS	A corresponding list of names which VM uses to identify the target systems.
TAGS	A list of corresponding tags to enable files to be spooled via RSCS to other processing units. This information is only used if the corresponding OUTSYS is RSCS. Each tag consists of two 8 character fields, the node (processing unit) name and the virtual machine name.
SPLCLS	The reader spool class on which each system is expecting input from MUSIC.
NUM	The maximum number of records that MUSIC can send to a particular system.
MINUM	The corresponding minimum number of records.

Privileges Required: None.

WAITS: List System Wait Statistics

This program displays a list of all the system wait-time statistics. Every time MUSIC goes into a wait state, a record is made of why it is waiting and for how long it waits. This information can be useful to the system programmer and installation management personnel. The meaning of the wait times can be found in the file XWAITS.

The program can be run from a terminal by entering WAITS. The times are those accumulated since the last IPL.

Privileges Required: CREAD.

WHOACT, WHOALL, WHOSON: List Terminal Users

Three programs are available that display lists of terminal users. The program WHOALL (run by a entering WHOALL) displays a list of all terminals (ports) defined on the system. For each port, it gives the virtual and real physical line address, terminal control block (TCB) number, user code and subcode, and terminal identification. At the end of each line, an asterisk is shown if the port is currently in use. If the port is not in use, the code, subcode, and terminal identification fields are those of the last active user. Fields of vertical strokes indicate a port where the last activity was a new user coming on but no valid sign-on has yet been received.

The program WHOACT (run by entering WHOACT) is similar to WHOALL except that it lists only active ports. (See Figure 17.7 for a sample run of WHOACT.)

The program WHOSON runs in a conversational mode. Simply type in the userid or groups of userids that you want information on. This program also shows how long it has been since the user last asked for user region service (TLAT). It gives an 8-char program name, which is the member name (for /LOAD XMON jobs) or the /LOAD name (for other jobs). Also, just before the program name in the output, there is a "flag", giving the job state: OFF (TCB is not signed on), RUN (running a job), *GO, RD (waiting for user input), SPI (spooled input). In the case of OFF and *GO, the program name is that of the previous job on that TCB.

```
/exec whoact
*In Progress

     W H O   A C T    26APR83
     - - -   - - -     14.55

   SEQ  TCB  VIRT REAL  USERID


    33   33  043  03C  MD69000
    34   34  044  03D  ML55000
    35   35  045  03E  DPG0888
    37   37  047  040  AD91SCR
    50   50  054  04D  AD87000
    52   52  056  04F  ED34000
    54   54  058  089  CFGC000
    55   55  059  08A  MT29000
    56   56  05A  08B  CCSO000
    57   57  05B  08C  DPJM000
    58   58  05C  08D  CCTW000
    85   85  0A1  5D9  CCSO000
*End
*Go
```

Figure 17.7 - Sample Run of WHOACT

Privileges Required: INFO and LSCAN.

XTELL: Sends Messages

This program is similar to the TELL command except that it delivers the message regardless of whether the target user has set MESSAGES ON or OFF.

Privileges Required: SYSCOM.

MUSIC/SP Administrator's Reference

Part V - Chapter 18 - 23 (AR_P5.cm PS)

Part V. MUSIC/SP Internals

Chapter 18. System Internals

MUSIC/SP's Main Storage

Figure 18.1 shows the relationship between MUSIC/SP's real storage and virtual storage layouts.

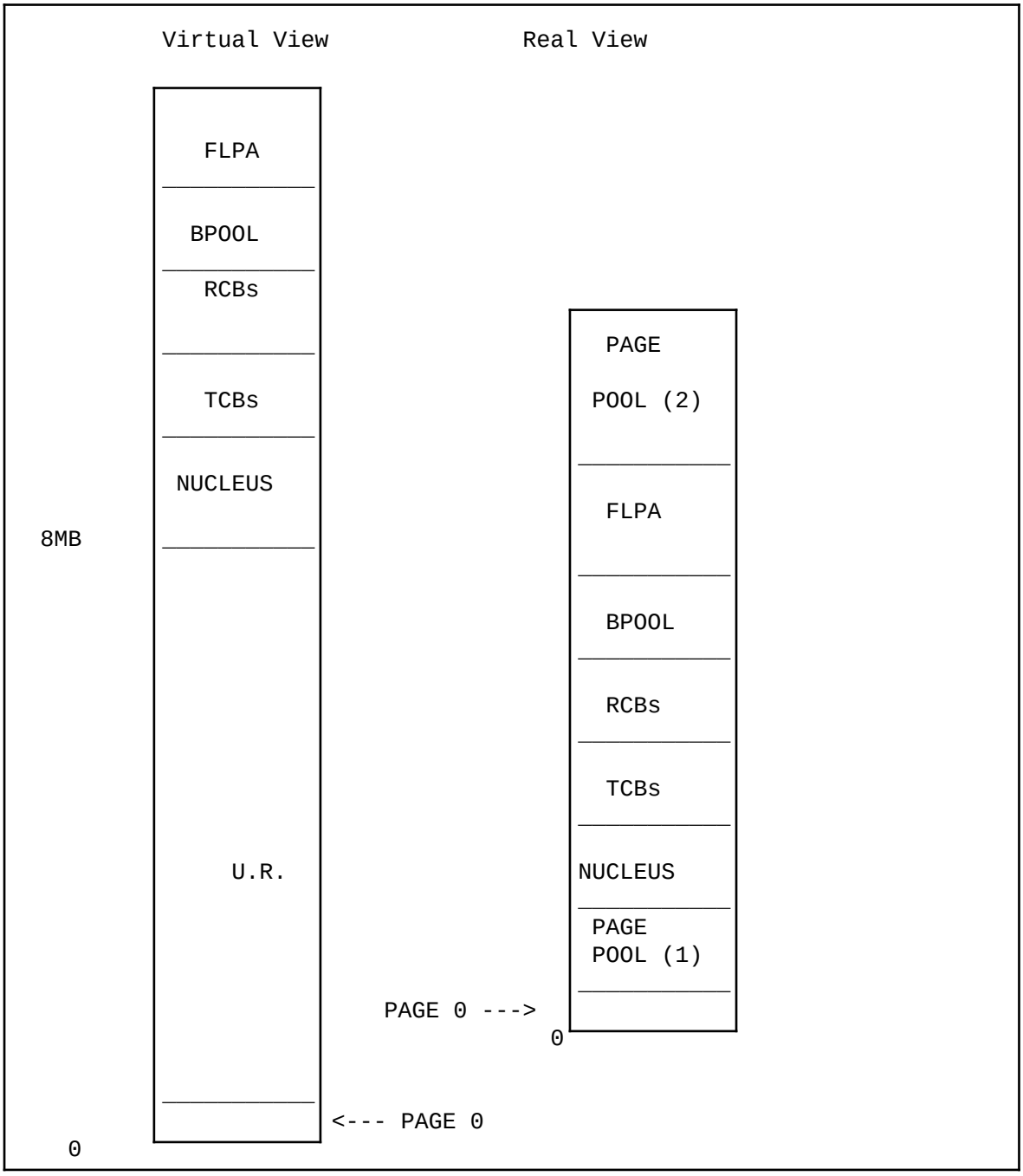


Figure 18.1 - Main Storage Layout of MUSIC

MUSIC/SP uses virtual storage. Page and segment tables are used to map the real storage layout to the

virtual storage layout. When MUSIC/SP is running the virtual storage layout is used by the majority of the system. The real storage layout is only of concern to those components of the system handling the page tables and low level I/O operations.

In the virtual view, the first 8 megabytes are taken up by an area called the User Region. Only a single user task can occupy this area at any given instant. The nucleus starts at address 8 megabytes and is followed by control blocks, buffers, and the Link Pack Area.

In the real view, the nucleus starts at address 30000 (hex). Thus any virtual addresses in the nucleus can be converted to a real address by subtracting 7D0000 (hex). The User Region component for each user is provided from the two page pool areas. At any instant only the pages for one user task are mapped to the virtual User Region area. If there is not enough space in these page pools to store the User Region pages for all user tasks, inactive tasks are paged or swapped to disk. The nucleus, control blocks, buffers and LPA are never paged or swapped.

Nucleus: contains the MUSIC/SP operating system supervisor program.

TCBs: contains the Terminal Control Blocks. There is one TCB for every terminal I/O device defined to MUSIC/SP.

RCBs: contains the Region Control Blocks. There is one RCB for every possible user job that is in the user region area.

BPOOL: contains the *Buffer Pool* (buffer pool) used to transfer information between the user regions and the terminals.

Fixed Link Pack Area:
contains high usage programs used by users. The use of this optional *FLPA* can improve MUSIC/SP's performance as you will see soon.

The User Region

The User Region itself has different sections as is illustrated in figure 18.2.

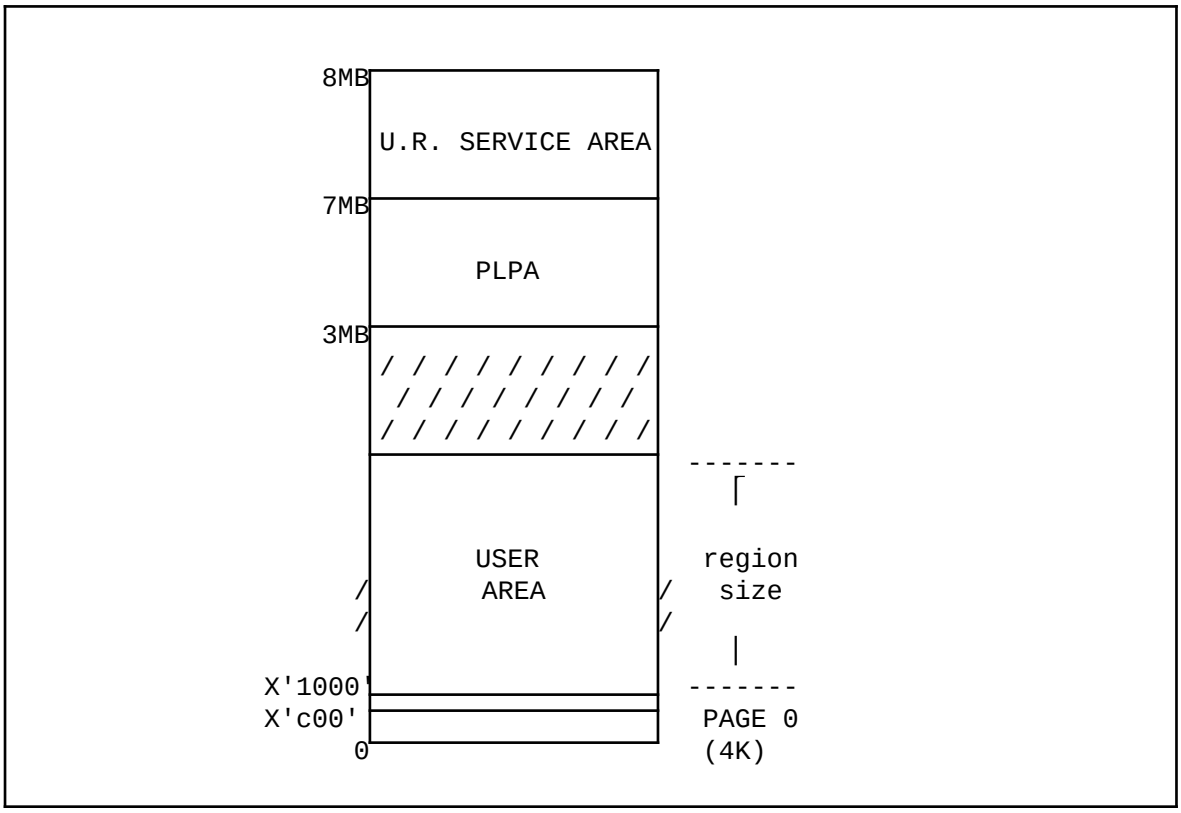


Figure 18.2 - User Region

User Program: This is where the users program is loaded. The maximum size of this area is 3 megabytes.

PLPA: The Pageable Link Pack area contains re-entrant modules that are called from programs in the user program area. This area extends from addresses 3 to 7 megabytes. The modules that can be loaded in this area have been preloaded on the systems page datasets during system initialization. This greatly reduces the loading overhead. Typically the PLPA contains compilers and system utilities that are commonly used.

Service Area: The User Region Service Area is used by the system supervisor in handling service requests from user tasks. It contains I/O buffers, save areas and stacks. One megabyte has been reserved for this area.

As mentioned above, not all users pages can be kept in memory at the same time. In addition, only one users pages can be mapped to the user region at any given time. A major goal of the system is to make sure that all the users get a fair share of the user region. There are three processes that work together to achieve this goal.

- The *scheduler* whose job it is to prioritize the requests for the user regions. The priority is based on two factors, one what kind of request it is and how long has the request been waiting. A job's priority is automatically set and readjusted by the system as the characteristics of the job changes.

- The *paging* process that attempts to keep only the most active pages of a job in main storage.

- The *swapping* process that can block page all the active pages of a job to auxiliary storage should the system determine that the storage can be better used by a higher priority job.

Nucleus Generation

The NUCGEN Utility is used to create the tape or punch file that when loaded writes MUSIC/SP's nucleus to disk. The output of the program is a sequential data set in card image format. The data set consists of the object decks for the MUSIC system and device configuration cards that define the physical I/O unit addresses for the terminals, disks, etc. These card images are preceded by a stand-alone loader utility.

The SYSGEN1 utility can be used to write the nucleus directly to disk while MUSIC is running. The following describes the process when this utility is not used.

The nucleus generation is initiated by an IPL from either the tape or the punch file produced by the NUCGEN utility. The IPL loads a small stand-alone loader utility. This loader then loads the object deck for the MUSIC system module SYGEN1 into main storage. SYGEN1 is a more powerful loader that uses disk work areas on the MUSIC IPL volume. SYGEN1 reads in and relocates the remaining MUSIC system modules. (Do not confuse this module SYGEN1 with the SYSGEN1 utility.)

Once the loading is complete, SYGEN1 moves a copy of the module storage map to the module HELLO. (This map is used by the main storage dump utilities.)

The module SYGEN2 gains control once the loading is complete. It reads the device configuration cards and builds device tables which will be used by HELLO when MUSIC itself is initialized.

SYGEN2 calls the module WMON to format and write the IPL information on the MUSIC SYSRES disk. Then WMON writes the nucleus in core-image format to the dataset SYS1.MUSIC.NUCLEUS on the MUSIC SYSRES disk. Control is then returned to SYGEN1 which writes a completion message to the console and enters the wait state. The new nucleus can now be used by IPLing from the SYSRES disk.

System Initialization

When the hardware LOAD function is performed, the MUSIC IPL text is read into main storage. This IPL text is located on track 0. It was written to disk by the module WMON during the nucleus generation procedure.

The module *MFETCH* reads in the MUSIC resident system from CKD disks and transfers to the initialization module *HELLO* which resides in what will become the user region. On FBA devices, the IPL text written on block 0 performs the function of MFETCH.

The following functions are among those performed. Main storage size and processing unit type are determined. Time and date are set. Free storage is zeroed and protection keys are set. Region Control Blocks (RCBs), Free page storage pool, I/O device Unit Control Blocks (UCBs) and Terminal Control Blocks (TCBs) are created. HELLO prints informative messages to the system operator concerning time, date and disk drive utilization. If any unusual conditions are found, appropriate error messages are issued and initialization is halted.

HELLO calls the module *DSINIT* to process the information in the system catalog. Its major functions are: Locate and read the system catalog, and allow the operator to modify it. Scan all ready disk packs, locate necessary system data sets and construct the Data Extent Blocks (DEBs), Pseudo Device Blocks and Statistics Blocks, positions the accounting dataset, load required modules into the Fixed Link Pack Area, and issue any VM commands found in the catalog.

When initialization is complete, the modules MFETCH, HELLO and DSINIT are no longer required and the

area they occupy is added to page pool 1. Control is then passed to the system scheduler/dispatcher module *URMON* to initialize servicing of user jobs. The first thing it does is run the module JOBONE. job.

JOBONE first scans the Save Library index and deletes any temporary files left around from the last time the system was run. JOBONE then loads the required modules from the load library into the PLPA (Pageable Link Pack Area). It loads each member into the virtual address specified on the RESPGM catalog entry which was read at IPL time. These addresses will be in the range X'300000' to X'6FFFFF'. Once loaded, each member is written out to the page data set. Users are not allowed to sign on to the system till JOBONE completes.

Interrupt Processing

The System/370 architecture defines five interrupt classes:

- Supervisor Call (SVC)
- Input/Output (I/O)
- External (EXT)
- Program Interrupts (PI)
- Machine Checks (MCK)

Supervisor Calls (SVC)

SVC interrupts are only caused by the SVC instruction. SVCs can be thought of as just a call to a routine. The caller need not be concerned with location of the routine in storage. SVC routines always start in the hardware Supervisor State with no I/O or External interrupts allowed.

When the hardware performs a SVC instruction it stores the current psw at storage location 20 (hex) and then loads a new psw from storage location 60 (hex). It will store a word at location 88 (hex) that contains the SVC number as the second half word.

When an SVC is issued control is transferred to the module *TRACE* which records the event in the system trace table. (This trace table is printed by the PRDUMP storage dump utility.) TRACE then branches to the module *SYSSVC*.

SYSSVC checks system status bits to determine if the SVC was issued in SYNC or ASYNC mode. If the SVC is issued by a system interrupt handler, it is in ASYNC mode. If the SVC is issued by a user program or by the system on behalf of a user program, it is said to be in SYNC mode. If in SYNC mode, transfer is made to the module *USRSVC* where the SVC is decoded and the appropriate service routines called to handle the request on behalf of the current user region. The user region service area (USRSVA) is used for buffers, save areas and the stack. Storage in the user region is directly addressable by the SVC service routines.

If in ASYNC mode, then SYSSVC handles the SVC itself. These are system requests and do not pertain to any particular user region, in fact the user region may not even be addressable at the time. Only a subset of the SVCs are valid in this mode.

Appendix C lists the SVCs that are allowed in sync and async modes.

SVC Handling in Trap Mode

There is another layer possible in the handling of SVCs. An user program can request control whenever an SVC is issued. This is known as TRAP mode. This mode is intended primarily for OS-Simulation, but is

also available for other applications. If an application is running in TRAP mode, then SVCs will cause control to be passed to an SVC processing routine that is part of the application. This SVC processor has no special supervisor powers as the system switches back to problem state with interrupts enabled before transferring control. The SVC is decoded and the appropriate action is taken to perform the required task.

For example, when a COBOL program is running, the COBOL object code issues OS SVCs. USRSVC will recognize that the program is running in trap mode and branch to the SVC handler OSTRAP which is in the user region. OSTRAP then may wish to issue an equivalent MUSIC SVC to perform the required function.

Most MUSIC SYNC mode SVC numbers are in the range of 126 and higher and the supported OS SVCs are from 0 to 125. This fact is used by the fast reflect SVC trap option. This fast reflect option is usually on in OSTRAP and so only the SVC numbers 0 through 125 return to OSTRAP for processing. So OSTRAP get control on OS style SVCs (0-125) but MUSIC's native SVCs (126-256) are handled directly by USRSVC.

I/O Interrupts

I/O interrupts are asynchronous interrupts. That is, they can occur at any time if the system mask in the current psw is enabled for I/O interrupts. This is the case when running user programs. I/O interrupts are not allowed while the system is processing SVCs on behalf of user programs except for those in OSTRAP.

When the hardware accepts an I/O interrupt it stores the current psw at storage location 38 (hex) and then loads a new psw from storage location 78 (hex). Location BA (hex) will contain the I/O address associated with the interrupt. The new I/O psw will cause the processing unit to transfer to the module *TRACE* which will record the event in the system trace table. (This trace table is printed by the PRDUMP storage dump utility.)

TRACE will check if the WAIT bit was on at the time of the interrupt and if so it will turn it off and update the wait statistical counters. (See the discussion under topic "Wait State" in this chapter for further details.)

TRACE will then branch to the module *DIOEX* that will check to see if the interrupt is for a disk or tape device. If so, it will update its queues and start more disk/tape I/O. (See the discussion under topic "Disk and Tape I/O" in this chapter for further details.)

If the interrupt is not for disk or tape it will branch to the module *MIOX*. MIOX checks if the interrupt is for a unit record device. Such devices are the card reader, card punch, batch line printer and the main system console. If so, it will process the interrupt.

If the interrupt is not for a unit record device, MIOX will branch to the module *TCS*. TCS handles the interrupts for the terminals. (See the discussion under topic "Terminal Handler" in this chapter for further details.)

External Interrupts (EXT)

MUSIC uses the external interrupts for the interval timer, the clock comparator and the CPU timer. Interfaces to VM services such as IUCV, VMCF, and Logical Device also use external interrupts. Interval timer interrupts are used only at system initialization time.

External interrupts are asynchronous interrupts. That is, they can occur at any time if the system mask in the current psw is enabled for external interrupts. These interrupts are enabled when the system is running user code.

When the hardware accepts an external interrupt it stores the current psw at storage location 18 (hex) and then loads a new psw from storage location 58 (hex). The halfword at 86 (hex) contains a code that identifies the kind of external interrupt. The new external psw will cause the processing unit to transfer to the module

TRACE which will record the event in the system trace table. (This trace table is printed by the PRDUMP storage dump utility.)

TRACE will check if the WAIT bit was on at the time of the interrupt and if so it will turn it off and update the wait statistical counters. (See the discussion under topic "Wait State" in this chapter for further details.)

TRACE will then branch to the module *URMON*.

Clock comparator interrupts are used only by the dispatcher. The clock comparator is set if the dispatcher found nothing to do at some point. When this interrupt comes through, the dispatcher will check its queues again. Outstanding clock comparator interrupts are not cleared if the dispatcher is entered for other reasons. Thus, it is possible to get these interrupts at a later time. Should this happen, they will be ignored.

CPU timer interrupts mean that a time slice has come to the end. URMON will then go to USRSVC to handle it. The transfer to USRSVC is handled as if an SVC 256 had occurred at the location pointed to by the external old psw.

When USRSVC processes the SVC 256, it will check to see if the job has used too much time. If so, then a bit will be set. The dispatcher is then called, in all cases, to end the current time slice.

When the job is dispatched again, and when it returns to problem state, it will check the bit to see if the job time has been exceeded. If so, a message will be printed and the job will be terminated.

Program Interrupts (PI)

Program interrupts (PIs) occur when the processing unit cannot perform an instruction because of (a) it cannot access one of the required storage pages or (b) it found bad data (such as division by 0) or (c) by the attempt to execute an instruction that is undefined or invalid.

Case (a) is known as a *page exception*. The PSW in this case will point to the actual instruction that caused the reference -- not the one following it as is the case with most other program interrupts. The hardware will store the virtual address of the page that could not be found at location 90 (hex). The low order 3 hex digits of this value cannot be counted on to be exact.

When the hardware detects a Program Interrupt it stores the current psw at storage location 28 (hex) and then loads a new psw from storage location 68 (hex). Location 8E (hex) contains the PI code.

This new program interrupt PSW will cause the processing unit to transfer to the module *PAGER*. This module checks to see if it was a paging exception. If so, then this module will resolve the paging exception if it can. Reference to a data page outside of the user region will cause the system to pass back a *protection check* interrupt. This will appear as PI code 4. Reference to an instruction on an odd address boundary that is outside of the region will be treated as *specification error*. This will appear as PI code 6.

If the PI was not be resolved as a paging exception then control is given to the module PITRAP. The action PITRAP takes depends on whether the PI occurred in Supervisor or Problem state. It determines this from the Problem State bit in the psw.

Supervisor State:	PITRAP will issue a console message and stop the system by entering a wait state with all interrupt masks off. A re-IPL of MUSIC is required to recover from this condition.
Problem State:	PITRAP enters the routine XERMON contained within PITRAP.
	XERMON contains routines to fix up the floating-point registers for program checks due to floating-point underflows or overflows.

If the user program has defined a program interrupt exit, then XERMON will branch to it. Otherwise the user program may terminate depending on the degree of severity of the error.

Machine Check Interrupt

Machine checks occur when the processing unit detects an error in its hardware. Examples include the detection of an unrecoverable parity error in storage. Channel checks may or may not cause a machine check interrupt, depending on the processing unit model. Channel checks that are not presented as machine checks are represented by certain bits on in the CSW presented with I/O interrupts. The module *DIOEX* checks for the presence of these channel check bits. If found, DIOEX will branch to the module MCHINT.

When the hardware detects a machine check condition, it stores the current psw at storage location 30 (hex) and then loads a new psw from storage location 70 (hex). This psw will cause the processing unit to transfer to the module *MCHINT*.

MCHINT logs the error. The action taken depends on the type of error:

Soft Machine Check: The system is allowed to continue if the error counts have not been exceeded.

Hard Machine Check: If the error can be localized to a specific user's job, then only that job is terminated. Otherwise, the system is shut down.

WAIT State

MUSIC enters a wait state when there is no processing unit work to do. (The wait state is indicated by the Wait bit on in the current PSW.) The following are the three main reasons for wait states on MUSIC.

- All available user region work has been performed. This is known as the MUSIC idle wait state. The last bytes of the wait psw will be FFFFFF in this case.
- All user regions are all in wait state. This usually means that each one has some I/O to do. The last bytes of the wait psw will be FF00FF in this case.
- Some system function cannot proceed until some I/O is completed. The last bytes of the wait psw will point to the virtual location of the wait request.

MUSIC provides a technique of recording how much time is spent in wait state and for what reason. A task that wishes to enter the wait state issues the MUSIC wait SVC followed by a 2 byte statistical counter number. (The system macro LOOP generates the required sequence of instructions.)

The module *SYSSVC* processes this SVC request. It forms a psw with an instruction address set to retest the completion of the event. The psw also has the wait bit on. This psw is then loaded and the system enters the wait state.

Upon either an I/O or External interrupt, the old psw is inspected in the module TRACE to see if the wait bit is on. If on, the bit is turned off and the time that the system was in wait state is recorded for statistical purposes.

After the system processes the interrupt, it will reload the old psw and thus resume what it was previously doing. If it was in wait state before, the old psw now has the wait bit off which will cause it to retest the completion of the event. If the event is now complete, the task will continue, otherwise the wait SVC will be

redone.

The WAITS utility program can be used to print the time information gathered by this process.

Job Dispatching and Scheduling

Scheduler

Requests for user region service are processed by the scheduler which is located in the module *URMON*. For example, when a terminal user enters a /ID command, TCS issues a $QUEIT SVC which is processed by URMON. Depending on the type of request, URMON then places the request into one of some 9 queues. These $QUEIT requests can come in at any time and do not have any immediate effect on the job currently running in any region.

The items within each queue are serviced in a first in, first out (FIFO) basis. That is, they will be taken in the order they requested service. The only exception to this is the futures queue which is ordered by the time the user wants service. The futures queue is entered by a call to the DELAY subroutine, etc.

Each queue has a bias number which determines the relative priority between queues. Take for example the following case. Suppose the conversational read queue has a bias of 10 and the full time slice queue has a bias of 200. The system would attempt to give response times in that ratio if reads were getting 0.01 second, then full time slice ones would get 0.2 seconds.

Installations can change the bias numbers by changing the table in URMON. These numbers can also be changed dynamically.

Dispatcher

The module *URMON* contains the MUSIC dispatcher. Its job is to keep the user regions busy. It does that by managing chains of Region Control Blocks (RCBs).

At IPL time the system builds a number of RCBs and chains them all into the free chain. Also at IPL time, the system establishes the maximum multiprocessing level (MAXMPL). This MAXMPL is the maximum number of RCBs that can be simultaneously in the active chain.

If any queue has some work that can be done, the system will try to add another RCB to the active chain if the MAXMPL has not been exceeded.

An RCB will stay in the active until one of the following happens to the job represented by the RCB:

- The job exceeds its processing unit time slice. This time slice is determined at IPL time. It is approximately one second divided by the MAXMPL number. Note that this time is pure processing unit time not elapsed time. The RCB is then placed in the dormant chain. The scheduler is called to add the job to the full time slice queue.

- The job exceeds its I/O time slice. This means that the job has performed more than 50 I/O operations in this time slice. The I/O count includes paging I/O. The RCB is then placed in the dormant chain. The scheduler is called to add the job to the full time slice queue.

- The job has ended. The RCB is then added to the free chain.

- The job requests input from the terminal. The RCB is then placed in the dormant chain. (TCS will call

the scheduler when this job is to be activated again.)

- The job has issued a tape rewind or other similar condition that specifically called the dispatcher to dispatch someone else. The RCB is then placed in the dormant chain.

Swapping and Paging

Region Control Blocks

The region control blocks (RCBs) contain information about the user regions that are currently in main storage. In addition to status information the RCBs contain I/O control areas and the page and segment tables. Associated with the page tables are the page frames that are mapped to the user region. An RCB can be active or dormant.

Active RCBs are those that are currently being given a time slice. The dispatcher manages the competition between these tasks for CPU and I/O service. The maximum number of active RCBs that you can have at one time is limited by the "Maximum Multi-Programming Level" (MAXMPL). This number is set during system initialization based on the amount of free storage. It is normally between 3 and 10.

When the time slice ends the RCB becomes dormant. The dormant RCB and any pages associated with it remain in main storage until the system needs pages to satisfy the requests of an active RCB. At this time the module *SWAPER* will be called to free up some pages. SWAPER will call URMON to scan through the list of dormant RCBs and select the one that it thinks will not be needed for a while. SWAPER will then write all the main storage resident pages associated with that job to the swap data set(s). It will write a copy of the contents of the RCB as part of the swap data. After the swap out operation is completed then the RCB can be added to the free chain and the pages added to the free page list.

When the scheduler decides to continue a job, it will check to see if its RCB is in the dormant chain. If so, it can be reactivated by simply adding the RCB to the active queue again. If the RCB cannot be found then it must have been swapped to the swap data set(s). In this case, the system must find a free RCB and sufficient free pages to hold the job again. This may require calling the SWAPER module to swap someone else out.

Maximum Real Region Size (MAXRRS)

The MAXRRS parameter limits the real storage available to a user task. The default value is 272K, but this can be overridden in the NUCGEN job. The value of MAXRRS can have a large effect on performance. If the storage requirements of a program are larger than MAXRRS, MUSIC/SP provides the required virtual storage using both demand and block paging techniques. The larger MAXRRS, the less paging is required. It is not a good idea to make MAXRRS too large however, since it can increase the swap load. The ideal choice for MAXRRS depends on the job mix, the available main storage and the disk and channel facilities.

Swapping

The swap data set is formatted into 4K blocks. It is suballocated in terms of multiples of 10 blocks. Each of these 40K areas is known as a *Swap Set*. By default the MAXRRS value is 272K. Adding the 8K for the RCB gives a maximum of 280K that must be swapped. This will take 7 swap sets. SWAPER will attempt to distribute them over the defined swap data sets. (SYS1.MUSIC.SWAPx). Since multiple swap data sets are usually on separate channels, this will allow for overlapping of the time to do one swap set on one channel with the time to do one on another.

Paging

Paging does not directly get involved in job scheduling or swapping operations. No matter how big the user's virtual storage size may be, the real storage allocated will never exceed MAXRRS. Once the MAXRRS limit is reached the task will page. The paging algorithm is designed so that a task will only page against itself and never take storage away from other competing tasks. This control has the following benefits:

- A job with large virtual storage demands will not overwhelm the system.
- It limits the amount of swap I/O that has to be done even for large jobs.
- It tends to smooth all user's response times by not being so dependent on the number or characteristics of other simultaneously running jobs.

At job start time, no pages are assigned. The job control information including the page tables is stored in the RCB. Pages will be assigned as page exceptions occur. The pages will be obtained from the system-wide page pool. Up to MAXRRS of pages can be obtained per job from this pool.

After the MAXRRS limit has been exceeded, the PAGER module will scan the assigned pages to decide which ones have not been used for a while. The state of the inactive page determines what to do next. The choices are:

- The inactive page is entirely filled with binary zeros so it will be immediately added to the free page pool.
- A copy of the inactive page is also on the paging data set and thus it does not have to be written to disk. This allows the system to be able to immediately add it to the free page pool.
- The inactive page cannot be reused until a copy of its contents are written to disk. It is removed from the active page mapping and is added to the list of pages that must be written to disk on behalf of this user but is not added to the free page pool. If the list has grown to 8 pages then write all 8 in one I/O operation and when done add all 8 pages to the free page pool.

The blocking of page-out operations is called *block paging* and is a significant performance benefit over performing a separate I/O operation for each.

Page-in operations are not blocked. This results in a better usage of the available main storage page frames.

PLPA members are loaded by simply updating the page tables in the RCB. When a specific PLPA page is referenced, then the system will use this information to read in that page from the page data set.

The paging system gets informed of specific MUSIC requests to zero storage. This includes deleting PLPA members and releasing storage at the end of processing phases such as compilers and loaders. Pages that get zeroed in these ways are immediately added to the free page pool.

Notice that users page against themselves and not against each other. This results in a more consistent response pattern. Also note that no paging is done if the user jobs never exceeds MAXRRS. The MUSIC editor only needs about 80K to run and so it does not normally page.

Terminal Handler

The module *TCS* supervises all terminal I/O in MUSIC. When no program is running (*Go mode) TCS reads a command from the terminal. With the exception of trivial tasks, executing the command usually involves running a program in the user region. TCS schedules the user region program and waits for terminal output or requests for input from the program. When the job has finished and all the output has been processed the user is again prompted for a command.

MUSIC uses data spooling techniques to reduce overhead in communicating with terminals. For example, when a program writes to a terminal, the data is simply moved into a buffer and the program allowed to continue as if the I/O had taken place. The module SIOCS is responsible for placing the data into these spool buffers. The terminal handler (TCS) fetches data from this spool and performs the physical I/O at whatever rate the terminal can accept. Automatic synchronization is performed when input is required and at end of job. The system informs TCS that data is pending on the spool by calling the routine WAKEUP. TCS then fetches the data from the spool, one record at a time, and processes it according to it's type. There are three basic types of spool records. The most common type contains output data which is sent to the terminal with the appropriate control characters. The second type of spool record is used to set various terminal options flags. These records are usually the result of calls to the $SETOPT SVC. The third type of spool record is used to initiate a read from the terminal.

TCS is entered on a terminal I/O interrupt (IOTRAP) or from a user region request (WAKEUP). If TCS is busy processing another terminal, the request is queued.

TCS calls several modules to perform part of the terminal support. The modules are:

TERMIO — to perform device dependent functions such as CCW construction and I/O interrupt handling for 2741, 1050 and TTY terminals.

TM3270 — to perform device dependent functions such as CCW construction and I/O interrupt handling for 3270s.

TRMCTL — contains control blocks specifying many parameters which control the way in which the system communicates with remote terminals. Many parameters, such as line length, carriage return rate, tab and backspace characters can be specified for many terminal types.

TCS refers to the information in this module by an index stored in the TCB. This index is set up by the SIGNON phase at /ID time based on the trmcls specification either given on the command or extracted from the user's userid code profile or the default one for the terminal type.

COMAND — Processes the control commands entered by the terminal users.

CHKPFK — is called by TM3270 to perform operations associated with program function keys such as multi-session requests, the retrieve function, and user defined operations.

DEFPFK — processes the /DEFINE command.

NEWTCB — manages the control block pool for multi-session support and provides the logic to add, delete, or move to a new session.

DSPOOL — provides access to the input and output spool and the buffer pool.

FSIO — serves as the interface between the terminal handler and the user region for full screen I/O requests.

TRANTB	contains all terminal translate tables except those used for 3270 APL.
APLTRN	contains the translate tables used for the 3270 APL support.
TABSET	inserts tab characters and idle characters in lines to be sent to a terminal. It uses the terminal's current tabs to ensure that the time needed to print program output is minimized.
TABCMD	processes /TABIN and /TABOUT commands entered from terminals. It translates the values on the commands into internal form and places them into the appropriate TCB.

Terminal Buffer Pool

During system initialization, an area of storage is allocated as the terminal buffer pool based on the BPOOL parameter specified in the NUCGEN. Each buffer in this pool has a 512 byte data area and a 4 byte pointer used in buffer chaining. With the exception of a small input area associated with each terminal, all terminal buffers space is acquired dynamically from this pool. The module DSPOOL and its entry points provide two levels of access to this pool.

The DSPUT and DSGET routines allow the caller to create and access chains of logical records in these buffers. (In-core spool files). Two such chains are provided for each terminal, the input and output chains. Terminal output, spooled input, and full screen I/O, transfer data between TCS and the user region using this type of access.

The GETBUF and FREEBUF routines allow the caller free access to the buffer pool without imposing the logical record format. The 3270 screen manager, conversational read, and the /DEFINE command use this type of access. The FREEBUF routine is also used to free entire buffer chains when a job is cancelled or abnormally terminates.

Since the buffer pool is a limited resource, users are limited as to the number of buffers they can own at one time. When a user exceeds this limit, the user's job is temporarily stopped until more buffers become available. This limit is dynamically adjusted so that it is reduced as the number of available buffers is reduced. It is possible for the system to run out of buffers. When this happens system performance will be effected since all operations that require the buffer pool will have to wait for buffers to become available and almost all terminal I/O requires the buffer pool.

Allocate at least four buffers for each active terminal. The COUNTS and BPOOL utilities can be used to monitor buffer pool usage. If the total number of buffers used consistently approaches the number allocated it is recommended that more buffers be allocated. If the total number used is consistently much less than you have allocated the storage is being wasted. As noted above, it is normal for the user limit exceeded count to be non-zero.

Disk and Tape I/O

The module *DIOEX* supervises the MUSIC disk and tape I/O. DIOEX handles the actual I/O execution on the selector channel(s). It will handle tape and disk, read and write requests, and special purpose functions.

DIOEX is entered by an SVC when a disk/tape I/O operation is required. It will validate the UCB number given in the caller's argument list. If the UCB number is greater than the highest disk/tape UCB, DIOEX will check if it corresponds to a valid Data Extent Block (DEB) number. (Most DIOEX requests use these DEB numbers.) (The DEB's are created at MUSIC IPL time by DSINIT with the exception of those used to

access User Data Set files. UDS DEB's are contained in the XFCB module and are initialized when the user job starts.)

Once the request has been validated, DIOEX adds the request to the channel queue depending on what physical channel the disk/tape is on. If the channel is free, DIOEX will see if it needs to construct a channel program. Most requests require that disk channel programs be constructed using information from the caller's request block and the appropriate Unit Control Block (UCB) and DEB. When the channel program is constructed, DIOEX will start the I/O.

DIOEX also performs the I/O interrupt handling for selector channels, including error detection, correction and logging. If the I/O request signals that the I/O is complete, DIOEX will update the channel queue and process the next request in the queue.

If an I/O error is indicated, a retry operation will be attempted.

The module *UIO* preprocesses all disk and tape I/O coming from user regions. It performs many of the functions that DIOEX does such as looking up DEBs. The CCWs prepared by this module use indirect addressing to access the real storage pages involved in the I/O. It might have to call the PAGER module to page-in part of a buffer. When the CCWs are prepared, URIO will call DIOEX to schedule the I/O. It will then call the dispatcher to run another user until the I/O is complete. When the I/O is complete, the original task will be redispatched and will then return to URIO so it can check the ending conditions and unlock the pages involved in the I/O.

User Region I/O

Most user region I/O requests result in an SVC that is handled by the module *MFIO*. Full details of the MFIO SVC interface can be found in *Chapter 20 - File System*. This single SVC provides the interface to the file system, the UDS file system, terminal input and output, tape, printers, punches and card readers. There are two basic ways that an application can access files and I/O devices.

Applications can access files dynamically. When the open request is issued, MFIO returns an *Internal Unit Number* that the application uses on all subsequent I/O requests. Some devices have unit numbers pre-defined and need not be specifically opened.

The /FILE statement is used to define a file or I/O device for the duration of a job. In this case, the module CTL opens the file and saves the internal unit number in one of two tables in page zero of the user region. The application specifies either a *DDNAME* or *Logical Unit Number* when accessing the file. The system translates this to an internal unit number using the tables set up by CTL.

I/O requests can specify either the internal unit number (1-255) or the logical unit number (1 - 15). Since the logical unit number corresponds to a /FILE statement, the actual physical device can be changed by changing the definition on the /FILE statement. Thus a program that reads from the terminal keyboard using logical unit 9 can be changed to read from a tape by adding a /FILE statement that defines unit 9 as a tape. A program using internal unit 9 will always read from the keyboard.

Unit numbers less than 16 are assigned to specific I/O devices or files. Unit number greater than 16 are assigned when files are opened by MFIO dynamically. The following describes the handling of internal unit numbers. Numbers 1 through 10 also correspond to the default logical unit numbers.

Units 1,2,3,4 These unit numbers are used to access the User Data Sets and tapes. MFIO calls the module *FIOCS* to process this request. FIOCS uses the control blocks in the module *XFCB* to buffer and control these four units. ($CTL, the execution start phase, processes the user /FILE cards and initializes fields in the XFCBs. The XFCBs are part of the module URSRVA.)

Unit 5 This is the system input unit. Commands, programs and input data are read from this unit. MFIO calls the module *SIOCS* to process the I/O. SIOCS starts out reading the terminal or batch input spool. If a /INCLUDE statement is encountered it switches to reading the file named on the /INCLUDE statement.

Unit 6 This is the system output unit. Program output and messages are by default sent to this unit. MFIO calls the module SIOCS to handle the I/O. SIOCS writes the output to the terminal or batch output spool. On batch the output is sent to the printer. On a terminal the output is displayed on the terminal.

Unit 7 This unit is the system punch and it is rarely used these day. On batch it represents the card punch, on terminals the paper tape punch. Again SIOCS is called to do the I/O and the data is written to the batch or terminal output spool.

Unit 9 This is used when a program wants to read interactively from the terminal. On batch it is treated like unit 5. Again MFIO calls SIOCS. SIOCS writes a special record to the terminal output spool and then calls *URMON* to end the time slice. When the module that handles terminal (TCS) detects this special record, it reads the input from the terminal and reschedules the program execution. Once the program regains control, MFIO passes it the terminal input.

Unit 10 This is the known as the HOLDING file. Some compilers write object programs to this file. MFIO will check if this is the first I/O on unit 10 in this job. If this is the case then it will create a file called @HOLD.sss. (*sss* is the subcode.) Any previously existing file with this name will be purged. Subsequent writes to unit 10 just add records to this file.

Units 16-255 These unit numbers are assigned when a file is dynamically opened using the OPEN request of the MFIO SVC.

Batch Spooling

The module *SPAM* supervises the batch spooling in MUSIC. It logically consists of two processes: (1) reading cards and writing them to the batch spool area on disk and, (2) reading the batch output spool area and printing or punching the records.

Batch Input Spooling

SPAM constructs the CCWs to read from the card reader. It calls the module *MIOX* to initiate the I/O operation and perform most I/O error recovery. It calls the module *XTXT* to compress the card images into the same format as used with /INPUT files. (Refer to the topic "Batch Input File Format" for a description of the packing scheme.)

SPAM calls the module *IOBUFR* to buffer these compressed records into physical spool block format. IOBUFR will perform the actual disk request when a spool buffer fills.

When a /ID card is detected the module *CKCDE* is called to perform the userid (code) checking function. CKCDE calls the module *CDSRCH* to locate and read in the userid (code) record. (The job time and page limits, etc, are not processed at this time. They will be handled later by CTL when the job starts running.)

When SPAM reads a /END card, the input spool file is closed and the user region job scheduler is called to queue the job request.

When SPAM reads a /PAUSE card, a console message is sent and spooling continues. When a /END card is later detected, SPAM will check if the operator has responded to the PAUSE message yet. If not, the job will not be queued at this time.

Batch Output Spooling

SPAM calls IOBUFR to read the output spool disk data set. Each logical record is prefixed with a one byte output route code which indicates if the record is for the printer or punch. (The output records are in compressed form with trailing blanks removed.)

SPAM forms the required CCWs to print or punch the record and call MIOX to perform the actual I/O.

Batch Internal Reader

Jobs submitted to the internal reader get written to the system data set SYS1.MUSIC.SUBMIT. Control information exists in the first block of this file. The data set is divided into a number of sections (also called queues), each representing one of the internal reader classes (1, 2,...). A submitted job occupies one block in the appropriate queue.

The subroutine SBMJOB ($SUB:SBMJOB.S) places a job into the submit data set. It also initializes the submit data set, if necessary. The routine SBREAD in the module SPAM reads the job from the submit data set. The module CAR processes the /RDR console command, used by the operator to control the internal reader.

For more information, refer to the comments and coding in SPAM and $SUB:SBMSET.S.

Batch Input File Format

Lines entered through batch as card images (80-character records) are compressed and blocked prior to being written on disk. This packing conserves disk space and reduces data transfer time.

Each card image is compressed by removing high-order (trailing) blanks from the following 3 fields:

Field 1: positions 1 to 6
Field 2: positions 7 to 72
Field 3: positions 73 to 80

Four 1-byte binary counts are appended to the compressed text to enable the line to be unpacked. The final format is

```
C  L1  L2  Text  C
```

where:

C total length of packed line, including control bytes. The C at the end of each line is used by the /DELETE command.
L1 length of field 1 after high-order blanks removed.
L2 length of field 2 after high-order blanks removed.
Text fields 1 to 3 in packed format.

Example:

```
 Column 1      7                    73
        17     FORMAT(I5)           FORT0003
```

appears as

```
        24  2  10  17FORMAT(I5)FORT0003  24
```

A blank line is stored as 04000004 (hexadecimal, 4 bytes).

The compressed lines are blocked into 512-byte physical blocks. A line is not split over a physical block. The last line in a block is always followed by a zero byte to indicate end of buffer. Thus, each 512-byte block may contain from 6 to 128 lines, depending on the amount of information in each line.

Console I/O

The module *CAR* manages the I/O to the MUSIC system console.

CAR is entered by an SVC when a message to the console is requested. The console queue is updated and if possible, the I/O started.

CAR is entered from MIOX upon a console I/O interrupt. The interrupt may indicate the completion of some I/O activity. It could also indicate that the console REQUEST button has been pressed.

Accounting Log

The module *DICDOC* handles writes to the MUSIC accounting log. Records are written to the log by using an SVC. DICDOC buffers these records and does disk I/O when required through calls to DIOEX.

The ACTDMP utility program is used to read this detailed log and produce single accounting records for each session. The format of the detailed log records is described below.

SPAM calls DICDOC to write the information from the /ID card read from the card reader. It also calls DICDOC to write out details of a job's output such as how many lines were printed and how many cards were punched.

SIGNON issues an SVC that calls DICDOC to write information from the /ID commands processed. TCS calls DICDOC to record the number of characters transmitted to the terminal, information about /OFF commands.

URMON calls DICDOC to write end of job records that contain the total amount of time used in the user region.

Accounting Record Formats

The accounting data is stored on the system data set SYS1.MUSIC.ACCT. Each block on the file is 4096 bytes in length. The first block in the file contains pointers into the file. The remainder of the blocks contain the actual accounting records.

The accounting data is written block by block to the disk file by the system module DICDOC. The blocks are written in ascending block number order until all the blocks have been used. Subsequent blocks will be written to block numbers 2, 3, etc. The pointers in the first block contain information that will stop the system from destroying blocks which have not been processed by the ACTDMP utility.

Each block ends with a four byte sequence number. This number is always one greater than the last block written. A discontinuity in these numbers indicates the logical end of the file.

Several record formats are used to hold accounting information in the disk accounting file:

System Initialization Record

```
      Time of day      Date           Clock

K     H H M M    M M / D D / Y Y   C C C C
```

Notes:

1. *K* represents the character count of the record (all counts include themselves).

2. The *CCCC* represents the time of day clock which is in units of two's-complement three-hundredths of a second. The value is stored in 4 bytes.

