MUSIC/SP User's Reference Guide

Part 2 - Chapters 5 and 6 (UR_P2.PS)

Chapter 5. MUSIC Commands

MUSIC Commands - Overview

MUSIC commands cause the specified action to occur immediately. (They are sometimes referred to as immediate commands.) Most commands are entered in command mode or from the command line of many programs. The workstation is in Command Mode when the *Go message is displayed. Variable parameters are separated by blanks or commas as indicated in the command description.

If a command is entered in Break mode, the "/" is required. The slash may be required in front of a MUSIC command when it is entered from the command line of a currently running program. The slash distinguishes MUSIC commands from commands valid for the current program. For example, while using the Editor program, use "/COPY" for the MUSIC command (COPY is also an Editor command).

A large number of MUSIC commands have meaning only when used from a workstation. Consult *Chapter 3. Using Batch* for a list of commands that may be used from batch.

Many MUSIC commands can be combined in command procedures as described under the REXX Command Executor topic in *Chapter 8. Processors*. REXX can be used to effectively tailor MUSIC commands and Job Control Statements to suit each user. (See *Chapter 6. Job Control Statements* for more information.)

Workstation Modes

Some commands are valid only at certain times. For example, a command to cancel a job is only valid when the job is running. The various possible modes are defined below.

Command Mode — The workstation is in this mode when the *Go message appears. After you have signed on MUSIC, you are normally in this mode. From this mode you can start running a job, list a file, etc. This is also known as *Go mode.*

Execution Mode — The workstation is in this mode when MUSIC is running a job for you. Your workstation may display the message `*In progress` as it goes from command mode to execution mode.

Reading Mode — A program that is running may ask for some input from the workstation. At this point the workstation is in reading mode waiting for you to respond to the read. The job does not continue until after you enter the required information. Once you have entered the required line, your workstation goes back into execution mode. You may type /CANCEL when your workstation is in reading mode to cause your job to be terminated.

Break Mode — (This is also called attention mode.) The workstation enters this mode when you press the BREAK or ATTN key on your workstation (PA1 on a 3270-type workstation) and the system responds by allowing you to enter a break mode command. At this time you can enter a /CANCEL command to stop your job and return to *Go mode, or one of many other commands allowed in break mode, in which case, your workstation returns to execution mode. A preceding slash "/" is required for a

---

* Some users may not go directly to command mode, but instead, into a subsystem program such as FSI or TODO. In this case, MUSIC commands can be entered from the command line on the screen.

command entered during break mode. On a 3270-type workstation in break mode (indicated by `**Attn**` in the lower right corner), you can press the PA1 key to skip the remaining output. MUSIC does not stop your job just because your workstation has gone into break mode.

Functional Summary of Commands

The following commands are divided into groups by function, each with a brief description. Details about each command is given later.

Accessing Menu Facilities

CI	Invokes the Class Information facility for students.
FSI	Invokes the FSI (Full Screen Interface) subsystem for general users of MUSIC.
PROG	Invokes the PROG (Programmer's Menu) subsystem for student users.
CM	Invokes the Instructor's Course Management Facility.
TODO	Invokes the TODO (Time, Office, and Documentation Organizer) subsystem for electronic office functions.

Editing

BASIC	Edits a VS BASIC program.
DW370	Invokes the optional IBM DisplayWrite/370 word processing product.
EDIT	Invokes the Editor.
BIGEDIT	Invokes the Editor with a large work file.

Controlling Workstation Input/Output

/CANCEL	Terminates whatever activity is in progress.
/COMPRESS	Indicates whether or not to delete leading blanks and to compress multiple blanks to one blank in subsequent output lines.
/DEFINE	Defines 3270 program function keys.
/RECORD	Controls the writing of session recording file.
/SKIP	Skips all or part of output on the workstation.
/WINDOW	Sets the range of columns to be displayed on the workstation for the subsequent output lines until the *Go message appears.

Manipulating Files

AMS	Invokes AMS (Access Method Services) that supports VSAM.
ATTRIB	Displays the attributes of the file specified.
BROWSE	Similar to EDIT except that you are in Read Only (RO) mode.
COMPARE	Compares the lines in two different files and reports the differences.
COPY	Copies one file into another, using the same attributes, while leaving the original file unchanged.
COUNT	Displays the usage counts and creation dates for files.
DECRYPT	Decodes a file that has been encrypted by the ENCRYPT program.
DISPLAY	Displays all or part of a file at the workstation. Each line displayed is preceded by a line number.
ENCRYPT	Encrypts (codes) files for added security.
EVIEW	Invokes the VIEW program for viewing encrypted files.
FINDTEXT	Searches some or all of your files for a specified string.
FLIB	Lists the names of your files and directories. This program is one of the selections of FSI.
LIBRARY	Lists names of all files currently owned by the user.
LIST	Same as DISPLAY, except line numbers are not shown.
OUTPUT	Invoke OUTPUT program to inspect batch output files.
PRINT	Print a file on the system or auxiliary printer.
PURGE	Permanently removes (deletes) a file (aliases: DELETE, ERASE)
RENAME	Renames a file.
SORT	Sorts the records of a file.
SUBMIT	Submit a file to be executed by MUSIC batch or some other batch processor.
SUMMARY	Displays a brief summary of the contents of a file.
TAG	Displays or changes the tag information associated with a file.
VIEW	Invokes the VIEW program for viewing files of any record length, type, or size.
ZEROCNT	Resets the usage count for a file to 0 by archiving it to a temporary file, then restoring it.