/ID Record

```
      Logical  Job  Packed ID Line   Clock
CNT    Unit    Code

 K      U U   0 1   X X X - - - X   C C C C
```

General Accounting Information Record

```
      Logical  Job   Request
CNT    Unit    Code    Type     Clock

 K      U U     J       R     C C C C
```

Job Code

02xx Job Request(xx is Request Type--see below)
03 Job Selection
06 Job Cancel or Job Cutoff (see below)
07 End of Output (See below)
08 /OFF Record

Request Type (code 2 records only)

01 /RUN or /EXEC
02 /UPDATE
03 /SAVE
04 /PURGE
05 Jobone system initialization phase
06 /DISPLAY OR /LIST

07 /ID
08 /RENAME
09 /INPUT

End of Output Record (#7) (Terminal)

```
Cnt  Logical      # of    Clock
      Unit        chars

 K     UU    7    XXXX    CCCC
```

End of Output Record (#7) (Batch)

```
cnt             crds rd  crds pch lns prt tape, disk mounts

 K      0     7   XXXX    XXXX     XXXX    X   X
```

Job Cutoff (#6) Records

```
     Logical    Job   Cutoff   Info     Execution
CNT    Unit    Code    Code    Bytes      Time     Clock

 K     U U      6       C       P KK     T T T T   C C C C
```

Job Code

Same as for General Accounting Information record.

Cutoff Code

01 Job Overtime
02 Not used
03 Normal EOJ
04 Not used
05 Abnormal EOJ
06 System I/O Error
07 Abnormal End of phase
08 Not Used
09 Cancel

Info Bytes (#6 records only)

P is processor identification set by CTL.

KK is the number of 4K pages allocated to the job. Minimum is 27 representing 108K.

UDS Delete Record (#10) (See module PST for details)

```
Cnt  Cde         Cde cont'd  Vol   Ucb Type  Details

K    CC    A     CCCCC    VVVVVVV    U  .....
```

Shutdown Record

```
      Shutdown
CNT     Code      Clock

 K    FF FF FF   C C C C
```

Ordinarily the accounting records for terminals will be recorded in logical sequence. However, they are recorded in time sequence, i.e., in order of their occurrence in the system. This means that /ID records for other terminals can be recorded before another job occupying the processing unit is done. The end of output rcd (#7) for batch may occur before or after the end of job record or after the next /ID record. The #7 records for terminals can occur anytime.

Accounting Record Notes

1. The execution time portion of a job cutoff entry is in service units times 300. It is the total chargeable time for the job.

2. /ID's are placed in the Accounting File in packed input format.

3. The accounting records are packed into 4096-character blocks on disk. A record with CNT = 0 indicates end-of-buffer. The last four bytes contains a sequence number. This number is incremented by one for each subsequent record. A discontinuity of these numbers indicates the logical end of the file.

4. Number 6 records usually occur after number 3 records. The exception is that a number 1 record (/ID) can occur between a number 3 and a number 6.

User Code Table

The User Code Table contains the information required by MUSIC for each userid authorized to use the system. This information includes passwords, time limits, options, privileges, etc. (Userids on MUSIC are sometimes called the user's code.) The table resides in two system data sets: SYS1.MUSIC.CODINDX, which is a 2-level index, and SYS1.MUSIC.CODTABL, which contains the individual userid records. Each userid, consisting of 16 characters with trailing blanks, is represented by a code record, which is pointed to by an entry in the index. The index provides quick access to any record, and also allows records to be added and deleted easily.

An SVC ($GETCOD, processed by module CDSRCH) is used to get the code record for a given userid. The SVC searches the index for the userid, then reads the corresponding code record into main storage. For example, at sign-on time, the module SIGNON uses this SVC to verify the user's password and fill in fields in the user's Terminal Control Block (TCB). It is also used by module CKCDE for batch jobs, and by the CODUPD and PROFILE programs.

The Code Table Update utility (CODUPD) is a conversational program which adds, changes, displays, and deletes code records. Usage of CODUPD, along with the various code record fields, is fully described in the *Chapter 17 - System Utility Programs*. A restricted version of this program (PROFILE) enables users to change and inspect some of the fields in their code records, such as passwords, default time limit, auto-program name, etc. PROFILE usage is described in the *MUSIC/SP User's Reference Guide*.

The Code Table Dump/Restore utility, CODUMP, is used to dump the entire code table to tape or a sequential file, for backup purposes. It can also restore the code table from a dump. The restore function recreates the index, compressing it at the same time. CODUMP is described in the *Chapter 17 - System Utility*

Programs.

Source for CODUPD, PROFILE and CODUMP is stored under code $COD in the Save Library (or on the MUSIC source tape). Code $COD also contains some subroutines that are useful for scanning and modifying the code table. File $MCM:CODENTRY.M contains the macro that defines the code record contents.

Code Table Structure

The code table index (SYS1.MUSIC.CODINDX) contains the master index record (the first record of the data set) and the second level index records. The data set is referred to by Data Extent Block (DEB) number 234. Each record is normally 4096 bytes, but a shorter length may be used. The master index record is kept in main storage, for faster code lookup.

Data set SYS1.MUSIC.CODTABL contains the actual code table records, one for each userid, pointed to by entries in the index. It is referred to by DEB number 233. Each record is 512 bytes.

Both DEBs use a pseudo device format, with 10 logical tracks per logical cylinder and 40 (code table) or 11 (index) records per logical track. Use of a pseudo format does not affect the number of records per real track.

The userid is a 16-byte key that is used for indexing the code records. Each index record has one or more 18-byte entries, each entry being a userid followed by a 2-byte pointer. The entries are in increasing order by key, and (unless the record is full) the last entry is followed by 18 hex FF bytes. An entry in the master index record contains the highest key of an index record and the pointer is the number of that index record (the index records after the master are numbered 1,2,3...). An entry in an index record points to a code record (the records in the code data set are numbered 0,1,2,...).

The maximum number of code records is limited by the number n of entries per index record. The maximum number of code records is n*n (assuming that the code data set is big enough). Index records are normally 4096 bytes long, so n = 4096/18 = 227, and the biggest code table could hold 227*227 = 51529 records.

The CODUPD program keeps track of unused index and code records by means of bit maps, which it builds at the start of each CODUPD run. CODUPD is the only program allowed to add or delete code records. All other programs can only get or change records (but they must not change the 16-byte userid at the beginning of the code record).

To avoid wasted space in index records when adding a large group of consecutive codes or subcodes, they should be added in increasing sorted order, i.e. the ADD commands should be sorted by userid.

Code Search SVC

The $GETCOD supervisor call instruction is used to search the code table index for a given code/subcode and read the code record into main storage. The SVC can also be used to request information about the code table, and to mark the main storage copy of the master index record as invalid. For the calling sequence and return codes, refer to the source of system module CDSRCH (under code $SYS). Subroutines CDSVC, CDPRM and INVAL (source under code $COD) can be called from a Fortran program to issue the SVC.

Chapter 19. Direct Access Storage

Overview

Two classes of direct access storage devices are supported by MUSIC: count-key-data (CKD) devices such as 3375 and 3380, and fixed block architecture (FBA) devices such as 3310 and 3370. Data records on CKD packs are accessed by cylinder, track and record number, and the records may be of various lengths. Records on FBA packs are accessed by a single block number (0,1,2,...) and are all of fixed length 512 bytes.

All disk packs used by MUSIC are in the standard OS/VS format. That is, they use normal volume labels and volume table of contents (VTOC). MUSIC CKD packs differ from OS in that most data sets are preformatted, which allows for more efficient access techniques. Furthermore unallocated tracks on UDS packs are preformatted in anticipation of future allocation, as described later on in this chapter. Preformatting is not required on FBA volumes since all blocks are of fixed length.

For CKD packs, the VTOC on MUSIC's IPL pack must not be greater than one cylinder. Format-5 DSCBs on CKD disks are used to keep track of free space, and the free space entries are maintained in order by disk address.

The format-5 DSCBs on FBA disks are not used (there is a single dummy format-5).

For purposes of data set allocation, blocks on FBA devices are grouped into *logical tracks*. A logical track is defined by MUSIC as a set of 32 512-byte blocks, in which the block number of the first block is a multiple of 32. All MUSIC FBA data sets must start and end on logical track boundaries. MUSIC's allocation routines automatically adjust to these boundaries.

Data sets allocated by MUSIC fall into one of two categories: SDS and UDS.

System Data Sets (SDS) contain all system controlled data such as spooling and swap areas, accounting files, Save Libraries and system residence data sets. A system data set must occupy only one extent. A system data set has a name of the form SYS1.MUSIC.xxxxxxx, where *xxxxxxx* is from 1 to 7 characters. The MUSIC/SP file system, the Save Library, is contained within these data sets.

User Data Sets (UDS) can be created by users by means of a /FILE statement. A UDS may occupy from 1 to 5 extents. A UDS has a name of the form ccccxxxx, where *cccc* is the 4-character sign-on code of the owner and *xxxx* is from 1 to 4 characters. Despite the name, most users never use User Data Sets. The MUSIC/SP Save Library is where most user files are stored.

These two types of data sets can be intermixed on the same physical pack. However, to allow for easier dump/restore operations, it is recommended that they not be intermixed. If user allocation is allowed on SDS volumes on CKD disks, then you must ensure that all the free space on the pack is preformatted to conform with the UDS requirements (ie. 512-byte records, no key). For example, before deleting a system data set, use the FORMAT utility to format its space to 512-byte records. The FMFREE utility can be used to format all the free space on a CKD volume.

All SDS packs are permanently mounted at IPL time. UDS packs normally are permanently mounted but batch jobs may make use of temporarily mounted disk packs.

A UDS CKD pack must be pre-formatted using the MUSIC disk format utility before it can be used for user data sets.

A SDS pack is not normally pre-formatted, but any data set allocated on the pack must be formatted prior to

its use.

At IPL time, the system initialization routine reads the system catalog (a data set on the IPL volume). This catalog provides (among other things), the data set names of all system data sets to be used. The volume label of the pack on which each SDS is to be found may also be listed. All ready disk packs are scanned. All SDS are located and control blocks are constructed to later allow system routines to use them. Any pack on which a SDS is located is marked as a SYSTEM pack. All other disk packs are marked as UDS packs.

System Data Sets

The following is a discussion of each type of SDS. This topic is useful if the installation wishes to expand or optimize these data sets for the configuration. Unless otherwise mentioned, the following rules should be assumed.

1. A SDS name must begin with the three characters SYS. The usual naming convention is to start a SDS name with SYS1.MUSIC.. The names used on the distribution system are shown at the beginning of each section.
2. The block size specified for each data set *must* be used. It cannot be altered. If the wrong block size is used, system errors may occur.
3. On the first line of each description, following the data set name, is the following information:

   ```
   data-set-name       identification,  block-size
   ```

   The identification and data set name are specified in the catalog. Certain data sets are not listed in the catalog. For these, no identification is shown.
4. All data sets listed are needed for MUSIC operation.
5. All MUSIC SDS must be one extent as multiple extent SDS data sets are not allowed.

Save Library Index

SYS1.MUSIC.UIDX **U000,512**

This data set contains the index of the MUSIC Save Library. Each file in the library is located through as reference to the data set. The usual size for this data set is 15552 records.

Save Library Space

SYS1.MUSIC.ULnn **Unnn,512**

This type of data set contains MUSIC files. They are numbered starting from 01 up to a maximum of 180. Each data set is referred to as a Library space. The *nn* above refers to the space number. At least one must be present. The starter system comes with several defined. New library volumes must be added in numeric order. Once a volume is added to the Library, it becomes part of the set and there is no quick way to remove it. Each library space can be up to 114,504 records in size.

Batch Input Spooling

SYS1.MUSIC.BATCHIN **B001,512**

This data set is used for input spooling for batch card reader/printer jobs. If it is not available to the system, batch jobs can not be processed. The number of records allocated to this file should be at least one sixth the number of cards to be read in for any single job. In practice, more than six cards will normally be placed in each record. The maximum number of records which will be used is 32767.

Batch Output Spooling

SYS1.MUSIC.BATCHOT **B002,512**

This data set is used for both printer and punch output spooling for batch jobs. It should be available if batch jobs are to run. Output lines are stored in compressed format with trailing blanks removed. A single record may hold several lines. The output from any executing programs may exceed the size of the data set. In this case, the job is temporarily suspended until all output already spooled to disk is printed or punched. Then the job is continued, re-using the spooling area. The maximum number of records of this data set that will be used by the system is 32767.

User Region Swap 1

SYS1.MUSIC.SWAP1 **F203,4096**

This data set is used to store user jobs while they are being executed and not resident in main storage. The block size used is always 4096 bytes. The system divides the blocks into groups of 10. Each group is called a *swap set*. A user's job can use up to nK of real main storage, where *n* is MAXRRS specified in NUCGEN. That size together with the RCB control block of 8K totals (n+8)K. Thus a job may require a total of (n+8)/40 swap sets. These swap sets can be divided up among the swap data sets SWAP1, SWAP2 and SWAP3. A job may use considerably less real storage, resulting in fewer space sets used. Each swap data set size can range from 10 to 64,000 blocks.

User Region Swap 2

SYS1.MUSIC.SWAP2 **F204,4096**

This data set and SYS1.MUSIC.SWAP3 are optional. They are of benefit only when each of the swap data sets are located on different disk channels. When this condition is met, the swap load is distributed among the channels such that they may be concurrently transferring parts of a single user's task. This can provide a significant performance enhancement. All the swap data sets should be approximately the same size, as the system will attempt to divide the swap load evenly between them and when it runs out of space of any one it will not be able to perform the swap operation.

User Region Swap 3

SYS1.MUSIC.SWAP3 **F205,4096**

See notes under SYS1.MUSIC.SWAP2 above.

Temporary User Data Sets

SYS1.MUSIC.SCRATCH **F206,512**

If a user job specifies UDS(&&TEMP) on a /FILE statements, this is an indication that the user wants a scratch file. This is one which is to be deleted at end of job. Instead of going to the trouble (and overhead) of allocating and then deleting a UDS, the system gives to the user a portion of this SDS. It in effect sub-allocates the UDS within this SDS. This data set is the exception to the normal SDS pack rule. It may reside on a normal UDS pack. A pack will not become a SYSTEM pack solely due to this data set. Of course, if any other SDS is found on the pack, it then becomes a SYSTEM pack. This data set can be up to 8176 tracks long. (On an FBA device, 1 track is considered to be 32 blocks.) Note that any job currently running may use this space and that any job may request up to 8,000 512-byte records of space (although most jobs do not use this much).

Page Data Set 1

SYS1.MUSIC.PAGE1 **F207,4096**

This data set is used to store infrequently used pages of user jobs while they are being executed. The block size used is always 4096 bytes. The system divides the blocks into groups of 8. Each group is called a *page set*. The maximum amount of page space that could be used per job depends on the maximum region the job has requested. This size is given on the /SYS REGION= statement associated with the job. Some additional storage beyond this size can also be paged. This holds items such as Save Library I/O buffers.

The first page data set must be big enough to hold all pages associated with the pageable link pack area modules. If your installation has fixed head devices such as a 2305, then it would be desirable to have this data set on it.

Each page data set size can range from 8 to 64,000 blocks.

Page Data Set 2

SYS1.MUSIC.PAGE2 **F208,4096**

This data set and SYS1.MUSIC.PAGE3 are optional. They are of benefit only when each of the page data sets are located on different disk channels. Unlike the swap data sets, the page data sets need not be the same size.

Page Data Set 3

SYS1.MUSIC.PAGE3 **F209,4096**

See notes under SYS1.MUSIC.PAGE2.

Batch Internal Reader

SYS1.MUSIC.SUBMIT **F225,512**

This data set is used with the internal batch reader support. Its minimum size is 200 blocks. A typical size is about 1800 blocks, which allows up to 250 jobs in queue 1 and up to 100 jobs in each of queues 2 to 16.

System Load Library

SYS1.MUSIC.LOADLIB **F226,2048**

This data set holds the processors and phases that may be loaded into the user region or link pack areas. It should contain at least 4000 records. Typical size is 12000 records. The LDLIBE utility program is used to initialize and maintain the load library. The directory should allow for at least 600 members.

Resident System Residence

SYS1.MUSIC.NUCLEUS **F227,note below**

This is the SDS in which the core-resident part of MUSIC resides. It is read into main storage at IPL time. The block size is 6400 for count-key-data devices and 512 for FBA devices. It should be at least 80 records long (1000 records if FBA device). It must reside on the IPL volume and its name must be as shown.

MUSIC Accounting File

SYS1.MUSIC.ACCT **F232,4096**

This data set is used to record all accounting data regarding computer and terminal usage. Records are entered here at such times as terminal sign-on and sign-off, job start and job end. The system will use whatever size data set is allocated. This data set must be initialized using the =RESET special IPL option before it can be used for the first time.

User Authorization Code Table

SYS1.MUSIC.CODTABL **F233,512**

This data set contains one record for each userid authorized to use the system. The maximum number of user code records in the code table depends on the block size B of the SYS1.MUSIC.CODINDX data set: the maximum number is (B/18)**2. Normally B is 4096, resulting in a maximum of 51529 codes or the number of records in SYS1.MUSIC.CODTABL, whichever is less. The total number of records in the data set must not exceed 52000. The number of records must be at least 40.

User Authorization Code Table Index

SYS1.MUSIC.CODINDX **F234,4096**

This data set contains the various index records which enable the system to quickly locate a specific code table entry. A completely full code table should require only (B/18)+2 records in this data set, where B is the block size of SYS1.MUSIC.CODINDX. However, due to fragmentation it is recommended that more records be allocated. This is particularly required when many additions and deletions are made to the code table. A recommended size is 1.5 * (B/18) records (i.e. about 340 records for the standard blocksize 4096). The Code Table Dump/Restore (CODUMP) utility

can be used to compress the code table to eliminate fragmentation.

The total number of records in the data set must not exceed 700. The number of records must be at least 11.

The standard and maximum block size of the index is B = 4096. A smaller block size may be used, but there is no advantage in doing so.

Spooling Area Status

SYS1.MUSIC.HPOOL -,6400

This data set is used by the system initialization at a system START to record the spooling areas and Save Libraries in use. It must reside on the IPL volume and its name must be as shown. It is only one track long.

Permanent Volume List

SYS1.MUSIC.HVLIST -,80

This data set is used to record the list of permanent volumes mounted at IPL time. Its name must be as shown, and it must reside on the IPL volume. The minimum number of records required is one plus the number of permanently mounted volumes.

System Data Set List

SYS1.MUSIC.DSLIST -,6400

This data set logs the data set names and volumes of all current system data sets in use at IPL time. It may be used by statistical and error logging or analysis routines. It is one record long, its name must be as shown, and it must reside on the IPL volume.

Generation Load Utility

SYS1.MUSIC.GENLOAD -,446

This data set is used during the time the system nucleus is being loaded on disk. It is not required during normal operation of MUSIC, but since it is rather small in size, it is recommended that it always be present. Its size is at least 100 records, its name must be as shown, and it must reside on the IPL volume of the generated system.

System Catalog

SYS1.MUSIC.CATALOG -,80

This data set is used to contain the MUSIC system catalog which describes the MUSIC SDS files and their location. This catalog is used only at MUSIC IPL time and must be located on the IPL volume. It should have at least 300 records.

Accessing Data Sets

Data on disk can be accessed in a variety of ways. The most basic method is to invoke the disk I/O scheduler passing the Unit Control Block (UCB) number of the disk to be used, and the absolute cylinder, head, and record of the data on the pack. This method is in general only used at system start-up time and by certain error and recovery routines.

The more common method is not to refer to the UCB, but to a Data Extent Block (DEB) number. There is one such number for every system data set used by the system. In addition to the DEB number, the routine indicates which record(s) of the data set it wishes to read or write. It may use a record number for this (first record of data set is number one, etc), or it may pass a pseudo-device CCHHR disk location. In the latter case, the program assumes that the data in the SDS is arranged as if it were a disk with a specific number of records per track, and heads per cylinder. It constructs what it considers the correct cylinder, head, record address. The disk scheduler converts this to a record number according to the pseudo-device format of the

data set (that is, the number of pseudo records/track and tracks/cyl.). The record number is then normally processed. Each data set which may be accessed in this way has a pseudo-device format specified in the catalog.

For more information about the System Catalog, refer to the EDTCAT utility in Chapter 17.

Allocating MUSIC System Data Sets

MUSIC system data sets are allocated in a similar manner to user data sets. However, several extra parameters are available on the /FILE statement.

All MUSIC system data set have data set names of the following form:

```
SYS1.MUSIC.xxxxxxx
```

The last identifier is a 1-7 character alphanumeric string, the first of which is alphabetic. Since the /FILE statement only allows a maximum of 8 characters in a data set name, the percent character (%) is used to abbreviate 'SYS1.MUSIC.'. A system data set is limited to one extent.

Instead of defining the size of the data set in number of records, the number of tracks is specified via the NTRK parameter. This parameter is only valid when the data set starts with the % character. The NREC parameter may not be used to allocate system data sets. For a data set on an FBA (fixed block architecture) device, one track is considered to be 32 512-byte physical blocks.

The BLK parameter must be used to specify the block size of the data set. For an FBA data set, this is the *logical* block size; a logical block is composed of one or more physical 512-byte blocks, and each logical block starts on a new physical block.

The MUSIC VIP code privilege is required to allocate or delete a system data set.

An example of a job to allocate a system data set follows. This will be a new batch input spooling data set. The BUFNO parameter avoids the unnecessary allocation of buffers in the user region.

```
/ID    appropriate parameters
/FILE 1 UDS(%BATCHIN) NTRK(250) BLK(512) VOL(MUSICX) NEW BUFNO(0)
/LOAD IEFBR
/END
```

Note that IEFBR is a dummy program which simply returns control to the system.

Extra parameters are available (although not often used). They allow a data set to be allocated at a specific absolute address on a disk pack. For example:

```
CC(cc) HH(hh)
```

The values in parenthesis are the starting cylinder and head addresses of the new data set (in decimal). For this allocation to be successful, sufficient free space must be available at the specified address. The 'LISTV' function of the DSKDMP utility may be used to inspect the position of free extents on disk packs. On FBA devices, specify cc as 0 and hh as the logical track number (the starting physical block number divided by 32).

The figures below are useful for calculating the number of tracks needed for data sets. Figures 19.1 and 19.2 give the track capacity (number of records) for various size records. Figures 19.3 - 19.5 review some disk

device characteristics.

RECORD SIZE (IN BYTES)	RECORDS PER TRACK 2314	3340	3330	2305M1	2305M2	3350 NAT
80	40	34	61	28	53	72
446	13	13	22	16	23	30
512	11	12	20	15	20	27
1024	6	7	11	10	12	15
2048	3	3	6	5	6	8
4096	1	2	3	3	3	4
6400	1	1	2	2	2	2
VTOC DSCBS	25	22	39	18	34	47

Figure 19.1 - Disk Device Track Capacity (*Part 1 or 2*)

RECORD SIZE (IN BYTES)	RECORDS PER TRACK 3375	3380	3390	9345		
80	75	83	78	67		
446	43	49	52	44		
512	40	46	49	41		
1024	25	31	33	28		
2048	14	18	21	17		
4096	8	10	12	10		
6400	5	6	8	6		
15476		3				
17600	2					
VTOC DSCBS	51	53	50	45		

Figure 19.2 - Disk Device Track Capacity (*Part 2 of 2*)

	2314	3340 M35	3340 M70	3330 M1,2	3330 M11	2305 M1	2305 M2	3350 NATIVE
Max BLKSIZ	7294	8368	8368	13030	13030	14136	14660	19069
Trks/Cyl	20	12	12	19	19	8	8	30
No. of Cyl	200	348	696	404	808	48	96	555
Capacity mb	29.1	35	70	100	200	5	11	317
Capacity mb Blksize 512	22.5	25.6	51.3	78.6	157	2.9	7.8	230
Capacity mb Blksize 4096	16.3	34.2	68.4	94.3	188	4.7	9.4	272
Xfer Rate kb	312	885	885	806	806	3000	1500	1200
Av Access ms	60	25	25	30	30	0	0	25
Rot Delay ms	12.5	10.1	10.1	8.4	8.4	2.5	5.0	8.3

Figure 19.3 - Disk Device Characteristics (*Part 1 of 3*)

	3375	3380 /arm	3380E /arm	3380K	3390	3390 .2	3390 .3	9345	9345 .2
Max BLKSIZ	35616	47476	47476	47476	56664	56664	56664	46546	46546
Trks/Cyl	12	15	15	15	15	15	15	15	15
No. of Cyl	959	885	1770	2655	1113	2226	3339	1440	2156
Capacity mb	409	630	1260	1890	946	1892	2838	1000	1500
Capacity mb Blksize 512	235	312	625	937	418	837	1256	453	678
Capacity mb Blksize 4096	377	543	1087	1631	820	1641	2461	884	1324
Xfer Rate kb	1859	3000	3000	3000	4200	4200	4200	4400	4400
Av Access ms	19	16	16	16	9.5	12.5	12.5	10	11
Rot Delay ms	10.1	8.3	8.3	8.3	7.1	7.1	7.1	5.6	5.6

Figure 19.4 - Disk Device Characteristics (*Part 2 of 3*)

	3310	3370-1 per arm	3370-2 per arm	9332	9335
Block Size	512	512	512	512	512
Capacity mb	64.5	285	364.9	184	412
Capacity bk	126016	558000	712752	360036	804714
Xfer Rate kb	1031	1859	1859	2500	3000
Av Access ms	27	20	19	19.5	18.0
Rot Delay ms	9.6	10.1	10.1	9.6	8.3

Figure 19.5 - Disk Device Characteristics (*Part 3 of 3*)

Adding Disks to MUSIC

All MUSIC packs use standard OS/VS type formats. That is, they have standard labels and VTOCS. This label and VTOC must be created before attempting to use a new disk on MUSIC. The Device Support

Facility program (DSF) can be used to perform this initialization. In addition, it is useful in detecting and correcting disk errors. A copy of DSF can be loaded by IPLing from the SOURCE tape. For more information see Device Support Facilities User's Guide and Reference (GC35-0033).

The OS/VS IEHDASDR or standalone IBCDASDI programs can be used to initialize non-FBA disks. The MUSIC utility INITFBA can be used to initialize FBA devices.

When initializing a pack any volume name may be used, as long as no two packs (to be used at the same time) have duplicate names. The user is cautioned against changing the name of the MUSIC1 UDS volume as certain execution procedures (for example PL/I) refer to it.

If a volume is to be used for User Data Sets, it should be formatted as documented under the Disk Format Utility (FORMAT). A VOLUME specification should be added to the MUSIC system catalog to enable users to allocate on this pack. All that is then required is to ensure that it is online when MUSIC is next loaded. In addition, the DSLIST file should be changed to include this new UDS pack (last statement of file). Take care to ensure that the file remains execute-only (XO). If the DSACT program is used, its control statements should also be changed to include the new volume.

If a volume is to be used for System Data Sets (SDS), it must be mounted and ready at IPL time. Any required data sets should be allocated as detailed in the previous topic of this chapter. The data sets should then be formatted using the Disk Format Utility (FORMAT).

The DSCOPY program can be used to move the contents of system data sets from one disk to another without using tape. The catalog statements in file SYSCAT should be modified to reflect the new data set locations. The catalog creation utility (EDTCAT) should be run to create a new catalog. The system should then be reloaded. If the code table was newly created, a current copy (dumped before reloading) should be restored using the code table dump/restore program. If a new subroutine library was allocated, it should be recreated by performing a subroutine library edit. If new Save Library data sets were allocated, formatted, and specified in the catalog, they are automatically incorporated into the Save Library at IPL time.

The case of moving existing Save Library data sets is a somewhat special case. Care must be taken to ensure that *nothing* on the entire Save Library changes from the time any of the data sets is copied until the time the system is loaded using this new library.

It is recommended that the starter system (after generation) be preserved so that in the case of any failure of the production system, a workable, compact system is still available.

Migrating MUSIC to a Different Type of Disk

Take the following steps to convert DASD.

1. Initialize the new volumes, using e.g. DSF as in the *MUSIC/SP Administration Guide*, "Installation" chapter. Configure your existing MUSIC system so that the new volumes are accessible to it, as extra volumes. You will probably need to assign different volume names to the new volumes for now, to avoid duplicate names -- e.g. MUSNWX, MUSNW1. You will rename the new volumes later, to their standard names MUSICX, MUSIC1.

2. Format the new volumes like UDS volumes, i.e. 512-byte blocks. Use FORMAT program with /INC FMTUDS. See control statements in description of FORMAT utility in MUSIC Admin. Reference. This job (and most jobs that follow), requires the /VIP ON command to be entered before you run it.

3. Allocate the new system data sets. These are all the SYS1.MUSIC.xxxxxxx data sets that MUSIC needs. See the chapter "Direct Access Storage" in the Admin. Reference. Use your existing data sets as a guide.

4. Format any system data sets which do NOT have blocksize 512, using the FORMAT program. They are SWAPn, PAGEn, LOADLIB, NUCLEUS, CODINDX, HPOOL, HVLIST, DSLIST, GENLOAD, CATALOG.

From this point on, keep all users off of MUSIC, and stop all BTRMs, until the migration is complete.

5. Use DSCOPY to copy any system data sets which contain data: UIDX, ULnn, SUBMIT, LOADLIB, CODTABL, CODINDX. (Do not copy NUCLEUS and CATALOG, since they will be generated below.)

6. Copy UDS files:

   Note: If your users do not use UDS files, or if you have only a few UDS files, you can skip this step and copy the UDS files manually, using DSCOPY.

   a. Scan old VTOC's (using STDSCB routine) to produce job stream of DSCOPY jobs, preserving LRECL, backup, and number of records.
   b. Temporarily REP the module PRE to not update the date of last access: DS1REFD field in format-1 DSCB, near label PREB106.
   c. Submit the DSCOPY jobstream to batch.
   d. Run a program to copy some format-1 DSCB fields from old VTOC to new:
      - date and time of creation
      - date and time of last accounting
      - date of last access
      - opened-for-write bit

7. Prepare the new system catalog (with ultimate volume names) and use EDTCAT program to write it to the new IPL volume.

8. Prepare a NUCGEN tape for the new system: new unit addresses, device types. This involves running your $GEN:NUCGEN.JOB file (updated as required), with output to tape. See the NUCGEN utility in the Admin. Reference.

9. Rename the new volumes to their final names, by using the NEWID option of the FORMAT utility. This needs /VIP ON.

10. Shutdown MUSIC and remove the old volumes.

11. IPL the new NUCGEN tape.

12. IPL the new system and test.

Chapter 20. File System

Save Library

MUSIC/SP's file system is called the *Save Library*. Physically the Save Library consists of a group of system data sets named SYS1.MUSIC.ULnn and an index data set called SYS1.MUSIC.UIDX. The system manages this common pool of disk space to provide file space for the users. Users need not be concerned about the physical organization of these data sets. Logically, each user has their own private set of files. The ownership of these files is based on an *ownership id*, which in most cases, is the userid that is used to logon to the system. The exception is for userids that have subcodes - the subcode is excluded for file ownership since these users share the same set of files.

File Names

A file name consists of a 1 to 17 character name. These file names can be stored in directories. These directories give the file system a tree-structured hierarchical structure.

The file system can refer to any file in the system by using what is called a "fully qualified" file name. All files in the system have a unique fully qualified name. An example of a fully qualified name is:

```
ABCD:CS100\NOTES
```

In the above example, ABCD refers to the ownership id. Note the colin (:) is used to separate the ownership id from the rest of the name. The file name is NOTES and it is located in the directory called CS100.

The file system has reserved 64 characters to hold the fully qualified name field. The directory and file name portion including any "\" characters can be up to 50 characters in length in total.

Usually users only use the *filename* part when referring to their own files. The ownership id and the colon can be prefixed in front of the directory and file name portions to refer to files belonging to other users. If a file has been stored with the share (SHR) attribute then any user can access that file. An administrator with the FILES privilege can access any file by prefixing file names with an ownership id.

Files can also placed in something called the *Common Library*. In this case a special index entry is created so that all users can reference the file without having to prefix the file name with the ownership id. Files for MUSIC/SP commands and utility program are in this Common Library. Files that are to be accessible to all users would normally be placed in this library. There is an option in the user profile that can be set to allow or disallow saving files in the Common Library.

Record Formats

The following record formats are supported (FC is the most used format):

F Fixed Format. All records are the same length and are physically stored on disk that way. Direct access is possible with such a file since each record occupies a fixed and predictable position on disk. The system maps the fixed format records to the physical 512 byte blocks. Records less than 512 in length are packed into the physical blocks but do not span those blocks. Records greater than 512 may span blocks but, each new record must start at the beginning of a physical block.

FC Fixed Compressed. To the application, all records appear to have the same record size. The record

size is defined when the file is created. Internally the file is compressed with repeated character sequences of four or more of the same character reduced to two bytes. Trailing blanks at the end of each record will never be written to disk. Logical records may span 512-byte disk boundaries though the user never needs be concerned about the internal representation of this file type.

V Variable Length. Each record is written to disk without compression. The length of each record is also written so that the length may be returned when it is read. The system keeps track of the size of the largest record written. The maximum record length is 32760. Trailing blanks at the end of each record may be truncated, by user option, so that they will never be written to disk. Logical records may span 512-byte disk boundaries though the user never needs be concerned about the internal representation of this file type.

VC Variable Compressed. Internally the file is compressed with repeated character sequences of four or more of the same character reduced to two bytes. The length of each record is also written so that the length may be returned when it is read. The system keeps track of the size of the largest record written. The maximum record length is 32760. Trailing blanks at the end of each record may be truncated, by user option, so that they will never be written to disk. Logical records may span 512-byte disk boundaries though the user never needs be concerned about the internal representation of this file type.

U Undefined. This format allows the user to use any style data handling desired, although the user must do the deblocking or blocking since the logical record routines will not attempt to handle it.

Record Size

The maximum record size is 32760 bytes. The minimum length is 0 for V and VC, and 1 for F and FC.

File Sizes

The minimum file size is four 512-byte records for a total of 2048 bytes.

The first 512-byte record of a file is used by the system to keep control information about the file. This record is called the *header*.

There is no absolute limit to the size of a file. A file's initial allocation must be satisfied in one library space. Since a single library space can hold up to 57 million bytes, that limits the initial size of a file. For any particular user, the system administrator can set a personal limit to be smaller than this.

Direct Access Support

Regardless of the format, records in all files can be accessed directly by telling the file system to position itself at a particular location in the file. This feature is particularly useful with fixed format files, where the application knows the positioning information in advance.

For all formats except U, file positioning information is given in the form of a relative byte address (RBA). The RBA is relative to the first block of the file. The RBA of the first usable byte in the file is 512 since the file's header occupies the first 512 bytes. The RBA of the second usable block in the file is 1024. Record format U files are positioned by physical block number.

The access methods supported by MUSIC allow direct access based on record number on fixed format files. OS-style NOTE and POINT operations can be done on all files excepting format U. A *Write Rule* is used to avoid the security exposure of accessing the old contents of file written by the previous user. (See Usage Notes for details.)

Dynamic Access

Files can be dynamically created, deleted, opened, closed, expanded and read or written into during the execution of a program.

RAS

Reliability and serviceability is enhanced through the following techniques:

The bitmaps, used to manage file space, are checked with a check-sum technique. When a purge operation is done, the system checks when bits are added back to the maps to detect the case of freeing space already freed. In this case, the total bit count is maintained correctly. If the checksum fails, the system will remove this space group from future allocations. The system will keep running.

Should a part of the library be unavailable, (due to failure of disk drive, for example), the system will still be able to operate. Attempt to use files in the unavailable space will result in a message to the user.

Should the master index get destroyed, it is possible to recreate it in most cases by scanning the library spaces since the file header has an unusual character sequence at the start of it. Headers of purged files are zeroed.

User Controls

Associated with each userid is a User Control Record (UCR) that controls the amount of file space that the user can have. Even privileged users cannot save a file under a userid that has no UCR record. The UCR record contains the maximum amount of space the user can allocate per file and maximum total space for all the user's files. The record also contains the amount of space currently used. All space amounts are expressed in units of K bytes (K=1024). (Since the basic allocation unit is currently set to 2K, the space used amounts will be even multiples of 2.)

Usage Dates

The system maintains three dates: date last opened for writing, date last opened for reading only, and the date the file's header was created. The time of day that the file was last opened for writing is also maintained.

Audit Information

With each file is maintained the full userid of the creator and last writer.

There is an option to maintain a usage count for each file recording the number of accesses.

Access Control

The basic options are PRIVATE and PUBLIC. A private file can be accessed only by its owner. Public files can be read by any user but only written by the owner. A user can select that a file can be added to but not read or written in any other way (APPEND). A file can be EXECUTE-ONLY, meaning that it can only be read by the systems job startup routines. User's can run the programs in these files but cannot look at them. The access control can differentiate between two classes of users: file owners and others. Thus a file can be execute only to others but not to the file owner. A number of combinations of access controls are available. The utility FILECH can be used to set or modify them.

Tag Field

Each file has a 64 byte tag field. This field can be used for any purpose the user wishes. It is not read or written as part of the data of the file but is part of the system information in the file's header. For example, it could be used to hold a phrase to describe the contents or usage of the file. Directory names can also contain a tag field.

Installation Field

Each file has a 16-byte installation dependent field that can be used for any purpose. It is reserved for use by installations other than McGill or for features that McGill wishes to use and never be part of the distributed MUSIC system.

Extendable

The Save Library index is a separate data set allowing it to be expanded or put on a faster device should that be desirable. This data set contains the hash numbers used with the index.

The number of concurrently open files can be expanded by the re-assembly of MFIO and MFIOCB. The space used per open file is 108 bytes. The number of open files bears no relation to the number of available buffers.

The size of the buffer pool used by the SL routines is also expandable. Each buffer requires 512 bytes.

Naming Conventions

Refer to the *MUSIC/SP Users Reference Guide* publication for a complete description of file naming conventions and directories. The following is a summary of the conventions.

File Names

The file name can be up to 17 characters in length. The user's current directory name is prefixed to the file name and can result in a string of up to 50 characters long. A "\" character will separate the directory prefix from the file name portion.

The user can prefix the userid (excluding subcode) and/or a directory to a file name. This is useful to refer to files owned by other users or to files in other directories. The general form of the fully qualified name is:

```
USERID:directory\filename
```

Character Set

The following characters are allowed in the ownership id and file name fields:

letters: A through Z (Upper Case Characters Only)
numbers: 0 through 9
special characters: the five (5) characters: # $ @ _ .

The special character "\" can be used when it follows the rules of the tree-structured directory naming

conventions.

The file name can include more special characters than ownership ids. They are:

```
- + % ! & ~
```

The first character of the file name cannot be any of these special charaters or a period (except for the special name of &&TEMP).

National Language File Suffix

File names can be suffixed by the 3 special character sequence "{@}" when used to open old files. This is used in the support of national languages. It allows programs to refer to different files depending on the user's language preference.

At open time, the system will replace these 3 characters with a single character taken from the national language option in effect. For example, if a user has selected the national language option of Portuguese, then opening with the character string "MSG{@}" will result in the attempt to open the file "MSGP". Should that file not exist, then an attempt will be made to open file "MSG". If the current language selected was English, then the file suffix is ignored. The 3-character file name suffix is not counted as part of the 17 character name although the generated name must be 17 characters or less.

The national language suffix characters are:

blank	English
F	French
K	Kanji (Japanese)
P	Portugeuse
E	Spanish
G	German

Reserved File Names

File names should not start with the commercial 'at' sign (@) since certain system utilities will use names of this form. The search order for files starting with this character is also different. For example:

@ADMIN.xxx	Used by ADMIN system facility
@ADMIN.CONFIG	Configuration file in the NUCGEN
@AMS.sub	Used by VSAM's AMS
@APLW.wsname	VS APL workspaces
@APLv.vvvvvv	Used to store VASPL external variables
@APL.DUMP.nn	Used to hold dump of VS APL
@AUTHSCHED	Used by SCHED
@CF.xxxx	Used to hold CONF xxxx
@CI.xxxx	Used by CONF
@CONF.sub	Used by CONF utility to store user's full name
@CONLOG	Used by CONLOG
@CW.sub	Used by Waterloo C
@ELOG.sub.nn	Editor restart file.
@GNJOB	Used by GENPCH
@HOLD.sub	Holds the user's write 10 output
@INPUT.sub	Holds the user's /input file
@date.SCHED.	Used by TODO to hold user's monthly schedule.
@INI.xxx	Used by FSI

@INI.LISTING	Used to hold job output run under FSI
@INI.sub	Used to hold the FSI options.
@LEARN	Holds the restart info of the LEARN command of IIPS
@LIB	Used by /library command to hold list of files
@MEMO	Used by the now obsolete MEMO facility
@NAMES	Used to hold the MAIL directory.
@PRT	Used by the print screen utility (F9)
@REMIND	Used by REMIND
@REXX.sub	Work file used by REXX
@EXEC.sub	Work file used by the exec command of REXX.
@SENDFILE	Used by the SENDFILE program.
@SORT.TMP	Used by SORT macro for editor.

Qualification

The 17-character file name can have the period characters anywhere *within it* (ie not at the beginning or end). Examples of valid names are RM.LETTER.05DEC77, PROG.V1.L1, THISISALONGNAME.

For example, MAIN.SRC could contain the source and MAIN.OBJ could contain the object deck for the program MAIN.

The qualification can be used to give the appearance of a PDS as in the example: SCRIPT.MAIN, SCRIPT.FORMAT, SCRIPT.SAMPLE, etc.

Notice that CMS's naming conventions of filename/filetype can be directly mapped to filename.filetype.

Specifying Ownership Ids

The ownership id is the userid excluding any subcode. It can prefix the file name as in the example: OWNERID:FILENAME. Note the use of the colon (:) to separate the ownership id from the file name as well as specifying the presence of an ownership id.

If the ownership id starts with an asterisk (*), it takes on the following meanings:

*com:filename	means look into the common library only
*usr:filename	means look into the user's private library only

Temporary Files

Temporary files exist only for the duration of a single job. The only data set name that specifies a temporary file is &&TEMP.

Temporary files are always deleted when closed unless a rename to some permanent name is done at that time.

Internally the system will use a filename of the form .nnnnnn where nnnnn is a number that starts at 1 at IPL time and is increased by one each time a temporary file is allocated. The system will purge any temporary files that exist at IPL time through a routine in the transient phase $PUS.

Save Library Usage Notes

- When one user has a file open for writing, others are locked out as with the UDS system. DISP=WSHR can be used for valid multiple write situations.
- Write Rule. Only affects direct access operation. The system remembers the highest block number written. You may never write beyond this point unless it is the next highest record. This scheme prevents a security exposure that would otherwise result.
- Only one end-of-file (EOF) mark is permitted per file. The UDS system allows any number of end of file marks. There is no special character string that cannot be written to the file. The location of the eof mark is stored in the file's header and is thus not part of the file's data.
- Tape I/O will normally be buffered by the system to correspond to the LRECL and BLKSIZE given on the /FILE command. If RECFM=U is given, then each IO request reads and writes a complete block.
- The minimum record length for reads from file types RDR and TERM is 80 bytes. For other file types the minimum is one byte for RECFM=F and FC files and zero bytes for RECFM=V and VC files.
- Should the file run out of space during a write operation, the system will automatically add secondary extents. The amount of secondary space added as well as the maximum size is controlled by attributes stored with the file.

Assembler Language Interface

Overview

This section describes the macros and control blocks that constitute the assembler language interface to the MUSIC file system. A high level language subroutine interface is also available. It is described in Chapter 22.

In order to use a file the user must OPEN the file. If the file does not exist, there is an option that will have the open create one at this time. Several other options exist on the open and are discussed below. Once the file has been successfully opened, the user can then read and write records to and from the file. Two general techniques exist to read and write to the file. One is the unbuffered technique. With this technique, the user must provide the block number that starts the read or the write request. Logical record deblocking with this form of request is the user's responsibility.

The other, more common access method, is to allow the system to do the logical record deblocking. With this technique, the user need not be concerned with how the logical records are mapped into the physical disk records. There are four record formats: F, FC, V and VC. The meanings of these four formats were discussed in a previous section.

Once the read and write operations have been completed, you then CLOSE the file. On the close operation you can specify that the file be deleted from the system, or kept, or renamed to some other name.

All the requests reference a MFIO argument list. The user may choose to use the same argument list for all files that will be referenced in a program, or the user can choose to have different ones for each file. The argument list is composed of two sections: one a fixed section which must always be given. This is followed by a list of optional argument pointers. Pointers can be given to optional arguments which have no meaning for the specific request currently being used.

WARNING: Do not attempt to use the MFIO interface from within the MUSIC resident system modules. They are designed to be used by code in the User Program or Link Pack areas.

Macros

The following five macros are used. These not only give access to the Save Library file system, but are used to access tapes, UDS files, terminals, and other I/O devices.

MFARG	This macro is used to set up an argument list. Several MFARG's can be given to set up one argument list. The argument list is not generated until a MFGEN macro is used. It is quite common to have several MFARG macros in sequence terminated with a MFGEN macro.
MFGEN	This macro completes the argument list definition defined by a series of MFARG macros.
MFVAR	This macro can be used to create any of the argument blocks pointed to by the MFARG macro.
MFSET	This macro alters the request options in an argument list. It performs no system request but just sets the required bits for a future request. *Register 1 is altered by this macro*.
MFREQ	This macro issues the SVC that performs the request defined in the argument list. *Register 1 is altered by this macro*.

req-name parameter

The *req-name* parameter is used in several macros to specify the type of request. The valid req-name parameters are given below:

OPEN	The open function is to be performed.
CLOSE	The close operation is to be performed.
DIRSRV	A directory service function like CD, MD, or RD is to be performed.
FSIO	Perform 3270 full screen operations.
IO	Perform a read or write or control request using the logical record deblocking routines. The file's RECFM determines how the deblocking will take place.
EXTRACT	Extract information about this file but do not open it. The user must have enough privileges to at least read the file in order that the extract operation be done.
JOBI	Perform the job initialization functions. This call is used internally and can never be done from a user program.
JOBT	Perform the job termination function. This call is used internally and can never be done from a user program.
LIBR	Return the parameters which enable the caller to perform a LIBRARY operation.
MISC	Performs miscellaneous functions.
MSG	Get error message text (treated like an I/O read).

UIO Perform an unbuffered read-write request. The read/write request always starts at the beginning of the 512-byte disk block and continues for as many blocks as required to satisfy the length field. The file's RECFM has no effect on this type of I/O.

USERCTL Read or write a user control record (UCR).

MFARG Macro

The following are the operands that may be placed on this macro:

req-name Specifies the request code to be assembled into the argument list. The request code of numeric 0 may be used if none is to be assembled into the list.

(option1,option2,...)
Specifies the options to be assembled into the argument list. Options are separated from each other by commas and the entire list must be enclosed in parentheses. Refer to the comments under the various request types for valid option names.

optional parameters
Specify the optional parameter name in keyword format followed by the Assembler label of the optional argument. Refer to the separate list below for the valid names.

ULAB= Specifies the Assembler label that will be generated on the one byte internal unit number in the argument list.

U=n Specifies that a unit number of n is to be assembled into the argument list.

RLAB= Specifies the Assembler label to be generated on the one byte return code returned by the various requests.

PICT=Y This parameter can be used to have the macro generate a diagram showing the layout of the argument.

The label field of this macro will generate a corresponding label if it is used on the first MFARG macro in a sequence. Otherwise the label will be associated with the optional parameter pointer given on the macro. Because of this second usage, the user should use separate MFARG macros when the user wishes the labels to be associated with the specific optional parameter names.

The MFARG instructions (one or more) are followed by an MFGEN macro instruction, which causes the argument list to be generated. An optional operand *AREA=location* on MFGEN causes MFGEN to generate executable instructions to create an MFIO argument list at the specified location, using information from the MFARG instructions. This makes it easier for reentrant code to set up an MFIO argument list in a work area defined by a DSECT. A label may be used on MFGEN if the AREA option is present. When AREA is used, MFGEN modifies register 1.

Optional Parameters

The optional parameters on the MFARG macro are given in a keyword format. For example, XNAME=FNAME would be used to specify that the optional parameter XNAME should point to a label called FNAME in the user's program. The valid optional parameter names are given below:

ARG Points to the buffer location and length associated with the current read or write operation. For certain types of I/O, this control block will also contain the starting record number or RBA locator.

BUPNUM	Points to the argument that contains the backup tape number associated with the last backup operation done for this file. This number may be zero indicating no backup done yet for this file. This number can be set by privileged users via the SETBFG option on the CLOSE request.
EOFPT	Points to the argument area which contains the highest block number written and the displacement of the last written byte within that block.
EXTNTS	Points to the argument area which contains the extent list of the file.
FSARG	Points to the 3270 full screen I/O argument. This contains the buffer addresses, lengths and control options used by the interface.
INFIN	Points to the information to be assigned to a new file. Such information includes its records size, its record format, and its access control.
INFOUT	Points to the location of the control block which is to be filled in by the system to contain information about an existing file. INFOUT may point to the same location as INFIN.
LIBR	Points to the optional argument area used in conjunction with the LIBR request.
XNAME	Points to the file name or the ddname of the file (64 characters).
NAME	Points to the file name or the ddname of the file (22 characters). The XNAME parameter should be used for new code as it can handle longer names.
RNAME	Points to the returned file name (64 characters).
PHYS	Points to the control block that will contain the current record length and RBA information after a read or write operation.
TAG	Points to the tag field to be associated with a new file or the location where the system is to move in the tag information of the existing file. The tag information is only returned if the user is the owner of the file.
UCTL	Points to the argument area used to access and modify the user control record for a specific library code. A non-privileged user can only read the user's own UCR record.
HINFO	Points to the argument area that contains the usage information from this file's header. This argument replaces UINFO used before ownership ids could be longer than 4 characters. The only case to use UNIFO is for EXTRACT requests that use the header location on input.
UINFO	This argument was used before ownership ids could be longer than 4 characters. The only case to use UNIFO is for EXTRACT requests that use the header location on input. Use HINFO for other cases.
XINFO	Points to the 40-byte argument area that contains additional usage information from the file's header. The first 4-byte word contains the time of day (in units of 1/300 sec) of creating or last open for write.

MFVAR Macro

This macro is used to generate any optional argument that may be required. The argument name that is to be generated is given as the first operand. The name is the same as the req-name given on the MFARG macro. A keyword parameter of PRE=prefix, may be used to specify a four character prefix to be placed in front of

the generated labels within the argument. The keyword parameter of PICT=Y, can be used to generate a diagram to show the layout of the various fields in this argument.

MFSET Macro

The operands in this macro are as follows:

arg-name This specifies the Assembler label of the argument list to be used for the request. The label is the one given of a MFARG macro.

req-name This specifies the request type to be moved into the argument. Omit this parameter if the request type is not to be changed from a value previously set in the argument.

R=(option1,option2,...)
This is used to specify the request options to replace those in the specified argument.

M=(option1,option2,...)
This specifies the request options that are to be modified in the specified argument. Notice that this form only modifies the specified option bits and keeps the remaining ones untouched. You can omit the req-name operand, meaning that you do not wish to alter the request code in the argument list. You may use a minus sign immediately in front of any option to specify that option is to be turned off in the current argument list.

MFREQ Macro

The options for this macro are as follows:

arg-name Points to the Assembler label associated with an argument list generated by a MFARG macro.

EOF=eofadr Specifies the Assembler label to branch to should an end-of-file condition be found during a read operation.

BAD=badadr Specifies the Assembler label to branch to if an unusual condition occurred during the processing of this request.

OPEN Request

The OPEN request passes a data set name, a ddname, or a PDS member name to the SL and upon the successful completion, the system will set the one byte internal unit number in the argument list.

An OPEN by data set name (dsname) is independent of any /FILE JCL definitions. It is a true dynamic OPEN. Various options are available at open time to say whether an existing file with that name is acceptable and whether a new file is to be created if none exists with that name.

An OPEN by ddname requires that a valid /FILE JCL definition be in effect for this ddname. This OPEN retrieves the internal unit number associated with the ddname. The EOFB field in the EOFPT argument is returned for SL and UDS files. The RSIZ and RFM fields of INFOUT are also returned. For files, the PRM field of INFOUT returns the current file size. The file has already been opened when the /FILE JCL was checked. Therefore, the OKOLD option must always be given. The LU option can be given to mean that the first word of the ddname field is the binary logical unit number.

An OPEN by PDS member name uses the root name given on a valid /FILE JCL definition to form a

dsname. Once formed, the OPEN proceeds similarly to an OPEN by dsname. If the dsname cannot be found, it will create a new file if OKNEW is given. If OKNEW is not given, other root names will be tried in the order specified on the /FILE JCL.

At least one of the options RDOK, WROK must be given.

The various options that can be specified with the OPEN request are as follows:

DDNAME	The name provided is a ddname. Only the ddname OPEN will be attempted.
DDORDS	An attempt will be made to perform an OPEN by ddname. If the ddname is not found, an OPEN by dsname will be done.
MEMBER	An OPEN by PDS name will be done. The name field contains the ddname as the first eight characters followed by the member name as the next eight characters.
LU	Used with DDNAME. The OPEN is to be done on a logical unit number. The full word logical unit number is in the first 4 bytes of the name argument.
OKNEW	Specifies that the OPEN will create a new file if one does not exist.
OKOLD	Specifies that an existing file can satisfy the OPEN request.
RDOK	Specifies that the user will be reading from the file.
WROK	Specifies that the user will be writing to the file.
APPOK	Specifies that the user wishes to write records only to the end of the file. The file will be positioned to the end in anticipation of these writes. The WROK option must also be given.
NODATE	Specifies that the date last opened for reading will not be updated. This option is only valid for privileged users. When the file is opened for writing, this option has no effect.
SETUI	Specifies that the usage information given with the open request is to be used with the new file. This option requires that the user be privileged. This option can use the HINFO or UINFO parameters. Do not use both.
POSEND	Specifies that the file pointers are to be set so that writes done will add records to the file.
NOEFPUP	Do not update the EOF pointer at CLOSE or TCLOSE time. This option is ignored if the RLSE option is used.
NOEFPLO	Do not lower the EOF pointer at CLOSE or TCLOSE time.
ENQSHR	Forces the enqueue operation to be done in share mode. Normally the enqueue operation will be done in share mode only when WROK is not used.
ENQEXCL	Forces the enqueue to be done in exclusive mode. This means that no other user can access the file while this one is open. A user who is not allowed write access to the file cannot use this option. Normally the enqueue operation will be done in exclusive mode when WROK is used.
XONLY	Used only from the module SIOCS to inform MFIO that reads of execute only files are valid at this time.
JOBOPN	Specifies that the file is to be kept open until the end of the job. CLOSE requests on the file

will be handled as if the TCLOSE option was also given. This option is used by the module CTL when opening a file specified on a /FILE statement.

EOJDEL This option is used in conjunction with JOBOPN to delete a file at the end of a job.

EOJREL This option is used in conjunction with JOBOPN to perform the RLSE option at the end of the job.

PGMP Use merged program and user privileges for file access.

DIRNOK Allows the user to create and read the directory file. It also allows the creation of files into a non-existent directory. This option requires the MAINT privilege, otherwise this option is ignored.

NORAM Do not use RAM disk copy of the file. Valid with OKOLD.

CLOSE Request

The following options are defined for the CLOSE operation:

DEL Specifies that the file is to be deleted from the library.

REPL Specifies that a new file name is to be associated with the existing file. If a file already exists with that name, it is to be deleted. If a file does not exist, then this option will have the same effect as the RENAME option. New access control flags are assigned to the new name from the INFIN argument that should be provided when this option is used. Other parts of the INFIN and TAG arguments are not used by CLOSE. The new name cannot be the same as the original file name. This option does not set the RNAME field.

RENAME Specifies that the file is to be renamed to another name only if that new name does not exist on the library. New access control flags are assigned to the new name from the INFIN argument that should be provided when this option is used. Other parts of the INFIN and TAG arguments are not used by CLOSE, except for logical record length when the SETEFP option is used. This option does not set the RNAME field.

RLSE Specifies that any unused space at the end of the file is to be released. This means that the file is to be made as small as possible to hold the existing data.

TCLOSE Specifies that the CLOSE function is to be performed but that the file is to remain open for further requests. The current end of data pointer is updated in the file's header. RLSE is the only other option that can be given when TCLOSE is used.

SETBFG Specifies that the backed up bit in the index entries for this file are to be set. The backup tape number specified by the BUPNUM is also to be set in the index. This option is only valid for privileged users.

SETEFP Specifies that the pointer to the end of the written data is to be updated with the value specified in the EOFPT argument. The file's record size will be set to the value given in the RSIZ field of the INFIN argument. This allows the maximum record size to be set for a variable length file. The SETEFP option is only valid for privileged users and exists primarily for the MFREST utility and the COPY command.

GETEFP Refresh the EOF pointer. The file remains open. This option does not decrease the EOF pointer. This option cannot be used with other CLOSE options.

RESET Do not do pending I/O, but update header block, setting line count to zero and end-of-file pointer to the start of the file. This sequence is logically equivalent to, but faster than, a rewind operation followed by a write end-of-file. The file is closed.

CTAG Set a new tag field when the file is closed. The new tag is taken from the TAG argument.

PGMP Use merged program and user privileges for file access. This option is used with REPL or RENAME.

DIRNOK Allows the user to create and read the directory file. It also allows the creation of files into a non-existent directory. This option requires the MAINT privilege, otherwise this option is ignored.

The above options can be used in combination. Should a request such as RENAME fail to work, the user may issue additional CLOSE requests. When the CLOSE requests is successful, the unit number in the argument list will be set to 0.

IO and UIO Requests

These requests are used to perform read, write and control operations on the file. The valid options are as follows:

LU The unit number in the argument list is a logical unit 1 to 15. The system will use this number to index into $LUXTAB to determine the internal unit to use. This option is used mainly by FORTRAN to perform its I/O. ($LUXTAB is initialized by CTL and may be altered by giving /FILE JCL with numeric ddname.)

RD Read a record from the file.

WR Write a record to the file.

WEOF Specifies that an end-of-file operation is to be done on the file. Not valid with the UIO request. (Notice that with the SL, only one end-of-file marker can exist on the file.)

REW Specifies that a rewind operation is to be done on the file. Not valid with the UIO request.

BSP Specifies that a backspace operation is to be done on the file. Not valid with UIO request and SL files.

FILL Specifies that the user's buffer is to be filled with blanks should the current record read from the file be smaller than the user's buffer. Not specifying this option is no guarantee that the fill function will not be done. Not valid with the UIO request.

TRUNC This specifies that blank characters are to be truncated from the record that is being written to the file. Only meaningful for RECFM=V or VC files. The record will then appear to have no trailing blanks on it. Not valid with the UIO request.

RBA The IO request is to start at the specific location within the file given by the relative byte address (RBA) contained in the ARG parameter. Use this operation with care if write operations are being performed. This option is not valid with UIO requests.

USERL The caller's request length is to override the file's record size for the current IO operation of RECFM=F files. This option is used by FORTRAN when doing direct access operations.

EXTRACT Request

This request is used to obtain information about a file without opening it. (An OPEN request can also obtain the same information.) The options are as follows:

DIRLOC Specifies that the location of the file's header is given in the UINFO argument. Do not use the HINFO parameter. (Note the "header" used to be called the file's "directory". The name has been changed to avoid confusion with directories that the user see.) In this case the UINFO argument is 4 bytes in the form 00XXYYYY where XX is the library number and YYYY is the space unit number (as in the index entry). This saves the index search that would otherwise be done. This option can only be used from a privileged program. It is used by the LIBRARY command. Note that the file name argument must still be provided when DIRLOC is used.

PGMP Use merged program and user privileges for file access.

DIRSRV Request

The Directory Service request usually takes a directory name as input. Upon the successful completion, the system will return a 64 byte RNAME field as a result.

The directory name can either be passed to the DIRSRV request in the NAME field if the directory is 22 characters or less, or in the XNAME field if the directory is 64 characters or less.

Directory names prefixed by with strings like "..\", ".\", and ".." will be resolved by using the current directory.

The various options that can be specified with the DIRSRV request are as follows:

DIRCD Changes directories from the current directory to the specified directory.

DIRMD Creates a new directory.

DIRRD Removes an empty directory.

DIRQD Queries the current directory. Works like DIRCD but does not change the current directory.

MSG Request

This request is used to obtain the message text corresponding to the error code set in the argument list. Should not be called if error code is 0. The ARG argument defines the location and maximum length for the returned message. A length of 70 is considered ample for message texts. Unused location in the message buffer will be blanked.

USERCTL Request

This request is used to read and write a UCR record. A non-privileged user can only read the user's own UCR record. The options are as follows:

SETUCR Specifies that the maximum allocation limit fields in the UCR are to be replaced with the ones specified in the calling argument.

REPUCR Specifies that the user control record is to be replaced entirely with the one specified in the

calling argument.

CODE Specifies that the ownership id associated with the UCR record is to be taken from the NAME argument. Otherwise the ownership id will be that of the current user. Maximum length is 16 characters with a blank terminating the field if shorter.

DELUCR Specifies that the UCR record for this ownership id is to be deleted from the index.

LIBR Request

This request returns arguments to enable the caller to do a scan of the Save Library Index. For example, the LIBRARY command issues this request to determine the first and last block numbers of the index it will have to read. If the caller passes a 4-character ownership code, then only the block numbers corresponding to that segment will be returned.

LIBNAM Gets name from XNAME field instead of from the LIBR parameter.

MISC Request

This request is used to perform miscellaneous functions as follows:

SETLUX Takes the one byte internal unit number given in the basic argument and places it in the logical unit cross reference table (LUXTAB) at location corresponding to the full word logical unit number given in the NAME argument.

SETDDN Adds a ddname to the ddname table (if not already there), and associates an internal unit number with the ddname. The step number in the table entry is set to zero, so that the ddname will apply to all steps of the job. The ddname is taken from the first 8 bytes of the file name argument. The internal unit number is taken from the basic argument block. If there is not enough room in the ddname table, the ddname is not added, and error 31 is returned.

FSIO Request

This request is used to perform full screen I/O operations to 3270 type terminals. The basic request block should contain pointers to the FSARG and PHYS blocks. U=9 should also be specified. The FSARG block contains lengths and pointers to the 3270 data stream buffers. The actual length of the input read is returned in the PHYS block. *Chapter 22 - System Programming* contains detailed information of the assembler and subroutine interfaces to this facility.

File System Messages and Return Codes

Special Conditions

```
Error  1  END OF DATA SET ENCOUNTERED (see also error 44)
Error  2  INCORRECT LENGTH
```

Errors in Calling ARG

```
Error 10  INVALID REQ
```

```
Error 11  INVALID REQ PARAMETER
Error 12  FILE NAME INVALID
Error 19  INVALID ARGUMENTS IN CALL TO SERVICE SUBROUTINE
```

Violations of MUSIC Conventions

```
Error 20   TOO MANY OPEN FILES
Error 21   NOT YOUR LIBRARY
          (Not authorized to store under this code.)
Error 22   NOT YOUR FILE
Error 23   VIOLATION OF WRITE RULE
Error 24   ATTEMPT TO READ BEYOND END OF WRITTEN INFO
Error 25   WRITE THEN READ SEQ INVALID
Error 26   YOUR USERID CANNOT CREATE FILES ACCESSIBLE BY OTHERS
Error 27   YOUR USERID CANNOT CREATE FILES IN THE COMMON INDEX
```

File Existence Errors

```
Error 30   FILE NOT FOUND
Error 31   DDNAME NOT FOUND (see also error 35)
           (on a SETDDN request, error 31 means that there
           is not enough room in the ddname table)
Error 32   FILE ALREADY EXISTS
Error 33   FILE IN USE
Error 34   COMMON NAME USED BY SOMEONE ELSE
Error 35   UNIT NUMBER NOT DEFINED
           (or ddname undefined by user request such as by
           specifying UNDEF on a /FILE statement.)
Error 36   SUBDIRECTORY DOES NOT EXIST
```

Restrictions Imposed by FILE

```
Error 40   SPACE QUOTA EXCEEDED FOR THIS USERID
Error 41   SPACE QUOTA EXCEEDED FOR THIS FILE
Error 42   CANNOT ADD SPACE TO THIS FILE
Error 43   REQUESTED ACCESS OR OPERATION NOT ALLOWED
Error 44   REQ BEYOND EXTENT OF FILE
Error 45   FILE RECFM NOT DEFINED
Error 46   FILE CANNOT BE READ SEQUENTIALLY
Error 47   INSUFFICIENT SPACE FOR BUFFER ALLOC
Error 48   MIN RECORD LEN IS 80 FOR THIS FILE TYPE
```

System Temporary Limits

```
Error 50   FILE NOT ON-LINE
Error 51   NOT ENOUGH FREE DISK SPACE
Error 52   NOT ENOUGH FREE DISK SPACE (IDX)
```

System Integrity Errors

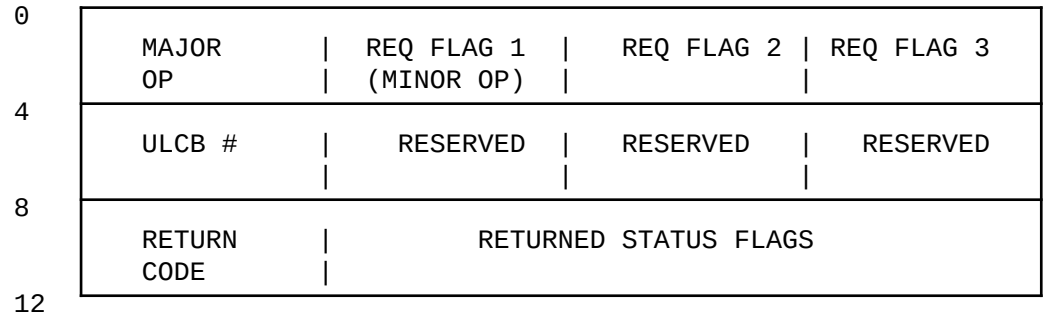
```
Error 60   RD I/O ERROR IN FILE
Error 61   WR I/O ERROR IN FILE
Error 62   RD I/O ERROR IN SYSTEM AREA
Error 63   WR I/O ERROR IN SYSTEM AREA
Error 64   INDEX IN ERROR
ERROR 65   HEADER IN ERROR
Error 66   MAP INTEGRITY ERROR
Error 67   INDEX/HEADER MISMATCH
```

System Coding Errors

```
Error 70   SYSTEM FILE ERROR (logic error in system module)
```

Control Blocks

Basic Caller's Request Block



NAME--Data Set or DDNAME (old format 22 character name)

```
    Same as XNAME but is only 22 characters long

          Label     DC    CL22' '            NAME
```

XNAME---Data Set or DDNAME

```
    CAN BE IN 4 FORMS:

       1) DSNAME   EG. CODE:FILENAMEXXXXXXX (MAX 64 CHARS)
                   EG. FILENAMEXXXXXXX (MAX 50 CHARS)
                   EG. DIR1\DIR2\DIR3\FILENAME (MAX 50 CHARS)
       2) DDNAME   EG. DDDDDDDD (MAX 8 CHARS)
       3) DDNAME AND MEMBER NAME EG. DDDDDDDDMMMMMMMM (MAX 16 CHARS)
                                 EG. DDDD    MMMMMM   (MAX 16 CHARS)
       4) BINARY LOGICAL UNIT NUMBER (4 BYTES)

          Label   DC    CL64' '            XNAME
```

RNAME---Returned NAME from MFIO

RNAME is a 64 byte long field

General format is

```
CODE:DIR1\DIR2\FILENAME
```

One use for this arg is to return the filename actually found on an open request.

```
Label   DC    CL64' '            RNAME
```

INFIN/INFOUT---Size and Attributes

```
0
      +-------------------------------------------------------+
      | INFIN: PRIMARY SPACE ALLOC IN K BYTES                 |
      | INFOUT: CURRENT FILE SIZE IN K BYTES                  |
4     +-------------------------------------------------------+
      | SECONDARY SPACE ALLOC.  IF>0 THEN IS AMT IN K BYTES   |
      |                         IF=0 THEN AMT IS CURRENT + 50%|
      |                         IF-N THEN AMT IS CURRENT + N% |
8     +-------------------------------------------------------+
      | MAX SPACE LIMIT FOR FILE IN K BYTES                   |
12    +-------------------------------------------------------+
      |  INFIN: RSIZ IF RECFM=F,FC |   RECFM      |  (RESERVED)|
      |  INFOUT: MAX RSIZ WRITTEN  |              |            |
16    +-------------------------------------------------------+
      | <----------------FILE ACCESS CONTROL FLAGS------------->|
      | GEN CTL INFO| OWNER ACCESS  | NOBODIES ACC | RESERVED |
20    +-------------------------------------------------------+
```

```
Label     DS    0F
 **FILE SIZE SPECIFICATIONS
xxxxPRM   DC    A(0)             PRIMARY SPACE WANTED IN K BYTES
xxxxSEC   DC    A(0)             IF >0 THEN IT IS AMT IN K BYTES
                                 IF =0 THEN AMT IS CURRENT + 50 %
                                 IF -N THEN AMT IS CURRENT + N %
xxxxMAX   DC    A(0)             SPACE LIMIT IN K BYTES
                                 -1 MEANS UNLIMITED

 **LOGICAL RECORD CONTROL
xxxxRSIZ  DC    AL2(0)           RSIZ IF RECFM=F OR FC
xxxxRFM   DC    AL1(0)           RECFM
                                 =X'00' RECFM=U (NO LOGICAL RECFM DEF)
                                 =X'01' RECFM=F (FIXED)
                                 =X'02' RECFM=FC (FIXED COMPRESSED)
                                 =X'03' RECFM=V (VARIABLE)
                                 =X'04' RECFM=VC (VAR COMPRESSED)
          DC    AL1(0)           RESERVED FOR CARR CTL TYPE
```

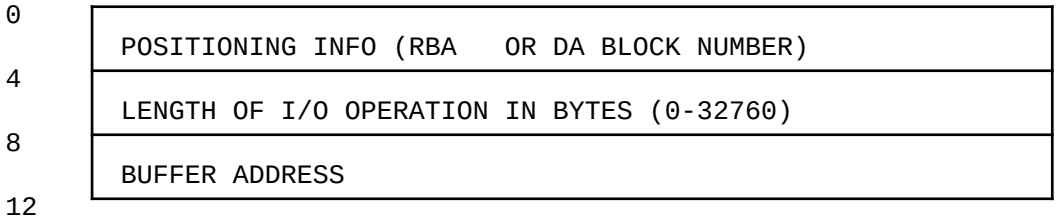
GENERAL CONTROL FLAGS

```
          EQU   B'10000000'      FILE IS IN COMM INDEX
          EQU   B'01000000'      MAINTAIN USAGE COUNTS IN DIRECT
          EQU   B'00100000'      FILE IS A VSAM FILE

xxxxGCTL  DC    X'00'            GENERAL CONTROL FLAGS
```

ACCESS CONTROL BITS (USED FOR NEXT 3 BYTES)

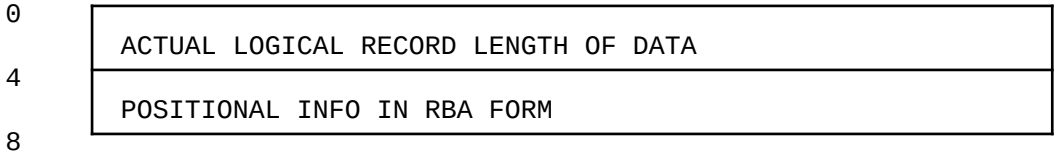
```
      NOTES:
         -BITS NORD+NOWR MEANS NO ACCESS
         -IF XO TO OWNER, NO ATTRIBUTES CAN BE CHANGED
         -ONLY THE OWNER (OR A PRIVILEGED USER) CAN REPLACE OR
          DELETE THE FILE

 ****

          EQU   B'10000000'        NO KIND OF READ ACCESS ALLOWED
          EQU   B'01000000'        NO KIND OF WRITE ACCESS ALLOWED
          EQU   B'00100000'        ONLY RD ACCESS IS EXEC-ONLY
          EQU   B'00010000'        ONLY WR ACCESS IS APPEND

xxxxACOW  DC    X'00'              ACCESS CONTROL FOR OWNERS
xxxxACNB  DC    X'00'              ACCESS CONTROL FOR NOBODIES
xxxxACPJ  DC    X'00'              ACCESS CONTROL FOR PROJECT MEMBERS
```

ARG---Read/Write Arg



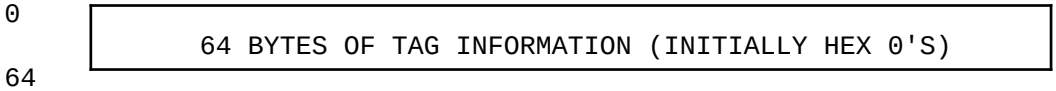

```
Label     DS    0F
 **THE FOLLOWING WORD IS USED TO PASS POSITIONING INFO
xxxxSBNU  DS    0F                 STARTING BLK NUMBER (UIO ONLY)
xxxxIRBA  DS    0F                 RBA FOR RWB REQUESTS ONLY
          DC    A(0)               POSITIONING INFO (SEE ABOVE)

xxxxRLEN  DC    A(0)               LENGTH OF REQ (BYTES)
xxxxRBUF  DC    A(0)               LOCATION OF BUFFER
```

PHYS---Returned Physical Info



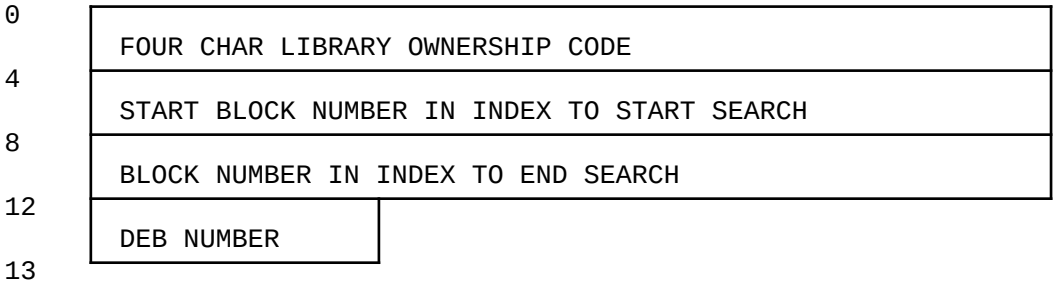
```
Label     DS    0F
 *THESE ARGS ARE VALID ONLY FOR BUFFERED READ/WRITE REQUESTS
xxxxARSZ  DC    A(0)               ACTUAL LOGICAL RCD LEN ON DISK
                                  (IE LENGTH OF VAR RECORD BUT NOT
                                  NUM OF BYTES IN COMPRESSED RCD)
xxxxORBA  DC    A(0)               RBA OF START OF THIS RECORD
```

TAG---Tag Information



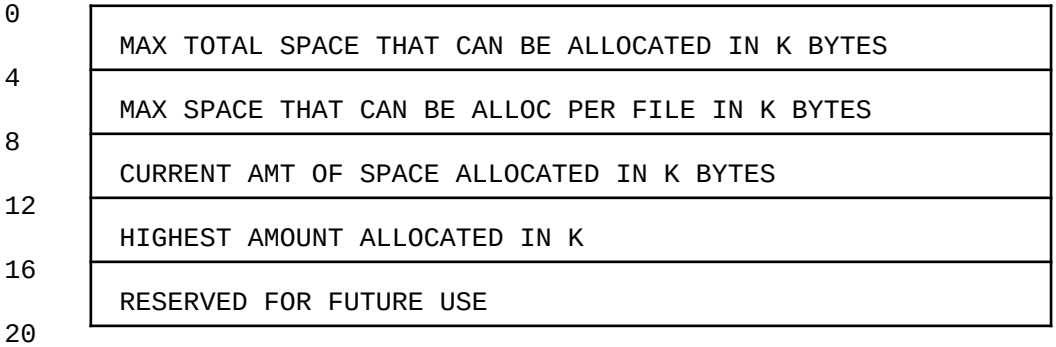

```
Label     DC    XL64'00'           TAG FIELD
```

LIBR---Arg for LIBRARY Req



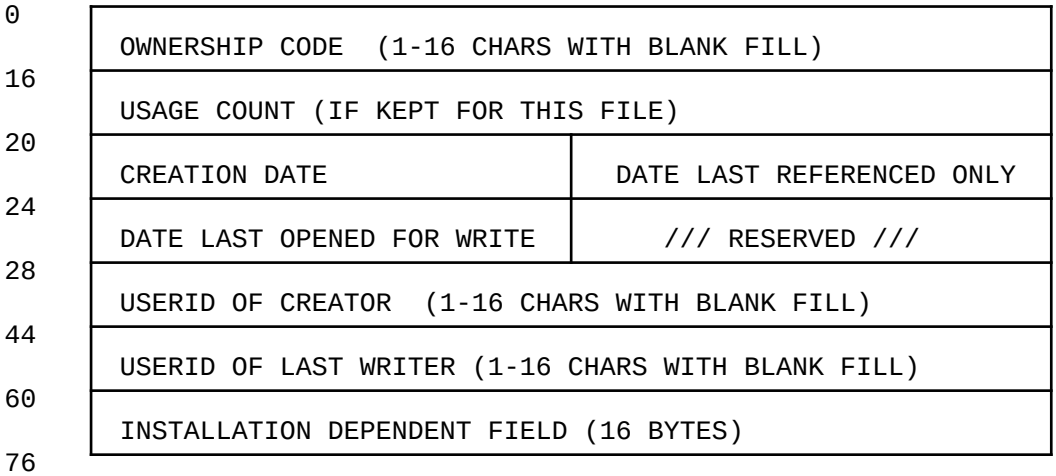

```
Label     DS    0F
xxxxLBCD  DC    CL4'   '           OWNERSHIP ID FOR LIBR OPERATION
xxxxLBSB  DC    A(0)               START BLOCK NUMBER OF SEGMENT
xxxxLBEB  DC    A(0)               ENDING BLOCK NUMBER OF SEGMENT
xxxxLBDB  DC    AL1(0)             DEB NUM OF INDEX DATA SET
```

UCTL---User Control Record Arg



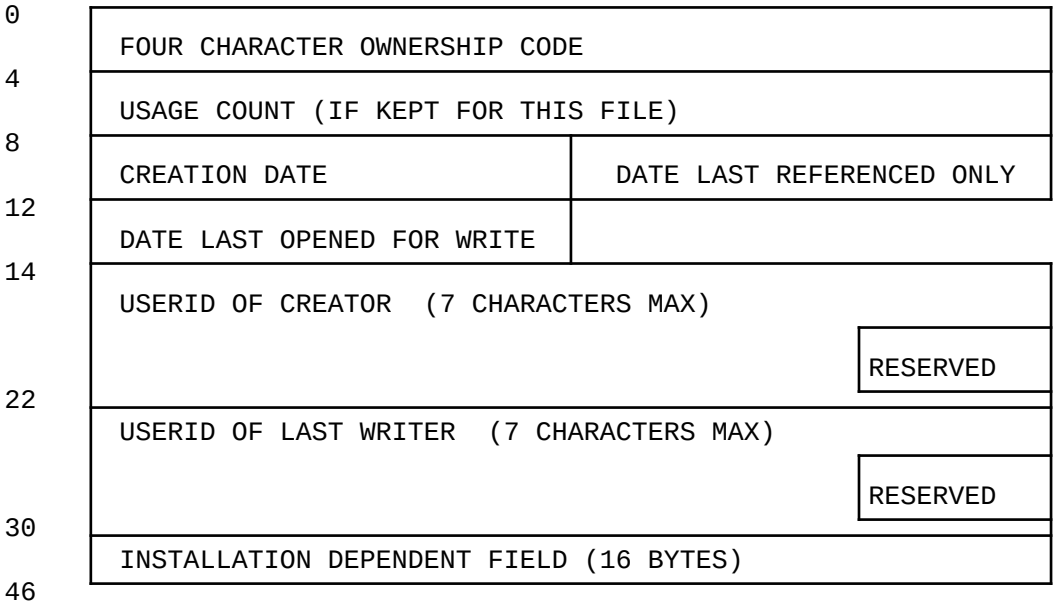
```
Label     DS    0F                **ALL UNITS ARE K BYTES (K=1024)
xxxxMAXS  DC    A(0)              MAX SPACE THAN CAN BE ALLOC TOTAL
xxxxMAXF  DC    A(0)              MAX SPACE PER FILE

xxxxACUR  DC    A(0)              AMT CURRENT ALLOCATED
xxxxAHWM  DC    A(0)              HIGHEST AMOUNT ALLOCATED
          DC    A(0)              RESERVED FOR FUTURE USE
```

HINFO--Usage Information from File's Header for 16 char userids



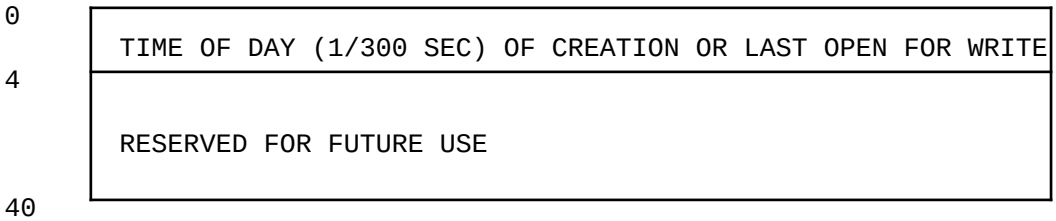

```
 *NOTE
 *This argument replaces UINFO used before ownership ids
 *could be longer than 4 characters.
 *The only case to use UNIFO is for EXTRACT requests that
 *use the header location on input.

 Label    DS    0F
 *NOTE:THE OWNERSHIP AND USERIDS ARE 1 TO 16 BYTES
 *       LONG WITH BLANK FILL.
 xxxxHIFC DC    CL16' '           OWNERSHIP ID OF CREATOR
 xxxxHIUC DC    F'0'              FILE USAGE COUNT (IF KEPT)

 **DATE INFORMATION.
 *   DATE NUMBER IS DAYNUM+(YEAR-1970)*366
 xxxxHICD DC    AL2(0)            DATE THIS DIRECTORY ITEM WAS CREATED
 xxxxHIRD DC    AL2(0)            DATE FILE WAS OPENED ONLY FOR READING
 *                                (MAY BE =0)
 xxxxHIMD DC    AL2(0)            DATE FILE WAS OPENED FOR WRITE
          DC    AL2(0)            RESERVED

 xxxxHICI DC    CL16' '           USERID OF CREATOR
 xxxxHIWI DC    CL16' '           USERID OF LAST WRITER

 xxxxHIID DC    XL16'00'          INSTALLATION DEPENDENT FIELD
```

UINFO--Usage Information from File's Header



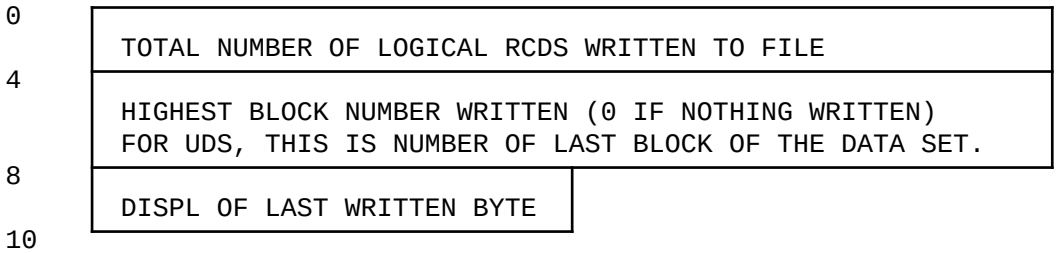

```
Label     DS    0F
 *NOTE THE FIRST WORD IS USED TO PASS THE HEADER LOC ON
 *CERTAIN TYPES OF EXTRACT REQUESTS.
xxxxUIFC  DC    CL4' '           OWNERSHIP ID PART OF FILE NAME.
xxxxUIUC  DC    F'0'             FILE USAGE COUNT (IF KEPT)

 **DATE INFORMATION.
    DATE NUMBER IS DAYNUM+(YEAR-1970)*366
xxxxUICD  DC    AL2(0)           DATE FILE WAS CREATED
xxxxUIRD  DC    AL2(0)           DATE FILE WAS OPENED ONLY FOR READING
                                 (MAY BE 0)
xxxxUIMD  DC    AL2(0)           DATE FILE WAS OPENED FOR WRITE

xxxxUICI  DC    CL7' '           USERID OF FILE'S CREATOR
          DC    C' '             RESERVED
xxxxUIWI  DC    CL7' '           USERID OF LAST WRITER
          DC    C' '             RESERVED

xxxxUIID  DC    XL16'00'         INSTALLATION DEPENDENT FIELD
```

XINFO--Extra Usage Information from File's Header



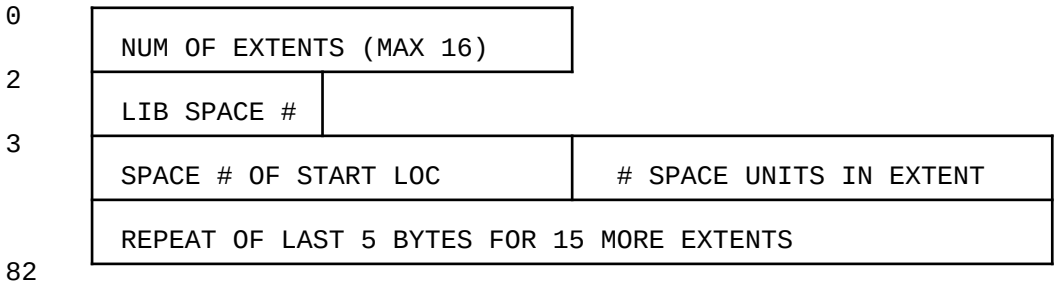
```
Label     DS    0F
xxxxXITD  DC    F'0'          TIME OF DAY (1/300 SEC) OF CREATION
                              OR LAST OPEN FOR WRITE (OR 0 IF BEFORE
                              TOD SUPPORT BEGAN)
xxxxXIRS  DC    9F'0'         RESERVED FOR FUTURE USE
```

EOFPT--Pointers to End of File



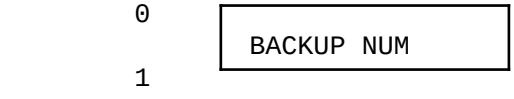

```
Label     DS    0F

      THE FOLLOWING IS THE NUMBER OF LOGICAL RECORDS WRITTEN TO THE
      FILE.
      THIS NUMBER MAY BE 0 EVEN IF THE FILE CONTAINS INFORMATION AS
      THIS FIELD CANNOT BE CORRECTLY MAINTAINED IN ALL CASES.

xxxxNLRC  DC    A(0)

     THE FOLLOWING IS THE 512-BYTE BLOCK NUMBER LAST WRITTEN.
     IF NUMBER IS 0, THEN THE FILE IS EMPTY.

xxxxEOFB  DC    A(0)

     THE FOLLOWING IS DISPLACEMENT INTO THE BLOCK OF THE 1ST BYTE
     BEYOND THAT LAST WRITTEN.
     FOR EXAMPLE, =512 IF BLOCK ALL WRITTEN.
     (CAUTION--THIS NUMBER MAY NOT BE 0 FOR AN EMPTY FILE.)
xxxxEOFD  DC    AL2(0)
```

EXTNTS--Extent List of File



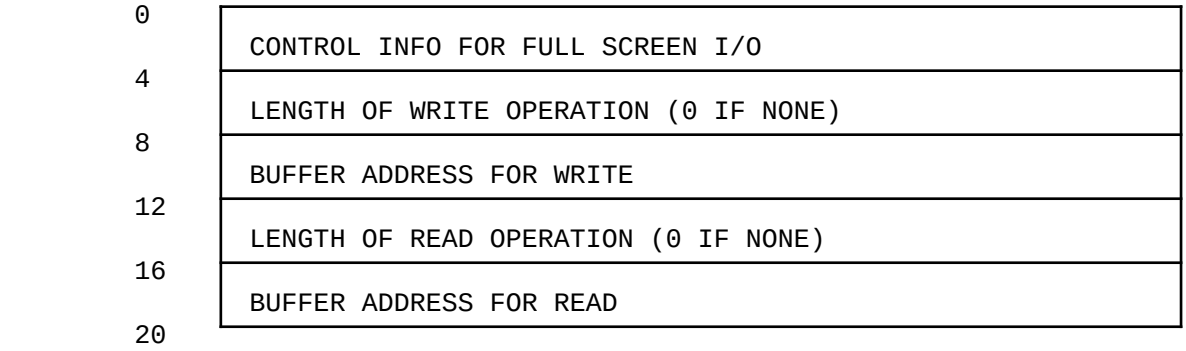
```
Label     DS    0H
xxxxEXTC  DC    AL2(0)           COUNT OF NUMBER OF EXTENTS (MAX 16)

 ** EXTENT INFO FOLLOWS.
       PER EXTENT: FIRST BYTE IS LIB SPACE NUMBER WHERE EXTENT IS
                   LOCATED (1,2,...)
                   NEXT HALF WORD IS STARTING SPACE UNIT NUMBER OF
                   EXTENT (1,2,...)
                   NEXT HALF WORD IS NUMBER OF SPACE UNITS IN EXTENT.
xxxxEXTI  DC    16XL5'00'        EXTENT INFO
```

BUPNUM--Backup Tape Number




```
Label     DC    AL1(0)             BACKUP TAPE NUMBER
```

FSARG--FSIO ARG




```
    Label     DS    0F
    xxxxFSFG  DS    XL4'00'        FLAGS TO CONTROL FULL SCREEN I/O
    xxxxFSWL  DC    A(0)           WRITE LENGTH
    xxxxFSWB  DC    A(0)           WRITE BUFFER LOCATION
    xxxxFSRL  DC    A(0)           READ LENGTH
    xxxxFSRB  DC    A(0)           READ BUFFER LOCATION
```

The control flag definitions are described in the FSIO section of Chapter 22.

Internals - Save Library

Tree-Structured Hierarchical Processing

By default, the current directory is set to the root directory when the user signs on to MUSIC. The character string associated with the current directory is stored in the XTCB control block. It can be set and queried by the DIRSRV service request of MFIO or by the CD command. If a user issues the "CD \ABC\DEF" command, the field in the XTCB will be set to:

```
AL1(8),C'ABC\DEF\'
```

A count of hex 0 would indicate the root directory.

Directory names are stored in the file system like ordinary user files. Their name always ends in a backslash (\) character. Suppose a user creates a directory using the "MD SAMPLE" command. This will cause the file system to create a file called "SAMPLE\". This file is stored like any other file that a user may own. The contents of the file are not used. This is very different to many other systems which actually store the names of all the files that are part of a directory with a directory file. The main use of the directory file on MUSIC is to establish that the user has issued a MD command.

You cannot normally create or delete a directory file. That should be done with the MD or RD commands or by the equivalent DIRSRV MFIO request. The MUSIC backup and restore utilities can directly create and delete directory names. This is done through the use of the DIRNOK special request on some MFIO calls.

Suppose a user creates a file called TEST after he has set the current directory to "SAMPLE". The file will be stored under the name of "SAMPLE\TEST". This allows for direct access to all files on the system without having to read any directory files first.

Data Set Overview

The Save Library consists of a number of data sets. All these data sets have a blocksize of 512 bytes. There is one index data set that contains the names of all the files. The remaining data sets contain the actual data. The Save Library can be expanded by adding more of these data sets, however, there is only one index data set.

Index Data Set

The index data set contains the names of all files and a pointer to the start of each one. It also contains the UCR records. The first few blocks in this data set contain the system control data that is used at IPL time. Such information includes hashing numbers and a field that contains the number of data spaces. The remainder of the index data set is split into several sectors. All the file names belonging to any one given user will be found entirely within one sector. Thus, the LIBRARY command need only search one sector to find the names of all the files stored under one code. There is an overflow sector in the index data set that contains records which could not fit within the proper sector because there was no space in the specific record that was pointed to by the hashing process.

In order to locate a specific file the system first takes the code part of the file name and determines what sector it maps to. It then takes the file name, and using the hashing process, locates the specific disk block within that sector that would contain the specified file name. Upon reading the block, the system can determine whether the file does in fact exist. (The hashing scheme is set so that if multiple users who are sharing the same index sector use the same file name, it will probably map to different index blocks.)

To locate an item in the common library, the system considers all the primary sectors as being one large sector and hashes the file name to locate a specific index block.

Space Data Set

Each space data set starts off with several blocks that contain control information. One of the blocks contains a bitmap which indicates which space units are used within this data set.

The remainder of this data set is used to contain users' data. The first block of any user file is reserved for system use. This block contains the file header. In the file header are the various attributes and

characteristics of the user's file. (The header block used to be called the directory block. It was renamed to avoid confusion with directories that the user can create.)

Bitmap

The bitmap is stored in up to 7 512-byte blocks that start at block 2. The first four bytes contain a check sum. This field is used to check the validity of the map and is calculated by an arithmetic process involving the addition of the remaining words in the map. (Consult the system code in module MFIO for a complete description of how this is calculated.)

The check sum is followed by a two byte bit count containing the number of free space units in this data set.

The bit count field is followed by a bitmap. The bitmap shows exactly which units are free. Each bit corresponds in order with the space unit number available. Thus the bit after the counter is the status of space unit one, the next, space unit two. If the bit is on, then the space unit is free and hence, can be allocated.

File Header

The first block of each file contains its header. This header contains such things as the access control information, the ownership id (userid without subcode), usage dates and the file's extent information.

ULCBs

A set of User Library Control Blocks (ULCB) exist in the service area of the user region. These control blocks contain information about open files. They are contained in the system module MFIOCB.

Buffer Pool

A buffer pool exists for the use of Save Library I/O. This pool is located in the system module MFIOCB. Each buffer in the pool can be assigned to any file. They are assigned dynamically as required using a scheme similar to a paging scheme.

Logical Record Formats

The following describes how the logical records are stored on the Save Library's 512-byte physical blocks.

Fixed Format (F)

With this format, the first logical record contained in any disk block always starts at the first byte of the disk block. When the record size is not greater than 512, then no logical record spans a disk block. Unused space may therefore exist at the end of each block.

When the logical record size is greater than 512, each logical record starts at the beginning of a disk block. Overflow from one block is continued at the first byte of the next block. Unused space may exist at the end of the last block used for each record.

Variable Format (V)

The first two (2) bytes of each disk block are used to contain the displacement to the start of the first logical record in that block. The remaining 510 bytes are always used. Logical records exceeding one disk block will continue at the third byte of the following block. Each logical record is preceded by a 2 byte count containing the number of bytes in the record plus two.

Compressed Format (FC and VC)

The only difference between FC and VC is the meaning of the file's record size. For FC files, the record size means that all records will appear to be this size. Trailing blanks will always be removed from FC files before compression and supplied on expansion. For VC files, the record size means the length of the largest record in the file so far.

The first two (2) bytes of each disk block is used to contain the displacement to the start of the first logical record in that block. If no logical record starts in this block, the first two bytes are zero. The remaining 510 bytes are always used. Logical records exceeding one disk block will continue at the third byte of the following block. Each logical record may be composed of a number of segments. Each starts with a 1 byte descriptor. A repeated text segment is used when four or more bytes in sequence are the same. One byte segments are also represented by the repeated text format. The three types of segments are:

1. *Non-repeated text*. Format: D string
   The D byte is the length of the string minus one. D may range from X'01' to X'7F' representing string lengths of 2 through 128 bytes. D may not be zero.

2. *Repeated text*. Format: D C
   The D byte is the number of times to repeat the character C plus X'7F'. D may range between X'80' and X'FE' representing a repeat count of 1 to 127.

3. *Record end*. Format: X'FF'
   This segment indicates the logical end of the record.

Working with Files

- Reference to another user's private file is done by prefixing the ownership id to the file name. The ownership id is usually the same as the userid. The only exception is when subcodes are used. Thus ABCD:HISFILE refers to the file HISFILE belonging to userid ABCD. This ownership id prefix is accepted in EDIT, RENAME, SAVE, PURGE and other similar commands.

- To change ownership of files you can use the RENAME command. To change a long list of files, use the FILECH utility. The COPYCOL command of the Editor is useful in the preparation of the input to this program.

- Use the TAG command of the Editor to help keep track of the contents of files. This is particularly useful when you own a large number of files.

- A COM option can be specified on the Editor FILE and SAVE commands. This has the effect of forcing the file name to be placed in the common index. A private file in the common index can be accessed only by its owner or by a privileged user. This feature can be used to reserve names in the common index, or to share files between privileged users without the need for public files.

- The "LIBRARY abcd:*" command lists all the files belonging to the ownership id ABCD. The

"LIBRARY ab*:*" command lists all the files belonging to the set of ownership ids that start with the letters AB.

You can select only certain names within a ownership id. The "LIBRARY *:@learn" command lists all the files called @LEARN belonging to anyone.

- Use "LIBRARY *:* COM" to list all files in the common index. These files were saved with the COM or PUBL option.
- Use the ATTRIB utility to find the owner of a public file. Thus "ATTRIB xxxx" displays who owns the file *xxxx.*
- The FPRINT utility is useful to list a large number of files with page headings.
- To purge a number of files, first create a file that contains a list of the file names. Then use the "PURGE <list" command.
- Even privileged users will be unable to store files under unused userids. This is because a UCR record for the userid must exist before files can be stored with that userid. UCR records are automatically added by CODUPD when userids are added. You can use the UCR utility to add UCR records directly without requiring a corresponding sign-on userid.
- The following technique can be used to activate the usage counting of a file. Edit the file and type the editor command FILE * CNT.
- The FIXINDEX and MFHASH utilities are useful for direct inspection and alteration of the Save Library Index should that be required.
- Note that when you set up an execute-only file, you should put in a /SYS NOPRINT statement in the file so that the system will know not to print any /FILE statements that follow.
- You can specify a ownership id on the CD command by using the following format:

```
CD \ownerid:\dir
```

Working with UDS Files

This topic contains tips and techniques for working with User Data Set (UDS) files. These are OS style data sets that are supported by MUSIC. Each file has its own VTOC entry and may occupy from one to five extents. Despite the name, most users' data is stored in the Save Library and UDS files are only kept around for compatibility with past releases. An entry in the user's profile sets the maximum number of tracks that can be used for a UDS file.

- The LISTV command of the DSKDMP utility is used to list the contents of the VTOC. This gives a comprehensive list of all UDS (and SDS) files on any disk. It is also useful in determining the amount of free space on the disk.
- The EDITOR can edit UDS files. Define the UDS file as logical unit 4. Remember to specify the OLD option on the /FILE statement if you want to file the changes.
- When UDS files are deleted, their contents are not erased. Therefore, subsequent users who allocate UDS files may be able to read the information stored there by the previous user of that disk space. (This is not the case with regular files.) The ZERO.FILE utility described in the *MUSIC/SP User's Reference*

Guide, can be used to clear the contents of a file that contains sensitive information.

- To be able to list the UDS files owned by specific users, first create a file SYSDSL which is the same as the file DSLIST with the last line removed. Then run the following job:

```
/INC SYSDSL
VOL='volume1,'volume2',DSNS='userid*'
```

where *volume1*,*volume2* are the names of the UDS volumes to scan and *userid* is the file ownership id of the files that you want to list. The DSNS parameter can specify a data set name pattern, containing wild characters * and ?.

Chapter 21. Load Library and Link Pack Area

Load Library

The system load library is in the dataset SYS1.MUSIC.LOADLIB. Private load libraries can be created in UDS files. These libraries contains relocatable machine language programs that are loaded into memory using the SVC $LODSVC. The OS LOAD and LINK macro instruction can also be used to load programs from these libraries. The system load library contains the modules for system utilities, the editor, the assembler, compilers, system interfaces and commands. Private libraries are used when specific applications require large program libraries. A example of this is the PL/I transient library.

The LDLIBE utility program is used to create and maintain load libraries, LDLIST lists information about the contents of load libraries, and SYSREP can be used to make machine language modifications (ZAPs or REPs) to the programs in a load library.

System performance can be significantly improved by placing programs from the system load library in either the Fixed or Pageable Link Pack Areas. In addition to the loading overhead being reduced, this also reduces system paging and swapping loads.

Modules in the Fixed Link Pack Area (FLPA) are loaded into memory during system initialization. If the module is reentrant it can be executed directly from the FLPA. The pages of the module are never involved in paging or swapping. If the module is not reentrant, it is copied to the user region before execution. This memory to memory load is much less overhead than loading from the load library on disk.

Modules in the Pageable Link Pack Area are copied from the load library to the system's page data sets during system initialization. To be eligible for the PLPA a program must be reentrant. When a PLPA module is loaded, the system simply adjusts the user's page tables to point to the area of the page data set that contains the module. The pages are brought in to memory as required by demand paging. This greatly reduces loading overhead. Paging overhead is also reduced since the PLPA pages never have to be paged out.

Entries in the system catalog determine which modules are placed in the FLPA and the PLPA.

Load Library Member Formation Procedures

This section contains the job control statements used for creating the modules in the system Load Library. Most of the members are formed by running two jobs. The first job uses the Linkage Editor (/LOAD LKED) to create a load module file, from the object decks as input. The second job uses LDLIBE to copy the member from the load module file to the Load Library. Refer to the LDLIBE utility for an explanation of the options.

The LPA copies and main storage directory entries for these members will not be updated until the next time MUSIC is IPLed. Some of the jobs that follow use the REPL=T option. Using REPL=T means that an attempt will be made to use the same space occupied by the previous version. It may be advisable to rename old members and then use REPL=F if you are not sure the new module will work or if you do not plan to re-IPL MUSIC right away.

$CTL Module Formation

Refer to files $SYS:CTL.LKED and $SYS:CTL.APPLY for the two jobs required.

$FNPAK1 Re-entrant Subroutine Package

Refer to files $SUB:$FNPAK1.LKED and $SUB:$FNPAK1.APPLY for the jobs required.

$INP Module Formation (/INPUT Command)

Refer to files $SYS:INPUT.LKED and $SYS:INPUT.APPLY for the two jobs required.

$LST Module Formation (/LIST Command)

Refer to files $SYS:LIST.LKED and $SYS:LIST.APPLY for the two jobs required.

$OSTRAP Module Formation

Refer to files $CMP:OSTRAP.LKED and $CMP:OSTRAP.APPLY for the two jobs required.

$PRE Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ORG 1000
   ENTRY PRE
<---------object for PRE. (Source stored under code $SYS)
   NAME $PRE

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='$PRE',RENT=F,REPL=T
```

$PST Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ORG 1000
   ENTRY PST
<---------object for PST. (Source stored under code $SYS)
   NAME $PST

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='$PST',RENT=F,REPL=T
```

$PUR Module Formation (/PURGE Command)

Refer to files $SYS:PURGE.LKED and $SYS:PURGE.APPLY for the two jobs required.

$REN Module Formation (/RENAME Command)

Refer to files $SYS:RENAME.LKED and $SYS:RENAME.APPLY for the two jobs required.

$REX Module Formation

Refer to files $REX:REXX.LKED and $REX:REXX.APPLY for the two jobs required.

$ROUTING Module Formation (Route Name Table)

Refer to file $PGM:$ROUTING.S for instructions.

$SAV Module Formation (/SAVE Command)

Refer to files $SYS:SAVE.LKED and $SYS:SAVE.APPLY for the two jobs required.

$SIGNON Module Formation

Refer to files $SYS:SIGNON.LKED and $SYS:SIGNON.APPLY for the two jobs required.

$UPD Module Formation (/UPDATE Command)

Refer to files $SYS:UPDATE.LKED and $SYS:UPDATE.APPLY for the two jobs required.

$XMON Module Formation

Refer to files $LKD:XMON.LKED and $LKD:XMON.APPLY for the two jobs required.

APL Module Formation

See file $APM:MUSAPL.LKED for the link edit job. See file $APM:MUSAPL.PUTSYS for the LDLIBE job.

APLVSXEC Module Formation

ASM Module Formation

See files $CMP:ASM.LKED and $CMP:ASM.APPLY for the two jobs required.

ASMLG Module Formation

See files $CMP:ASMLG.LKED and $CMP:ASMLG.APPLY for the two jobs required.

BLKLET Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ORG 0000
<---------object for BLKLET. (Source stored under code $CMP)
   NAME BLKLET

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='BLKLET',RENT=F,REPL=T
```

CBMANIP Module Formation

Refer to files $VSM:CBMANIP.LKED and $VSM:CBMANIP.APPLY.

CMPMON Module Formation

Refer to files $SYS:CMPMON.LKED and $SYS:CMPMON.APPLY for the two jobs required.

COBOL Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ORG 4600
<---------object for COBOL. (Source stored under code $CMP)
   NAME COBOL

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='COBOL',RENT=F,REPL=T
```

DICT1 Module Formation (Spelling Dictionary)

Refer to *Chapter 12 - TODO Facilities* under the topic "Installing a System Word Dictionary in the PLPA".

DMSREX Module Formation

Refer to files $REX:DMSREX.LKED and $REX:DMSREX.LD for the two jobs required.

EDIT Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,PRINT,STATS,NOSEGTAB,MODE=OS
.ORG 1000
<---------object for EDIT. (Source stored under code $EDT)
 NAME EDIT

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='EDIT',RENT=F,REPL=T
```

EDITOR Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,PRINT,STATS,NOSEGTAB,MODE=OS
.ORG 1400
<---------object for EDITIO. (Source stored under code $EDT)
<---------object for EDITOR. (Source stored under code $EDT)
<---------object for EDTFIX. (Source stored under code $EDT)
<---------object for TSUSER. (Source stored under code $EDT)
<---------object for SUBMIT. (Source stored under code $PGM)
<---------object for PRINT.  (Source stored under code $PGM)
<---------object for QSCAN.  (Source stored under code $PGM)
<---------object for DUMMY.  (Source stored under code $EDT)
   NAME EDITOR

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='EDITOR',RENT=T,REPL=T
```

EDTDSP Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,PRINT,STATS,NOSEGTAB,MODE=OS
.ORG 0000
<---------object for EDTDSP. (Source stored under code $EDT)
 NAME EDTDSP

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='EDTDSP',RENT=T,REPL=T
```

EXEC Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ORG 1000
<---------object for EXEC. (Source stored under code $LKD)
   NAME EXEC

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='EXEC',RENT=F,REPL=T
```

FORTG1 Module Formation

See file $FG1:FG1.GEN for generation procedure.

IBCOM Module Formation

See file $LM1:LM1.GEN (Fortran Mod I library) or $LM3:LM3.GEN (Fortran Mod II library) for generation procedure.

IEFBR Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ORG 0000
<---------object for IEFBR. (Source stored under code $CMP)
   NAME IEFBR

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='IEFBR',RENT=T,REPL=T
```

IFOXnn Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
/INCLUDE $CMP:VSASM.LKED

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='IFOX00',RENT=F,REPL=T
NAME='IFOX01',RENT=F,REPL=T
NAME='IFOX02',RENT=F,REPL=T
NAME='IFOX03',RENT=F,REPL=T
NAME='IFOX04',RENT=F,REPL=T
NAME='IFOX05',RENT=F,REPL=T
NAME='IFOX06',RENT=F,REPL=T
NAME='IFOX07',RENT=F,REPL=T
NAME='IFOX11',RENT=F,REPL=T
NAME='IFOX21',RENT=F,REPL=T
NAME='IFOX31',RENT=F,REPL=T
NAME='IFOX41',RENT=F,REPL=T
NAME='IFOX42',RENT=F,REPL=T
NAME='IFOX51',RENT=F,REPL=T
NAME='IFOX61',RENT=F,REPL=T
NAME='IFOX62',RENT=F,REPL=T
```

IGIALL Module Formation:
IGIEXT Module Formation:
IGIGEN Module Formation:
IGIPAR Module Formation:
IGIUNF Module Formation:

See file $FG1:FG1.GEN for the generation procedure.

IISRENT Module Formation

See file $IIS:IIS.GEN for the generation procedure.

IKFCBLnn Module Formation

See file $GEN:COBOL4.GEN or $GEN.VSCOBOL.GEN for the generation procedure.

ILBOPRM0 Module Formation

See file $GEN:VSCOBOL.GEN for the generation procedure.

JOBONE Module Formation

Refer to files $SYS:JOBONE.LKED and $SYS:JOBONE.APPLY for the two jobs required.

LIBRARY Module Formation (/LIBRARY Command)

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ORG 1000
   ENTRY LIBCMD
<---------object for LIBCMD. (Source stored under code $SYS)
<---------object for MFINDX. (Source stored under code $SUB)
<---------object for MATCH. (Source stored under code $SUB)
<---------object for SSORT. (Source stored under code $SUB)
   NAME LIBRARY

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='LIBRARY',RENT=F,REPL=T
```

LKED Module Formation
LKEDPHS1 Module Formation:
LKEDPHS2 Module Formation:
LKEDPHS3 Module Formation:

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEARCH,MODE=OS    (note: option NOSEGTAB not used)
.ORG 0F80
   OVERLAY A
<---------object for SYSIO. (Source stored under code $LKD)
<---------object for LKDMON. (Source stored under code $LKD)
<---------object for SBLOPN. (Source stored under code $PGM)
   OVERLAY B
<---------object for PHASE1. (Source stored under code $LKD)
<---------object for PREP1. (Source stored under code $LKD)
   OVERLAY B
<---------object for PHASE2. (Source stored under code $LKD)
<---------object for PREP2. (Source stored under code $LKD)
   OVERLAY B
<---------object for PHASE3. (Source stored under code $LKD)
<---------object for PREP3. (Source stored under code $LKD)
   NAME LKED

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='LKED',SEG=2
NAME='LKED',RENAME='LKEDPHS1',SEG=3
NAME='LKED',RENAME='LKEDPHS2',SEG=4
NAME='LKED',RENAME='LKEDPHS3',SEG=5
```

LKEDEXEC Module Formation

See $LKD:LKEDEXEC.LKED and $LKD:LKEDEXEC.APPLY for the two jobs required.

LOADER Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ADD  004300    STRTUC
.ADD  001000    IBCOM#
.ADD  002394    FIOCS#
.ADD  002840    ADCON#
.ADD  003CE8    IHNERRM
.ORG  001000
<---------object for LLOADER. (Source stored under code $LDR)
   NAME LOADER

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='LOADER',RENT=F,REPL=T
```

OLOADER Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ADD  004800    STRTUC
.ORG  001000
<---------object for OLOADER. (Source stored under code $LDR)
   NAME OLOADER

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='OLOADER',RENT=F,REPL=T
```

PLI Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ORG 4600
<---------object for PLI. (Source stored under code $CMP)
   NAME PLI

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='PLI',RENT=F,REPL=T
```

QLOADER Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ADD  004300    STRTUC
.ADD  001000    IBCOM#
.ADD  002394    FIOCS#
.ADD  002840    ADCON#
.ADD  003CE8    IHNERRM
.ORG  001000
<---------object for QLOADER. (Source stored under code $LDR)
   NAME QLOADER

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='QLOADER',RENT=F,REPL=T
```

RLOADER Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
<---------object for RLOADER. (Source stored under code $LDR)
<---------object for SBLOPN. (Source stored under code $PGM)
   NAME RLOADER

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='RLOADER',RENT=T,REPL=T
```

SORT Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,MODE=OS   (note: NOSEARCH is not used)
.ORG 0000
<---------object for SRTMRG. (Source stored under code $CMP)
   NAME SORT

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='SORT',RENT=F,REPL=T
```

VSAM Module Formation

Refer to files $VSM:VSAM.LKED and $VSM:VSAM.APPLY.

VSAPL Module Formation

See file $APV:VSAPL.LOAD.LKED for the link edit job. See file $APV:VSAPL.LOAD.PUTSYS for the LDLIBE job.

VSBASCMP Module Formation

See file $BAV:VSBASIC.GEN for the generation procedure.

VSBASIC Module Formation

```
/FILE LMOD NAME(xxx) NEW(REPL)
/LOAD LKED
/JOB MAP,NOGO,NOSEGTAB,NOSEARCH,MODE=OS
.ADD  005000    CMPADR
.ORG 1000
   ENTRY SYSIO
<---------object for SYSIO. (Source stored under code $BAV)
<---------object for VSBMON. (Source stored under code $BAV)
   NAME VSBASIC

/FILE 1 N(xxx)
/INCLUDE LDLIBE
LIBE='SYST',IN=1
NAME='VSBASIC',RENT=F,REPL=T
```

VSBASLIB Module Formation

See file $BAV:VSBASIC.GEN for the generation procedure.

Link Pack Area Considerations

The Link Pack Areas are used to hold resident copies of transient system and compiler modules. These modules are read in from the system data set SYS1.MUSIC.LOADLIB.

The Fixed Link Pack Area (FLPA) and Pageable Link Pack Area (PLPA) are loaded at during system initialization based on the names given on RESPGM statements in the system catalog. These copies are subsequently used by the system avoiding the I/O overhead involved in loading them each time they are required. The LPA program can be used to display the current contents of the FLPA and PLPA.

FLPA modules are loaded into real memory and remain resident there. There is obviously not enough room in real memory to load all load library modules. High usage modules are the best candidates for the FLPA. By default, a number of system utility modules such as SIGNON and EDITOR are put in the FLPA. If an FLPA module is reentrant, then it can be used directly by the user program. This not only reduces the load

time but also cuts down on the swap and paging load. If it is not reentrant, a copy is made in the user region when requested.

PLPA modules are placed on the system's paging datasets during the initialization process. There is no real storage penalty for putting a module in the PLPA, but the module must be reentrant to be eligible. When a PLPA module is requested the system only need adjust the page table of that user. Any required page is then read in as required using the system page mechanism.

The utility LDCNTS can be used to determine how often a module is used. Most installations will find that the following are the most commonly used and therefore should be in the FLPA: $CTL, $PRE, $PST, $LST, EDIT, EDTDSP, EDITOR, and $SIGNON. Other good candidates are $OSTRAP, $XMON, EXEC, IBCOM, $REX, and DMSREX.

The following members must be in either the FLPA or PLPA: RLOADER, EDITOR, EDTDSP.

The member $ROUTING must always be in the FLPA. Even though it is reentrant it will not work in the PLPA as some nucleus routines refer to it.

Keeping VSBASCMP and VSBASLIB routines resident means that about 50K larger VS BASIC programs can be handled in the same size user region.

Keeping all C/370 modules resident has shown a marked reduction in compile times. The routine names are: EDCP, EDCO, EDCT, EDCXV, EDCX24, and IBMBLIIA.

The current sizes of modules on your system can be listed by running the LDLIST program. Non-reentrant modules in the FLPA will typically use 15% more space than indicated here due to space needed to hold the relocation information. The items marked RENT are reentrant modules and hence are suitable for the PLPA as well as the FLPA.

Chapter 22. System Programming

Overview

This chapter contains information of interest to system programmers who want to add new functions or make modifications to MUSIC/SP. Before undertaking a programming project there are some basics you should know.

The source for MUSIC is available on the source tape. This source is not only useful when you want to make modifications, but contains many valuable comments and programming examples. Some features of MUSIC internals that are not documented in manuals are described in detailed comments in the source modules themselves.

There are a number of choices available when adding a new program or command. The command could be written in REXX and interpreted directly from the REXX source. It could be written in a high level language or assembler and run from a load module or object deck. A load module could be run from the system Load Library or the LPA. A new command could be added to the system nucleus itself. The method chosen depends of the type of application and required performance characteristics.

You should keep track of any local modifications or additions so they don't get forgotten when applying maintenance or going to another release of the system. Also keep your own source separate from the base system source.

Naming Conventions For System Files

MUSIC system files follow a set of naming conventions based on the suffix used. The most common suffixes used are the following:

Source files end in .S

Object decks end in .OBJ

Files containing linkage editor control cards end in .LKED

Files containing load modules end in .LMOD

Files containing program data end in .DATA

Files containing macro definitions end in .M

Usage of $ Userids on MUSIC

All system files are stored under an ownership userid that starts with a dollar ($) sign. Files associated with different system functions are stored under different userids as follows. (For the latest table of $ userids see the file CATALOG.)

$ACT Contains the files associated with the session accounting program.

$ADM	Files associated with the ADMIN program for the system administrator.
$ADS	Files for the site's Ads Facility. These files are not distributed with MUSIC. They are created by the site.
$ADZ	Data files for the SAMPLE ADS (classified ads) Facility.
$APM	Contains the files associated with MUSIC/APL.
$APV	Contains the files associated with VS APL.
$AXF	Contains the files associated with VS Assembler (XF).
$BAV	Contains the files associated with the VS Basic compiler.
$BBS	The BBS utility program, which is used for HELP, ADS, CWIS, and other applications. Files with names starting $BBS:@CIO are a SAMPLE CWIS.
$BMK	Benchmark programs, for comparing and measuring machine and system speeds.
$CIC	Contains the files associated with IBM CICS.
$CMP	Contains the files associated with OS simulation and the VS Assembler.
$COD	Contains files associated with sign-on code utilities.
$CON	Contains the files associated with the CONF and CONFMAN utilities.
$CSL	Contains the Contributed Software Library.
$DEM	Contains the files containing demonstration programs and games.
$DW2	Contains the files associated with IBM DisplayWrite/370.
$EDT	Contains the files associated with the Editor.
$EMD	Contains files associated with the Mail facility operation.
$EML	Contains the source for the MAIL facility.
$EMM	Userid owning the mailbox containing the site's mail profile defaults that are copied to new mailboxes at creation time.
$EMP	Contains the files associated with the postmaster for the MAIL facility.
$EMS	Electronic Marks Submission facility.
$EMn	$EM1,$EM2,... Contain the outgoing mailboxes for the RDMAILER tasks that process incoming mail.
$FG1	Contains the files associated with the FORTRAN IV (G1) compiler.
$GDM	Contains the files associated with IBM GDDM.
$GEN	Contains the files used by system generation functions.

$GPS	Contains files associated with the GPSS product.
$HLP	Contains the files used by the HELP commands. Files with prefix "@ED." are used with the Context Editor HELP facility. Files with prefix "@GO." are used with the *GO mode HELP facility. Files with the prefix "@FC." are used with the MUSIC/FCS facility.
$IBB	Contains the files associated with IBM Basic.
$IBC	C/370 and C/370 Version 2, Compiler and Library.
$IIS	Contains the files associated with the Interactive Instructional Presentation System (IIPS) and with the Interactive Instructional Authoring System (IIAS).
$IMD	Contains the files associated with IBM GDDM IMD.
$INT	Contains the files associated with the Full Screen Interface (FSI).
$ITS	Contains the files associated with the indexed text searching utilities.
$IVU	Contains the files associated with IBM GDDM IVU.
$JCL	Model job control card files for the SUBMIT program.
$KRM	KERMIT file transfer (part of the CSL - Contributed Software Library).
$LDR	Contains the MUSIC loaders.
$LKD	Contains the MUSIC Linkage Editor.
$LM1	Files to generate the FORTRAN Mod I Library.
$LM3	Files associated with the FORTRAN Mod II Library.
$MAN	Contains the files associated with the online manuals.
$MBx	$MBA,$MBB,... Contains mailbox files for the MAIL facility.
$MCC	Contains CMS macros. (CMS is a component of VM.) These macros are needed to assemble IIS source.
$MCM	Contains the MUSIC system macros. They are used when assembling system source.
$MCS	Client-server programs for interfacing with PC workstations.
$MCU	Contains the MUSIC user macros (mainly OS macros).
$MDxx	($MDAA,...$MDZZ, $MD00,...$MD99) Mail data files.
$MEM	Contains files associated with the MEMO utility.
$MON	Contains some auto sign-on programs.
$PAN	Contains the files associated with the 3270 PANEL subsystem.
$PCW	Files associated with the Personal Computer Workstation (PCWS).

$PGF	Contains the files associated with IBM GDDM PGF.
$PGM	Contains the files associated with miscellaneous main programs.
$PIP	Contains the files associated with the pipe cmd.
$PLI	PL/I Optimizing Compiler and Libraries.
$PL2	PL/I Version 2 Compiler, Library and Test Facility.
$PLK	Contains the files associated with IBM GDDM (PCLK feature).
$PRT	Print queue and print files.
$PUB	Default anonymous FTP code.
$RDR	Contains the batch jobs submitted to the MUSIC internal reader.
$REX	Contains the files for the REXX command executor.
$RG2	RPG/370 Compiler and Library (5688-127).
$RPG	Contains files associated with the RPG II compiler product.
$SCR	Contains the files associated with MUSIC/SCRIPT.
$SRV	Contains the files for applying service and installing optional products.
$STP	Files associated with STATPAK. (Not distributed with MUSIC/SP.)
$SUB	Contains the files associated with the subroutine library.
$SYS	Contains the MUSIC system source.
$TCP	Contains the files associated with TCP/IP.
$TDO	Contains the files associated with the Time, Office, and Documentation Organizer (TODO).
$TPC	Files associated with the PC co-axial connected file transfer program.
$TUT	The TUTORIALS facility: tutorials on programming languages such as Rexx, Fortran, etc.
$VCT	Vector Facility Simulator (5798-DWF).
$VC2	Contains the files associated with VS COBOL II.
$VF2	Contains the files for VS Fortran, Version 2.
$VMR	Contains the reader queue used by the VMREADX utility.
$VP2	Contains the files associated with VS PASCAL.
$VSC	Contains the files associated with VS COBOL.
$VSF	Contains the files associated with the VS FORTRAN compiler product.

$VSM	Files associated with the VSAM (Virtual Storage Access Method).
$VSP	Contains the files associated with the VS PASCAL compiler product.
$WWS	Sample data files for Web server.
$XXX	Contains miscellaneous dummy files.
$001	Contains workspaces distributed with VS APL for lib 1. (Not distributed with MUSIC.)
$002	Contains workspaces distributed with VS APL for lib 2. (Not distributed with MUSIC.)
$003	Contains workspaces distributed with MUSIC for VS APL lib 3.

Source Key File

The file SOURCE.KEY contains a list of all the system files. All of these files can be found on the MUSIC/SP distribution tapes. Not all of these files are included in the Save Library when you install MUSIC/SP. For example source code is not loaded onto your system to save disk space.

An * beside a file name means that the file is on the tape but is not restored during system generation.

The LIBRARY command can be used to obtain a list of the files currently on your system. For example, type the command "LIBRARY $PGM:*" to obtain a list of all the files under the ownership userid of $PGM.

Use the MFREST utility to restore selected files from the source tapes as required. Consult the file CATALOG for sample usage of MFREST. This file also contains information on how to locate which tape specific files are on.

Modifying MUSIC's Nucleus

You must recreate the system nucleus to change the I/O configuration, modify a nucleus module or add a new module to the nucleus. The ADMIN facility can be used to make routine configuration changes. You should understand what goes on behind the ADMIN menus however, if you want to actually change the nucleus code.

Change the Source

If required, restore the source for the existing module from the distribution tape using MFREST. Source for the nucleus modules is stored under the userid $SYS. Make your changes using the editor. Use the FLAG command of the editor to identify your changes. This is done by issuing the following Editor command: "FLAG DEL=*/,COL=73". File the changed version using a different name from the original.

Assemble the Module

The file SYSASM is setup to assemble system modules. The following JCL is used to assemble the system module XXXX and save the object deck in the file XXXX.OBJ.

```
/FILE SYSPUNCH NAME(XXXX.OBJ) NEW(REPL)
/INCLUDE SYSASM
/OPT DECK
=XXXX
```

Modify the NUCGEN Job Stream

This can be done using the ADMIN facility or by simply editing the file $GEN:NUCGEN.JOB. The NUCGEN utility reads the base system object decks from the file $GEN:NUCLEUS. It then reads the device control cards and replacement object decks from the input data stream. As output, the NUCGEN program produces a file which contains an IPLable loader, the object decks for the system modules including the replacements, and the new device control cards. The output can be written to a file or to tape. The tape option is useful in that it provides a good backup copy of the nucleus. This output file can optionally be used as the base system input file for a subsequent run of the NUCGEN program.

To replace a system module an /INCLUDE statement pointing to the new object deck should be placed after the DEVEND statement. If more than one copy of that module is found, the first one is used.

If you are adding a new module to the nucleus, edit the file $GEN:MDLSYG and add)OBJ statement for the new module where you you want it to appear in the nucleus. It should be placed somewhere between the modules URMON and PROTND.

Install the Nucleus

Before installing the new nucleus make sure that you have a backup copy of the old nucleus on tape. If the nucleus was written to tape, attach a tape drive to the MUSIC virtual machine, shutdown MUSIC and IPL from the tape drive. If the nucleus was written to a file use the SYSGEN1 utility to install it directly. The ADMIN facility also uses this technique. Once the nucleus is installed reload the system and test out your changes. If you are making changes to system code, the tape method is preferred, since if something goes wrong you always have the tape from the previous NUCGEN as a backup. This is only true of course if you used the tape method the last time.

Backup copies of the MUSIC nucleus can also be kept in CMS files. Punch the file created by the NUCGEN job to a CMS machine, read it in and save it in a CMS file. To install the nucleus, punch a copy of the file to MUSIC and IPL the file from the virtual card reader. (Specify the no header option on the punch command).

Under VM, you can also setup a second copy of MUSIC to be used as a test system. This is very useful in debugging and testing system changes during normal working hours. If you are considering making major modifications to the nucleus this option should not be overlooked.

Modifying Applications, Utilities and Commands

This section gives a general overview of the steps required to modify or add applications, utilities or commands. The basic procedure involves three steps: get the source off the distribution tape if required; make the changes; install the new version.

Restore Source from Tape

You can use the ADMIN facility to restore source files as described in the *MUSIC/SP Administrator's Guide* under the topic "Restoring System Source Files".

Alternately, the MFREST utility is used to restore files from the distribution tapes. Consult the file CATALOG to find out what tape a specific file is on and sample jobs to restore them. When restoring groups of files with MFREST, never specify REPL=T unless you really mean it and always specify FIXUP=' ' unless you specifically want it to do name fixup. The files you restore from the distribution tape depend on what you want to do. If you have a lot of disk space you may want to restore all the files and have the convenience of having everything online.

Install the New Version

The procedure to follow here depends on the language used and on whether the program is run from a load module or from the load library.

If the program is written in REXX there is very little to do after the source changes are made since REXX runs directly from source. You can make REXX programs more efficient by removing comments and compressing the source using the REXDCOM and REXCOMP commands. This is only important for large programs or programs that are used frequently. (Remember to keep the original copy with the comments around). To install the new version simply replace the contents of the original file with the new REXX source.

Programs written in other languages must be compiler or assembled. They can then be made into load modules. Load modules can be executed directly or put in the Load Library. Files have been set up to assist you in performing these steps. The source files for many programs contain the control statements required to compile or assemble that program. Simply executing the file will automatically produce the object file. You create the load module by running the LKED file for the program. Sometimes an LD or APPLY file is also available to copy the load module to the system load library. (The load library is described elsewhere in this manual).

Suppose you are changing the AUTOPR utility.

$PGM:AUTOPR.S	Contains source. Executing this file compiles the source and produces the object program.
$PGM:AUTOPR.LKED	Executing this file creates the load module.

Making Changes with REPs

The Replace (REP) statement is a convenient method of making small changes to the MUSIC system without the need for reassembling the modules.

The REP statement is considered by the MUSIC loaders as a special form of a TXT statement found in object modules. The REP statement is normally inserted in the object module after the TXT statements and before the RLD statements. It will also work when placed just before the END statement of an object module if the information in the REP statement is not subject to relocation.

The first character on REP card can be punched by the combination of the 12-2-9 punches using the multi-punch key on a card punch. Alternately it can be entered while using the Context Editor's *hexadecimal input* feature. The 12-2-9 punch is hexadecimal 02. For example, type the following commands to place a hexa-decimal 02 in position 1 of a line:

```
xin;o 02;ain
```

The format of the REP cards is as follows:

```
card col.   0000000001111111111
            1234567890123456789

            .REP  aaaaaa sssccccc,cccc,cccc  comments
```

where:

col 1 is a 12-2-9 punch (hexadecimal 02)

col 2-4 are the characters 'REP'

col 7-12 is a 6 digit hexadecimal address relative to module and obtained from the far left column of an assembled listing. The address must be a multiple of 2.

col 14-16 is the control section id, normally 001. This number is hexadecimal. This can be verified by the SD number given in the External Symbol Dictionary on the first page of an assembled listing

cols 17-70 contain a maximum of 11 4-digit hexadecimal fields separated by commas.

Example:

```
12
2
9REP  000452  014700,0000 MAKE IT A NO-OP
```

REP cards can also be included in the NUCGEN jobstream after the DEVEND statement. A special)REP statement is used to indicate which module the REPs apply to.

```
)REP TCS
.REP  000100  010700 PUT NOPS IN TCS
.REP  000120  014700
)REP URMON
.REP  00024C  0147F0 FORCE BRANCH
```

")NOREP xxx" removes all reps previously made in module xxx.

SYSREP Utility

The SYSREP utility can be used to apply REPs to load library members. The format of the REP statement used by SYSREP is different from the one described above.

System Control Blocks and DSECTs

In understanding any system, knowledge of the layout of the system control blocks is very helpful. You can get an assembled listing of MUSIC's control blocks by assembling the module @MUSBK. This module is on the source tape under the userid $SYS. @MUSBK contains dsects of many of the system control blocks. It is provided to assist in interpreting core dumps and system code. Some bits and bytes may not be currently maintained. Avoid writing user code that refers to any bits or bytes directly as this may make it system level dependent. Most of the control blocks and DSECTs and mapped by macros. These system macros are stored on the user id $MCM.

Program Chaining and Multi-Tasking

MUSIC provides subroutines to control program chaining and multi-tasking. In program chaining a series of programs are run one after another. The parent program must finish before the child program starts. Multi-tasking allows the parent program to run in parallel with the child program it created. The parent may wait for the child to finish before continuing.

Program Chaining

The subroutines NXTCMD and NXTPGM can be used to schedule another program or a command when the current program finishes running. These are documented in the User's Guide. It is also possible to designate a particular program as a user's *ALWAYS* program either in the user profile, or dynamically using an option of the NXTPGM subroutine. When there are no other programs scheduled, rather than return the terminal to *Go mode, the system runs the *ALWAYS* program. This mechanism allows programs such as REXX and TODO to schedule the execution of other commands and programs but automatically regain control when the scheduled jobs are finished. *ALWAYS* programs are started from the beginning each time and must be responsible for saving information that must be preserved while scheduled commands and programs are running. Also, scheduling a new *ALWAYS* program replaces the previous one. When executing a chain of programs under the control of an *ALWAYS* program the /CANCEL command can be used to cancel the individual programs. The /CANCEL ALL command will remove the *ALWAYS*, program if it is flagged as cancellable. Obvious applications are in the area of providing restricted or subset views of the system.

Multi-Tasking

The NXTCMD subroutine can be called to create a new task. The underlying mechanism uses MUSIC's multi-session support to add a new session and start the new program running in that session. The parent task controls how the user views this new session. The calling sequence for NXTCMD is as follows.

```
    CALL NXTCMD(command,len,opts)

command - command or program to execute
len     - length of the command
opts    - options
```

The bits in the low order byte of the options parameter are defined as follows.

X'80' Set to 1 for multi-tasking request

X'40' Parent task will wait for child to terminate.

X'20' Delete the child task when the scheduled program finishes.

X'10' Display job startup messages.

X'08' Leave the parent task active on the terminal. The child task will run in the background until the user activates it with multi-session function keys or commands.

X'04' Don't let the user see the parent task while the child task is running. This is normally used in conjunction with X'40' to prevent the user returning to the parent task prematurely using multi-session function keys or commands.

X'02' Create the new task as a BTRM and disconnect it from the parent. The BTRM is deleted when the

new task finishes.

X'01' Make child task non-cancellable.

Defining BTRMs and Auto-Sign-on

In implementing various features of the MUSIC system there was a requirement to be able to run programs in the background without a terminal attached (BTRM), and to be able to run programs on terminals without keyboards (Auto-Sign-on). Since both these areas involve starting a program without user intervention, they have been addressed in the same way.

The problem of running background jobs was solved by introducing the dummy terminal class BTRM. A BTRM is specified as a device in the NUCGEN. This causes MUSIC to construct the appropriate control blocks during system initialization as if a terminal was defined at that address. The device address specified for the BTRM is NOT related to any physical device, but is only used as a mechanism to determine which code to sign on. By default the system should be set up to have BTRMs defined on 005 and 010-019. These are used to run system utilities that communicate with VM. The fact that there is also virtual punches defined on these addresses is not a problem.

The lowest (first) BTRM address (for example, 005 in the standard system) should be used for the System Log Message Server BTRM ($PGM:SYSLG). Any BTRMs (such as RDMAILER) that may send log records to the Log Server BTRM should come after it. This is so that at MUSIC startup time, the Log Server BTRM will start before any of the other BTRMs.

In the NUCGEN, Auto-Sign-on terminals are defined in the same manner as regular terminals. They also have the SIGNON option specified. As with BTRMs the device address plays a role in determining which code to sign on.

By default BTRM or Auto-Sign-on terminal is automatically signed on to the code $MONsss, where the subcode *sss* is the device address of the BTRM or terminal as specified in the MUSIC NUCGEN. If the code does not exist in the system code table the sign-on will fail and the BTRM or Auto-Sign-on terminal will not work. Alternately the userid can be defined in the file $PGM:AUTO.CONTROL. Each entry in the file contains a device address range and a userid. For example:

```
100-13F CWIS000
200-23F INFO000
250-250 PRT1000
```

Anything in the virtual address range 100-13F is signed on with the userid CWIS000, 200-23F uses INFO000, and 250 uses PRT1000.

During the sign-on process time limits, file space limits, privileges, terminal type, and operation default are all set from the user profile. Finally if an AUTOPROG is specified in the profile, and one should be, it is run. This allows the BTRM and Auto-Sign-on terminals to be signed on and start running the appropriate program.

The following steps should be taken when adding a new BTRM or Auto-Sign-on terminals to the system.

1. Decide on the device address and add the appropriate device specification statement to the NUCGEN job stream and create a new nucleus.

2. Allocate the code $MONsss, where *sss* the device address in question. When allocating the code set the appropriate password, time limits, terminal type, and privileges. The AUTOPROG parameter must also

be specified indicating which program is to be run. Use the CODUPD command "ADD $MON SC(sss)..." so that sss will be a subcode.

3. Set up the file to contain the auto-program defined in step 2.

4. Test the program by signing on to the code from a real terminal and verifying that there are no problems. This is very important since BTRMs have no terminals attached so you do not see any error messages when they are run in production. Auto sign-on terminals may be remotely located and again error messages may be difficult to access. Therefore testing the program before using it with the production BTRM or Auto sign-on terminal prevent problems later on.

5. Apply the new nucleus and verify the new BTRM of Auto sign-on terminal is functioning correctly.

BTRMs and Auto sign-on terminals respond to operator commands in the same way as do regular terminals. Normally the operator need not intervene, but under certain circumstances it may be necessary to cancel or restart one of the programs running on a BTRM or Auto sign-on terminal. Typically this situation occurs when the program or its parameters have to be changed, or the program is not functioning correctly. The following commands are recommended for these functions.

/CANCEL nnn	- the program is cancelled.
/RESET nnn	- the program is cancelled and restarted.

(nnn is the TCB number)

System Log Message Server BTRM (SYSLG)

Several MUSIC applications (for example MAIL and RDMAILER) need to be able to write many log and information records to a log file. Activity in such log files can be high - tens of thousands of records per day. In order to do this efficiently, these applications can send the log records as messages to a single log server task, via the Intertask Communications Queue (BPOOL buffers in memory).

The log server, running the program $PGM:SYSLG as a BTRM session, does all the work of managing the log files and writing the log records to the files. This is much more efficient, since each log file is only opened once, and many log records are batched together and written to disk with a single i/o operation. In addition, the log server can optionally redirect log messages to the operator console, to multiple files, to an id as e-mail, and to other programs. This is controlled by a definition file, $PGM:SYSLG.DEFINE. Log files can be defined so as to create a new file each day or month, with the date in the file name.

An application calls the SYSLOG system subroutine in order to send a log record to the log server. The application specifies an 8-character application name (e.g. MAIL and RDMAILER use the application name "MAIL"), a priority or class number (normally 0), and the text of the message (1 to 1024 bytes). Refer to the source file $SUB:SYSLOG.S for details. The calling program or user must have at least the SYSCOM privilege. The log server receives the message (at some later time up to 3 minutes after the SYSLOG subroutine was called) and uses the application name and priority/class number to route the message, according to the definitions in $PGM:SYSLG.DEFINE.

The format of the definition file $PGM:SYSLG.DEFINE is described in file $PGM:SYSLG.DEFINE.DOC. The define file is read once when the log server BTRM starts. If you change the define file and you want the change to take effect now, rather than at the next MUSIC IPL, you must stop the BTRM (by operator command "REPLY n STOP" where n is the TCB number - see below) and restart it (by "RESET n"). If the define file is missing or contains an error, the log server automatically appends all log records to file $000:@SYSLOG.MISC.

When setting up the log server, the lowest (first) BTRM address (for example, 005 in the standard system) should be used for it. Any BTRMs (such as RDMAILER) that may send log records to the Log Server BTRM should come after it. This is so that at MUSIC startup time, the Log Server BTRM will start before any of the other BTRMs, and no log records will be lost. Also, when shutting down MUSIC, in order not to lose any log records, stop BTRMs (such as RDMAILER) that generate log records, then stop the log server (see below), and finally shut down MUSIC.

The userid for the log server BTRM should be created by a command similar to the following, entered in the CODUPD utility. "xxx" is the 3-digit hex BTRM address, as specified in the MUSIC nucleus configuration. For example, in the standard MUSIC it is 005.

```
ADD $MON SC(xxx) PW(*NOLOGON) PROG($PGM:SYSLG) -
  PRIME(NL) NONPRIME(NL) BAT(0) DEF(NL) XSES(10) -
  FILES MAINT LSCAN SYSCOM -
  NAME(BTRM FOR SYST LOG SERVER)
```

The following MUSIC operator commands are used to communicate with the log server. "n" is the TCB number of the log server, which can be obtained by the command "enqtab syslog" entered in a MUSIC terminal session.

REPLY n HELLO	To verify that the log server is running, and to cause it to process any accumulated log messages.
REPLY n STOP	To cause the log server to process any accumulated log messages, then shut down the log server program. WARNING: After the log server is shut down, any log records generated by application programs will be discarded.
RESET n	To restart the log server, after stopping it. This command should only be used after "REPLY n STOP" has been used to shut down the log server.

MFIO Subroutine Interface

Two subroutines, MFIO and MFACT, are provided to give high level languages access to MUSIC's I/O interface. Before attempting to use these routines you should review the information on the assembler and macro interface in chapter 19.

MFACT - Open/Close a file

```
CALL MFACT(rc,'type options.','shortfilename ')
CALL MFACT(rc,'type options.',-1,'longfilename ')
```

rc	File system return code.
type	Operation type (OPEN, CLOSE, EXTRACT).
options	Options dependent on the type of operation (e.g. OKNEW OKOLD WROK RDOK APPOK REPL RENAME etc..) The option string is terminated by a period.
shortfilename	The file name (22 characters) terminated by a blank unless the file name is the maximum length, 22 characters long.
longfilename	The file name (64 characters) terminated by a blank unless the file name is the maximum length, 64 characters long.

Examples:

```
CHARACTER*22 FNAME1
CHARACTER*64 FNAME2
FNAME1(1:)=' '
FNAME2(1:)=' '
FNAME1(1:9)='ABCD:TEMP'
FNAME2(1:15)='ABCD:\WORK\TEMP'
CALL MFACT(IRC,'OPEN RDOK OKOLD.',FNAME1)
CALL MFACT(IRC,'OPEN RDOK OKOLD.',-1,FNAME2)
```

MFIO - Read/Write a file

```
CALL MFIO(rc,'type options.',pos,len,buf)
```

rc	File system return code.
type	Operation type (IO, UIO, FSIO).
options	Options dependent on the type of operation (e.g. RD WR WEOF REW FILL TRUNC RBA etc..) The option string is terminated by a period.
pos	RBA value used if RBA option is specified. Block number for UIO requests.
len	Length of buffer.
buf	I/O buffer.

The string '*.' can be used in the second parameter as shorthand for 'IO RD FILL.'. The string '$.' can be used for 'IO WR TRUNC.'.

MFGETU/MFSETU Subroutines

```
CALL MFGETU(unit)
CALL MFSETU(unit)
```

unit is the integer internal unit number.

When MFACT is called to open a file the internal unit number assigned to that file is stored in the basic request block used by MFIO and MFACT. Once the file is opened, all that is required during I/O operations is this unit number, the buffer address and the length. However, if MFACT is called again to open a second file, the control blocks are reused and the old unit number is lost. The subroutines MFGETU and MFSETU allow the programmer to have a number of files open at the same time by saving the unit number after opening a file (MFGETU) and resetting it whenever an I/O operation is to be done on that file (MFSETU).

MFIO Common Blocks

The following common blocks names are assigned to the MFIO control blocks used by these routines. Detailed information on the contents of these control blocks can be found in Chapter 20.

MFARGX	Copy of MFIO basic request block
MFTAG	Tag information
MFINFO	Infin/infout block
MFHINF	User information block for 16 character userids
MFUINF	User information block
MFEOFP	End of file pointers
MFBUPN	File backup number
MFLIBR	Argument for library request
MFUCTL	User control record
MFEXTN	File extent information

MFFSAR	FSIO argument
MFXINF	Usage information block
MFPHYS	Actual length read, RBA information
MFRNAM	Returned file name from MFACT

MFIO tracing

You can get full tracing of MFIO and MFACT subroutine activity. To activate this feature a /FILE statement with a ddname of MFTRACE is required. For example:

```
     /FILE MFTRACE N(GORK) NEW(REPL) RECFM(VC)
or   /FILE MFTRACE PRT
```

The tracing information looks as follows:

```
MFACT AT AAAAAA RC=RR U=UU REQ= PARM TEXT/XXXXXXXXXX
```

Where *aaaaaa* is the address from where MFACT or MFIO is being called from. *rr* is the return code. *uu* is the internal unit number. "parm text" is the keyword literal text passed in the call. *xxxxxxxxx* is the resulting basic user request block.

Each type request provides additional useful information.

OPEN and EXTRACT	- file name and block INFIN
CLOSE	- blocks EOFPT and TAG
IO and UIO	- block PHYS and 40 bytes of the I/O buffer

Enqueue/Dequeue Facility

The MUSIC subroutines ENQ and DEQ allow the user to gain shared or exclusive control of a resource (by calling ENQ) and subsequently to release control of the resource (by calling DEQ). The QUERY subroutine allows you to check if the resource has been enqueued. This facility is used to coordinate the concurrent use of resource by several users.

The resource is identified by an arbitrary 8, 10 or 24 character name, of which the first character must be a letter A to Z. Any number of users can have simultaneous shared control of the resource, but only one user at a time can have exclusive control of the resource. A user's control of any resource is automatically released when the user's job terminates. Each user may have control of no more than 2 resources at any one time. Improper use of the enqueue/dequeue facility may cause the user's job to be aborted.

A privileged system programmer may display the contents of the system's enqueue table by executing the ENQTAB utility, described in Chapter 17 in this manual.

The calling sequences are as follows:

```
CALL ENQ(TYPE,NAME,RETCD1)

CALL DEQ(TYPE,NAME,RETCD2)

CALL QUERY(TYPE2,NAME,RETCD3)
```

TYPE Integer specifying type of control and length of resource name:

1 request for shared control, 8-byte name
2 request for exclusive control, 8-byte name
3 request for shared control, 10-byte name
4 request for exclusive control, 10-byte name
5 request for shared control, 24-byte name
6 request for exclusive control, 24-byte name

In the call to DEQ, TYPE should be the same as in the corresponding call to ENQ.

TYPE2 Integer specifying the length of resource name:

0 request for 8-byte name
1 request for 10-byte name
2 request for 24-byte name

NAME Maximum length of resource name (8, 10, or 24 bytes long). The first character must be A to Z for a non-privileged user.

RETCD1 Integer return code from ENQ:

0 successful enqueue (the user has been given control as requested)
1 resource is not available (shared control was requested and a user already had exclusive control, or exclusive control was requested and a user already had shared or exclusive control)
2 error

RETCD2 Integer return code from DEQ:

0 successful dequeue (control has been released)
1 the user did not have control
2 error

RETCD3 Integer return code from QUERY:

0 resource is not enqueued
1 resource is enqueued (shared)
2 resource is enqueued (exclusive)
3 error

Unbuffered Tape I/O

Normally MUSIC will automatically buffer I/O to tape. This can be overridden by using RECFM(U) on the /FILE statement. When this is done, each read or write will be unbuffered. An automatic end-of-file will not be done for I/O done this way. The TAPUTIL utility uses this facility to handle variable length unspecified blocks. The subroutine TPIO can be used to issue the file system calls to do I/O in this manner. (The object deck for TPIO is in the file $PGM:TPIO.OBJ, not in the subroutine library). The calling sequence is as follows.

```
CALL TPIO(U,O,R,BUF,LEN)
```

U	Unit number of the tape.
O	Operation code. 0 - Read (max Length = 32760) 1 - Write 2 - Write EOF 3 - Read (max length = "LEN")
R	Return code. -1 - End of file 0 - Normal >1 - MFIO return code
BUF	Buffer address.
LEN	Length. On a read this value is set.

Logical Device Interface

This interface can create a VM terminal session using logical device support and thus allow a user to connect to other virtual machines directly from a MUSIC session. This gives the MUSIC user access to applications that are not running on his MUSIC machine. These are referred to as remote applications. The interface can control what goes on in the session it creates or turn control over to the MUSIC user's keyboard. In the typical situation a MUSIC application would create a session, issue the appropriate command sequences to connect the user to a remote application and then turn control over to the user. When the remote application terminates, the MUSIC application regains control and issues the commands to log the user off the remote system.

The interface consists of a suite of subroutines that are used by the application to create, manage, and terminate the remote session. In addition to providing a terminal session connection the interface can also do file transfer using the 3270 file transfer protocol.

CALL LDLOG(n)

Log operations within logical device support to the user file LDEV.LOG. Logging is off by default.

0 turn logging off
1 turn logging on

CALL LDINIT

Create a logical device.

CALL LDECHO(n)

Echo session on users terminal. The user can see what's happening but can't use the keyboard.

0 turn echo off
1 turn echo on

CALL LDTERM(kseq,lk,row,col,hseq,lh)

Pass control to the user's terminal.

kseq Keyboard escape sequence. Any commands prefixed by this are passed to the local MUSIC system for processing. The following exceptions are handled directly by the LDEV routine.

RECeive	- receive a file from the remote host
SEND	- send a file to the remote host
LOG	- log interrupts to LDEV.LOG
NOLOG	- stop logging interrupts
END	- return to controlling program (also QUIT and QQ)

lk Length of kseq
row row to scan for hseq
col column to scan for hseq
hseq Host escape sequence. When this character sequence is located in the specified location on the screen control is returned to the calling program. This is useful in regaining control to perform automatic logoffs. The scan follows the same rules as LDWAIT.
hl Length of hseq

CALL LDPUT(row,col,data,len)

Put data into a modifiable field. No checking is done on the validity of the data or the field.

row row number
col column number
data data
len length of the data

CALL LDCUR(row,col)

Set the cursor address prior to pressing a function key.

row row number
col column number

CALL LDKEY(key)

Press a function key.

key integer value representing the key

0	: enter
1-24	: PF key
-1	: PA1
-2	: PA2
-3	: PA3
-4	: CLEAR

CALL LDENT(data,len)

Put data in the input area and press enter. This is useful for applications that are not field dependant,

data data
len length of the data

CALL LDGET(row,col,buf,len)

Get data from the in core screen buffer. This need not be used on a field basis. The routine will pick out whatever part of the buffer you want. Note that currently keyboard input is not maintained in the buffer, only host transmitted data can be retrieved.

row row number
col column number
buf local buffer to receive the data
len length of the data

CALL LDWAIT(row,col,data,len,secs,rc)

Wait for a certain string of text to appear on the screen. This is useful in synchronizing events and preventing sending data before the host session is ready for it.

row row number
col column number
data data
len length of the data
secs The number of seconds to wait before returning to program empty handed.
rc This is non zero if the text was not found within the specified limit.

Note: If col=0 the screen is scanned starting from the specified row. If row and col are zero the entire screen is scanned.

CALL LDWINT(secs)

This causes the support routine to wait for an external interrupt or so many seconds, whatever comes first. It can also be used to force the interface to process any interrupts that are currently in the stack if the time limit is set to zero.

secs number of seconds to wait

CALL LDCLR

Clear the screen buffer.

CALL LDSEND('lname rname opt',len,flag,rc)

Send a file to the remote system.

lname local file name
rname remote file name
opt file transfer options
len length of first parameter
flag 1: display file transfer monitor.
rc return code

CALL LDRECV('lname rname opt',len,flag,rc)

Receive a file from the remote system.

lname	local file name
rname	remote file name
opt	file transfer options
len	length of first parameter
flag	1: display file transfer monitor.
rc	return code

Options If the remote system is a CMS system the the options must be separated from the file names by an open bracket "(". The options are the same as the 3270 file transfer program.

Note: The user can invoke the file transfer facility from the keyboard using the %RECEIVE and %SEND commands. ("%" is the escape character) The parameters on these commands are in the same format as the first parameter of the subroutines. For example:

```
%rec music.file cms file (crlf
%send music.file tso.file crlf
```

This sample FORTRAN program listed below is the source for the VM program that is documented in the *MUSIC/SP User's Reference Guide*.

```
CALL LDECHO(1)
CALL LDINIT
CALL LDWAIT(1,2,'VIRTUAL',7,3)
CALL LDTERM('%',1,0,0,0,0)
CALL EXIT
END
```

Logical device routines can be called from REXX, however, the format of the CALL statement in REXX is different from FORTRAN. The following is the REXX version of the sample program above.

```
/INC REXX
      CALL LDECHO  1
      CALL LDINIT
      CALL LDWAIT 1,2,'VIRTUAL',7,3,'RC'
      CALL LDTERM '%',1,0,0,0,0
```

Note that the 'RC' parameter must be present on the call to LDWAIT. Unlike FORTRAN, in REXX all parameters must be specified.

For a more elaborate REXX example see the file $PGM:LDEV.REX.

SIGNON Subroutine (REXX)

The following is a sample REXX application that can be used to access other systems in an automated way. The target could be on the same system, or accessed via VM-Passthru. A specific ID/Password can be used, or a 'shared' id, selected from the shared table can be utilized.

Calling Sequence: CALL SIGNON 'id' 'type' 'system' 'ipl' 'MSG(x)' 'TDISK(xxxx,yyy,z)' 'application'

Parameters:

The parameters are positional.

id SHARED, select ID/Password from table. UNIQUE(id/password) use the supplied ID/password.

type — LOCAL, local VM system is target system. PVM(node), target system accessed via VM-Passthru.

system — VM, target is a VM system. MUSIC(VMID), target is a MUSIC system running in the virtual machine VMID. the ID/Password used MUST be specified via the unique parameter.

IPL
NOIPL — If 'NOIPL' is used, then logon to VM id will be done with the 'NOIPL' option. When the first CP read is issued, the an IPL CMS will be done, and for the next read an 'acc 191 a (noprof' will be used, so CMS goes to command mode. If 'IPL' is specified (or is not specified), then an IPL of CMS is done and the profile exec (if it exists is executed).

msg — specifies the message levels that will be produced from the SIGNON call.
0 default if not specified. Issue the message about starting the automated sign on process.
1 no messages (except errors are displayed)
2 display a message at each major step of the sign on process
3 turn on echoing, let caller see all screens.

TDISK — allocate a TDISK, using:
xxxxx - the number of blocks
yyy - the virtual disk address
z - the file mode
(TDISK is only allowed if the SYSTEM is VM.)

application — specifies the name of application that is running. This parameter is not type-checked but is used to match the application name in the Shared ID file. If application is not specified, a Shared ID without an application name will be used. If application has a value, application must match a SHARED ID's application name for the SHARED ID to be used.

After processing, return will be made to the caller with echoing OFF and control has NOT been passed to the users terminal. If MSG(3) was specified, then echoing has been turned on. It is up to the caller to issue a LDTERM call. When going through Passthru, the LDTERM would normally be of the form:

```
CALL LDTERM '%',1,6,12,'APPLICATION TERMINATED',22
```

On return, if 'shared id' processing has been requested, the ID used will be returned in the RESULT. When the logical session returns to REXX, you MUST issue the following call, to free up the 'shared id'.

```
CALL DEQ 6, userid ,'RC'   /* remove name from enqtable */
```

where userid is the ID provided in the RESULT variable.

3270 Full-Screen I/O Interface.

Using PANEL is the best way to implement 3270 applications. This section describes the FSIO interface that PANEL uses to communicate with 3270 terminals and PCs.

To use the FSIO interface, the application program places output data and orders in a buffer, then issues an FSIO request. This sends the output data to the screen. The program is then delayed until the user presses an action key, at which time the input data (from a READ-MODIFIED operation) is transferred to the program's input buffer. The program then resumes execution and processes the data from the modified fields. This write-read sequence can be repeated many times. The format of the input and output data streams is documented in the publication *IBM 3270 Information, Display System Data Stream Programmer's Reference*.

The application program can address the screen by using a 1-byte row number followed by a 1-byte column number, instead of the 2-byte screen address used by the hardware. The FSIO interface automatically does the required conversion.

FSIO Assembler Language Interface

This interface allows full screen I/O to 3270 terminals. It does this through the MFIO SVC (MFREQ). Each request can result in a WRITE, READ, WRITE/READ or control operation. After setting up the arguments and buffers the MFREQ SVC is issued to perform the I/O. The system then takes the data from the WRITE buffer and places it into the system full screen buffer pool and stops the user's time slice. The user may be swapped out at this time. The data is then written to the terminal using the system buffers. When any action key is struck at the terminal, a read modified is done, the data placed in a system buffer and the user made eligible for user region service. When the user program next gains control, the system will transfer the data to the user's READ buffer.

Arguments

Three MFIO arguments are used.

```
Basic req block    -  a1   MFARG FSIO,U=9,FSARG=a2,PHYS=a3
                           MFGEN

FSARG block        -  a2   MFVAR FSARG,PICT=Y

PHYS block         -  a3   MFVAR PHYS,PICT=Y
```

The basic request block contains control information for MFIO, pointers to the other args and return codes.

The FSARG block contains control information, data lengths and buffer addresses. It expands as follows.

```
          4         4         4          4          4
     | control | wrt len | wrt adr | read len |read adr |

control byte 0 - x'80'  -  write erase
                 x'40'  -  erase all unprotected (after read)
                 x'20'  -  write structured field
                 x'10'  -  asynchronous read
                 x'08'  -  skip write operation
                 x'04'  -  skip read operation
                 x'02'  -  user supplies own WCC.
                 x'01'  -  convert data (x,y -> 3270 form)
                 x'00'  -  normal WRITE/READ operation.
                 x'0C'  -  immediate read modified
                 x'0D'  -  immediate read buffer

control byte 1 - x'80'  -  no data compression on this request
                 x'40'  -  no data compression on this request
                           and all following requests in this job
                 x'20'  -  no APL conversion on this request and
                           all following requests in this job
                 x'10'  -  remember x'40' and x'20' option bits
                           from this control byte, but do not do
                           any I/O operation
```

```
                  x'08'  -  no translation for 3270 model 5 terminal
                            (27 by 132 size screen)
                  x'04'  -  ignore screen not initialized error
                  x'02'  -  reserved
                  x'01'  -  reserved

control byte 2 - x'nn'  -  Overriding Channel Command

control byte 3 - reserved.
```

The actual length of the data moved to the users read buffer is saved in the first word of the two word PHYS block.

Return codes

The return code is stored in byte 8 of the basic req block. A return code of 0 is normal. If any error has been detected this code is set to 2 or 11 and in this case bytes 9 and 10 should be checked to find out what caused the error. Values are as follows.

```
Byte 9.

  x'80'  -  not a 3270.
  x'40'  -  bad screen.  Screen is not setup for full screen I/O.
            Probably caused because no full screen initialization
            was done or some non-full screen I/O was since then.
  x'20'  -  bad length on write (0 or too big)
  x'10'  -  bad length on read (0)
  x'08'  -  bad screen addr (conv x,y -> 3270 addr)
  x'04'  -  buffers not available for write (retry later)
  x'02'  -  reserved
  x'01'  -  no data from read. Test request was hit.

Byte 10.

  x'80'  -  input truncated (user buffer too small)
  x'40'  -  bad data stream from terminal (I/O error on read)
  x'20'  -  I/O error on write
```

Data Conversion

The interface can optionally perform some data conversion for you. This is requested by setting the X'01' bit in control byte 1. If this is not set raw 3270 data is expected from, and passed back to the program. Data conversion is done as follows.

Write buffer — The buffer is scanned for any occurrences of SBA, RA or EUA. In each case the next two bytes are converted from X,Y to 3270 addresses. An error condition is set if these are not on the screen.

Read buffer — Data returned to the user buffer will be in the format...

```
   aid,x,y     x,y,len,data     x,y,len,data...etc

   cursor         field 1          field 2
```

The length in front of each field is a halfword.

Note: Data conversion is not performed on data associated with a WRITE STRUCTURED FIELD command.

Notes on Usage

In addition to the standard write/read request a number of options can be selected using the control bytes. Either the write or the read part of the operation can be skipped. This allows programs to handle output and input as logically distinct. The system will automatically clear the input fields after a read if the erase all unprotected option is selected. This may be useful in data entry applications.

During a normal read operation the application program is stopped, and the system waits for an attention interrupt before doing the read and rescheduling the application. The asynchronous read option allows the application to continue running while the system is waiting for the attention interrupt. When the interrupt occurs, the system does the read and sets the post code to ATTN. The application can test the post code using the PSTCOD subroutine. If the application now issues a read request is given the data from the asynchronous read. If it issues a write, the read data is purged. The application could also choose to issue an immediate read buffer or read modified request at this time.

On completion of a full screen read any action key and associated modified fields are returned to the program with the exception of the TEST REQ key which is reserved as an escape mechanism and puts the terminal in ATTN mode so that BREAK time commands can be entered. In this case no data is returned and the error flag (byte 9 in MFARG) is set to x'01'.

Full screen I/O and regular I/O may be inter-mixed if certain rules are followed. Full screen I/O must begin with either a WRITE ERASE or a WRITE STRUCTURED FIELD. (If this is not the case a BAD SCREEN condition will be returned to the program.) From then on any full screen operations can be used. Any regular I/O from either the program or the system will automatically reset the screen to MUSIC mode and full screen I/O must be re-initiated.

The return flag should always be checked for error conditions.

Buffer Pool

The interface utilizes the terminal buffer pool which is allocated during system initialization. The BPOOL parameter in the NUCGEN program is used to inform the system of how many buffers should be allocated. Each buffer is 516 bytes long, 512 bytes for data and 4 for a chain pointer. These buffers are allocated to a particular terminal only when necessary. For example suppose a program wanted to write 1.2K of data to a terminal and read the response. FSIO would request a chain of three buffers from the pool, move the data there and schedule the write. When TM3270 completes the write it frees the buffer chain and waits for an attention interrupt. When that occurs, since the terminal could possibly transmit up to 2K of data during the read, TM3270 gets a chain of 4 buffers from the pool to do the read, unused buffers are freed immediately. The buffers that actually contain data cannot be freed until the application program is again swapped into the user region.

If a program issues a write request that cannot be accomplished due to lack of buffers, it is informed by a return code, and should re-try the operation after waiting some time. If this condition occurs often, systems personnel should allocate more buffer space by increasing the BPOOL parameter and doing a NUCGEN.

If a read operation cannot be done due to lack of buffers the system will automatically re-try it later.

FSIO Subroutines

The following subroutines can be used instead of the assemble macro interface.

WBUF1 This routine places 3270 orders and data strings into the output buffer. The buffer is contained in the common block /BUFCOM/.

FSIO After the output buffer is complete, this routine is called to issue the write-read request.

GETAID After the read is complete, this routine examines the input buffer to find which action key was pressed and the row and column position of the cursor. The input buffer is in common /BUFCOM/ (occupying the same storage as the output buffer). GETAID also does initialization required for calls to GETFLD.

GETFLD After the read, each call to this routine returns the next screen field in the input buffer, i.e. the next modified field. An alternate return is taken when there are no more fields. GETAID must be called before the series of calls to GETFLD.

TRANSL This routine translates (using the TR machine instruction) a string of characters according to a specified translate table. It should be used to remove screen control characters from output data strings, and to translate input to upper case when necessary. This routine is described in the *MUSIC/SP User's Reference Guide*. It is recommended that characters X'00' through X'3F' and X'FF' be translated to blanks in output data strings.

Calling Sequences

```
CALL WBUF1(N1,N2,...,LEN,DATA,&n)
```

N1,N2,... From 0 to 10 integer values, used for specifying order bytes (SBA, SF, etc.), attribute bytes, and row and column numbers. The 4th byte of each argument is placed into the buffer.

LEN The length of an additional character string (in DATA) to be placed into the buffer. LEN=0 indicates a null string. If LEN is -1, the string in DATA is ended by $$.

DATA Additional characters to be placed into the buffer, after any bytes specified by N1,N2,...

&n Alternate return taken if there is no more room in the output buffer.

Note: BUFPTR in common /BUFCOM/ must be set to 0 before a series of calls to WBUF1.

```
CALL FSIO(TYPE,WRBUF,WRLEN,RDBUF,MAXRD,RDLEN,RETCOD)
```

TYPE This 4-byte integer value contains the FSIO control bytes that specify the type of operation that is to be performed. The bit settings correspond to those defined in the control byte fields for the assemble macro interface. The bytes are reversed however. So the low order byte of TYPE corresponds control byte 0 and the next one to control byte 1.

WRBUF The write buffer.

WRLEN The number of bytes of data to be written.

RDBUF The buffer to be used for the input operation. If WBUF1 and GETAID/GETFLD routines are used, the read buffer is in common /BUFCOM/ (see below) and coincides with the write buffer.

MAXRD The size of the read buffer, i.e. maximum read length.

RDLEN This is set by FSIO to the length of data actually read.

RETCOD This 4-byte integer value contains the return code from the FSIO request. 0 indicates a successful operation. If an error condition occurs, RETCOD is in the format X'0Bxxyy00' where *xx* and *yy* correspond to bytes 9 and 10 of the basic I/O request block. The meanings of the individual bits are described in the assembler macro interface section.

```
CALL GETAID(KEYHIT,CROW,CCOL)
```

KEYHIT Set to an integer value indicating which action key was pressed:

0	ENTER key
1 to 24	Program function key
-1	PA1 key
-2	PA2 key
-3	PA3 key
-10	CLEAR key
-99	Some other key, or an error condition.

CROW Set to the row number of the cursor at the time of the read.

CCOL Set to the column number of the cursor at the time of the read.

Note: The variable RDLEN in common /BUFCOM/ must be set before calling GETAID. Also, GETAID must be called before a series of calls to GETFLD.

```
CALL GETFLD(FROW,FCOL,POS,FLEN,&n)
```

FROW Set to the row number of the first character of the modified field, i.e. the character following the attribute byte.

FCOL Set to the column number of the first character of the modified field.

POS Set to the position of the start of the data for the modified field in the read buffer. The first character of the buffer is considered to be position 1. POS may be set to 0 if FLEN is 0.

FLEN Length, 0 or more, of the data for this field.

&n Alternate return taken when there are no more modified fields.

Common Block Used

COMMON /BUFCOM/ BUFSIZ,BUFPTR,RDLEN,BUF

Routines for Scanning the Code Table

Code $COD contains some subroutines which can be called by privileged users to *scan* the code table, i.e. process all the records, one by one, in index order. For example, they could be used to change a particular code record field or option bit for all codes, or for all codes satisfying some condition. They could be used to print a code table listing in some desired format.

Refer to the comments in files $COD:NXTCOD.S and $COD:CDGETU.S for detailed descriptions.

NXTCOD	Returns the next code record from the table. This routine should be used when none of the records will be changed.
NXTCDA	Similar to NXTCOD, but only the next userid (7-byte code/subcode) and a pointer to the code record are returned. This routine should be used, with CDGETU and CDUPDT or CDFREE, when some of the records will be changed.
CDGETU	Gets a code record for updating. Enqueues for exclusive control of the code record. Either routine NXTCDA or the code search SVC (or CDSVC routine) must have been used previously to get the (pseudo) disk address of the code record.
CDUPDT	Writes an updated code record to disk and dequeues. The record must have been gotten by CDGETU.
CDFREE	Releases control of a code record (dequeue). This routine must be called instead of CDUPDT if a record obtained by CDGETU is not to be changed.

Defining a First-Time Program

The parameter FIRST(n) on a CODUPD ADD or CHANGE command can be used to define the ID number *n* (1 to 255) of a special program that is to be run when the user signs on for the first time. For example, the first-time program could prompt for information about the new user and store it in a log file.

The file names for the ID numbers must be defined by modifying a table in system module SIGNON (member $SIGNON in the Load Library, source in code $SYS). The file name must also be added to the table in module LOOKUP (source in $SYS), since the program must be privileged, with at least the CODES privilege.

Once the first-time program has done its processing, it must zero the ID number in the user's code record, to prevent the program from being invoked the next time the user signs on. Also, it must schedule the user's normal auto-program, if any. This can be done by calling subroutine NOPGM1. See the comments in file $COD:NOPGM1.S for details.

The system is distributed with one first-time program. It has an ID number of 1. Its source is stored under the name $PGM:FIRSTTIME.PGM.S.

Defining an Alternate System Catalog

Normally MUSIC will read the system catalog from the data set called SYS1.MUSIC.CATALOG. This can be changed by the use of the =CATxxxx special option at IPL time. The alternate name will be SYS1.MUSIC.CATxxxx in this case.

The alternate catalog data set must first be allocated on MUSIC's IPL volume, and formatted. Then the data is written to it by using the EDTCAT utility program.

Submitting Jobs Automatically

The AUTOSUB utility can be used to automatically submit batch jobs through VM at certain times of the day and even on certain days of the week. Examples of use include submitting the system usage information statistics programs IOTIME and COUNTS. Refer to the writeup of AUTOSUB in *Part IV - Utilities*.

Miscellaneous System Subroutines

CORZAP

This routine modifies (zaps) a string of bytes anywhere in main storage. The VIP privilege is required.

Calling Sequence: CALL CORZAP(frmbuf,len,loc)

Arguments:

frmbuf	Replacement bytes.
len	Number of bytes to be replaced (any value .ge. 0).
loc	Starting addr of main storage to be replaced.

FETCH

This routine fetches a string of bytes from main storage. Non-privileged users can fetch information from the user region, page zero and their own TCB. The CREAD or VIP privilege allows you to inspect any part of the job's virtual memory.

Calling Sequence: CALL FETCH(loc,len,tobuff)

or

CALL FETCHB(loc,len,tobuff,bufpos)

Arguments:

loc	Starting addr of core to be fetched

len	Number of bytes to be fetched (any value .ge. 0)
tobuff	Receiving field
bufpos	Starting position in tobuff (.ge.1) (fetch uses bufpos=1)

FSCHEK

This routine checks the status of 3270 screen. It is usually used by full screen multi-session applications to determine if the screen can be re-written without erasing something that the user has not yet read.

Calling Sequence: CALL FSCHEK(retcd)

retcd 4-byte return code field.
- 0 Terminal is in full screen mode
- 1 MUSIC mode but no data on screen
- 2 MUSIC mode and screen contains data

PMSG

This routine sets the message flag in the TCB to allow a program to handle the messages or to suppress the automatic display of messages until the flag is reset.

Calling Sequence: CALL PMSG('op')

op ON : program will handle messages
OFF: normal message processing

PSTCOD

A call to this routine retrieves and resets the post code. If you want to look at the post code without resetting it, you can find a copy of it in location x'C64' in low core. The post code is set by the /POST command, the WAKEUP subroutine, and a variety of asynchronous system events.

Calling Sequence: CALL PSTCOD(value)

value is a 8-byte value of the post code.

QRD

Read data from a spool buffer chain managed by DSPOOL. This can be used to read the message queue, the intertask communications queue, or the terminal input and output queues.

Calling Sequence: CALL QRD(qid,tcb,buf,len)

Arguments:

qid
- 1: output
- 2: input
- 3: message
- 4: task

tcb TCB number

buf buffer address

len data length

The SYSCOM privilege is required to access queues owned by another TCB.

QRDX

Read data from a spool buffer chain managed by DSPOOL. This can be used to read the message queue, the intertask communications queue, or the terminal input and output queues.

Calling Sequence: retcod=QRDX(qid,tcb,buf,len)

Arguments:

retcod A return code, set as follows:
0 operation complete
1 failed (not enough buffer space)
2 no record found (end of file)
3 routine already processing chain

qid 1: output
2: input
3: message
4: task

tcb TCB number

buf buffer address

len data length

The SYSCOM privilege is required to access queues owned by another TCB.

QWR

Write data to a spool buffer chain managed by DSPOOL. This can be used to write to message queue, the intertask communications queue, or the terminal input and output queues.

Calling Sequence: CALL QWR(qid,tcb,buf,len)

Arguments:

qid 1: output
2: input
3: message
4: task

tcb TCB number

buf buffer address

len data length

The SYSCOM privilege is required to access queues owned by another TCB.

QWRX

Write data to a spool buffer chain managed by DSPOOL. This can be used to write to message queue, the intertask communications queue, or the terminal input and output queues.

Calling Sequence: retcod=QWRX(qid,tcb,buf,len)

Arguments:

retcod A return code, set as follows:
0 operation complete
1 failed (not enough buffer space)
2 no record found (end of file)
3 routine already processing chain

qid 1: output
2: input
3: message
4: task

tcb TCB number

buf buffer address

len data length

The SYSCOM privilege is required to access queues owned by another TCB.

ROUTE

This routine returns information from the system routing table. See $PGM:$ROUTING.S for the format of the routing table.

Calling Sequences:

CALL ROUTE(1,rtname,retcod)
or

CALL ROUTE(rtname)
This returns the default printer name for the user running the program. The routine first checks if one is specified in the user profile. It then looks for a device address match in the routing table. If these fail *retcod* is set to 1 and *rtname* is set to the default route name from the beginning of the table. In the call with only 1 argument, no return code is set, but *rtname* is set the same way.

CALL ROUTE(2,rtname,retcod)
A request to check whether the specified *rtname* is in the table.

CALL ROUTE(3,rtname,userid)
userid returns the 4-char id of the user authorized to run the printer *rtname.* If the id is

longer than 4 chars, only the first 4 are returned. *userid* is returned as 1 or 2 (like retcod) if the route name cannot be found.

CALL ROUTE(4,rtname,retcod,info,buflen)
: A request to search the table for route name *rtname*, and return information from the table entry in *info*. This *info* is the table entry except for the 1st 12 bytes. The caller sets *buflen* to the length of the *info* argument (0 or more). The bytes of the table entry after the route name are returned in *info*, as much as will fit in the buffer provided, and the remainder (if any) of the buffer is set to blanks.

CALL ROUTE(5,rtname,retcod,info,buflen)
: Similar to call type 4, except that the sequential number (1,2,...) of the table entry to be returned is put into *retcod* by the caller. The route name from the entry is returned in *rtname*, and the entire table entry is returned in *info* (12 more bytes than for call type 4, since the entire entry is returned). Table entry 1 is the default route name; table entry 2 is the first "normal" route name entry; etc. This type of call can be used to scan the table, or to get all the entries one by one.

Arguments:

rtname
: 8-character route name (e.g. a destination name for job output, or the name of an rscs remote printer). It is output (set by this routine) for call type 1 and input for the other types.

retcod
: A return code, set as follows:
  0 OK (device addr or name found in table).
  1 Device addr or name not found in table; for call type 1 (or if only 1 arg), *rtname* is set to the default route name from the beginning of the table. For call type 5, retcod=1 means the input entry number is out of range.

2
: The table could not be loaded. For call type 1, the name is set to all blanks if retcod=2. For call type 5, *retcod* is also an input argument, specifying the number (1,2,...) of the desired entry.

userid
: For call type 3, set to the first 4 chars of the user id from the table entry.

info
: User's buffer where the remainder of the table entry is returned (or, for call type 5, the entire table entry).

buflen
: The length of the area provided by the caller in the *info* argument.

Notes:

1. If the calling sequence is incorrect or call type (1st arg) is wrong, the program stops with an invalid op-code (program interrupt 1) and r8=x'EEEEEEEE'.

2. The *info* returned from the type 4 call is in the following format:

```
displ.  len.  contents
  0      8     MUSIC userid authorized to run this printer,
               or blanks.
  8      1     type of table entry:
               1 = local VM system printer
               2 = rscs printer or another virtual machine
               3 = ignore (throw away) all output
               4 = output to MUSIC's print queue
               5 = mvs printer (mcgill only)
```

```
 9      1     VM class.
10      2     (reserved)
12      8     target VM userid.
20      8     VM forms name (or blanks).
28     24     VM tag text (or blanks).
```

total length of *info* 52

(For call type 5, the above info is preceded by the first 12 bytes of the table entry, which contain the low and high address and the route name. For entry #1 i.e. the default route name, entry length and number of entries are returned instead of low and high address.)

3. If entry point "ROUTE1" is called instead of "ROUTE", then this routine finds the address of the routing table itself (by searching the LPA directory), instead of issuing the $LODSVC SVC. For example, nucleus module SPAM calls ROUTE1, since $LODSVC is not supported in supervisor state. ROUTE1 does not work outside of supervisor state because the pointer (in module LODSVC) to the LPA directory is in fetch-protected main storage. For ROUTE1, the routing table must be in the FLPA (not PLPA).

4. The subroutine is re-entrant (it does not modify itself).

SETSVC

This routine lets programs in high level languages issue the $SETOPT SVC. This SVC is used to move parameters or set options in areas of protected storage. The actual function performed depends on the argument passed to the routine.

Calling Sequence: CALL SETSVC(arg)

A number of different functions can be performed depending on the contents of the argument string. These are listed below.

A0 Set a user option bit on or off in the TCB.

```
ARG  DC    X'A0',AL1(x,y,z)
```

Where x is 0 to set bit off, 1 to set bit on. y is option byte number (1 to 4). z is bit number (0 to 7).

A1 Wake up MUSIC batch (SPAM) to process a job which has just been put into the submit data set. The SYSCOM privilege is required.

```
ARG  DC    X'A1',AL1(N)
```

Where *n* is the submit *q* number (must be 1 to 32). The *q* bit in SBMCOM is turned on, and, if *n* is the current reader class, a $CSTART SVC (reader start) is done.

A2 Force the specified terminal to be signed off. This requires the MAINT privilege.

```
ARG  DC  X'A2',AL3(TCBADR)
```

A3 Scan the ENQ table for any TCBs enqueued on a specific name and issue the equivalent of a /POST command to those programs (see WAKEUP). The calling program must have SYSCOM privilege.

```
ARG  DC  X'A3',CL8'ENQENT',CL8'POSTCMD'
```

A4 Set a user defined CVT pointer in location X'10' and a TCBOLD pointer at X'21C'.

```
ARG   DC   X'A4',AL3(USERCVT),A(USRTCB)
```

A5 Set the VIP bit on/off in the XTCB for this user this will only be done for user who are authorized for for VIP.

```
ARG   DC   X'A5','AL3(TCBADDR),X'N'
```

n - 0 set vip off
n - 1 set vip on

A6 Set the debug control block pointer.

```
ARG   DC   X'A6',AL3(BDGPTR)
```

A8 get/put from spool buffer chains. The caller must have SYSCOM to access another tasks buffer chains. (see QRD/QWR).

```
ARG   DC    X'A8',AL1(OPT),H'TCBNUM',A(BUF),A(LEN),A(PTR)
```

A9 Wakeup a task that has called delay.

```
ARG   DC    X'A9',H'TCBNUM'
```

AA Set message flag bits on or off (see PMSG).

```
ARG   DC   X'AA',H'TCBNUM',X'YY',X'ZZ'
```

yy - 80 or, 00 and
zz - data byte (only 80, 40, and 20 bits can be changed)

UTEST

Tests a userid, a userid pattern or a user type (TSUSER option 15) against a range.

Calling Sequence: CALL UTEST(string,strlen,userid,utype,rc)

Arguments:

string	a 1 to 32 character string in the form of t=n1-n2 where n1 and n2 are an integer range and/or a userid/userid pattern.
strlen	length of *string*.
userid	16 character userid.
utype	integer*4 value obtained from TSUSER option 15.
rc	is the return code. =1 hit (utype is in the range n1-n2) =0 no hit (utype is NOT in the range n1-n2) =-1 invalid integer for either n1 or n2

VMCMD

Issue a VM CP command via diagnose 8.

Calling Sequence: CALL VMCMD(cmd,lcmd,resp,lresp,retcod,outlen)

Arguments:

cmd — Command string. May contain multiple CP commands separated by x'15'.

lcmd — Length of command string, 1 to 240.

resp — Response buffer.

lresp — Length of response buffer, 1 to 8192, or 0 if response buffer should not be used. If lresp=0, VM sends the response lines to the MUSIC VM console.

retcod — Return code:

- 0 Command was issued, and executed successfully.
- >0 Command was issued but was not successful. *retcod* is a VM error number.
- <0 MUSIC error code. Command was not issued.
- -1 Interface is busy; calling program should delay for a short time and retry.
- -2 Invalid request.
- -3 Not under vm.

outlen — Set to indicate length of the response:

- >0 If the response fit in the buffer: *outlen* is the length of the response text. Response lines are separated by x'15' character.
- =0 If lresp=0 or if retcod<0.
- <0 If the response did not all fit in the buffer: outlen=-(number of bytes that would not fit). Note that this does not affect retcod.

Notes:

1. The user or the program must have the "maint" privilege.

2. High-order bit x'80' must be on in the last argument word, to indicate exactly 6 arguments.

3. The calling sequence is similar, but not identical, to that for the mug-mods-tape version of this routine, which is called "cpcmd". Return codes are a bit different, the 6th argument is added, and lresp is not modified.

4. If the SVC is busy, this routine automatically retries up to 5 times before setting retcod=-1. Therefore, the caller should not normally have to worry about testing for busy condition.

5. This routine is re-entrant.

WAKEUP

Issue $SETOPT SVC request to wake up other jobs. The other jobs are identified by an entry in the enqueue table. The SYSCOM privilege is required.

Calling Sequence: CALL wakeup(name,post)

Arguments:

name — 8-byte name. This is matched with the last 8 bytes of the 10-byte names in the enqueue table.

post — 8-byte post code to be put into the XTCB of the jobs that are woken up.

REXX Function Packages

These allow MUSIC subroutines to be called directly from REXX programs. Three function packages are supported by REXX.

```
RXSYSFN  - Provided with MUSIC/SP.
RXLOCFN  - Local function package.
RXUSERFN - User function package.
```

Only RXSYSFN is actually distributed with the system. If you want to make additional functions or subroutines available to your users, you can do so by creating RXLOCFN or RXUSERFN.

There are three major components to the function packages.

The Function Interface

The program in the file $REX:REXX.RXSYSFN.S is the interface between the standard linkage used in MUSIC routines and the parameter lists used by REXX. When a function is called this routine converts the REXX parameter list to standard linkage form, calls the subroutine or function, and moves any results back to the appropriate REXX variables.

The Function Table

The parameter conversion is done based on a table that describes the functions and subroutines that are part of the package. This table is defined by a group of PARMS macros. The file $REX:REXX.FUNCTABL.S contains the function table used for the distributed function package. Each subroutine or function and its calling sequence is defined. If a routine has variable calling sequences, each one must be defined. Comments in the macro ($REX:REXX.PARMS.M) describe how the macro is used.

The Functions

The object code for standard MUSIC functions and subroutines is linked in with the interface module and the function table to form the function package. Not all routines are eligible. For example, routines that use OS I/O access methods, have multiple return addresses, pass floating point parameters or use non-standard linkage, will not work.

Creating a Function Package

1) Create a new function table. Use $REX:REXX.FUNCTABL.S as a model. Assemble it and save the object module.

2) Link the new table with the interface module and the subroutines. The file $REX:REXX:RXSYSFN.LKED provides an example of control statements to do this. Although the interface object code ($REX:REXX.RXSYSFN.OBJ) should be used you should change the name of

the load module file and load module name to RXLOCFN or RXUSERFN, depending on which function package you are creating.

3) Put the new function package into the system load library. The file $REX:REXX.RXSYSFN.LD shows how this is done.

Segmented Function Packages

When a function is called from a REXX program the entire function package is loaded into memory. This causes overhead proportional to the size of the function package. To reduce the overhead it is possible to break a function package into segments containing related functions. Only the segment containing the required function is loaded.

The distributed function package (RXSYSFN) contains five segments.

RXSYSFN	- Standard functions (root segment)
REXITCM	- Intertask communications routines
REXLDEV	- Logical device routines
REXMFIO	- MFIO subroutines and control areas
REXSOCK	- TCP/IP Socket routines

Creating a segmented function package is slightly more complicated than an unsegmented one since each segment has to be created as a separate module.

The function table ($REX:REXX.FUNCTABL.S) contains the definitions of all functions in the package. Those that are not contained in the root segment are identified by the EXTTYP=ENTRY parameter on the PARMS macro. In addition an entry is added for each extra segment specifying EXTTYP=MEMBER and defining the name of the segment module. This function table is linked in with the RXSYSFN code to form the root segment. See the file $REX:REXX.RXSYSFN.LKED for an example.

Each extra segment consists of a stub module, created using the FHEADER macro, that defines all the routines in the segment. This is linked in with the routines that form that segment of the package. See the files $REX:REXX.REXLDEV.* for examples. The segment modules must be in the load library.

Calling Functions From REXX

To avoid problems in parameter substitution, each individual parameter must be quoted and in upper case. The parameters should be separated by commas. For example:

```
call TSUSER '4','TCBNUM'
```

Care must also be taken in how some of the routines are used. For example, when REXX schedules a MUSIC command, the system automatically closes all files and terminates any logical devices. This means that the MFIO and Logical Device routines lose control whenever the REXX program issues a MUSIC command. This problem can be avoided by using multi-tasking (NXTCMD routine), to execute MUSIC commands rather than the standard chaining technique that is used by REXX.

As well the following GT and ST take a single parameter, which should be the name of the variable that contains the BINARY version of an MFIO control block. REXX thinks that they are characters, so you must be very careful about inserting and extracting information. Typical calling sequences are:

```
call GTMFARGX 'MFARGX'     (or call GTMFARGX 'GORK' - the name is
                           arbitrary)
call STMFARGX 'MFARGX'
```

The following subroutines and available in the distributed system function package. Some routines are documented in the *MUSIC/SP User's Reference Guide*, others are documented elsewhere in this manual.

CLRIN	Clear the INPUT area on the screen.
COPY	Copy one file to another.
COUNT	Issue count SVC.
DEBUG	Call interactive debug routine.
DELAY	Stop program for a length of time.
ECHOIN	Echo input to screen.
ENQ	Call enqueue facility
EOJ	Set return code and terminate a job.
FETCH	Fetch data from storage.
FILMSG	Return file system message.
FSCHEK	Check screen status.
FSIO	Do full screen I/O.
GETRET	Get return code.
GTMFxxxx	retrieves control block information xxxx can be one of the following: ARGX, TAG, INFO, HINF, UINF, EOFP, BUPN PHYS, LIBR, UCTL, EXTN, FSAR, XINF
ITxxxx	Intertask communication subroutines (ITCOM) are described below, they include: ITSERV, ITRECV, ITSEND, ITFIND.
KEEPIN	Preserve terminal input.
LDCUR	Logical Device Interface
LDECHO	" " "
LDENT	" " "
LDGET	" " "
LDINIT	" " "
LDKEY	" " "
LDLOG	" " "
LDPUT	" " "
LDRECV	" " "
LDSEND	" " "
LDTERM	" " "
LDWAIT	" " "
LDWINT	" " "
MATCH	Pattern matching with wild card characters.
MFACT	Open/Close file dynamically.
MFGETU	Get file internal unit number.
MFIO	Do I/O to file.
MFSETU	Set file internal unit number.
NOECHO	Don't echo data to screen.
NOSHOW	Don't display input data.
NOTRIN	Don't translate terminal input.
NXTCMD	Execute multi-tasked or chained command.
PARM	Fetch parameter string.
PSTCOD	Fetch post code.
PURGE	Purge files.
QRD	Read data from a buffer chain.
QRDX	Read data from a buffer chain.
QUERY	Call query routine.
QWR	Write data to a buffer chain.
QWRX	Write data to a buffer chain.
RENAME	Rename a file.
ROUTE	Get information from routing table.

RXQFOP	Quick reading and searching of large files.
SHOWIN	Cancel call to NOSHOW
SOCKET	Socket routines are described in the section "MUSIC Socket Interface to TCP/IP" later in this chapter.
STMFxxxx	stores control block information xxxx can be one of the following: ARGX, TAG, INFO, HINF, UINF, EOFP, BUPN PHYS, LIBR, UCTL, EXTN, FSAR, XINF
SYSENV	Return information about system environment.
TRIN	Cancel call to NOTRIN
TSDATE	Get current date information.
TSTIME	Get time of day information.
TSUSER	Get user/terminal information.
VMCMD	Issue a VM command.
WAKEUP	Wakeup a task and set the post code.

ITCOM: Intertask Communication

This set of subroutines provides a simple intertask communications protocol designed for client/server applications within a MUSIC system. The SYSCOM privilege is required to use these routines.

In a typical application the server defines itself to the system using the ITSERV routine. The server name must be unique. The system reserves names beginning with "$S" for applications with the SYSCOM privilege. The server then enters its main processing loop where it calls ITRECV to wait for incoming requests from clients.

The client program locates the server application by calling ITFIND. This returns the TCB number of the server whose name is specified in the call. Once the TCB number of the server is known the client can use ITSEND and ITRECV to exchange data with the server application.

ITSEND and ITRECV are used to exchange data between tasks. Although intended to be used in conjunction with ITSERV and ITFIND in a client/server model, they can be used quite independently since they use only the TCB number to identify the other task.

ITSEND allows a program to send packets of up to 8K to another task. The data is chained in buffers off the receiving tasks TCB. An eight byte header is added to the buffer to identify the sending task. This is used internally by the subroutines and need be of no concern to the application program since it is stripped off by the ITRECV routine.

ITRECV received data from the "intertask-communications buffer chain". Application designers should note that this buffer chain is used for a number of different functions (IUCV, TCP/IP) and is reset at sign-off but not at end-of-job. ITRECV completes when data is available or the specified time out expires. If the data is from ITSEND, the TCB number of the sender is filled in, otherwise the return code is set to indicate that the TCB number is unknown. If the receiving buffer is too small as much data as possible is returned and the rest is lost.

Calling Sequences:

```
call itserv(name,rc)
```

defines a server name to the system

name	server name for enq (1-24 characters)
rc	0:name defined -1:name in use

```
call itfind(name,tcb,rc)
```

finds the TCB number of a server

name	server name for enq (1-24 characters)
tcb	tcb number of server
rc	0:server found -1:server not defined

```
call itsend(data,len,tcb,rc)
```

sends send data to a specific tcb

data	data buffer
len	length of data (1-8k)
tcb	tcb number
rc	0:data sent -1:invalid length -2:no buffers available

```
call itrecv(data,blen,rlen,tcb,time,rc)
```

sends data to a specific tcb

data	data buffer
blen	buffer length (1-8k)
rlen	length of data received
tcb	tcb number of sender
time	time to wait for data
rc	0:data received, sender's TCB number set -1:data received but sender's TCB number unknown -2:time out

MUSIC Socket Interface to TCP/IP

A number of structures and constants are used in the TCP/IP socket interface. A few of the common ones are explained below. Others are detailed in the applicable socket calls. Note that the information applies to the AF_INET domain (internet).

Network socket address or name

This is used in the calls that establish connections or send data to specific places.

```
    2      2       4              8
+------+------+----------+------------------+
| 0002 | pppp | aaaaaaaa | 0000000000000000 |
+------+------+----------+------------------+
```

0002	indicates the AF_INET (internet) domain
pppp	16 bit port number
a..a	32 bit internet address of host
0..0	binary zeros

Clientid

This is used in exchanging sockets between applications.

```
      4              8                  8              20
+----------+------------------+-----------------+-----/-----+
| 00000002 | vvvvvvvvvvvvvvvv | tttttttttttttt | 0.......0 |
+----------+------------------+-----------------+-----/-----+
```

0..2	indicates the AF_INET (internet) domain
v..v	virtual machine name
t..t	task name specified when connecting to TCP/IP
0..0	binary zeros

In the descriptions below each subroutine is known by two names. The first one in mixed case is the official name. The second name is the six character name that can be used to call the routines from languages that don't support long subroutine names like FORTRAN, ASSEMBLER, and REXX.

Accept/ACCEPT

Accept() is used by a server to complete a connection with a client. It assigns a new socket number (ns) to the connection and fills in the client's network address in the fields provided. If there are no pending connections accept() will block until one occurs unless non-blocking mode is in effect. The select() routine can be used to test for pending connections prior to issuing a blocking accept(). Once the accept() is complete the server can accept subsequent connections on the same socket (s).

```
ns  = accept(s, name, namelen)
```

ns	New socket number assigned to the connection
s	Original socket number created by the server
name	The port number and network address of the connecting client
namelen	The length of the name.

If an error occurs ns is set to a negative value. Common errors are:

```
EBADF            -9  : s is not a valid socket descriptor
EINVAL          -22  : listen() was not called
EOPNOTSUPP      -45  : s is not a stream socket
EWOULDBLOCK     -35  : non-blocking mode with no connections
                       pending
```

Bind/BIND

Bind() defines the local network address and port number that is to be associated with a specific socket.

```
rc  = bind(s, name, namelen)
```

rc	Return code

s	Original socket number created by the server
name	The port number and network address to bind to
namelen	The length of the name

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EADDRNOTAVAIL   -49  : address is not available or invalid
EAFNOSUPPORT    -43  : address family is not supported
EADDRINUSE      -48  : address is currently in use
EINVAL          -22  : socket is already bound to an address
```

Close/CLOSE

This shuts down the socket, frees all resources associated with it and closes the TCP connection.

```
rc  = close(s)
```

Connect/CONECT

With stream sockets connect() is used by client applications to connect to servers. The server must have set up a passive open at the other end by issuing bind() and a listen() before the connect() is issued. The server should respond with an accept() to complete the connection.

In blocking mode the connect() blocks until the connection is established or an error occurs. In non-blocking mode EINPROGRESS (-36) is returned and select() must be used to determine when the connection is actually complete.

Connect() can also be used with UDP sockets to establish peer. This basically allows the caller to send datagrams using calls like read() and write() that have no room to specify a remote address. In other words connect() simply defines a default remote address.

```
rc  = connect(s, name, namelen)
```

rc	Return code
s	Socket number
name	The port number and network address of the server to connect to.
namelen	The length of the name

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EADDRNOTAVAIL   -49  : cannot reach the remote host
EAFNOSUPPORT    -43  : address family is not supported
EALREADY        -37  : previous connection has not completed
ECONNREFUSED    -61  : remote host rejected the connection
EINPROGRESS     -36  : connection is in progress
EISCON          -56  : socket is already connected
ENETUNREACH     -51  : remote host cannot be reached
ETIMEDOUT       -60  : connection timed out
EINVAL          -22  : addrlen is incorrect
```

Fcntl/FCNTL

Fcntl() is used to control whether a socket operates in blocking or non-blocking mode. In blocking mode (the default) the application does not receive control back from the subroutine until the operation is complete. For example if a program issues a read() it is stopped (blocked) until some data actually arrives on the socket. In non- blocking mode an error (EWOULDBLOCK) is given in response to the read() if no data is present.

```
rc = fcntl(s, cmd, data)
```

rc	Return code
s	Socket number
cmd	Indicates what to do
data	Data depending on cmd

The are currently three possible calls:

```
rc = fcntl(s, 4, 4)   : put socket into non-blocking mode.
rc = fcntl(s, 4, 0)   : put socket into blocking mode.
rc = fcntl(s, 3, 0)   : rc is set to 4 if socket is in non-blocking
                        mode, and is set to 0 if the socket is in
                        blocking mode.
```

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EINVAL          -22  : data is not a valid flag
```

GetClientId/GCLNID

This routine is used to find out the client ID by which an application is known to TCPIP. It is used in passing sockets from one application to another through the givesocket() and takesocket() routines.

```
rc  = getclientid(domain, clientid)
```

rc	Return code
domain	set to 2 for AF_INET
clientid	format is defined previously

Common errors:

```
EPFNOSUPPORT    -46  : domain in not AF_INET (2)
```

GetHostId/GHSTID

This returns the default internet address for the host running the application that calls the routine.

```
id  = gethostid()
```

id	32 bit internet address

GetHostName/GHSTNM

This returns the name of the host running the calling application. Note, this is the actual character name, not the port number/ internet address combination that are referred to in other calls as the name.

```
rc  = gethostname(name, namelen)
```

rc	Return code.
name	The returned name of the host
namelen	On input this specified the maximum size of the name buffer. On output it contains the length of the name.

GetPeerName/GPRNM

This routine retrieves the name of the remote host. The name contains the port number and internet address.

```
rc  = getpeername(s, name, namelen)
```

name	The port number and network address of the application that the caller is connected to.
namelen	The length of the name.

Common errors:

```
EBADF            -9  : s is not a valid socket
ENOTCONN        -57  : the socket is not connected
```

GetSocketName/GSCKNM

This routine retrieves the name associated with the local socket. This contains the port number and internet address.

```
rc  = getsocketname(s, name, namelen)
```

name	The port number and network address of the application that the caller is connected to.
namelen	The length of the name.

Common errors:

```
EBADF            -9  : s is not a valid socket
```

GetSockOpt/GSCKOP

This is used to get the options associated with a particular socket.

```
rc  = getsockopt(s, level, optname, optval, optlen)
```

s	Socket number.
level	Set to -1 for sockets.
optname	Integer indicating which option to get. See setsockopt() for a list of valid values.
optval	The value returned for that option.
optlen	The length of the option value.

If an error occurs rc is set to a negative value. Common errors are:

```
EBADF            -9  : s is not a valid socket descriptor
ENOPROTOOPT     -42  : optname is invalid
```

GiveSocket/GIVESK

This is used to pass a socket to another application. For a complete discussion of the givesocket() and take-socket() routines see the comments in the file $TCP:INETD.S. This shows the steps that must be taken to start up a new task and pass a socket on to it. The socket should not be closed until the other task has successfully acquired it using the takesocket() routine.

```
rc  = givesocket(s, clientid)
```

s	The socket number to be given the the other task.
clientid	The client ID of the task that you wish to give the socket to.

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EBUSY           -16  : listen() has been called for the socket
EINVAL          -22  : clientid does not specify a valid client ID
ENOTCONN        -57  : the socket is not connected
EOPNOTSUPP      -45  : s is not a stream socket
```

IoCtl/IOCTL

Ioctl() provides access to the operating characteristics of the socket.

```
rc  = ioctl(s, cmd, data)
```

s	Socket number.
cmd	The command to perform.
data	Information passed or returned.

The following table lists the supported requests:

Name	Value (HEX)	Function
FIONBIO	8004A77E	Clear or set non-blocking mode. Data=0 to clear, 1 to set.
FIONREAD	4004A77F	Returns number of byte available for reading in data.
SIOCATMARK	4004A707	Queries if current data is out of band. Data=1 if yes, 0 if no
SIOCGIFADDR	C020A70B	Gets network interface address. Data=IFREQ
SIOCGIBRDFADDR	C020A712	Gets network interface broadcast address. Data=IFREQ
SIOCGIFCONF	C008A714	Gets network interface configuration. On input data should be set to the length

		of the output area. On output it will have an IFREQ structure for each interface.
SIOCGIFDSTADDR	C020A70F	Gets network interface destination address. Data=IFREQ
SIOCGIFFLAGS	C020A711	Gets network interface flags. Data=IFREQ
SIOCGIFNETMASK	C020A715	Gets network interface mask. Data=IFREQ

The IFREQ structure that is returned has a 16 byte request name followed by the requested information.

Listen/LISTEN

Listen() is used by a server to indicate it is ready to accept calls from clients.

```
rc  = listen(s, backlog)
```

s	Socket number.
backlog	Length of pending connection queue.

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EOPNOTSUP       -45  : s is not a stream socket
```

Read/READ

This call reads data from a connected socket. The number of bytes read is returned in the return code (rc). In blocking mode this call will block until some data becomes available.

```
rc  = read(s, buf, len)
```

s	Socket number.
buf	Buffer to receive the data.
len	Buffer length.

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EWOULDBLOCK     -35  : set in non-blocking mode if no data to read
```

Readv/READV

This call is similar to the read() except that the data is read into a group of buffers pointed to by an I/O vector structure. Each entry in the I/O vector contains a buffer address and a length. The buffers are filled in order.

```
rc  = readv(s, iov, iovcnt)
```

s	Socket number.
iov	I/O vector containing buffer address / length pairs

iovcnt Number of entries in the iov.

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EWOULDBLOCK     -35  : set in non-blocking mode if no data to read
```

Recv/RECV

This receives data from a connected socket. The return code (rc) is set to the number of bytes received. It is very similar to read() except for allowing certain flags to be set.

```
rc  = recv(s, buf, len, flags)
```

s	Socket number.
buf	Buffer to receive the data.
len	Buffer length.
flags	Flag value of MSG_OOB (1) or MSG_PEEK (2)

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EWOULDBLOCK     -35  : set in non-blocking mode if no data to read
```

RecvFrom/RECVFM

This receives data from a datagram socket. The return code (rc) is set to the number of bytes received.

```
rc  = recvfrom(s, buf, len, flags, name, namelen)
```

s	Socket number.
buf	Buffer to receive the data.
len	Buffer length.
flags	Flag value of MSG_OOB (1) or MSG_PEEK (2)
name	Port number and network address of the sending application
namelen	Length of name.

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EWOULDBLOCK     -35  : set in non-blocking mode if no data to read
```

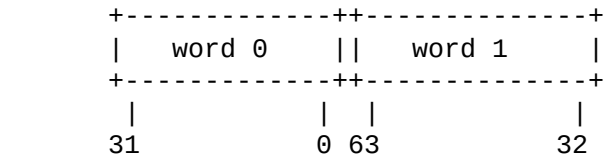
Select/SELECT

The select() routine is used to wait for an interrupt from the socket interface and to determine which socket caused the interrupt. It is used to wait for connections to come and for data to arrive. It is usually used by applications that field interrupt from a variety of sources or handle a number of different sockets and can't afford to get into a blocking situation. The select() is used to test in advance for an event before issuing the socket call to handle that event.

Select() can test a group of sockets for reading, writing, and exceptional conditions. Reading implies that data is available and a read() call will not block. Writing implies that the socket is available and is set for example when a connection comes in for a socket with a pending listen(). Exceptional conditions are events

like out of band data.

On input three bit masks are used to select which sockets should be tested. On output the return code (rc) is set to the number of sockets that have pending conditions and the bits are set in the masks to indicate which sockets they are. The bit masks are organized into 32 bit words that are counted from the low order end. It's a bit weird.




```
rc  = select(nfds,readfds,writefds,excptfds,timeout)
```

nfds	Number of sockets to test
readfds	Bit mask for read
writefds	Bit mask for write
excptfds	Bit mask for exceptional conditions
timeout	Timeout value (integer seconds, integer microseconds)

Common errors:

```
EBADF            -9  : one of the descriptor sets is invalid
EINVAL          -22  : the timeout value is invalid
EMUSINT       -2000  : a non-socket interrupt occurred
```

Send/SEND

This sends a data to a connected socket.

```
rc  = send(s, buf, len, flags)
```

s	Socket number
buf	Buffer containing the data
len	Length of the data
flags	Option flags. Valid values are MSG_OOB (1) and MSG_DONTROUTE (4)

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
ENOBUF          -55  : insufficient buffer space to send the data
```

SendTo/SENDTO

Used to send packets on a datagram socket.

```
rc  = SendTo(s, buf, len, flags, name, namelen)
```

s	Socket number
buf	buffer containing the packet
len	length of the packet
flags	Option flags. Valid option is MSG_DONTROUTE (4).
name	The port number and internet address of the application that you are sending to.

namelen The length of the name

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
EINVAL          -22  : namelen is invalid
EMSGSIZE        -40  : the message is too big to be sent as a
                       single datagram.
ENOBUFS         -55  : No buffer space available to do send
```

SetSockOpt/STSKOP

This sets various options associated with a socket. It allows you to control broadcast messages, out of band data, and tell the socket interface how to handle the close and bind operations.

```
rc  = setsockopt(s, level, optname, optval, optlen)
```

s	Socket number.
level	Protocol level (set to -1 for socket level)
optname	Option name
optval	Option value
optlen	Option length

The following table of lists the supported options. This list also applies to the getsockopt() routine. For those calls that toggle an option on or off, optval=1 sets it on and optval=0 sets it off. In both cases optlen should be set to 4.

Name	Description
SO_OOBINLINE X'00000100'	Toggle reception of out of band data. If on, out of band data is placed in the normal input queue.
SO_REUSEADDR X'00000004'	Toggle address reuse. Setting this on forces bind() to reuse an address
SO_LINGER X'00000080'	Causes interface to linger on close() if data remains to be sent. optval consists of two fullwords. The first one is a toggle. The second is the number of seconds to wait.

ShutDown/SHUTDN

This shuts down all or part of a full duplex connection.

```
rc  = shutdown(s, how)
```

s	Socket number
how	0:ends communication from socket 1:ends communication to socket 2:ends all communication with socket

Socket/SOCKET

This routine creates a socket and assigns the socket number (s).

```
s   = socket(domain, type, protocol)
```

domain	Set to 2 for the AF_INET domain (internet)
type	Type of socket.(1-stream, 2-dgram, 3-raw)
protocol	Protocol to use or 0. 0 is the default and is determined by domain and type. AF_INET and STREAM implies TCP. Valid values include:

```
ICOMP     1
TCP       6
UDP      17
Raw IP  255
```

Common errors:

```
EIBMIUCVERR    -1004  : error contacting TCP/IP through IUCV
EPROTONOTSUPPORT -35  : protocol not supported
```

TakeSocket/TAKESK

This is used by a program receiving a socket from another program. The subroutine GETSOC performs the handshaking required to get a socket from another program, including issuing the takesocket() call. For further details see the comments in $TCP:GETSOC.S and $TCP:INETD.S.

```
s   = takesocket(clientid, hisdesc)
```

clientid	The client ID of the application giving the socket.
hisdesc	The socket number used in the application giving the so

Common errors:

```
EACCESS       -13  : the other application did not give this socket
EBADF          -9  : hisdec is not a socket owned by the other pgm
EINVAL        -22  : clientid does not specify a valid client
EMFILE        -24  : socket descriptor table is full
ENOBUFS       -55  : TCP/IP has run out of socket control blocks
EPFNOSUPPORT  -46  : domain field of the clientid is not AF_INET (2)
```

Write/WRITE

This writes data to a connected socket.

```
rc  = write(s, buf, len)
```

s	Socket number
buf	Buffer containing the data
len	Length of the data

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
```

```
ENOBUFS         -55  : no buffer space for write
```

writev/WRITEV

This call is similar to the write() except that the data is written from a group of buffers pointed to by an I/O vector structure. Each entry in the I/O vector contains a buffer address and a length.

```
rc  = read(s, iov, iovcnt)
```

s	Socket number.
iov	I/O vector containing buffer address / length pairs
iovcnt	Number of entries in the iov.

Common errors:

```
EBADF            -9  : s is not a valid socket descriptor
ENOBUFS         -55  : no buffer space for write
```

MUSIC Specific Routines

The following routines are specific to the implementation of the socket interface on MUSIC/SP.

a2e/A2E

Convert ASCII to EBCDIC. Note that the default representation for character data on the internet is ASCII whereas IBM mainframes tend to use EBCDIC.

```
call a2e(buf,len)
```

buf	buffer containing ASCII data
len	number of characters in the buffer

CARGCALL

Convert socket routines calling sequences to that defined in the Berkeley sockets header file "MANIFEST.H".

```
rc = CARGCALL()
```

cnvd2x/CNVD2X

Convert a dotted decimal character format internet address to a 32 bit binary number. For example 132.206.120.2 is converted to X'84CE7802'.

```
call cnvd2x(daddr,xaddr)
```

daddr	character format dotted decimal address terminated by a
xaddr	32 bit binary internet address

cnvx2d/CNVX2D

Convert a 32 bit binary internet address to the dotted decimal character format. For example X'84CE7802' is converted to 132.206.120.2.

```
call cnvx2d(xaddr,daddr)
```

xaddr	32 bit binary internet address
daddr	16 character decimal address

e2a/E2A

Convert EBCDIC to ASCII. Note that the default representation for character data on the internet is ASCII whereas IBM mainframes tend to use EBCDIC.

```
call e2a(buf,len)
```

buf	buffer containing EBCDIC data
len	number of characters in the buffer

e2e/E2E

This removes control characters from an EBCDIC character string. Control characters are converted to blanks. Printables are left as is.

```
call e2e(buf,len)
```

buf	buffer containing EBCDIC data
len	number of characters in the buffer

Getcon/GETCON

This routine is used by a server application to wait for a connection on a specific port. It supports only one connection on the port and is intended to be used in debugging servers that would normally be run in the background by INETD. This allows the server to establish a connection without intervention of INETD and thus be debugged as a foreground task. Usually once the server has been debugged this call would be replaced by a call to getsoc() and the server would be set up to run under INETD.

```
s = getcon(port)
```

s	Socket number assigned to connection.
port	Port number

Getsoc/GETSOC

This routine used by a server application that has been scheduled by INETD to get the socket passed from INETD. It automatically handles the protocol to successfully transfer a socket from one application to another.

```
s = getsoc()
```

s	Socket number of passed socket.

Common errors:

```
EMUSBADPARM   -2001  : Parameter passed from INETD is invalid
EMUSINUSE     -2002  : TASKID is already in use
```

GtPort/GTPORT

This function gets a unique port number from the system that can be used to create unique port-number/internet-address combination that is referred to as the socket name.

```
num = gtport()
```

num unique port number between 1000-32768

Resolv/RESOLV

Call a domain name server to convert a host name into an IP address.

```
call resolv(hostname,ipaddr)
```

hostname character host domain name terminated by a blank.
ipaddr 32 bit internet address.

TrSock

This routine turns on socket level tracing (basically detailing the interface between the socket calls and IUCV). Two files are created: $TCP:@TCPIP.LOG, which contains trace records for each socket call, and $TCP:TCPIP.BUFFERS, which contains the inbound and outbound buffers at the socket level.

```
call trsock
```

Note that no parameters are needed.

Errors: none.

XPath/XPATH

This routine defines the TASKID field that is used when connecting the application to TCP/IP. This forms part of the client ID that is used in the givesocket() and takesocket() routines.

```
call xpath(taskid)
```

taskid An 8 character string that uniquely identified the task

Socket Errors

The error number is returned as a negative value as the return code of a socket call.

```
EPERM           1               Not owner
ENOENT          2               No such file or directory
ESRCH           3               No such process
```

```
EINTR           4               Interrupted system call
EIO             5               I/O error
ENXIO           6               No such device or address
E2BIG           7               Arg list too long
ENOEXEC         8               Exec format error
EBADF           9               Bad file number
ECHILD          10              No children
EAGAIN          11              No more processes
ENOMEM          12              Not enough core
EACCES          13              Permission denied
EFAULT          14              Bad address
ENOTBLK         15              Block device required
EBUSY           16              Mount device busy
EEXIST          17              File exists
EXDEV           18              Cross-device link
ENODEV          19              No such device
ENOTDIR         20              Not a directory
EISDIR          21              Is a directory
EINVAL          22              Invalid argument
ENFILE          23              File table overflow
EMFILE          24              Too many open files
ENOTTY          25              Not a typewriter
ETXTBSY         26              Text file busy
EFBIG           27              File too large
ENOSPC          28              No space left on device
ESPIPE          29              Illegal seek
EROFS           30              Read-only file system
EMLINK          31              Too many links
EPIPE           32              Broken pipe
EDOM            33              Domain error
EWOULDBLOCK     35              Operation would block
EINPROGRESS     36              Operation now in progress
EALREADY        37              Operation already in
ENOTSOCK        38              Socket operation on
EDESTADDRREQ    39              Destination address required
EMSGSIZE        40              Message too long
EPROTOTYPE      41              Protocol wrong type for
ENOPROTOOPT     42              Protocol not available
EPROTONOSUPPORT 43              Protocol not supported
ESOCKTNOSUPPORT 44              Socket type not supported
EOPNOTSUPP      45              Operation not supported on
EPFNOSUPPORT    46              Protocol family not
EAFNOSUPPORT    47              Address family not supported
EADDRINUSE      48              Address already in use
EADDRNOTAVAIL   49              Can't assign requested
ENETDOWN        50              Network is down
ENETUNREACH     51              Network is unreachable
ENETRESET       52              Network dropped connection on
ECONNABORTED    53              Software caused connection
ECONNRESET      54              Connection reset by peer
ENOBUFS         55              No buffer space available
EISCONN         56              Socket is already connected
ENOTCONN        57              Socket is not connected
ESHUTDOWN       58              Can't send after socket
ETOOMANYREFS    59              Too many references
ETIMEDOUT       60              Connection timed out
```

```
ECONNREFUSED    61              Connection refused
ELOOP           62              Too many levels of symbolic
ENAMETOOLONG    63              File name too long
EHOSTDOWN       64              Host is down
EHOSTUNREACH    65              No route to host
ENOTEMPTY       66              Directory not empty
EPROCLIM        67              Too many processes
EUSERS          68              Too many users
EDQUOT          69              Disc quota exceeded
ESTALE          70              Stale NFS file handle
EREMOTE         71              Too many levels of remote in
ENOSTR          72              Device is not a stream
ETIME           73              Timer expired
ENOSR           74              Out of streams resources
ENOMSG          75              No message of desired type
EBADMSG         76              Trying to read unreadable
EIDRM           77              Identifier removed
EDEADLK         78              Deadlock condition.
ENOLCK          79              No record locks available.
ENONET          80              Machine is not on the
ERREMOTE        81              Object is remote
ENOLINK         82              the link has been severed
EADV            83              advertise error
ESRMNT          84              srmount error
ECOMM           85              Communication error on send
EPROTO          86              Protocol error
EMULTIHOP       87              multihop attempted
EDOTDOT         88              Cross mount point
EREMCHG         89              Remote address changed
EIBMBADCALL        1000
EIBMBADPARM        1001
EIBMSOCKOUTOFRANGE 1002
EIBMSOCKINUSE      1003
EIBMIUCVERR        1004
EMUSINT            2000         Select interrupted by non-socket int
EMUSBADPARM        2001         Bad parameter
EMUSINUSE          2002         Resource is already in use
```

MUSIC IUCV Interface

MUSIC supports VM's IUCV interface. This allows communications and data exchange between MUSIC applications and other virtual machines and also between MUSIC and CP system services such as:

*BLOCKIO	- DASD Block I/O
*IDENT	- Identify
*LOGREC	- Error Recording
*MSG	- Message System
*MSGALL	- Message All System
*RPI	- Access Verification
*SIGNAL	- Signal Service
*SPL	- Spool Service

In addition IUCV can be used for communication between tasks on the same MUSIC system.

Security

In order to use IUCV, the MUSIC virtual machine must have the appropriate authorization. This involves specifying the required parameters on the IUCV control statement in the VM directory entry for MUSIC. In addition you may also want to allow more than the default number of connections by specifying the MAXCONN keyword on the OPTIONS statement. Remember that MAXCONN specifies the number of connections that can be made by the entire MUSIC machine, not the individual users. As an additional level of security, MUSIC applications must have the SYSCOM privilege to establish an IUCV connection.

Interface

The interface is essentially the same as that described in the publication *VM/ESA CP Programming Services for 370* (SC24-5435) or *VM/SP System Facilities for Programming* (SC24-5288). These manuals should be consulted to find complete information on how to write IUCV applications.

The ASSEMBLER language interface provides two macros.

The IUCV macro is used to execute IUCV functions.

```
        IUCV  function_name,options
```

Function_name The name of the specific IUCV function.

options Optional information that is function dependent.

The following functions are supported:

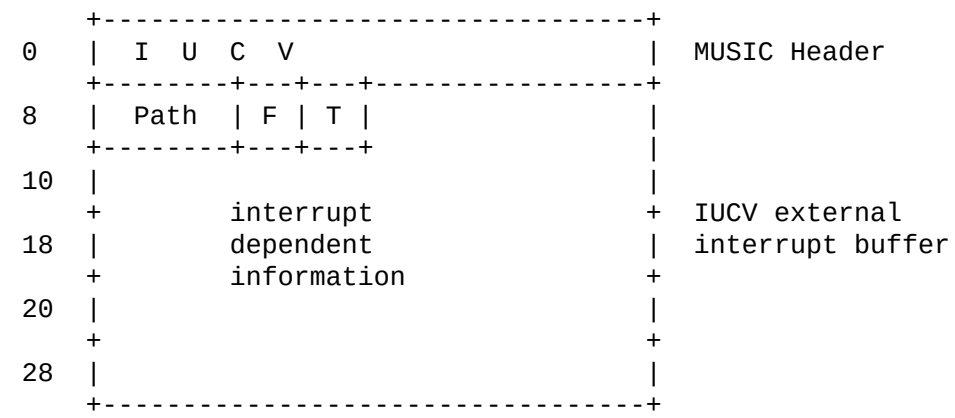
ACCEPT
CONNECT
DECLARE BUFFER*
DESCRIBE
PURGE
QUERY
QUIESCE
RECEIVE
REJECT
REPLY
RESUME
RETRIEVE BUFFER*
SEND
SET CONTROL MASK*
SET MASK*
SEVER
TEST COMPLETION
TEST MESSAGE

* These functions are supported as no-ops since MUSIC does not let applications redefine the IUCV interrupt buffer or control interrupts.

The IPARML macro used to map the IUCV parameter lists and interrupt buffers. This is copied into the assembler program.

One difference between the MUSIC and VM implementation is in the processing of interrupts. The interface module in the MUSIC nucleus gets control when an IUCV external interrupt occurs. The interrupt information is then passed to the application through the inter-task queue. The application retrieves the information

using the QRD routine. The format of this data is the same as the regular IUCV external interrupt buffer preceded by the 8 byte identifier 'IUCV '.



Path Pathid of the connection causing the interrupt

F Flags indicating the interrupt status

T Indicates the interrupt type

The IUCV external interrupt buffer is mapped by the IPARML macro.

The other difference between MUSIC and VM is how a MUSIC server application identifies itself to the system. The server program is identified by a name that it advertises in MUSIC's ENQUEUE table. A client wishing to connect to this application issues a CONNECT function specifying "MUSIC" in the IPVMID field of the connect parameter and the name of the server in the IPUSER field.

Examples

The files IUCV.SERVER and IUCV.CLIENT provide ASSEMBLE language examples of IUCV applications. In addition to showing the use of the IUCV macros, they show how to user the DELAY and QRD routines to handle the interrupts. The SYSCOM privilege is required to run these programs.

Internals

The interface handles the IUCV requests from MUSIC applications, keeps track of what application owns a conversation and passes the external interrupts to the appropriate application.

During system initialization, a DECLARE_BUFFER is issued to define MUSIC's IUCV external interrupt buffer. There is only one of these for the MUSIC virtual machine. When a task issues a DECLARE_BUFFER to initiate IUCV, it is basically treated as a NOP operation. When an external interrupt arrives that applies to that task, the interrupt handler passes the information to task via the inter-task communications queue. This information can be retrieved using the QRD routine. This handling of the external interrupt buffer data makes the MUSIC implementation a bit different from CMS.

When a task issues a CONNECT function the assigned PATHID is saved with the TCB number the PATHID_TABLE. When an external interrupt occurs, this table is used to find the TCB number of MUSIC task that is to receive the interrupt information.

If an outside task is to initiate a connection with a MUSIC task it must know the resource name associated with the MUSIC task. The MUSIC task sets the resource name in the ENQ table. So when an interrupt

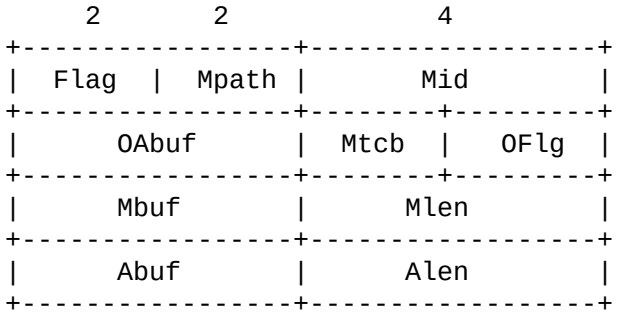
comes in requesting a connection, the TCB number of MUSIC task is retrieved from the ENQ_TABLE.

In addition to executing the IUCV instructions and passing on interrupts this module must also deal with MUSIC's virtual storage. The request parameters are copied from the user region to system storage before the request is made. The results are copied back afterwards. If a request involves data buffers, these are allocated in the system area and the virtual addresses replaced by real addresses. Once the request is complete, the data is copied to the user's virtual storage.

Message Control Blocks (MCB)

These are used to keep track of outbound messages. When a SEND or REPLY is issued an MCB is allocated. Information about the message, such as the buffer pointers, length and message ID are saved in the MCB. When IUCV finishes processing the message, it causes an IUCV external interrupt. The message ID field from the interrupt is used to locate the MCB. The MCB and any allocated buffers are then freed.

They are mapped by the MCB Dsect and are laid out as follows..



ABUF and MBUF point to buffer chains. The first buffer is used to contain a buffer list in the format used by IUCV. Subsequent buffers contain data. This allows a 32K message to span up to 64 buffers.

When an IUCV external interrupt occurs, URMON passes control to the IUCV external interrupt routine in the module IUCV. The function of this routine is to pass the external interrupt and any associated data to the appropriate MUSIC task. Since the tasks address space may be inaccessible to us, the interrupt buffer data is passed via the inter-task communications queue.

If the interrupt is for a connection pending, the ENQ table is scanned for a match on the resource_id from the interrupt buffer. If a match is found the interrupt information is passed to the task that enqueued on the name and an entry is made in the PATHID TABLE to permanently associate the PATHID with that task.

Otherwise the PATHID table is used to find out the TCB number of the task. Once the TCB number is known the interrupt buffer is sent to it via the inter-task communications queue.

Chapter 23. Performance and Tuning

Overview

This section discusses how you can improve the performance of your MUSIC/SP system. The basic steps involve observing the performance of your system and making changes and seeing the results. MUSIC comes with a number of performance tools that can help you measure your system's performance. It is strongly recommended that you regularly monitor your system's performance and not just at times of poor performance. This will allow you to see the compare the times of poor performance to those of good performance and see what has changed.

Basic Resources

There are 3 basic resources that are available to MUSIC. They are:

- Processor cycles. When MUSIC is not run under VM, then this is simply the speed of your processor. Under VM, it also refers to the amount of time available that MUSIC will get from VM. There are several performance options under VM that relate to how much time MUSIC will get.
- Main storage. This is the amount of main storage that MUSIC can use. Under VM, this is its "virtual machine size".
- Disk. This includes the number of channel paths available to the disks as well as the number of disks.

To tune the system it is important to see how MUSIC is using these resources and to determine which one can be improved and at what cost.

VM Performance Considerations

Refer to *Chapter 2 - Running MUSIC/SP Under VM* for very important information on how to best configure MUSIC under VM. In particular note that you must always run a production system in either a V=R or V=V locked environment. Just locking part of MUSIC's pages WILL NOT WORK correctly. The reason for this is that VM will stop the entire MUSIC system on ANY page exception within MUSIC and not process ANY work for MUSIC until the page exception has been resolved by VM reading in the page from disk. This can take 50 milliseconds or longer on a busy machine and will cause very uneven performance. Remember MUSIC is supporting many users in the one virtual machine and not just a single user as in the case of CMS. It is therefore quite reasonable to give MUSIC much better performance options that you give to a single CMS user.

Note that running MUSIC in a V=R or locked page environment will not give MUSIC preferential treatment over your other virtual machines. What it will do is save cycles that VM would otherwise have to use to resolve page exceptions on behalf of MUSIC. In the V=R case, VM will also save cycles by not having to translate MUSIC's I/O channel programs. These cycles saved are a benefit to ALL your virtual machines. This is because it gives effectively makes the processor faster. Running MUSIC V=R can sometimes buy back 10 or 20% of your processor cycles.

Dynamic View Your System's Performance

The SSTAT utility is valuable for seeing what is happening to your system now. It can be run by typing SSTAT as a command or from the ADMIN facility's "Information and Statistical" menu and selecting "Display System Status". Its display is updated several times per minute. You should run at different times in the day to view of the load before making any changes based on its output.

The following notes point out key items of interest in performance and tuning work.

- The storage size gives the amount of storage MUSIC has available to it.
- The MAXRRS shows how much storage each user region can get to run applications. The MAXMPL shows how many regions can be active at one time.
- The number of users show how many users are currently signed on. The number of sessions show how many sessions are active. A single user can have several sessions active at one.
- The response time is the response time this program is seeing. It is an indication of the system response time but is not an average response time people are getting. (The response time people will see depends on the queue the system has placed them in. This depends in turn on what type of work they are doing.)
- The CPU usage gives the % of the CPU MUSIC is using. This includes any VM overhead on behalf of MUSIC.
- The "running" number shows the number of tasks that are running. Many of them may be waiting for input from users. That number is shown in the "idle" field. The number actually running in the user regions is shown in the "active" field. The number that is waiting to get into a user region is shown in the "queued" field.
- The "in core" field shows how many of the running tasks are in main storage. The "swapped" field shows how many tasks the system had to swap out to disk.
- The table in the display shows the total number of specific events that have occurred since your system was last IPLed. It also shows the rate of these events. The rate is the more interesting number.
- The number of I/O operations per second is an indication of the how busy your I/O devices are. If you see this number never gets higher than a certain number at peak times, then this could mean that you have an I/O bottleneck. The IOTIME utility can be used to analyze your I/O activity.
- The number of page events per second is an indication of how much paging you are doing. It does not directly show your paging rate in terms of number of pages read or written per second as explained below.
- The "page wr" figure shows the number of page write operations being done per second. Each of these operations typically writes 8 4K pages to disk in one disk I/O operation.
- The "page rd" gives the number of page reads done. Page reads are done one at a time. Page reads done from the PLPA area are shown separately. Large paging rates here are an indication that some users are running jobs do not fit well in the MAXRRS area. You can increase the size of the MAXRRS area by doing a NUCGEN but that may increase the amount of swapping done.
- PLPA pages are for routines that have been placed in the PLPA area. They are typically reentrant compilers. PLPA page reads are done one at a time. A high PLPA rate would be an indication that you should move some items into the fixed link pack area. Consult the section "Link Pack Area Considerations" in *Chapter 21 - Load Library and Link Pack Area* for information on this topic.

- The number of swap reads and writes shows the rate of swapping. Each swap will typically take several I/O operations. This is because each swap piece is 40K in size. Refer to section "Paging and Swapping" in *Chapter 18 - System Internals* for more information.

Main Storage Usage

Many sites have found that they are not using main storage in an optimal fashion. The output from the MAPMEM utility shows you how your main storage is used. It can be run by typing MAPMEM as a command or from the ADMIN facility's "Information and Statistical" menu and selecting "Display Memory Map". The numbers displayed will not change until some system parameters are changed and the system re-IPLed.

The following notes point out key items of interest in performance and tuning work.

- The first information listed is similar to that shown with the SSTAT utility described above. Take note of the page pool size shown on the first line. That is the amount of storage available for user regions. User regions are the area that user jobs run in. A specific user job cannot use more any more real storage than shown in the MAXRRS number. The size of the page pool is determined by what is left over when all the other storage areas have been allocated.

- The length column in the table of addresses is the most important field for performance and tuning considerations. The amount used by the BPOOL area is important. Run the BPOOL utility at the end of a typical day to see the maximum number used. (The system must have been up during your peak times to have useful numbers here.) You can change the number of buffers in this pool by a parameter in the NUCGEN utility. Refer to the topic "Terminal Buffer Pool" in *Chapter 18 - System Internals* for more information.

- The LPA area shows the amount of storage being used by your fixed link pack area. Increasing this area reduces the amount of size of the page pool. Consult the section "Link Pack Area Considerations" in *Chapter 21 - Load Library and Link Pack Area* for information on this topic.

- The RAMDSK area shows the memory reserved for use as a RAM disk. Files are loaded into here when the system starts up and can be subsequently referenced without any physical I/O. Usually significant performance benefits can be gained by reserving about 400K of memory and loading in frequently used files. See the section "Configuring your RAM Disk" for more details.

- Cache all the Save Library space bit maps in memory, by specifying ULMAPS=1 in the NUCGEN utility.

- The trace area is set by a parameter in the NUCGEN utility and is usually no more than 20k.

- The nucleus area and low core areas are used by the system for its control code and buffers. These areas are not changed for tuning reasons.

- Ignore the page pool 1 and 2 numbers for tuning purposes. Use the page pool total on the first line instead.

Tuning Examples

The following are suggestions of what to do to improve specific performance bottlenecks.

Poor Response, CPU Usage High (eg 90%)

The demand for CPU cycles is greater than can be provided by your CPU. If this situation exists considerately throughout the day, you should consider reducing the number of users or upgrading your CPU. If it occurs periodically it may be caused by a group of users (like a student lab) running a specific piece of software. Have them use that software during non-prime time or spread its use out over the day rather than having them all run it at once. Maybe change from a lab format to a "work on your own" format.

Poor Response, CPU=40%, Swapping High

In this case the system may be I/O bound on the SWAP data sets. The output from the IOTIME program will indicate the amount of time spent swapping. There are various things that can be done to improve this. They are listed below in the order that we feel will have most effect.

- Give the MUSIC machine more main storage. This will enable the system to keep more users in storage and therefore reduce the swap load.

- Spread the swap load over more than one channel if possible. It is best to place the swap data sets on disks that have very little other activity on them. These disks should be on channels used exclusively by MUSIC. Some sites have found that some improvement can be obtained if the second (or third) SWAP data set is placed on a "low usage" VM channel. If this is attempted monitor the SWAP times very closely since if the channel becomes busy, the "solution" could be worse than the original problem.

- If you have specified a large MAXRRS value in the NUCGEN you may want to try reducing it. If a lot of people are using large region sizes making MAXRRS smaller can significantly reduce the amount of data that is swapped. This will increase the paging overhead for those using large user regions but will improve system performance for the majority of users.

- Optimize the location of the SWAP data sets if possible.
  - MUSIC owned channel (no interference from VM)
  - Dedicated disk (not minidisk)
  - Low usage pack if possible
  - Near center of pack

Poor Response, CPU=5%, I/O Rate Low

This is probably a problem in the I/O configuration or VM environment. MUSIC is spending most of its time in WAIT state. The key is to figure out what it is waiting for. Usually it is either waiting for an I/O interrupt from a terminal or a disk, or an external interrupt from a system timer. Whatever the case this situation should be reported as a system problem.

Response Ok Except for a Few Users

Look at the paging rates. It will usually indicate a significant amount of paging activity. The jobs that take a long time probably use a large user region and have a working set (of pages) larger than the MAXRRS. They are "thrashing". That is, spending the majority of time paging and doing very little useful work. Jobs like this do not seriously effect the performance of the rest of the system, however it can really be frustrating for the user since they have no idea of why their job is taking so long due to the exponential nature of the effect.

e.g. Waterloo Script Job (MAXRRS=232K)

100 page document.....5 minutes
300 page document.....2 hours
600 page document.....Cancelled after 18 hours.

There are two possible solutions.....

- If the thrashing program is reentrant then placing it in the FLPA will reduce the user region storage requirements by the size of the program and thus increase the available working storage.

- Increase the MAXRRS so that it is larger than the working set of the program. The value of MAXRRS effects other aspects of system performance. These should be monitored to determine whether the gains obtained by increasing MAXRRS are not outweighed by loses in other areas.

    - Increasing MAXRRS will reduce the number of tasks that can be kept in the page pool thus potentially increase the swap load. If you have sufficient main storage this should not be a problem.

    - If you have insufficient storage increasing MAXRRS will reduce the MAXMPL (maximum multiprogramming level). Usually a MAXMPL of 3 or 4 is sufficient.

    - The MAXRRS value indicates the maximum data transfer amount on a SWAP operation. If a lot of users are using large region sizes, increasing MAXRRS will increase the data transfer overhead for each swap operation. This may be offset however by substantial savings in paging overhead.

Poor Response at Peak Times

Performance is ok during most of the day except for the peak times. These peaks may last for half an hour or more. The response time monitor will show that the user region is active 100% of the time during these periods. CPU and I/O rates are not excessive and are similar to the ones you normally got from the machine at peak times before response started getting poor. The current situation is that the number of users (or the jobs they are running) have exceeded the capacity of your CPU current configuration. There is probably no obvious bottleneck. You must look at output of the performance measuring tools in more detail. The following is a list of things that you can try. The list is in order of what we feel will give you the most improvement.

- Give MUSIC as much main storage as possible and optimize its use as follows:

    - Place any high usage processors in the FLPA. Note the usage of processors such as VS/FORTRAN, and VS/PASCAL may vary depending on the time of year at academic sites. If you cannot fit them all in the FLPA, at least put the most used ones there. If you do not have any high usage reentrant processors, do not waste the storage by putting infrequently used modules in the FLPA. Use the output of the LDCNTS program to see which modules are being used the most. Use the LDLIST program to see the sizes of the individual modules. Refer to the topic "Link Pack Considerations" in *Chapter 21 - Load Library and Link Pack Area* for details on what processor each module is associated with.

    - If you have not already done so, reserve about 400K of memory for use as a RAM disk and load any frequently used files into it. Use the RAMREP utility to determine if the files loaded in RAM are being used frequently enough and replace any low usage files with others that you feel may have a higher hit count. A properly configured RAM disk should be able to handle 15% - 25% of file opens.

    - The number of RCBs effects the swap load. In adding main storage you should hope to increase the number of RCBs. Try to keep the number of RCBs to at least 25% of the number of active users. Once the key modules have been placed in the FLPA, additional storage is best spent in

additional RCBs. The number of RCBs is automatically determined from the size of the pageable storage pool. This pool contains the storage that is not allocated to other things like the FLPA, BPOOL, trace table, etc.

- Try to keep the MAXRRS to at least 232K. The system will reduce this value if there is insufficient storage left due to trying to put too much in the FLPA.

- If storage is at a premium remove any terminals, BTRMs and extra sessions (XSES) that are not required. Set the size of the BPOOL to be slightly higher than the high water mark indicated by the BPOOL program. Never let the BPOOL size be considerately equal to this high water mark. Reduce the trace table size (MAXTRC) to 8K.

- Improve MUSIC's virtual machine environment where possible.
  - Run MUSIC V=R or lock ALL its pages
  - Use dedicated disks not minidisks
  - If disks are on separate real channels, configure than as such on MUSIC.
  - Put all your MUSIC disks on different virtual channels. For example, if you disks are addresses 130, 131, 132, 134, try defining them to VM as 130, 230, 330, 430. This allows VM to better service the disks.
  - Do not use VM's multi-path support to access disks. This has shown to have a major negative effect on throughput.

- Improve MUSIC's disk configuration by adding channels or disks or moving high access data sets to channels or disks that are not so busy. If multiple real channels are used the swap and page load should be spread over all the channels. High access data sets such as SYS1.MUSIC.UIDX should be placed in the center of low activity disks on low activity channels. The IOTIME utility is useful in determining I/O activity on a data set by data set basis.

- Force a reduction in the user load. This is usually the least desirable though often the only resort once all of the above has been tried. There are two basic methods:

  - Restrict the usage of certain software to certain users or certain times. Subset environments can be set up using TMENU or REXX to restrict users to the assigned task and eliminate game playing or private projects which can consume resources.

  - Limit the number of users by limiting the number of terminals defined on the system. This may seem cruel but, when the peak time of the year rolls around and you have tried all of the above and response is still poor, it is probably better to have 100 people getting work done and 20 waiting for a terminal than to have 120 sitting at terminals getting work done slowly.

Appendixes

Appendix A. Console Messages and Wait State Codes

System Initialization and Generation Messages

This section lists the console messages that may be received from MUSIC during system initialization or system generation. They are listed in numerical order.

M001 ACCOUNTING FILE ERROR x. RE-IPL TO RETRY.
USE =RESET SPECIAL OPTION AS LAST RESORT

Explanation**:** An error has occurred when the system is processing the accounting data set. The code *x* further identifies the problem:

1 Blocksize is not 4096 bytes.
2 Accounting file not at least two records long.
3 The first record on the accounting file is incorrect. This record contains header information.
4 The accounting data set has changed size since last =RESET operation. (This message is issued by module DSINIT.)

Procedure**:** Reload MUSIC to retry. If error persists, inform system support personnel. The =RESET option will clear the file.

M002 AVAILABLE DISK DRIVES ARE:

Explanation**:** Self-explanatory. (This message is issued by module HELLO.)

M003 BAD BITMAP IN SAVE LIB DATA SET - LIB NO.=nnn

Explanation**:** The bitmap integrity check failed or bitmap's length was incorrect. As a result no user will be able to allocate space in this part of the Save Library. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel. The bitmap's length is in block 1, the bitmap starts at block 2, and can be up to 7 blocks in length.

M004 BAD FORMAT-4 DSCB - DRIVE=cuu VOL=vvvvvv

Explanation**:** The volume on drive *cuu* has a bad VTOC. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M005 CARD ALREADY DELETED - card image

Explanation**:** System response to a second request to delete the same card during a catalog edit. (This message is issued by module DSINIT.)

Procedure**:** None.

M006 CARD ALREADY DELETED DUE TO FORMAT ERROR - card image

Explanation**:** On catalog edit, card specified for deletion already been deleted due to a format error. (This message is issued by module DSINIT.)

Procedure**:** None.

M007 CARD DELETED-card image

Explanation**:** On catalog edit, the displayed card has been deleted. (This message is issued by module DSINIT.)

Procedure**:** None.

M008 CARD NOT FOUND IN CATALOG

Explanation**:** On catalog edit, the card requested for display or deletion could not be found. (This message is issued by module DSINIT.)

Procedure**:** Specify correct label for card to be listed or deleted.

M009 CATALOG CONTAINS NO ENTRIES

Explanation**:** No entries could be found in the system catalog data set. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M010 CATALOG DATA SET NOT FOUND - VOL=vvvvvv DRIVE=cuu DSN=dsn

Explanation**:** Catalog data set could not be found on the named volume. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M011 CATALOG TOO LARGE FOR WORKSPACE

Explanation**:** Catalog data set entries cannot all fit in the work area inside the module DSINIT. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M012 DATA SET NOT FOUND - VOL=vvvvvv DSN=dsn

Explanation**:** SYS1.MUSIC.NUCLEUS data set could not be located on the IPL volume. (This message is issued by module MFETCH and module WMON.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M013 DATA SET NOT FOUND - VOL=vvvvvv DSN=dsn
SYSTEM CANNOT RUN

Explanation**:** The named data set could not be found on the specified volume. If the second portion of the message does not appear, the system will attempt to run without it. The catalog entry referring to this data set may be in error, or the data set may not exist. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M014 DATA SET WILL NOT BE USED

Explanation**:** This is issued in conjunction with other Save Library error messages. (This message is issued by module DSINIT.)

Procedure**:** None.

M017 DO YOU WISH TO REPEAT THE EDIT PROCEDURE?

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Affirmative reply is 'YES'. Negative reply is 'NO' or blank line.

M018 DUPLICATE CATALOG ENTRY FOR system data set name

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M019 DUPLICATE CATALOG ENTRY FOR SAVE LIB DATA SET -
LIB NO.=nnn

Explanation**:** Two entries in the system catalog refer to the same Save Library extent number. (This message is issued by module DSINIT.)

Procedure**:** Reload MUSIC. Use the catalog edit procedure to fix the incorrect entry.

M020 DUPLICATE PSEUDO DEVICE CARDS FOUND -
PSEUDO TYPE=xxx

Explanation**:** Two catalog pseudo device entries specify the same pseudo device type. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M021 DUPLICATE VOLUME LABEL -
vvvvvv PACK ON DRIVE cuu NOT PROCESSED

Explanation**:** Two disk packs have the same volume name. The volume on named drive will not be used by the system. (This message is issued by module DSINIT.)

Procedure**:** If pack not processed had been mounted by mistake, no action is needed. Otherwise, stop the processing unit, mount the correct disk packs, and reload MUSIC.

M022 EDIT COMPLETE

Explanation**:** The system catalog edit operation has ended. (This message is issued by module DSINIT.)

Procedure**:** None.

M023 ENABLE THE INTERVAL TIMER

Explanation**:** Self-explanatory. (This message is issued by module HELLO.)

Procedure**:** Enable the interval timer.

M024 ENABLE TOD CLOCK TO CONTINUE

Explanation**:** The system is waiting for the switch on the processing unit to be placed in the 'ENABLE TOD' clock position. (This message is issued by module HELLO.)

Procedure**:** Perform the required operation.

M025 ENTER ADDITIONAL CATALOG CARDS (AS SHOWN BELOW),
OR BLANK LINE
LABEL... XNNN NN XXX NNN VVVVVV DATA.SET.NAME.....

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Enter card image or blank line if no additions are to be made to the system catalog.

M026 ENTER CARD LABELS TO BE DISPLAYED OR "ALL" OR
BLANK LINE

Explanation**:** System response to request to edit the catalog. (This message is issued by module DSINIT.)

Procedure**:** Enter labels of catalog entries to displayed, or 'ALL' if all catalog entries are to be displayed on the system console.

M027 ENTER CARDS TO BE DELETED OR BLANK LINE

Explanation**:** Self-explanatory message issued during catalog edit. (This message is issued by module DSINIT.)

Procedure**:** Enter labels of entries to be deleted if any, or blank line if none are to be deleted.

M029 ERROR DURING DSCB SEARCH - VOL=vvvvvv DRIVE=cuu DSN=dsn

Explanation**:** An error has occurred during a search of the VTOC of volume mounted on specified drive for the displayed data set name. Other explanatory messages may be issued. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M030 ERROR IN CATALOG CARD(S)

Explanation**:** Errors have been found in one or more catalog entries. Other explanatory messages precede this one. (This message is issued by module DSINIT.)

Procedure**:** Re-enter correct cards if known, otherwise, retry, and if error persists, Inform system support personnel.

M031 ERROR ON SYSTEM DEVICE - DRIVE=cuu

Explanation**:** An error has occurred on the system residence device on the named drive. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel and hardware support personnel.

M032 ERROR READING F-4 DSCB - DRIVE=cuu VOL=vvvvvv

Explanation**:** An error has occurred during an attempt to read the VTOC of the pack on the named drive. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M033 ERROR READING LABEL - DRIVE=cuu

Explanation**:** An error has occurred during an attempt to read the volume label of the pack on named drive. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M034 ERROR READING LOAD LIBRARY DATA SET

Explanation**:** An I/O error has occurred during an attempt to read the directory in the Load Library data set. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M035 ERROR...RE-ENTER

Explanation**:** The time or date is incorrect. Year must be in the range of 1980 to 2010. (This message is issued by module HELLO.)

Procedure**:** Correct the time or date and if necessary consult a calendar for the correct day of the week.

M037 INSUFFICIENT CATALOG ENTRIES FOR system data set name MINIMUM=nnn

Explanation**:** The specified data set type requires at least nnn catalog entries. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M038 INSUFFICIENT MAIN STORAGE TO INITIALIZE SYSTEM.

Explanation**:** Insufficient main storage for terminals, functions and devices. (This message is issued by module DSINIT.)

Procedure**:** Rerun the NUCGEN Utility with corrected configuration cards.

M039 INSUFFICIENT MAIN STORAGE FOR LOAD LIBRARY DIRECTORY

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Decrease main storage requirement of MUSIC.

M040 INSUFFICIENT MAIN STORAGE FOR RESIDENT PROGRAM TABLE

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Decrease main storage requirement of MUSIC.

M041 INVALID BLKSIZE IN SAVE LIB DATA SET - LIB NO.=nnn BLKSIZE=bbbbb SHOULD BE 512

Explanation**:** The blocksize of the Save Library data set was not 512 bytes. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M042 INVALID BLKSIZE IN SAVE LIB INDEX DATA SET - BLKSIZE=nnnnn SHOULD BE 512

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M044 MUSIC REQUIRES ECMODE ON VM

Explanation**:** Self-explanatory. (This message is issued by module HELLO.)

Procedure**:** Use the VM "SET ECMODE ON" command or change MUSIC'S VM directory entry to

specify ECMODE.

M045 USING NEW CATALOG DATASET

Explanation**:** The =CATxxxx special option has been used. (This message is issued by module HELLO.)

Procedure**:** None.

M046 INVALID DISK DEVICE TYPE - DRIVE=cuu

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Rerun the NUCGEN Utility with corrected configuration cards.

M047 INVALID FORMAT-4 DSCB

Explanation**:** The VTOC of the IPL volume is invalid. (This message is issued by module MFETCH and module WMON.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M048 INVALID VOLUME LABEL

Explanation**:** The volume label of the IPL volume is invalid. (This message is issued by module MFETCH and module WMON.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M049 INVALID HEADER IN LOAD LIBRARY DATA SET

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M050 INVALID LABEL IN SAVE LIB DATA SET -
LIB NO.=nnn LABEL=label

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M051 INVALID LABEL IN SAVE LIB INDEX DATA SET - LABEL=label

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M052 INVALID TIME/DATE STAMP IN SAVE LIB DATA SET -
LIB NO.=nnn TIME=hh.mm.ss DATE=ddmmmyy

Explanation**:** The date stamp on the Save Library index does not match that on all the extents. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M053 INVALID VTOC EXTENT

Explanation**:** MUSIC system disk pack VTOC's must occupy only one cylinder. Probably caused by specifying incorrect specifications when the pack was initialized. It is most unlikely that this error will result for any other reason. (This message is issued by module MFETCH and module WMON.)

Procedure**:** Retry. If the error persists, inform the system support personnel. The system support personnel will have to run the DSF utility to correct this error.

M054 I/O ERROR

Explanation**:** Self-explanatory. Issued in conjunction with other messages. (This message is issued by module DSINIT.)

Procedure**:** None.

M056 I/O ERROR READING FORMAT-4 DSCB

Explanation**:** Self-explanatory. (This message is issued by module MFETCH and module WMON.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M057 I/O ERROR READING VOLUME LABEL

Explanation**:** Self-explanatory. (This message is issued by module MFETCH and module WMON.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M058 I/O ERROR SEARCHING VTOC

Explanation**:** An I/O error occurred while searching for the SYS1.NUCLEUS data set on the IPL volume. (This message is issued by module MFETCH and module WMON.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M059 I/O ERROR WRITING DATA SET NAME LIST TO SYSRES IGNORED

Explanation**:** Self-explanatory. System will still run. (This message is issued by module DSINIT.)

Procedure**:** None, the system support personnel should be notified of this message.

M060 MISSING CATALOG ENTRY FOR system data set name

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M061 MISSING CATALOG ENTRY FOR SAVE LIB DATA SET - LIB NO.=nnn

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Reload MUSIC and edit the catalog to provide the missing entry.

M062 MISSING INDEX DATA SET FOR SAVE LIBRARY

Explanation**:** Self-explanatory. Probably caused by a system disk pack not available. (This message is issued by module DSINIT.)

Procedure**:** Make the required disk available and reload MUSIC.

M063 MISSING SAVE LIB DATA SET - LIB NO.=nnn

Explanation**:** Self-explanatory. Probably caused by a system disk pack not available. (This message is issued by module DSINIT.)

Procedure**:** Make the required disk available and reload MUSIC. If not possible, MUSIC will run but some user files will not be available. It is not recommended to run in this fashion.

M064 MORE THAN ONE EXTENT - VOL=vvvvvv DRIVE=cuu DSN=dsn

Explanation**:** Specified data set occupies more than one extent on the named volume. System data sets must only occupy one extent. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M065 NEW SAVE LIB DATA SET HAS NONZERO DATE STAMP - NOT ADDED. LIB NO.=nnn

Explanation**:** During initialization, the system found a new Save Library data set in the catalog, which, according to its number, should be a new data set. However, the time/date stamp in block 1 of the data set was not zero, which indicates that the data set is not a new data set. The data set was not added to the library. (This message is issued by module DSINIT.)

Procedure**:** Verify that the catalog is correct. If the data set actually is new, verify that it was formatted (via the FORMAT utility) and initialized (via the ULINIT utility) correctly. Make any necessary corrections and re-IPL.

M066 MUSIC/SP, LEVEL=aaaa

Explanation**:** Informative message. The level identification is that set when the nucleus generation

procedure (NUCGEN) was performed. (This message is issued by module HELLO.)

M067 NEW SAVE LIB DATA SET ADDED - LIB NO.=nnn

Explanation**:** Informative message indicating a new extent has been added to the Save Library. (This message is issued by module DSINIT.)

Procedure**:** None.

M068 NEW SAVE LIB INDEX DATA SET INITIALIZED - DATE=ddmmmyy TIME=hh.mm.ss

Explanation**:** Informative message. (This message is issued by module DSINIT.)

Procedure**:** None.

M069 NO ATTEMPT TO ALLOC SPACE WILL BE MADE ON THIS LIB EXTENT

Explanation**:** Informative message. Issued in conjunction with other messages. (This message is issued by module DSINIT.)

Procedure**:** None.

M070 NO DISK DRIVES ARE AVAILABLE FOR MOUNTS.

Explanation**:** Informative message. (This message is issued by module HELLO.)

M071 NOT ENOUGH MAIN STORAGE FOR CONFIGURED SYSTEM.

Explanation**:** Insufficient main storage for terminals, functions, and devices. (This message is issued by module HELLO.)

Procedure**:** Rerun the NUCGEN Utility with corrected configuration cards.

M073 TEMPORARY WORKFILE DATA SET AVAILABLE

Explanation**:** Self-explanatory. System will run, but in a degraded mode. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M074 NO TERMINALS WILL BE STARTED

Explanation**:** Response to a =NOTERM request (This message is issued by module HELLO.)

Procedure**:** None. To start terminals at a later time issue the /ADD console command.

M075 NO VOLUMES ARE PERMANENT.

Explanation**:** Self-explanatory. (This message is issued by module HELLO.)

Procedure**:** None

M076 (c) Copyright 1989-19xx, McGill University, Montreal, Canada

Explanation**:** Copyright message. For full text of copyright message see file $SUB:COMUC.S as well as other locations in the system. (This message is issued by module HELLO.)

Procedure**:** None

M077 ENTER OPERATOR ID OR SPECIAL OPTIONS OR HELP

Explanation**:** Self-explanatory. (This message is issued by module HELLO.)

Procedure**:** Enter operator ID. The ID is not checked but must be non-blank. Possible special options are =CATxxxx, =RESET, =NOTERM, =EDIT, and =CONFIG. Entry of all blanks means that all the default options will be used. Refer to *Chapter 3 - Loading the System* for further information.

M078 PSEUDO DEVICE TYPE=xxx NEEDED FOR DSN=dsn

Explanation**:** Catalog entry for listed data set name requires pseudo device type xxx. There is no entry for this pseudo device type, or pseudo device type requested is incorrect. (This message is issued by module DSINIT.)

Procedure**:** Correct catalog entry.

M080 REQUIRED DATA SET NOT FOUND ON IPL VOLUME - DSN=dsn

Explanation**:** Required data sets named cannot be found on the IPL volume. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M081 REQUIRED IPL VOLUME DATA SET - IMPROPER FORMAT - DSN=dsn

Explanation**:** Format of required IPL volume data set is incorrect. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M082 REQUIRED VOLUME NOT MOUNTED - VOL=vvvvvv

Explanation**:** A catalog VOLUME card specified that this named volume was to be mounted at IPL time. (This message is issued by module DSINIT.)

Procedure**:** Mount the named volume, if required, and perform an IPL operation.

M086 SAVE LIB DATA SET IS TOO SMALL -
LIB NO.=nnn NREC=xxxxx

Explanation**:** Self-explanatory. This error is most likely caused by having a 3330 pack mounted on a device address that was defined to MUSIC as a 2314 or similar mismatch of device types. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel. The system support personnel can use the reconfiguration option at IPL time to correct any device type mismatches and then rerun the NUCGEN utility to permanently correct the problem.

M087 SAVE LIB INDEX DATA SET IS TOO SMALL - NREC=nnnnn

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M088 NO GOOD SWAP DATA SET FOUND--
NO USER REGION SWAPPING WILL BE DONE

Explanation**:** Self-explanatory. System errors will result. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M089 SWAP DATA SETS MUST HAVE BLOCKSIZE OF 4096 BYTES

Explanation**:** Self-explanatory. The following M106 message will identify the incorrect data set. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M090 CONFIGURATION INCORRECT: NO DISK DRIVES DEFINED

Explanation**:** Self-explanatory. (This message is issued by module HELLO.)

Procedure**:** Rerun the nucleus generation program (NUCGEN).

M091 DEVICE NOT OPERATIONAL OR I/O ERROR - DRIVE=cuu

Explanation**:** Problems accessing the disk on the specified device. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel and hardware support personnel.

M092 SYSTEM INITIALIZATION COMPLETING.

Explanation**:** Informative message to indicate initialization completing. This message is given when the startup routines (HELLO and DSINIT) have finished their work and are passing control to the main MUSIC resident supervisor. At this time the JOBONE job is started to perform some more initialization

functions such as the loading of the PLPA area. The system will be ready to run normal user jobs as soon as either M300 BATCH IDLE message is given or the M306 message is printed to show the /ID of the first batch job. (This message is issued by module HELLO.)

Procedure**:** None

M093 SYSTEM INITIALIZATION TERMINATED

Explanation**:** Summary message. System initialization could not be accomplished. (This message is issued by module DSINIT.)

Procedure**:** Follow procedure for messages issued prior to this one.

M094 SYSTEM INITIALIZATION TERMINATED DUE TO MISSING DATA SETS OR VOLUMES

Explanation**:** System initialization could not be accomplished because required data sets or volumes could not be found. (This message is issued by module DSINIT.)

Procedure**:** Follow procedure for messages preceding this one.

M095 IPLD DEVICE OF uuu NOT IN I/O CONFIGURATION

Explanation**:** The disk drive from which the system was loaded was not defined in the MUSIC nucleus. (This message is issued by module HELLO.)

Procedure**:** Mount the system residence pack (MUSICX) on a disk drive defined in the nucleus or use the re-configuration (=CONFIG) feature at IPL time to allow for this address.

M096 THE FOLLOWING SYSTEM DATA SETS WILL BE USED

Explanation**:** Informative message. Following this message, all data set names for which logging has been requested in their catalog entries will be listed on the system console. (This message is issued by module DSINIT.)

Procedure**:** None.

M097 THE FOLLOWING VOLUMES ARE PERMANENT:

Explanation**:** All permanently disk volumes are listed. (This message is issued by module HELLO.)

Procedure**:** None

M098 PAGE DATA SETS MUST HAVE BLOCKSIZE OF 4096 BYTES

Explanation**:** Self-explanatory. The following M106 message will identify the incorrect data set. (This message is issued by module DSINIT.)

Procedure**:** Inform system support personnel.

M099 PAGE DATA SETS MUST BE AT LEAST 8 PAGES LONG

Explanation**:** Self-explanatory. The following M106 message will identify the incorrect data set. (This message is issued by module DSINIT.)

Procedure**:** Inform system support personnel.

M100 TWO OR MORE FORMAT-4 DSCBS - DRIVE=cuu VOL=vvvvvv

Explanation**:** Named volume has a bad VTOC. Alternately, it is a VM pack. In either case, the volume will not be used by MUSIC. (This message is issued by module DSINIT.)

Procedure**:** Ignore this message if the pack is not required for MUSIC. Otherwise, retry, and if error persists, inform system support personnel.

M101 UNABLE TO LOAD PROGRAM x. REASON=rrr

Explanation**:** Problems where encountered trying to load the named program from the system load library. This message is given when the system is loading the FLPA and PLPA areas. The reason field identifies the cause of the problem. Reasons are attempting to load a non-reentrant program into the PLPA area, program not found on the load library, the page data set is full, I/O error during load and insufficient main storage to hold the program. (This message is issued by module DSINIT.)

Procedure**:** The system will keep running. Inform the system support personnel.

M102 VM COMMAND ERROR. RC=rrr

Explanation**:** VM detected error in one of the /CP commands in the MUSIC catalogue. The command in error was printed in the last M103 message. The VM return code is printed as part of the message. (This message is issued by module DSINIT.)

Procedure**:** Inform system support personnel.

M103 VM COMMAND ISSUED: command

Explanation**:** The listed VM command was issued. This commands from the system catalog. (This message is issued by module DSINIT.)

Procedure**:** None.

M104 VM TIME OF DAY CLOCK LOOKS BAD

Explanation**:** The date obtained from VM was not in the range 1980-2010. This message will also be given when the STORE TOD CLOCK (STCK) did not get an acceptable condition code under VM. (This message is issued by module HELLO.)

Procedure**:** Either reload VM with the correct date or you may enter the time and date to MUSIC in the usual fashion.

M105 vvvvvv ON UNIT nnn ....

Explanation**:** Informative message. Follows M097 and lists the disk volumes by name and address. Additional information may be shown. SYS means that the volume contains some MUSIC system data sets, UDS means that the volume does not contain active MUSIC system data sets, NDS means that users can allocate new UDS files on this volume, PRV means that users cannot refer to data sets on this volume even when the data set naming convention is followed. R/O means the volume is accessed in read-only mode. The NDS, R/O and PRV attributes are taken from the VOLUME card specified in the MUSIC system catalog. (This message is issued by module HELLO.)

M106 VOL=vvvvvv DSN=dsn

Explanation**:** Issued in conjunction with other messages. (This message is issued by module DSINIT.)

Procedure**:** Follow procedure (if any) for messages issued in conjunction with this one.

M107 VTOC CI TOO BIG FOR MUSIC

Explanation**:** The VTOC has a control interval of greater than 1024 bytes. This message is issued for FBA disks only. (This message is issued by module DSINIT.)

Procedure**:** Inform the system support personnel.

M108 SWAP DATA SETS TOO SMALL TO HOLD EVEN ONE SWAP SET

Explanation**:** A swap set consists of 10 blocks of 4096 bytes. The following M106 message will identify the incorrect data set. (This message is issued by module DSINIT.)

Procedure**:** Inform system support personnel.

M109 NO GOOD PAGE DATA SETS FOUND--NO PAGING WILL BE DONE.

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Inform system support personnel.

M110 WRONG HASH NUMBERS IN SAVE LIB INDEX DATA SET

Explanation**:** Integrity check failed. (This message is issued by module DSINIT.)

Procedure**:** Retry. If the error persists, inform the system support personnel.

M111 2314 ON nnn
3340 ON nnn
2305.1 ON nnn
2305.2 ON nnn
3350 ON nnn
3375 ON nnn
3380 ON nnn
F512 ON nnn

Explanation**:** The device type and the physical address of an available disk drive is displaced in the message. This drive may be used by batch programs for mountable User Data Set (UDS) volumes. (This message is issued by module HELLO.)

Procedure**:** None.

M112 DISK I/O ERROR

Explanation**:** An I/O error was detected while attempting to read in the MUSIC system nucleus. (This message is issued by module MFETCH and module WMON.)

Procedure**:** Retry. If error persists, rewrite system nucleus.

M113 BATCH NOT AVAILABLE. NO DISK SPACE ALLOCATED

Explanation**:** No spooling space has been allocated for batch. (This message is issued by module HELLO.)

Procedure**:** None.

M115 TEMPORARY SYSTEM I/O RE-CONFIGURATION.
ENTER ALL ADDRESSES IN THE FORM 'CUU'

Explanation**:** The =CONFIG special option has been selected. (This message is issued by module HELLO.)

Procedure**:** Follow the prompt messages that will be printed.

M116 INVALID BLOCKSIZE FOR SPOOL DATA SET

Explanation**:** The batch spooling data sets must have a blocksize of 512 bytes. (This message is issued by module DSINIT.)

Procedure**:** Change the blocksize of the data set.

M117 DATA SET NOT ON 32 BLOCK BOUNDARY. DSN=dsn

Explanation**:** All MUSIC data sets to be used by MUSIC must begin on 32 block boundaries on FBA disks. (This message is issued by module DSINIT.)

Procedure**:** Correct the incorrect data set.

M118 PREPARING TO RESET (WIPEOUT) ACCOUNTING DATA SET

Explanation**:** Operator has used =RESET special option. This message warns the operator that the accounting data set will be cleared. (This message is issued by module DSINIT.)

Procedure**:** None, if the reset is to be performed. If the reset is not wanted, do another IPL.

M119 INVALID RESIDENCE AREA

Explanation**:** An internal table describing the layout of the MUSIC nucleus is invalid. (This message is issued by module MFETCH.)

Procedure**:** Rewrite system nucleus.

M123 MUSIC RUNNING V=R, NOTRAN SET ON

Explanation**:** MUSIC has detected a V=R environment and has set NOTRAN on. CCW translation will be bypassed. (This message is issued by module HELLO.)

Procedure**:** None.

M124 PRINTER BUFFERS WILL BE LOADED

Explanation**:** Issued as a response when the =LOADPRT option is specified at IPL time. (This message is issued by module HELLO.)

Procedure**:** The system will prompt for further information later in the initialization sequence.

M125 ENTER FCB NAME

Explanation**:** The operator is requested to enter a four character name of the FCB to be loaded into the printer. (This message is issued by module DSINIT.)

Procedure**:** Enter the FCB name. See the topic on "Initializing the Printer" in *Chapter 3* for valid responses.

M126 PRINTER SPOOLED, UTILIZE VM LOADBUF COMMAND

Explanation**:** =LOADPRT was specified at IPL time but the printer is a spooled device under the control of VM. (This message is issued by module DSINIT.)

Procedure**:** Printer should be initialize using facilities offered by VM. See *Chapter 2 - Running MUSIC/SP under VM* for details on the VM LOADBUF command.

M127 NO FCB WILL BE LOADED

Explanation**:** A blank line was entered in response to message M125. (This message is issued by module DSINIT.)

Procedure**:** None, unless the blank line was entered accidentally, in which case the IPL procedure should be repeated.

M128 FCB IMAGE NOT FOUND, RE-ENTER

Explanation**:** The response to M125 was not a valid FCB name. (This message is issued by module DSINIT.)

Procedure**:** Enter a valid FCB name. See Chapter 3, the topic on "Initializing the Printer" for valid responses.

M129 ERROR ON BUFFER LOAD

Explanation**:** An error occurred while loading the printer buffer. The IPL sequence will continue since terminals can function without the printer. (This message is issued by module DSINIT.)

Procedure**:** Correct the problem with the printer and re-try the buffer load operation. The buffer can be loaded using the SETBUF program, or by an IPL.

M130 ENTER UCS NAME

Explanation**:** The operator is requested to enter a four character name of the UCS to be loaded into the printer. (This message is issued by module DSINIT.)

Procedure**:** Enter the UCS name. See Chapter 3, topic "Initializing the Printer" for valid responses.

M131 NO UCS WILL BE LOADED

Explanation**:** A blank line was entered in response to message M130. (This message is issued by module DSINIT.)

Procedure**:** None, unless the blank line was entered accidentally, in which case the IPL procedure should be repeated.

M132 UCS IMAGE NOT FOUND, RE-ENTER

Explanation**:** The response to M130 was not a valid UCS name. (This message is issued by module DSINIT.)

Procedure**:** Enter a valid UCS name. See Chapter 3, topic "Initializing the Printer" for valid responses.

M133 UNKNOWN PRINTER TYPE - NOTHING LOADED

Explanation**:** The printer is not supported by MUSIC. (This message is issued by module DSINIT.)

Procedure**:** Inform the systems personnel. For a list of supported printers, see Chapter 3, the topic on "Initializing the Printer".

M134 WRONG UCS FOR THIS PRINTER, RE-ENTER

Explanation**:** The UCS name specified as a response to M130 is not valid for this type of printer. (This message is issued by module DSINIT.)

Procedure**:** Enter a valid UCS name. See Chapter 3, the topic on "Initializing the Printer" for valid responses.

M135 FOLD OPTION ON BUFFER LOAD - YES/NO (DEFAULT NO)

Explanation**:** The operator is asked if the FOLD option is to be used while loading the printers buffer. The FOLD option causes lower case characters to be printed as upper case. (This message is issued by module DSINIT.)

Procedure**:** Respond YES or NO as required. A blank line is treated as NO.

M136 UNRECOVERABLE PRINTER ERROR NOTHING LOADED

Explanation**:** Printer is not operational. The IPL will continue since terminals still function without it. (This message is issued by module DSINIT.)

Procedure**:** Fix the problem with the printer.

M137 PRINTER NOT READY, READY DEVICE AND RESPOND GO

Explanation**:** The printer was not ready during the buffer load. (This message is issued by module DSINIT.)

Procedure**:** Ready the printer and reply GO.

M138 KEYS ARE NOT SUPPORTED ON DISKS aaa

Explanation**:** Data sets with keys are not supported on MUSIC. (This message is issued by module DSINIT.)

Procedure**:** None.

M139 STORAGE SIZE xxxxK. PAGEABLE STORAGE nnnnK

Explanation**:** The size of the main storage size available to MUSIC is shown. Also given is the amount of main storage that can be dynamically assigned to user programs. (This message is issued by module HELLO.)

Procedure**:** None.

M140 MAXMPL= nn, MAXRRS= nnnk, NUM RCBS= nnn

Explanation**:** At IPL time, the system sets a number of items based on the available storage. This message display some of these items.

The maximum multi-programming level (MAXMPL) number is the number of concurrently active jobs the system will handle. The maximum real region size (MAXRRS) shows the maximum real storage size the system will dynamically assign to any active job. The number of Region Control Blocks (RCBS) is also shown. The system uses the RCBs to contain information of jobs that are actively running or are in inactive status and are resident in main storage (i.e. not swapped out). (This message is issued by module HELLO.)

Procedure**:** None.

M141 NOT ENOUGH MAIN STORAGE LEFT FOR USER REGIONS.

Explanation**:** The size of the main storage size available to MUSIC is insufficient to run jobs. (This message is issued by module HELLO.)

Procedure**:** Give MUSIC more storage or remove items from the FLPA.

M142 WARNING: SAVE LIBRARY INDEX OVERFLOW AREA ALMOST FULL.
NUM FREE BLOCKS=nnnn

Explanation**:** Self-explanatory. (This message is issued by module DSINIT.)

Procedure**:** Inform system support personnel. They can run the following cleanup programs: $PGM:LIBINDEX.CLEAN1.S and $PGM:LIBINDEX.CLEAN2.S.

M143 Batch internal reader has been turned off

Explanation**:** Informative message in response to the =RDROFF IPL option.

Procedure**:** None.

M144 Starting RAM DISK load

Explanation**:** The RAM DISK area in memory is being loaded. This is issued by JOBONE at startup time if a RAM DISK area is defined. It is also issued when you run the RAMDLD utility.

M145 RAM DISK load finished

Explanation**:** The RAM DISK area has been loaded.

M146 Cannot open file nnnnnn

Explanation**:** The file nnnnnn could not be opened during the RAM DISK load. Check that the file exists.

M147 No space left to load file nnnnnn

Explanation**:** There is not enough RAM DISK space left to load the file nnnnnn. Increase the RAMDSK parameter in the NUCGEN if you want to load more files. The system will attempt to load subsequent files in the list and will issue this message for each file that is too big to fit.

M148 Error reading file nnnnnn

Explanation**:** An error occurred while loading the file nnnnnn into the RAM DISK area. The most likely cause of this is that the RAMDLD utility is being used to load the disk but VIP has not been turned on. (VIP is required to read the directory blocks of the files)

M149 RAM DISK is to small to use

Explanation**:** A RAM DISK area has been defined in the NUCGEN but it is too small to be of any use. The RAMDSK parameter should be at least 32K.

M150 Cannot open RAMDISK.LIST

Explanation**:** A RAM DISK area has been defined in the NUCGEN but the RAMDISK.LIST file can not be opened. This file should contain the list of files that are to be loaded into the RAM DISK area. Check that the file exists.

M151 Error reading RAMDISK.LIST

Explanation**:** A file error occurred reading the file RAMDISK.LIST during the RAM DISK load. Check that the file exists and is readable.

M152 RAM DISK index area is filled

Explanation**:** The memory reserved for the RAM DISK index is filled. Increase the RAMDSK parameter in the NUCGEN if you want to add more files.

M153 Starting TEMP file cleanup

Explanation**:** Issued by JOBONE at startup time. The system scans the file index and deletes any temporary files that were perhaps left allocated due to a system failure or sudden shutdown.

M155 Loading PLPA

Explanation**:** Issued by JOBONE at startup time. This informs you that the Pagable Link Pack Area (PLPA) is being loaded. Selected modules are copied from the Load Library to the Page Datasets.

Terminal I/O Messages

M200 I/O ERROR ADDR=020 CAW=xx CC=y CSW=zzzz SI=ss OPCD=oo

Explanation**:** An I/O error occurred on the specified terminal address. The channel address word (CAW), sense information byte (SI), the channel status word (CSW) are given if available. The condition code associated with the SIO is given in the CC portion. The op code of the failing CCW is identified in the OPCD portion. This message may be followed by a DROPPED message. (This message is issued by module TCS.)

For non-3270 terminals, the sense bit meanings are as follows:

80 Command Reject. Invalid CCW command(s) issued by MUSIC to a terminal. The sequence of events causing the error should be reported to the system administrator.
40 Intervention Required. A telecommunications data set has become not ready either due to the telephone connection being severed or by the data set being powered off. This is not an unusual condition unless it happens very often on a particular line or many lines simultaneously.
20 Bus Out Parity Error. This is a hardware error and should be reported to the service personnel.
10 Equipment Check. An error has occurred within the 270x or 370x communications controller. Report this occurrence to the service personnel.
08 Data Check. Errors were detected in the data being transmitted to or from remote terminals. The error was probably caused due to noise on the communications channel.
04 Overrun. The computer could not handle the data as fast as it was being received. Report this problem to the service personnel as it is probably caused by hardware malfunction or configuration error.
02 Lost Data. Data was received from a terminal when MUSIC was not expecting any. Most likely caused by typing input at a terminal which has not requested any input.
01 Timeout. A reply was expected from a terminal within a specific time but was not received. This can be caused by a user failing to sign on within the allowed 30 second limit. This error may also be caused by a terminal being powered off or a malfunction of the remote terminal.
00 Probably reading paper tape and no tape off character after the line end signal.
FF Not Operational. The telecommunications control unit has become non-operational. If the control unit was not intentionally disabled, then this should be treated as a hardware malfunction.

For 3270 terminals, the sense bit meanings are as follows:

80 Command Reject. Invalid CCW command(s) issued by MUSIC to a terminal. The sequence of events causing the error should be reported to the system administrator.
40 Intervention Required. The device is in *not ready* status.
20 Bus Out Check. A parity check was detected. This is a hardware error and should be reported to the service personnel.
10 Equipment Check. This is a hardware error. Check the device for a blown fuse, etc.
08 Data Check. A cursor or parity check was detected.
04 Unit Specify. Indicates a hardware error. It might just be a transient condition. It might also require that the control unit be powered off and then on.
02 Control Check. The terminal failed to respond in a specified time. This may indicate a hardware error or that the device is in test mode.
01 Operation Check. The terminal received invalid data such as an incorrect buffer address in a data stream. It can also be caused by incorrect double byte character sequence used with a DBCS language such as Japanese.

M201 TERMINAL nnn DROPPED, ADDR=xxx

Explanation**:** See the above discussion. (This message is issued by module TCS.)

Procedure**:** See the above discussion.

M202 INVALID 3270 DATA STREAM FROM DEVICE xx

Explanation**:** An invalid 3270 data stream has been received from the specified device. An example of such an error is the data stream ending in the middle of the 2 byte field location. (This message is issued by module FSIO.)

Procedure**:** Report this to your system support personnel.

M203 TERMINAL AT xxx IS NOT OPERATIONAL

Explanation**:** This message is issued if a 3270 type terminal is detected to be not operational when the line is enabled. This situation could arise if the control unit is not on, if the terminal is not defined under VM. MUSIC will retry when the device becomes available. (This message is issued by module TCS.)

Procedure**:** Investigate why the terminal is not operational.

Batch Processing Messages

M300 BATCH IDLE

Explanation**:** All batch jobs have been processed and all output has been printed and the card reader is inactive. (This message is issued by module SPAM.)

Procedure**:** None.

M301 DISK ERROR, JOB SKIPPED

Explanation**:** Disk error occurred while writing on batch input area. (This message is issued by module SPAM.)

Procedure**:** Rerun the job. If the error continues, follow installation procedures.

M303 EXCESSIVE INPUT, JOB SKIPPED

Explanation**:** Batch job input has exceeded the allocated disk spooling space. (This message is issued by module SPAM.)

Procedure**:** Advise user. The disk spooling space can be enlarged by the system support personnel if the current size presents a problem. He needs enlarge the data set SYS1.MUSIC.BATCHIN.

M304 FLUSHING PURGE JOB

Explanation**:** A job has been read that contains a /PURGE card and the switch has been set to ignore such jobs. (This message is issued by module SPAM.)

Procedure**:** If the job that contained the /PURGE command should be allowed, then allow purge jobs by using the /CTL PURGE console command and re-read the job. The console command /CTL NOPURGE should be typed if you wish to return to the condition that purge jobs from batch will be ignored.

M305 HELP ddd aaa SI=ss

Explanation**:** Device type d (RDR, PRT, PCH) requires operator assistance. The device address is given in the *a* field of the message with the sense information in the *s* portion. (A "Help RDR" message will occur immediately after the system has been initialized if no batch reader is empty.) This message can also occur if the unit has become unavailable to the processing unit due to the fact it has been switched off the channel. Common sense bit patterns follow:

00 Device is not operational. Its address does not currently exist on the channel.
40 Intervention required. The device is in the not ready status either because the stop key has been depressed or the device is out of paper or a jam condition has occurred.
20 Bus-out check. Hardware service personnel should be notified.
10 Equipment check. Hardware service personnel should be notified.
04 On the printer means a UCS parity check. Reload the UCS buffer. If the error persists after this has been done, then the hardware service personnel should be notified.
02 No channel found. The printer's form control buffer does not have the required skip channel identi-

fied. (MUSIC uses channels 1 and 12.)

(This message is issued by module MIOX.)

Procedure**:** Ready the designated unit after correcting the error. The system will automatically resume input or output.

M306 /ID . . .

Explanation**:** Compressed listing of the /ID read from batch reader. Note some /ID's may not print if the jobs are being read in faster than the console can print. (This message is issued by module SPAM.)

Procedure**:** None

M308 LOADING UCS FOR uuu
LOADING FCB FOR fff

Explanation**:** These messages report that either the UCS or the FCB buffer has been loaded for the batch printer. The UCS buffer is normally loaded when the print chain is changed. The FCB (FORMS CONTROL BUFFER) indicates the length of the form on the printer and is normally loaded if a different size paper stock is to be used.

Procedure**:** Issue the "/REPLY 0" command once the correct chain or paper has been loaded on the printer.

M309 M nnn,xxxxxx
D nnn,xxxxxx

Explanation**:** M means mount, D dismount. *nnn* is the physical device address of the unit. *xxxxxx* is the volume name of the disk pack or tape that should be mounted or dismounted. (This message is issued by module MOUNT.)

Procedure**:** Mount or dismount specified volume. You should never mount a tape before the system specifically asks for it.

M310 /PAUSE text

Explanation**:** Batch job being run has a /PAUSE card for operator intervention. (This message is issued by module SPAM.)

Procedure**:** Follow instructions printed, then type /GO when ready to continue. Type /NOGO to resume batch processing and to cancel the job that has this control card in it.

M311 PUNCHING CARDS FOR xxxxxxx

Explanation**:** Job currently being run for user *xxxxxxx* has punched output. (This message is issued by module SPAM.)

Procedure**:** Remove cards from stacker when done.

M312 JOB DELETED DUE TO SYSTEM I/O ERROR

Explanation**:** Some I/O error was detected on either batch's input or output spooling areas. The batch job has been cancelled. (This message is issued by module USRSVC.)

Procedure**:** An I/O error message has also been printed. Follow the procedure associated with that message.

M313 BATCH JOB - CODE xxxxxxx NOT AUTHORIZED

Explanation**:** The code on the batch job being read is not authorized, or the code has expired, or the code is not allowed to run batch jobs (batch time limit = 0), or it requires the operator command "/CTL CD-ON". The job will not be run. (This message is issued by module CKCDE.)

Procedure**:** None.

M314 PUT UNIT ONLINE THEN TYPE "/VARY xxx ON"
CANCEL JOB WITH "/CAN"

Explanation**:** The specified unit was was offline. (This message is issued by module MOUNT.)

Procedure**:** Put it online. This may involve issuing a VM "ATTACH aaa TO MUSIC xxxx" command.

M315 responses to /BATCH command

Explanation**:** The time on batch refers to the length between the /ID was read and the current time. The time is is terms of hours and minutes. (This message is issued by module CAR.)

Procedure**:** None.

M316 /ID COMMAND NOT SUPPORTED FROM CONSOLE

Explanation**:** You cannot sign on to MUSIC from the operator console. (This message is issued by module CAR.)

Procedure**:** None.

System Console Log and User Messages

MUSIC writes log messages on the console each time a user signs on. Other messages such as the occurrence of someone entering an incorrect password are also printed. No direct operator action is required for any of these messages.

M400 BAD PASSWORD: uuuuuuu, UAD=uuu, TCB=ttt, RDEV=xxxxxxxx

Explanation**:** A user has specified an incorrect password for the code given in the *u* field. The *n* portion of the message is the terminal number and the *x* field is the terminal identification from the /ID line. (This message is issued by module SIGNON.)

M401 STAT BUFFER DUMPED uuu

Explanation**:** The hardware statistical log on device *uuu* has been dumped by the program BUFLOG. (This message is issued by module DIOEX.)

M402 SIGN-ON: uuuuuuu, UAD=aaa, TCB=ttt

Explanation**:** A user has signed on with the code *uuuuuuu* on unit address *aaa* on terminal control block (tcb) *ttt.* Some sign-on messages may be lost if the console cannot keep up with the printing of these messages. (This message is issued by module SIGNON.)

M404 nn uuuuuuu xx ... message from the user... ********

Explanation**:** A message has been sent from a remote terminal to the machine room operator. The code of the user is given in the u-field and the terminal identification specified on the user's /ID line is given in the x-field. Use the terminal number given in the n-field if it is required to send a message back to that terminal. (This message is issued by module TCS.)

M405 idno message from WTO

Explanation**:** SVC $WTO from user region. (This message is issued by module USRSVC.)

M406 STAT INFO uuu ... hexadecimal information ...

Explanation**:** The device at address *u* has sent a record of statistical information to the processing unit regarding such things as the number of seeks and the amount of data transfer. This message requires no operator action. The hexadecimal informal are usually the first 24 bytes returned in the sense command. For 8809 tape drives, they are the 24 bytes starting at the 8'th byte of the sense. (This message is issued by module DIOEX.)

M407 SIGN OFF: uuuuuuu, UAD=aaa, TCB=ttt, RDEV=xxxxxxxx

Explanation**:** The user on the indicated terminal has signed off or has been forced off by the operator or an I/O error.

M408 CMD FROM ttt uuuuuuu: ccc...

Explanation**:** A console command has been issued from the Auxiliary Console Facility.

System Error Messages

M500 DISK REQ ERR aaaaaaaa vvvv ....
DETECTED AT DIOEX+xxxxxx

Explanation**:** An invalid disk I/O request has been issued by some system module or by a user with system programmer privileges. The *a* field gives the address of the request block and the *v* field gives the contents. The *x* field gives the location within the module DIOEX at which the error was discovered. This can be used to determine the type of error. (This message is issued by module DIOEX.)

Procedure**:** Follow installation procedures.

M501 SYS ERR n TERMINATED JOB RUN BY uuuuuu FROM TERM ttt

Explanation**:** An error has been detected by the system in connection with the operation of a system utility. The system has terminated that job. (This message is issued by module USRSVC.)

Procedure**:** Note this occurrence for the MUSIC system support personnel.

M502 ABEOJ OF USER uuuuuuu. CAUSE=ccccccc

Explanation**:** A user's job has been terminated by the system due to an unusual condition. The user code and cause fields are shown. The user gets the message shown in the cause field. The reason for the failure could be entirely the user's fault. For example it could be the user was attempting to do an SVC which is not supported in MUSIC. On the other hand it could be an error in the SVC handler. (This message is issued by module USRSVC.)

Procedure**:** If these message occur very often with different user codes, then the system support personnel should be informed.

M503 OUT OF SWAP SPACE

Explanation**:** The system has run out of swap areas. The current job is cancelled. (This message is issued by module SWAPER.)

Procedure**:** No immediate action is necessary. Make sure that the systems administrator is informed so that this condition can be avoided in the future. The system data set(s) SYS1.MUSIC.SWAP1, SWAP2, etc. should be enlarged. See *Chapter 19 - Direct Access Storage* for further information.

M504 P.I. HANDLING JOB REQ j, PSW=hhhhhhhhhhhh,
PICODE=pppp. TERM tt xxxxxxxxxxxxxxxx

Explanation**:** P.I. (program interruption) has occurred in a system utility such as sign-on, save etc. The job is cancelled. The PICODE contains two parts: the first part contains the instruction length code, the second part contains the PI code. See message M508 for PI code meanings. The information following TERM is the terminal identification, user code, and the file name. The job request field identifies the system utility according to the following table:

1 /exec
2 /update
3 /save
4 /purge
5 initialization job 'jobone'
6 /display or /list
7 /id
8 /rename
9 /input

(This message is issued by module PITRAP.)

Procedure**:** Follow installation procedures. The SVC numbers are listed in *Appendix C. SVC Table*.

M506 SYSIN ABNORMAL COND xx USER uuu TERM ttt FILE f

Explanation**:** Some error has been detected during the reading of a Save Library, or input file. (This error message comes from the module SIOCS that handles the reading from MUSIC I/O unit 5.) The condition is identified in the *xx* portion of this message. The user's code is printed in the *u* field, the terminal number in the *ttt* field, and the file name in the *f* field. This message is often accompanied by an M517 message. This other message contains further information about the error. (This message is issued by module SIOCS.)

Procedure**:** Inform the system support personnel. If this error persists, with different file names, it might indicate that a section Save Library has been destroyed.

M507 SYSIN SECURITY VIOLATION USER u TERM t FILE f

Explanation**:** The user identified in the *u* field has tried to access the file *f*. The file is not accessible to the user due to the fact that it has been saved with the PRIVATE or EXECUTE-ONLY attribute. (This message is issued by module SIOCS.)

Procedure**:** If this message re-occurs often for the same user, and particularly, if different file names appear in the message, if could mean that the user is deliberately trying to violate the system security.

M508 MUSIC DOWN. PI IN SUPV. PSW=h.
PICODE=pppp. TERM tt xxxxxxxxxxxxxxxx

Explanation**:** A serious program interruption (PI) has occurred in MUSIC. System is automatically shut down. Information following TERM is the terminal number, user code, and file being read of the current active user. The error may not have been related to this user, however. The PICODE contains two parts: the first part contains the instruction length code, the second part contains the PI code. (This message is issued by module PITRAP.)

Common PI code meanings:

01 PSW points to invalid instruction
04 Instruction violates storage protection
05 Instruction refers to an address too large
06 Specification exception

Consult the manual *IBM System/370 Principles of Operation* (GA22-7000) for more information on program interruption codes.

Procedure**:** Take a main storage dump and then re-IPL MUSIC. Refer to the topic "System Errors and Restart Procedures" in Chapter 4 for details of how to do this.

M510 UNRECOVERABLE ERROR xx DETECTED AT MODULE mod+xxxxxx. MUSIC IS SHUT DOWN.

Explanation**:** A system routine has encountered a condition for which no recovery can be taken. The system is immediately halted. Note the module name and the error number fields for the system support personnel. (This message is issued by module XSTOP.)

Procedure**:** Take a main storage dump and then re-IPL MUSIC. Refer to the topic "System Errors and Restart Procedures" in Chapter 4 for details of how to do this.

M512 uuu OFFLINE

Explanation**:** The unit is not available to the processing unit due to the fact that it has been switched off the channel or the device address has been incorrectly specified when the MUSIC nucleus was generated (NUCGEN). (This message is issued by module DIOEX.)

Procedure**:** Determine the cause of the problem and if it is a tape drive use the /VARY command to retry.

M513 ACCOUNTING FILE IS FULL. RE-IPL AND RUN ACTDMP PGM TO CLEAR IT.

Explanation**:** The accounting data set (SYS1.MUSIC.ACCT) is full. The system will ignore accounting information until the file is emptied by running the ACTDMP utility program. (This message is issued by module DICDOC.)

Procedure**:** IPL MUSIC, specifying the =NOTERM special option, then run the accounting program ACTDMP. After ACTDMP has completed, do an IPL to reload MUSIC. Your system support personnel can increase the size of this data set to avoid the problem in the future.

M514 SYSTEM HUNG. WAITING FOR INTERRUPT ON UNIT=xxx

Explanation**:** The system is waiting for a disk or tape operation to end. Normally these operations take a fraction of a second. This message is printed after the system has waited for 30 seconds. (This message is issued by module DIOEX.)

Procedure**:** This condition can be caused by a job attempting to read a blank tape. The tape drive's SELECT light will be on solidly (no blinking). If this is the case, the operator should hit the RESET button on the tape drive whose address appears in the message.

Under VM, this message can occur when VM is waiting for an interrupt to come back from a device whose address is lower in value than the address shown in this message. The only way to determine the actual address in this case is to look at VM control blocks.

If none of the above, suspect a hardware error and report the problem to your hardware/software support personnel.

If this problem does not clear itself within a short time, it may be necessary to re-IPL MUSIC.

M515 uuu NOT READY

Explanation**:** Self-explanatory. (This message is issued by module DIOEX.)

Procedure**:** Ready the device. This message will also be issued if a tape mount has been requested and the job was cancelled before the mount was done. In this case, ignore the message.

Procedure**:** Refer to procedure with disk I/O error that was also issued.

M517 FILE I/O UNUSUAL COND n AT x USER u HDRLOC d

Explanation**:** The file system has detected an unusual condition. The *x* field points to the code within the file system module (MFIO) that found the problem. The code of the user is shown in the *u* field of the message. The pointer to the file's header is given in the *d* field of the message. The *n* field is explained below.

50 File not online. User tried to access a file that was in a part of the Save Library that was not online at IPL.
51 Not enough free space exists on the Save Library to satisfy the user's request for space. The system support personnel should add additional space.
52 Not enough free space in the Save Library index. The index and its overflow area are full.
60 Read I/O error in file.
61 Write I/O error in file.
62 Read I/O error in the index or header portion of the file.
63 Write I/O error in the index or header portion of the file.
64 The index contains incorrect information.
65 The file's header contains incorrect information.
66 The allocation maps contain incorrect information.
67 The index entry for a file does not point to a header that contains the same file name.
70 The file system has encountered an unusual situation. This can happen if the system enqueue table is full.

(This message is issued by module MFIO.)

Procedure**:** No action needed be taken immediately. Report the problem to the system support personnel.

M518 FATAL I/O ERROR ON ACCOUNTING FILE

Explanation**:** An I/O error occurred while writing to the accounting data set SYS1.MUSIC.ACCT. A system shutdown has been initiated. (This message is issued by module DICDOC.)

Procedure**:** Restart system.

M519 I/O ERROR ON VTOC OR VOLUME LABEL: UNIT=u VOL=v

Explanation**:** Self-explanatory. (This message is issued by module SPEP.)

Procedure**:** Consult the console listing for the accompanying message M600 for details of the error.

M520 I/O ERROR ON SCRATCH AREA BIT MAP

Explanation**:** Self-explanatory. (This message is issued by module SPEP.)

Procedure**:** Consult the console listing for the accompanying message M600 for details of the error. The scratch map is on the data set called SYS1.MUSIC.SCRATCH.

M521 ACCOUNTING IS DISABLED, RUN ACTDMP AND RE-IPL.

Explanation**:** The system has detected that the accounting file is full during the IPL sequence and disabled accounting. No accounting will be done, though batch and terminals can run. (This message is issued by module DICDOC.)

Procedure**:** Run the ACTDMP program to clear the accounting file and re-IPL.

M522 OUT OF PAGING SPACE

Explanation**:** Self-explanatory. (This message is issued by module PAGER.)

Procedure**:** The system may be working in a degraded fashion. Your system support personnel should allocate more space in the MUSIC/SP page data sets.

M523 IGNORING QUEIT #n REQ FROM uuuuuu. ALREADY IN Qm

Explanation**:** Informational message about an unusual scheduler request. (This message is issued by module URMON.)

Procedure**:** None required.

M524 BAD BUFFER CHAIN POINTER

Explanation**:** Self-explanatory. (This message is issued by module DSPOOL.)

Procedure**:** The system will attempt to recover automatically from this situation.

M525 FREE PAGE AT aaaaa HAS INCORRECT KEY OF kk

Explanation**:** Self-explanatory. (This message is issued by module PAGER.)

Procedure**:** None. This message is used for system debugging activities.

M526 USER AT TERM ADDRESS aaaa HAS NONZERO SWAP INFO OF xxxxxxxxxxxxxxx JREQ=

Explanation**:** This message is normally not issued. When it is, the information is for use by the system support personnel. (This message is issued by module URMON.)

Procedure**:** None. This message is used for system debugging activities.

M527 DUMP OK

Explanation**:** The stand-alone system dump program has successfully written the storage dump to disk.

M528 DS FULL

Explanation**:** The storage dump did not fit in the disk dataset (PGMDMP). What has been written may be of some use.

Procedure**:** Increase the size of the dump dataset by following the procedure in *Chapter 6 - System Reconfiguration*.

M529 DUMP DID NOT ALL FIT ON DISK

Explanation**:** The storage dump did not fit in the disk dataset (PGMDMP). What has been written may be of some use.

Procedure**:** Increase the size of the dump dataset by following the procedure in *Chapter 6 - System Reconfiguration*.

M530 BAD BUFFER QUEUE HEADER. LOC=aaaaaaaa, HDR=hhhhhhhhhhhhhhhh

Explanation**:** The Buffer queue chain has been corrupted. (This message is issued by the module DSPOOL.)

Procedure**:** The system will attempt to recover by abending the job that caused the problem to occur. If the problem occurs often it should be reported to the system support personnel.

M531 ACCOUNTING BUFFER OVERFLOW, WAITING FOR I/O, SOME RECORDS MAY BE LOST.

Explanation**:** This message is issued by module DICDOC.

Procedure**:** None required.

Hardware Error Messages

M600 I/O ERROR xxx DEB ddd
CAW aaaaaaaa CSW xxxxxxxxxxxxxxxx
REQ=nnn SEEK ssssssssrrrrrrrr SENSE dddddddd ....

Explanation**:** A disk or tape I/O error has occurred. The unit address, channel address word, and channel status word are given. The internal DEB number is also shown. If a disk drive caused the error, the cylinder and head of the track (4 hex digits each) involved are shown in *ssssssss*. If an alternate track seek was done, its cylinder and head are given in the *r* field. (The seek address field will be 0 for FBA disks.) Sense information follows. Either 5 or 24 bytes are printed, depending on device type. The REQ field displays the storage location that contained the I/O request block. The first two bytes of the sense information can be used to determine the general nature of the error. The following is a summary of the more common ones.

8100 An incorrect seek address was generated by MUSIC.

8000 Either the disk module has been incorrectly switched to 'READ-ONLY' status or an invalid CCW has been issued by MUSIC.

2000 A hardware error has occurred. Report this problem to the service personnel.

1000 Treat the same as 2000 above.

0800 A data check has occurred. It means that the unit cannot interpret the data on the drive either because it was recorded with the wrong density or it has been physically destroyed or deteriorated since its creation.

0200 Flagged track. MUSIC does not support flagged tracks on 2314 and 3340 devices.

0100 The hardware was unable to correctly perform a seek operation. Report this problem to the service personnel.

0008 A search command was performed for a record which was not there. Such would be the case if an attempt was made to search for the second record on a track which contains only one record. Certain MUSIC utility programs which can only be run by users with special system privileges can cause this error if used incorrectly. This error will also occur if, for example, a 2314 device has been defined as a 3330 drive in the MUSIC nucleus. This message can also be given when a User Data Set (UDS) file has been created in a portion of the disk volume that has not been preformatted.

0002 Same causes as 8000 above. Under VM, a minidisk in read-only status will also give this error code if a write is attempted.

(This message is issued by module DIOEX.)

Procedure**:** Note the physical location of the drive which had the error and the pack or tape reel that was mounted there. Then follow installation procedure.

M601 >>SERIOUS MCH CK xxxxxxxx, mmmmmmmm
MUSIC DOWN...CLEAR CORE...RESTART MUSIC

Explanation**:** An unrecoverable machine check has occurred on the system or too many soft machine checks have been recorded. The system has shutdown. The m field is the S/370 machine check interrupt code stored by the processing unit at location E8 (hex). The *x* field is the displacement into the MUSIC system module MCHINT at the point where it decided to abort the system. (This message is issued by module MCHINT.)

Procedure**:** Press the stop button on the processing unit to stop the messages from printing. If an IBM FE or CE is on site, then have the FE check the error condition. It is a good practice to run the IBM program EREP or an equivalent program to record the error condition for later study. (This model dependent program can be obtained through your local IBM FE office). Clear main storage after the error is recorded and IPL MUSIC to re-IPL the system. If the error persists, it will be necessary to call for service on your machine.

M602 >>CHAN CK UNIT=uuu, CSW=xxxxxxxxxxxxxxxx, LCL=ssssssss
MUSIC DOWN...CLEAR CORE...RESTART MUSIC

Explanation**:** A channel check has occurred on unit uuu or the channel connected to it. The error was detected by the presence of certain bits in the CSW field. Consult the *IBM System/370 Principles of Operations* (GA22-7000) manual for description of these status bits in the CSW. The LCL (for limited channel logout) reflects the contents of main storage field location B0 (hex). The LCL field is stored only on certain models of S/370 processing units and if this field is printed as all 00's (hex) or all FF's (hex), then the LCL was not stored. On some processing unit models this error will be preceded by the 'SOFT MCH CK' message. On others, channel checks might be always logged by the processing unit as a machine checks. (This message is issued by module MCHINT.)

Procedure**:** Press the stop button to stop the messages from printing. Determine if the error was caused by accidentally powering off control units while still on the channel or similar situations. If so, then re-IPL MUSIC after the problem is fixed. If not caused by the above, then follow the procedure specified under the 'SERIOUS MCH CK' message above.

M603 >>SOFT MCH CK mmmmmmmm,xxxxxxxxxxxxxxxx,tt,pppppppp,ff

Explanation**:** A recoverable machine check has occurred on the system. The *m* field forms the S/370 machine check interrupt code, the *x* field forms the machine check old PSW. The current user of the swap region has a sign-on code *p* and this user's TCB number (hex) is printed in the *f* field. The phase flag associated with the swap region is printed in the *t* field. The system will shut down if these errors occur too often. On certain processing unit models, the occurrence of soft machine checks may indicate that the processing unit internal circuitry is working in a degraded fashion. (This message is issued by module MCHINT.)

Procedure**:** Record the occurrence of the error and report it to the IBM service personnel. The *m* field in the message can be used to determine the general location of the error. Refer to the *IBM System/370 Principles of Operations* (GA22-7000) for a description of this field.

M604 >>HARD MCH CK mmmmmmmm,xxxxxxxxxxxxxxxx,tt,pppppppp,ff

Explanation**:** A machine check has occurred such that the error has been localized to a specific user's job and that the particular job has been terminated abnormally. MUSIC will attempt to keep running after this condition to allow other jobs to run successfully. See the explanation under 'SOFT MCH CK' message for the description of the variable fields in this message. (This message is issued by module MCHINT.)

Procedure**:** Verify that the system is still functioning normally. This can be done by typing in any

console command such as '/Q'. If the system does not respond to the request, then it is conceivable that other machine checks have occurred but the system is unable to print on the console. If the system appears to be working, then follow the procedure under 'SOFT MCH CK' message above, otherwise follow the procedure given under the 'SERIOUS MCH CK' message.

M605 >>CHANNEL CHECK ON A TERMINAL.

Explanation**:** A channel check has been detected by the terminal handler. The terminal has been dropped. Look at the CSW bits printed. (This message is issued by module TCS.)

Procedure**:** Report this problem to your hardware support personnel.

M607 >>CCW AREA OVERFLOW IN TM3270, DEVICE=xxx

Explanation**:** The module TM3270 has attempted to create a channel program larger than the area available in the IOCB. The unfinished channel program is used in the I/O operation to the terminal resulting in missing output.

Procedure**:** Report this as a system problem.

M608 >>INITIAL TAPE ERROR...

Explanation**:** This message is put out by the module DIOEX on some systems, to indicate a tape error occurred, before any error recovery is attempted. If error recovery is NOT successful, an M600 message will follow.

Procedure**:** Report this as a system problem.

Nucleus Generation Messages

M700 ERROR IN ABOVE RECORD

Explanation**:** Used to indicate a bad configuration item. (This message is issued by module SYGEN2.)

M701 ERROR ON INPUT DEVICE,
CORRECT PROBLEM THEN PRESS EXT KEY

Explanation**:** Reader error, feed stop, or unexpected end of file. To continue, correct error condition, ready card reader, and depress processing unit console interrupt (external) key. (This message is issued by module SYGEN1.)

M702 NUCLEUS GENERATION TERMINATED

Explanation**:** Correct errors shown by the other messages. (This message is issued by module SYGEN2.)

M703 DATA SET NOT FOUND

Explanation**:** This message indicates that a required data set cannot be found on the system residence volume being generated. If it is issued at the very start of the generation, it is SYS1.MUSIC.GENLOAD that is missing. If issued just after the configuration cards have been processed, it is SYS1.MUSIC.NUCLEUS which is missing. (This message is issued by module SYGEN1.)

M704 VTOC CI EXCEEDS LEN OF 1024

Explanation**:** The VTOC has a control interval of greater than 1024 bytes. This message is issued for FBA disks only. (This message is issued by module SYGEN1.)

M705 DUPLICATE CONTROL SECTION OR ENTRY

Explanation**:** Self-explanatory. (This message is issued by module SYGEN1.)

M706 ERROR READING NUCLEUS GENERATION INPUT

Explanation**:** Reader error, feed stop, or unexpected end of file. If the input was from a tape, then remake the tape. If card reader, correct error condition, ready card reader, and depress processing unit console interrupt (external) key. (This message is issued by module SYGEN2.)

M707 INVALID CARD

Explanation**:** Self-explanatory. (This message is issued by module SYGEN1.)

M708 INVALID ESD CARD

Explanation**:** ESD card contains invalid type code (This message is issued by module SYGEN1.)

M709 INVALID FORMAT-4 DSCB

Explanation**:** The F4 DSCB of the new system residence pack is not correct. (This message is issued by module SYGEN1.)

M710 INVALID VOLUME LABEL

Explanation**:** The volume label of the new system residence pack is not correct. (This message is issued by module SYGEN1.)

M711 EXPECTING SEQ FIELD OF nnn WHEN THE FOLLOWING RECORD WAS FOUND

Explanation**:** Sequence check failed when processing object decks. All object decks must have sequence numbers incrementing by 1. (This message is issued by module SYGEN1.)

M712 INVALID SLC OR ICS CARD

Explanation**:** Name must be specified on SLC and ICS cards. (This message is issued by module SYGEN1.)

M713 INVALID VTOC EXTENT

Explanation**:** Disk pack VTOC does occupy only one cylinder. (This message is not issued on FBA devices.) (This message is issued by module SYGEN1.)

M714 I/O ERROR READING FORMAT-4 DSCB

Explanation**:** An I/O error has occurred while reading the F4 DSCB. (This message is issued by module SYGEN1.)

M715 I/O ERROR READING VOLUME LABEL

Explanation**:** An I/O error has occurred while reading the volume label. (This message is issued by module SYGEN1.)

M716 I/O ERROR SEARCHING VTOC

Explanation**:** An I/O error has occurred while searching for the data set. (See message "DATA SET NOT FOUND".) (This message is issued by module SYGEN1.)

M717 LOADER WORKFILE OVERFLOW...LOADING ABANDONED

Explanation**:** System exceeds loader workfile capacity. Probably due to too many CSECTS, ENTRY or EXTRN statements. (This message is issued by module SYGEN1.)

M718 MINIMUM MUSIC SYSTEM REQUIRES DISK, AND CONSOLE DEVICE SPECIFICATIONS

Explanation**:** Add missing device specifications and recreate the nucleus. (This message is issued by module SYGEN2.)

M719 MISSING PARAMETER ON DISK OR TAPE DEVICE CARD

Explanation**:** Self-explanatory. (This message is issued by module SYGEN2.)

M720 MODULE OR ENTRY POINT (xxxxxx) MISSING

Explanation**:** Undefined external reference. (This message is issued by module SYGEN1.)

M721 NEW MUSIC NUCLEUS IS NOW ON DISK

Explanation**:** Indicates normal completion of the nucleus write operation. (This message is issued by module SYGEN1.)

M722 NUCLEUS DATA SET TOO SMALL

Explanation**:** SYS1.MUSIC.NUCLEUS data set is too small. (This message is issued by module WMON.)

M723 OBJECT PROGRAM TOO LARGE

Explanation**:** System object program exceeds allocated storage size. (This message is issued by module SYGEN1.)

M724 MUSIC NUCLEUS GENERATION ABANDONED

Explanation**:** Inconsistencies encountered in system nucleus deck. (This message is issued by module SYGEN1.)

M725 NUCLEUS GENERATION PARAMETER MISSING

Explanation**:** Required machine configuration card missing. (This message is issued by module SYGEN1.)

M726 SYSTEM STORAGE BOUNDARY ERROR

Explanation**:** The initialization routines (HELLO, DSINIT) have grown to the point that they overlap

the storage used by the system generation loader (SYGEN1). It can also mean that the number of CSECT and ENTRY point names has exceeded the maximum allowed by the table in SYGEN1. (This message is issued by module SYGEN1.)

M727 INVALID CHARACTER IN NUMBER OR ADDRESS FIELD

Explanation**:** Self-explanatory. (This message is issued by module SYGEN1.)

M728 TOO MANY DISK AND/OR TAPE UNITS DESCRIBED

Explanation**:** Number of tapes and disks exceeds system maximum of 64. (This message is issued by module SYGEN2.)

M729 TOO MANY TERMINAL DEVICES DESCRIBED

Explanation**:** Number of terminals exceeds system maximum of 250. (This message is issued by module SYGEN2.)

M730 UNRECOVERABLE DISK ERROR...LOADING ABANDONED

Explanation**:** Error reading or writing loader workfile. (This message is issued by module SYGEN1.)

M731 INVALID OR DUPLICATE UNIT ADDRESS

Explanation**:** Self-explanatory. (This message is issued by module SYGEN2.)

M732 DISK ADDRESS OFFLINE

Explanation**:** The disk address specified in the nucleus is not available. (The SIO returned the not operational status.) (This message is issued by module SYGEN1.)

Procedure**:** Correct the situation and retry.

M733 NUCLEUS DATA SET IS TOO SMALL

Explanation**:** Self-explanatory. (This message is issued by module WMON.)

Procedure**:** Increase size of SYS1.MUSIC.NUCLEUS data set.

M734 TABLE EXCEEDED IN MODULE WMON

Explanation**:** The table referred to is the one that contains the disk locations of where to write the nucleus. (This message is issued by module WMON.)

Procedure**:** Consult the module WMON to see how to resolve the problem.

M735 VTOC CI EXCEEDS LEN OF 1024

Explanation**:** The VTOC control internal size exceeds 1024 bytes. (This message is issued by module WMON.)

Procedure**:** Remake the VTOC on the IPL volume using an acceptable control interval size.

MFIO Error Messages and Codes

A number of error messages can originate from the MUSIC file I/O interface routine MFIO. (The actual text of the messages is in the $MCM:SYSMSG.M file.) These error codes and messages may appear when application utilities are running. Some programs report only the error code number.

```
 1 END OF DATA SET ENCOUNTERED
 2 INCORRECT LENGTH
10 INVALID REQ
11 INVALID REQ PARAMETER
12 FILE NAME INVALID
19 INVALID ARGUMENTS IN CALL TO SERVICE SUBROUTINE
20 TOO MANY OPEN FILES
21 NOT YOUR LIBRARY
22 NOT YOUR FILE
23 VIOLATION OF WRITE RULE
24 ATTEMPT TO READ BEYOND END OF WRITTEN INFO
25 WRITE THEN READ SEQ INVALID
26 YOUR USERID CANNOT CREATE FILES ACCESSIBLE BY OTHERS
27 YOUR USERID CANNOT CREATE FILES IN THE COMMON INDEX
30 FILE NOT FOUND
31 DDNAME NOT FOUND (see also error 35)
   (For CALL OPNFIL to define a ddname, error 31 means
   there is no more room in the ddname table.)
32 FILE ALREADY EXISTS
33 FILE IN USE
34 COMMON NAME USED BY SOMEONE ELSE
35 UNIT NUMBER NOT DEFINED
   (or ddname is undefined by user request such as by
   specifying UNDEF on a /FILE statement)
36 SUBDIRECTORY DOES NOT EXIST
40 SPACE QUOTA EXCEEDED FOR THIS USERID
41 SPACE QUOTA EXCEEDED FOR THIS FILE
42 CANNOT ADD SPACE TO THIS FILE
43 REQUESTED ACCESS OR OPERATION NOT ALLOWED
44 REQ BEYOND EXTENT OF FILE
45 FILE RECFM NOT DEFINED
46 FILE CANNOT BE READ SEQUENTIALLY
47 INSUFFICIENT SPACE FOR BUFFER ALLOC
48 MIN RECORD LEN IS 80 FOR THIS FILE TYPE
50 FILE NOT ON-LINE
51 NOT ENOUGH FREE DISK SPACE
52 NOT ENOUGH FREE DISK SPACE (IDX)
60 RD I/O ERROR IN FILE
61 WR I/O ERROR IN FILE
62 RD I/O ERROR IN SYSTEM AREA
63 WR I/O ERROR IN SYSTEM AREA
64 INDEX IN ERROR
65 HEADER IN ERROR
66 MAP INTEGRITY ERROR
67 INDEX/HEADER MISMATCH
70 SYSTEM FILE ERROR
```

Wait State Codes

MUSIC will enter the WAIT state when it cannot continue processing. Usually, a message is issued just before it enters this state. Sometimes this is not possible. This could happen, for example, if the console was not working.

This topic lists some of the wait states possible. The WAIT state is entered by loading a PSW with the WAIT bit on as explained in *Chapter 18 - System Internals*. The address portion (last 3 bytes) of the PSW form the *wait state code*.

Starter System Restore Program Wait Codes

FFFFFF	Normal end of restore of starter pack.
EEEEEE	Waiting for operator to press 'request' on console device.
0002FF	Unexpected SVC interrupt.
0003FF	Unexpected program interrupt, or attempt to run on System/360 or under VM with ECMODE off.
0004FF	Machine check interrupt.
0006FF	Condition code 1 from SIO (CSW stored).
0007FF	Condition code 2 from SIO (busy).
0008FF	Condition code 3 from SIO (device not operational).
0009FF	Error bit on in CSW unit status byte.
000AFF	Error bit on in CSW channel status byte.
000BFF	Insufficient main storage to load core image.
000CFF	Unable to find an unused disk UCB (Unit Control Block).

IPLable Main Storage Dump Program Wait Codes

00FF00	Main storage has been correctly written to the dump data set.
00FF01	Main storage did not all fit in the dump data set. As much as possible of main storage was written to the data set.
00FF04	The disk has become not operational.
00FF05	Some I/O error has been detected while trying to write the dump information to disk. Look at the CSW at location 40 (hex) for the ending status bits that indicated the error.
000003	An unexpected program interrupt occurred.
00FF0F	An unexpected program interrupt occurred.

MUSIC System Wait Codes

FFFFFF	System waiting for work in the user region. This wait state is normal during daily operation and is no cause for alarm. The system will automatically come out of this state when there is work to do.
FF00FF	All user regions are waiting for some I/O to complete. This wait state is normal during daily operation but should only occur for a fraction of a second.
EEEEFx	Some problem was experienced when the MUSIC system module MFETCH attempted to read the rest of the system from disk at IPL time (x is usually from 1 to 9). An error message is usually printed to explain the exact reason. It could also happen as a result of a machine check detected during the IPL of MUSIC (x=A). No message would be given in this case.
AAAAAA	Normal end after IPL to install a new MUSIC nucleus. This wait state is preceded by the message :M721 New MUSIC Nucleus is now on disk".
BBBBB1	Error after IPL to install a new MUSIC nucleus. This is preceded by an error message on the console.
F0F0F0	The system was shut down by the console command /STOP or /SYSTOP. This code can also occur during nucleus generation, when an error is detected in the device specification records.
000BAD	Some I/O error was detected during the MUSIC IPL operation. Consult the CSW at location 40 (HEX) for details.
00FF01	The system could not complete initialization correctly. A message is also given to state the reason.
00FF02	The system was shut down by an SVC 80 issued by one of the system modules.
00FF03	The system was shut down by a program check occurring in supervisor state.
00FF04	The system was shut down by a machine check.
00FF05	An SVC was detected that was greater than 255. The hardware (or VM) must have set location 8A (hex) to a non-zero value. This is a hardware or VM error. (The wait is issued by the module SYSSVC.)

Appendix B: System Module Descriptions

Resident Modules

APLTRN: APL 3270 Translate Table

This module contains the 3270 APL translate tables.

CAR: Console Handler

This routine performs the I/O to and from the processing unit console. Messages from other system modules are sent out and requests from the keyboard are interpreted and processed.

CDSRCH: User Code Table Search Routine

This routine performs a search of the User Code Table for a given code. If the code is found, the routine returns the complete code table entry. This routine can also provide the caller with general information about the table structure and its location on disk.

CHKPFK: Process Program Function Keys

TM3270 calls this module after a read to perform whatever processing is required due to the pressing of a program function key. Multi-session requests, the retrieve key, and any user defined function key operations are processed here. This module also contains the system default PF key definitions.

CKCDE: Batch User Code Verification

This routine verifies user codes for jobs submitted via the batch card reader. CDSRCH is called by SPAM to access the code table record. TCB and XTCB fields are filled in for the batch TCB.

COMAND: Terminal Command Processor

This module scans command lines entered from terminals. It is called from various points within TCS. The command table is included in this module.

DEFPFK: Define Program Function Keys

This module processes the /DEFINE command. It is called from COMAND if the command is issued from the terminal, or from USRSVC if the NXTCMD subroutine is used. The first time this routine is invoked for a user, a buffer is acquired from DSPOOL and the system default definitions copied from the module CHKPFK. This buffer will remain allocated to the user for the rest of the session.

DICDOC: System Accounting Routine

This routine handles the accounting for the system. When a job is started or terminated, this routine records the terminal number and the elapsed time used on the accounting data set. Entries are also made at sign-on and sign-off time and when a job's output at the terminal is complete.

DIOEX: Selector Channel Input/Output Executor

This routine handles the actual I/O execution on the selector channel(s). It will handle tape and disk, read and write requests, and special purpose functions. Most requests require that disk channel programs be constructed using information from the caller's request block. and the appropriate Unit Control Block (UCB) and Data Extent Block (DEB). DIOEX also performs the I/O interrupt handling for selector channels, including error detection, correction and logging.

DSINIT: System Data Set Initialization Routine

At IPL time, DSINIT locates and reads the system catalog, and allows the operator to modify it. It scans all ready disk packs, locates necessary system data sets and constructs the Data Extent Blocks, Pseudo Device Blocks and Statistics Blocks. It performs miscellaneous other data set initialization functions.

DSPOOL: Data Spool Manager

This routine handles the buffer pool that is dynamically used by the process that transfers data between terminals and the user regions.

ENQRTN: Enqueue and Dequeue Processor

ENQRTN 'Resident System Module' This routine provides the facility of reserving resources. These resources can be requested on a shared or exclusive basis. If the resource is busy, a return code is set. This processor is used to reserve resources such as Save Library files, User Data Sets, and the User Code Table (during updates.)

FIOCS: File Input/Output Control

FIOCS 'Resident System Module' FIOCS is the execution time I/O control program for sequential User Data Sets. It supplies all necessary file manipulation functions (READ, WRITE, BACKSPACE, ENDFILE, REWIND) as well as handling all blocking and de-blocking of logical records.

FMSG: Format Message Routine

Subroutine called by system routines to format a message by plugging in variables according to the given prototype. Used by USRSVC when formatting user error messages.

FSIO: Full Screen 3270 Interface

This is responsible for interpreting full screen requests, buffer allocation and moving data from user region to system buffers. It receives control from the conversational read routines in MFIO.

HELLO: System Initialization Module

This module is responsible for initialization of the system. The following functions are among those performed. Main storage size and processing unit type are determined. Time and date is set. Free storage is zeroed and protection keys are set. I/O device Unit Control Blocks and Terminal Control Blocks are created. The Accounting File is initialized. If a system restart is being done, user's input and output files are preserved. Additionally, on a System/370, the machine check handling routines are initialized. Flags are set-up at this time specifying whether the processing unit is a S/370 and what optional instructions it has and if running under VM it also determines if it is in a V=R region. HELLO prints informative messages to the system operator concerning time, date and disk drive utilization. If any unusual conditions are found, appropriate error messages are issued and initialization is halted.

IOBUFR: Generalized I/O Buffering Routines

This routine contains generalized I/O buffering and deblocking routines used by several resident system modules.

LODSVC: Dynamic Load Processor

This module dynamically loads the specified load module usually from the system data set SYS1.MUSIC.LOADLIB. If the phase is resident in the Link pack area, then appropriate action is taken.

LOGO: MUSIC Logo

This csect contains the MUSIC logo suitable for display on 3270 terminals. This logo is displayed when a 3270 terminal is connected to MUSIC and not signed on.

LOOKUP: Privileged Program Module

This module is called by MFIO when an execute-only Save Library file is read at the beginning of a user's job. It checks to see if the name is one of the special system names stored in this module. If so, flags are set in CKSAV which will be inspected by EXTSVC when certain types of SVCs are done. Should neither the user nor the program name have special privileges then the job will be aborted by the system if it attempts to perform certain privileged functions.

This module allows programs which require special privileges for execution (such as the User Profile program) to be run by non-privileged users.

MCHINT: Machine Check Handler

This routine is given control in the event of a machine check or a channel check. It logs the error. The action taken depends on the type of error. If it was a soft machine check, the system is allowed to continue if the error counts have not been exceeded. If the error was a hard machine check that can be localized to a specific user's job, then only that job is terminated. Otherwise, the system is shut down.

MFETCH: Monitor Fetch Routine

MFETCH is entered immediately following IPL from a CKD disk. It locates and loads the supervisor from the IPL device and transfers control to the system initialization module HELLO. (From FBA devices, the MFETCH function is handled by coding found in the WMON module.)

MFIO: User I/O Interface

This routine provides the interface to the Save Library and most I/O from the user region. It contains routines to OPEN, CLOSE, ALLOCATE, and do I/O to Save Library files.

MIOX: Multiplexer Input/Output Controller

This routine schedules the physical I/O for the Unit Record devices (console, reader, punch, printer), checks for errors and unusual conditions, and sends appropriate messages to the processing unit console.

MOUNT: Tape Mount Monitor

Handles the mounting and dismounting of magnetic tapes.

NEWTCB: Multi-Session Control Block Manager

This module manages the extra sets of control blocks used by the multi-session support and provides entry points to add, delete, and switch sessions.

PAGER: Paging Supervisor

This module is the paging supervisor. It can handle specific requests from other modules to do specific functions such as loading a module in the PLPA. It also resolves paging exceptions that are implicit paging requests. This module receives control on all program checks so that it can filter out the ones to do with paging. It passes the other program checks to PITRAP.

PAGE0: First 4k of Main Storage

This module resides in main storage starting at location zero. It contains all the data which must necessarily occupy storage that is accessible without using base registers. PAGE0 occupies the area corresponding to both System Locore and User Locore.

PITRAP: Program Interrupt Trap Module

This routine handles program interrupts that are not due to paging exceptions. On an error within the system, the entire system is halted. On any other program check, appropriate messages are issued to the processing unit console (if a serious error) and also the terminal user concerned.

A section of this module, XERMON, is used to service program interrupts from jobs in execution. It contains routines to fix up the floating-point registers for program checks due to floating-point underflows or overflows.

PROTND: Protected Main Storage End

A dummy module indicating the end of fetch-protected system main storage ended at the 2K boundary before or at this address.

PROTST: Protected Main Storage Start

A dummy module indicating the start of fetch-protected system main storage.

SIOCS: System Input/Output Control System

This routine processes SYSIN, SYSOUT, and SYSPCH requests. It is entered directly by various supervisor routines and via SVC's from transient system routines. For batch jobs, page and card counting is performed. Jobs are terminated if limits (set on the /ID card) are exceeded.

SIOCS also refers to a set of control blocks used by many system routines on a terminal (or batch) user's behalf. These control blocks are contained in the module URSRVA.

SPAM: Batch Spooling Monitor

This routine handles the I/O to the disk from the reader, and from the disk to the printer and punch. CKCDE is called to process /ID cards. After a job is read in, it is queued for execution (or save activity if a /SAVE card was entered). SPAM will also process input from a magnetic tape unit instead of a card reader.

SPEP: Device Allocation

SPEP performs pre-execution and post-execution device allocation/de-allocation for jobs using execution-time User Data Sets or tapes. It ensures that tapes and disks are mounted. This routine invokes transient modules PRE and PST to perform disk data set allocation and de-allocation.

STATS: System Statistics

This module contains tables used to maintain system statistics and use counts.

SWAPER: Swapping Control Module

This routine is used to write the user region to disk and to read it back in again. If the system disk and channel configuration allow it, the area is swapped in two or three segments, each on different disk packs, which can be on different channels.

SYGEN1: System Generation - Phase 1

This module is used during nucleus generation to read the system object decks from the card reader or magnetic tape, and to load (and link) these object decks. SYGEN2 is then called to configure the system. WMON is used to place the new system on the System Residence disk.

SYGEN2: System Generation - Phase 2

During nucleus generation, this routine processes system specification and configuration cards. The cards are checked for validity and converted into internal form for use by HELLO during system initialization.

SYMTBL: Symbol Table

This module contains symbol names and locations for use by certain command processors in the console handler CAR.

SYSSVC: System Mode SVC Processor

This module gains control on all SVCs. User mode SVCs are passed to USRSVC for action. System mode SVCs are processed in this routine or branched to from this routine.

TABCMD: Tab Command Processor

This routine is called by TCS to process /TABIN and /TABOUT commands from terminals. It translates the values on the commands into internal form and places them into the appropriate TCB.

TABSET: Output Tab Executor

TABSET is called by TCS to insert tab characters (and idle characters) in lines to be sent to a terminal. It uses the terminal's current tabs to ensure that the time needed to print program output is minimized.

TCS: Terminal Control System

This module supervises all terminal communications. Included in its functions are terminal spooling and code translation.

TCS calls the terminal device dependent modules TERMIO (for 2741, 1050, and TTY) and TM3270 (for 3270) to perform device dependent functions such as CCW construction and I/O interrupt handling. The actual I/O operations are processed by the internal subroutine called SMART.

Facilities are provided to allow both the running programs and the terminal user to specify certain terminal handling options. Such options as suppressing carriage returns, compressing output lines, and no output translation are available.

TERMIO: Term I/O Handler (all but 3270)

This routine handles the device dependent I/O functions for all terminal devices except 3270-type terminals.

TM3270: Term I/O Handler (3270 only)

This routine handles the device dependent I/O functions for all 3270-type terminals.

TRACE: System Trace Routine

TRACE intercepts all SVC interrupts, I/O interrupts and external interrupts. Before transferring control to the appropriate interrupt processor, it places an entry in the system trace table noting the type of event and information relevant to it, including time of day.

TRANTB: Translate Table

This module contains the terminal translate tables.

TRMCTL: Terminal Specifications

This module contains information about the characteristics of the different types of supported terminals. Refer to *Chapter 7 - Terminal Configuration and Tailoring* for further information.

URIO: User Region I/O Processor

This module receives disk and tape I/O requests in a similar fashion to DIOEX. The I/O request came from a user region which uses dynamic paging. Therefore this routine must lock pages and set up channel programs that use indirect addresses.

URMON: User Region Monitor, Dispatcher and Scheduler

Contains the user region monitor. Also includes the user region dispatcher and the job scheduler. It also supervises the timer and maintains a real-time clock.

URSRVA: User Region Service Area

This module defines an area at the top of the user's virtual storage. It contains buffers for Save Library I/O, SVC work areas, etc. No actual code is in this module as this module is not located in real storage.

USRSVC: User Mode SVC Processor

This module gets control from SYSSVC when an SVC occurs in user mode.

WMON: Write Monitor Routine

WMON is used at nucleus generation time to write the entire resident system to disk. If the nucleus is written to an FBA device, then coding in this module gains control at IPL time to load in the supervisor and then transfers control to the module HELLO.

XSTOP: System Dead Stop Routine

This routine types a message on the processing unit console and puts the machine into a *dead wait*. It is entered from either resident or transient system routines if error conditions are encountered for which no correction is possible.

XTCB: Terminal Control Block

This module contains a prototype Terminal Control Block TCB and several constants related to the number and type of TCB's. Each TCB contains information pertaining to the status of the terminal.

XTXT: Text Manipulation Module

This module contains routines to pack a card image into internal form, unpack a card, and convert a packed message into a printable form.

Note: Modules DSINIT, HELLO and MFETCH are resident only until the end of system initialization. Modules SYGEN1, SYGEN2, and WMON are used only at system generation time.

Transient Modules

CTL: Control Statement Processor

CTL processes control statements for jobs at execution time. This includes analysis of /FILE cards and allocation of special data sets for some /LOAD cards. Initial buffer allocation is performed for some language processors.

JOBONE: Initialization Job

This module is run by the system just after the system has been IPL'd and before any user job runs. Its function is to scan the Save Library and remove temporary files that may exist if the system was shutdown while jobs were running. It also loads the pageable link pack area from information the module DSINIT has set up at IPL time.

LIBCMD: /LIBRARY Processor

This module is called by CTL when a /LIBRARY command is encountered. This module runs in time sliced execution just like a problem program. It has the ability to directly read the Save Library index.

LIST: List/Display Processor

This routine displays a user's Input File or Save Library file. It is invoked from a terminal via the /LIST or /DISPLAY command.

PRE: Pre-Execution Allocation Module

This routine is loaded by SPEP to perform any UDS disk space allocation (temporary or permanent data sets) necessary for job execution.

PST: Post-Execution Module

PST is invoked after execution by SPEP if any UDS disk file de-allocation is needed.

PURGE: Purge Processor

This module processes the /PURGE command.

RENAME: Rename Processor

This module processes the /RENAME command.

SAVE: Save Processor

This module processes the /SAVE, /SV and /REPLACE commands.

SIGNON: Terminal Sign-On Processor

This module is called when a user first connects to the system or when a /ID command is entered. After checking that the sign-on is valid, SIGNON initializes the user's control blocks, displays any daily messages, and starts the user's auto program if this is required.

UPDATE: Update Module

This routine updates the file specified by the user. It can delete or replace existing records and insert new records. The file modified can be either the input file or a Save Library file.

Appendix C. SVC Table

Below is a list of MUSIC/SP SVCs. Most SVC's can only be done under certain conditions or from certain locations in the system.

Certain processors, such as those running under the OS/MUSIC Interface, intercept SVC's coming from the user area and may in turn issue one of the SVC's listed below. This is done during the compilation and execution of COBOL, Assembler, and APL jobs.

User mode SVCs are those that are issued in a user region and therefore are synchronized with the user job processing function. A number of system utilities such as the SIGNON phase runs in a user region but is recognized as special so that it can issue many SVCs that would not otherwise be allowed. User mode SVCs are initially processed by the module USRSVC.

System mode SVCs are those whose function is not synchronized with the processing of jobs in the user regions. These handle disk queueing and scheduling functions. System mode SVCs are initially processed by the module SYSSVC.

The flags in the description field in the following table indicate in which environment the SVC may be processed. The flags have the following meaning:

U	Valid in user mode
*U	Valid in user mode only when SUTIL is in control
S	Valid in system mode
OBS	Obsolete

DEC	HEX	Name	Description
0	00	OKEOJ	NORMAL EOJ (U)
4	04	SYSRIT	WRITE (U)
17	11	LDREXT	LOADER EXIT (U)
18	12	IBCRD	IBCOM READ (U)
19	13	IBCRIT	IBCOM WRITE (U)
64	40	PSTART	START BATCH PRT, PUN AND RDR (S)
65	41	CSTART	START BATCH READER (S)
66	42	QUEIT	ADD JOB REQUEST TO QUEUE (S)
67	43	DOIT	EXEC IN MASKED KEY 0 (*U)
68	44	DICDOC	MAKE ACCOUNTING ENTRY (*U/S)
71	47	DIOEX	DISK/TAPE REQUEST (*U/S)
73	49	OUTFUL	OUT OF DISK SPOOL SPACE (*U)
76	4C	SHUTDN	START SYSTEM SHUTDOWN (OBS)
79	4F	TRMSTR	START A TERMINAL (S)
80	50	STOP	ABORT THE SYSTEM (S)
81	51	ABEOC	ABNORMAL END OF COMPILE (U)
82	52	RD1052	READ FROM CONSOLE (S)
83	53	NORMEJ	NORM EOJ (SAME AS SVC 255) (U)
84	54	SYIERR	ERROR DURING SYSIN READ (OBS)
85	55	SYOERR	ERROR DURING SYSOUT WRITE (OBS)
86	56	TMW	SEND MSG TO TERMINAL (*U/S)
87	57	BEC	CONVERSION (S)
88	58	CMW	WRITE MSG ON CONSOLE (S)
90	5A	NORMEC	NORMAL END OF COMPILE (U)

91	5B	ABEOJ	ABNORMAL END OF JOB (U)
93	5D	JOBSTR	START OFF JOB (*U)
94	5E	PMSG	USER MSG FMTED WITH DMSG (*U)
95	5F	CMSG	CONSOLE MSG FMTED WITH DMSG (S)
126	7E	LODSVC	LOAD A PROGRAM (U)
127	7F	TODSVC	GET TIME OF DAY (U/S)
128	80	SIMSVC	SIMULATE ANOTHER SVC (U)
129	81	MFREQ	UL FILE REQ SVC (U)
202	CA	CMS202	RESERVED SVC NUMBER
203	CB	CMS203	RESERVED SVC NUMBER
214	D6	UDICDC	ACCOUNTING REC FROM USER PROG (U)
215	D7	CETI	CETI SVC (U)
216	D8	DEBUG	DEBUG SVC (U)
217	D9	TRACE	PUT ENTRY IN TRACE TABLE (U/S)
218	DA	VMRPRT	READ VM PRINTER SPOOL FILE (U)
219	DB	SYSIN	INTERFACE TO SIOCS (SYSINR) (U)
220	DC	SETOPT	SET USER OPTION BIT (U)
221	DD	EXREQ	SET /EXEC REQUEST (U)
222	DE	ENQDEQ	ENQUE/DEQUE (U/S)
223	DF	VMSPL	VM SPOOL PUN INTERFACE (U)
224	E0	USTIMR	USER 'STIMER' FACILITY (U)
225	E1	CLEAR	CLEAR CORE BETWEEN LIMITS (U/S)
226	E2	VMDIAG	DO DIAG INSTR TO VM (U)
227	E3	SYSDA	DO I/O ON A SYSTEM DATA SET (U)
228	E4	WTO	TERM WRITE TO OPERATOR (U)
229	E5	XWAIT	TERM WAIT (U)
230	E6	GETCRB	GET CONV READ BUFFER CONTENTS (U)
231	E7	GETCOD	CODE TABLE SEARCH (U/S)
232	E8	SETCOR	PUT LAST LOC USED IN XFCB (U)
233	E9	GETSER	MUSIC SER/MCH CK INTERFACE (U)
235	EB	CLSOUT	CLOSE SYSOUT (U)
236	EC	SETOP4	MISC FUNCTIONS (U)
237	ED	SETSAV	SET /SAVE REQUEST (U)
238	EE	SETABI	SET INPUT TABS (U)
239	EF	PSTCOD	FETCH AND ZERO POST CODE (U)
240	F0	IOWAIT	WAIT FOR LOCK OR I/O (U/S)
241	F1	SETBUF	SET UCS/FCB BUFFERS IN SPAM (U)
242	F2	DLYEXC	DELAY EXECUTION (U)
243	F3	BRTRAP	BRANCH AND SET SVC TRAP BIT (U)
244	F4	COUNTM	UPDATE A STAT COUNTER (U/S)
245	F5	SETABO	SET OUTPUT TABS (U)
246	F6	PWRDIX	ALL POWERFUL DIOEX (U)
247	F7	SETOP0	SET TCB OPT 0 (U)
248	F8	DAIO	DIRECT ACCESS REQUEST (U)
249	F9	UDIOEX	USER DIOEX (U)
251	FB	FETCH	FETCH CORE INTO USER REGION (U)
252	FC	XTIME	JOB TIME (U)
253	FD	BRANCH	BRANCH USING NO REGISTERS (U)
254	FE	ABEND	ABEND JOB (U)
255	FF	EOJ	NORM EOJ (U)

Appendix D. MUSIC TCP/IP for VM TCP/IP Version 1

A number of programs are provided with MUSIC to support TCP/IP Version 1. This section is for sites who do not run TCP/IP Version 2. It is better to use TCP/IP Version 2 since that support does not use CMS sessions to access TCP/IP, resulting in better performance and usability features.

These programs rely on access to the VM TCP/IP Version 1 through a set of CMS service machine. You must create entries in the VM directory that define the virtual machines that are used by MUSIC to access TCP/IP.

The two basic services currently supported by MUSIC are TELNET and FTP. When a MUSIC user starts one of these, a CMS session is started through logical device support on one of the virtual machines reserved for MUSIC TCP/IP support. In the case of TELNET, the CMS TELNET client application is started and the terminal session is turned over to the control of the user.

In the case of FTP, the CMS FTP client application is started, however the MUSIC interface remains in place between the user and CMS, acting as a command filter and handling the file transfer functions between MUSIC and CMS as they are required.

Setting Up to Use TCP/IP

1. Define the virtual machines for MUSIC TCP/IP in the VM directory.
   - One is required for each simultaneous TCP/IP user. Depending on usage, usually a pool of five of six will be sufficient for all your MUSIC users.
   - Each should be configured like a standard CMS user.
   - A 191 disk is required. If only TELNET is to be used this need only be big enough to hold the profile exec and can be read only. The FTP support uses the 191 disk as a staging area in the file transfer process and the size of this disk limits the size of the files that can be transferred. (It is possible to use T-Disk for this. See note later on.)

2. The IBM 3270 File Transfer Program is used to transfer files from MUSIC to CMS. If FTP is to be used, Version 1.1.1 of this program must be installed on CMS. It must be loaded as a nucleus extension before the FTP client application is started. This can be done by a profile exec. (Versions prior to 1.1.1 cannot run as nucleus extensions and will cause the FTP application to abend). Contact the MUSIC support centre for information about the exact PTF level that the 3270 File Transfer program must be at.

3. Create the profile execs for the virtual machines. The profile exec should establish access to the TCPIP program disk and load the file transfer program as a nucleus extension if required. The following is an example.

```
/*  Profile exec for TCPIP MUSIC machine */

/* Access the TCPIP disk */
CP LINK TCPMAINT 592 592 RR
'ACCESS 592 T/A'

/* Load 3270 file transfer program as nucleus extension */
NUCXLOAD IND$FILE
```

4. Add the USERIDs of the virtual machines to MUSIC's shared CMS ID table. The entries to be used for TELNET or FTP should use the application name TCPIP. ADMIN (4 10 10) allows you to change the table. The following shows some sample entries.

```
*
* Userid   Password Application
* |        |        |
  TCP0     password TCPIP
  TCP1     password TCPIP
  TCP2     password TCPIP
  TCP3     password TCPIP
  TCP4     password TCPIP
  TCP5     password TCPIP
```

Using TCP/IP

NET

The program NET displays a list of network nodes and allows the user to TELNET or FTP to a selected node. This list is from the file $PVM:NET.LIST. The distributed file is intended only as a sample and contains a list of about 100 internet nodes that allow anonymous FTP. You should replace this list with one more pertinent to your site.

The format of this file is straightforward. The first field in each record contains the network address that is used on the TELNET or FTP command. The rest should contain some sort of description, identifying the particular node and perhaps listing services that are available there. The network address can be either the numeric internet address or the symbolic node name.

When users runs NET, they can page up and down through the list or use the LOCATE command to locate the computer they want to talk to. They then select either TELNET or FTP by entering a T or F in the margin beside the selected entry. NET then issues a TELNET or FTP command with the selected network address. When the TELNET or FTP session finishes, the user is returned to NET.

TELNET

The TELNET command selects an available CMS userid from the pool of userids defined for TCPIP access in the shared userid table and logs on to a CMS session using that userid. A CMS TELNET command is issued and control of the CMS session is turned over to the MUSIC user. When the CMS TELNET session is finished, the CMS session is automatically logged off and the user returned to the MUSIC session. The format of the TELNET command is:

```
TELNET net_address
```

where *net_address* is the network address of the target computer.

FTP

The FTP command selects an available CMS userid from the pool of userids defined for TCPIP access in the shared userid table and logs on to a CMS session using that userid. A CMS FTP command is issued to start the FTP client program on CMS. Unlike TELNET, direct control of the CMS session is not turned over to the user. MUSIC's FTP interface remains in control. It intercepts output from CMS and displays it on the user's screen. Input from the keyboard is filtered and passed to the FTP application on CMS.

When the user issues a PUT command, the interface transfers the file from MUSIC to the CMS 191 mini disk and then issues a PUT command to the FTP application on CMS.

When the user issues a GET command, the interface sends it to the FTP application on CMS. This gets the file from the remote computer and puts it on the 191 mini disk. When this operation has completed the MUSIC interface automatically transfers the file from CMS to MUSIC.

Some commands are simply passed straight to CMS. Others are not supported at all. For example MGET and MPUT are not supported due the difficulties of implementing them with CMS acting as a surrogate for the MUSIC session.

When the CMS FTP session is finished, the CMS session is automatically logged off and the user returned to the MUSIC session. The format of the FTP command is:

```
FTP net_address
```

where *net_address* is the network address of the target computer.

Tailoring the FTP and TELNET Connection

Both the TELNET and FTP interfaces are written in REXX and use the SIGNON subroutine ($PGM:LDEV.REX) to make the logical device connection with the CMS virtual machines. There are a number of parameters in this routine that you may wish to modify. Full details of the calling sequence for this routine are in the Administrators Reference.

LOCAL The second parameter is a positional parameter and specifies where the service CMS machine is. The default is set to LOCAL, meaning on the same CPU as the MUSIC system. You could access TCP/IP services on another CPU through PVM by changing LOCAL to PVM(node), where "node" is the PVM node of the system in question. This is not recommended however, since the CMS to MUSIC file transfer will have to be done over the relatively slow PVM link.

MSG(n) Specifies the types of message that will be displayed during the connection process. By default this is set to zero and displays a minimum of messages. If you set it to 3, the entire connection process will be echoed on the users terminal.

TDISK This can be used to create TDISK space for use by FTP. see the next section for details.

Using TDISKS for FTP

You may not want to reserve permanent 191 mini-disks for your FTP users. If this is the case it is possible to use TDISKS. The only problem is the additional overhead during startup in allocating and formatting the disks. To use TDISKS you must modify the FTP program.

Edit $PVM:FTP and enter the following commands:

```
L SIGNON
I 'TDISK(nnnn,291,A)',
FILE
```

The value "nnnn" is the number of block of TDISK space that you want to allocate for the FTP session.

Note that a PROFILE EXEC is still required to access the TCP/IP disk and load IND$FILE as a nucleus extension. If TDISKs are used, this PROFILE EXEC could be on a small 191 mini disk that's shared in read only mode between the virtual machines.

Index

Index

Table of Contents

Figures

MUSIC/SP
Administrator's Reference
April 98

READER'S COMMENT FORM

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that the MUSIC Product Group may use or distribute whatever information you supply in any way it believes appropriate without inccurring any obligation to you.

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, institution, mailing address and date:

_____

_____

_____

_____

What is your occupation?_____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation.

MUSIC/SP Administrator's Reference (April 1998)

Reader's Comment Form

fold and tape please do not staple fold and tape

Place stamp here

MUSIC Product Group
McGill Systems Inc.
550 Sherbrooke St. West
Suite 1650, West Tower
Montreal, Quebec H3A 1B9
CANADA

fold and tape please do not staple fold and tape