Changing File Attributes

CHMOD	changes file attributes for a single file or a list of files.
MAKAPPO	changes file attribute to append only.
MAKCOM	includes file in the common index.
MAKPRIV	changes file attribute to private.
MAKPUBL	changes file attribute to public.
MAKSHR	changes file attribute to share.

Directories

CD	Changes directories.
DIR	Lists file names for directories.
MD	Makes directories.
RD	Removes directories
TREE	Provides a graphical display of your directories.

Displaying Information

CONF	Invokes the conference facility.
CONFMAN	Invokes the CONFMAN program for setting up and administering a conference.
IDP	Invokes the Information Display Program for creating help facilities and bulletin boards.
MAN	Invokes the word search facility for viewing MUSIC/SP manuals.

Signing On and Off

/DISCON	Disconnects the workstation from an active session and leaves a program running.
/ID	Signs on to MUSIC.
OFF	Signs off from MUSIC.

Getting System Information

HELP	Invokes the HELP program that provides help on a wide variety of topics.
NEWS	Lists current news items.
SESSIONS	Displays information about your multi-sessions.

/STATUS	Displays information about the user's workstation and the system.
SYSDATE	Displays the current time and date.
/TIME	Tells time of day and number of service units used for current job running up to that point.
/USERS	Displays number of users signed on to MUSIC.
VER	Displays current version of MUSIC.
WHOAMI	Displays information about your userid, TCB, ownership id, etc.

Changing Userid Attributes

NEWPW	Changes your MUSIC sign-on password.
PROFILE	Invokes the PROFILE program for displaying or changing userid attributes, passwords, and limits.

Internet Access (TCP/IP)

FINGER	Sends a one-line query to a remote Internet site.
FTP	Invokes the File Transfer Protocol program for transferring files between sites that are connected via TCP/IP to the Internet.
GOPHER	Connects to a GOPHER server for document search and retrieval.
IRC	IRC (Internet Relay Chat) allows you to carry on conversations with groups of people world-wide.
NET	Displays a list of network nodes and allows you to make a connection using FTP, GOPHER, or TELNET.
PING	The PING (Packet Internet Groper) command measures round-trip-times to internet sites.
PLAN	Invokes the Editor for updating your @PLAN file.
QFTP	Invokes Quick FTP for transferring files.
RN	Invokes the News Reader for Usenet.
TELNET	Allows you to establish sessions with computers on the TCP/IP Internet.
WEB	Invokes the line-mode browser for accessing Internet Web sites.

Sending Messages

CHAT	Invokes the CHAT facility for interactive conferencing.
GETMAIL	Invokes the GETMAIL utility to read your incoming mail and store the text in a file.
GETMINFO	Invokes the GETMINFO utility to list information about your incoming mail items and

	stores this in a file.
LM	LM (List Manager) helps you manage your discussion lists.
MAIL	Invokes the MAIL program for sending electronic mail.
MESSAGES	Control reception of messages.
REQUEST	Sends a one-line message to the console operator.
SENDFILE	Sends a file of any record length to MUSIC or CMS.
SENDMAIL	Invokes the SENDMAIL utility for a fast-track method to send a mail item.
TELL	Send a one-line message to a user.

Controlling Multi-Sessions

The following commands are available for 3270-type workstations only.

ADD	Signs the current workstation to another session.
DELETE	Deletes the current session. The DELETE command does not work if this is your only session. If the file name parameter is added, then it deletes files.
MS	Displays information about each session on your workstation.
NEXT	Goes forward to the next session in the chain.
PREVIOUS	Goes backward to the previous session in the chain.
SESSIONS	Displays information about your multi-sessions.

Graphics

GDDM	Invokes the optional IBM GDDM graphics utilities.

Personal Computer WorkStation

XTPC	Using the PCWS program, transfers files to the PC from MUSIC.
XTMUS	Using the PCWS program, transfers files to MUSIC from the PC.
PCEXEC	Using the PCWS program, starts up a PC application from MUSIC.

Miscellaneous Commands

ACCESS	Invokes the MUSIC Passthru interface to access other systems.
EXECUTE	Executes program contained in a file and passes parameters to the program when it is running.

CICS	Invokes the MUSIC interface to the optional IBM CICS product.
COBTEST	Invokes the interactive debugger for the optional IBM VS COBOL compiler.
DEBUG	Invokes the DEBUG utility for debugging programs at the machine language level.
LANG	Set National language option.
MEET	Invokes the program from the TODO facility that schedules meeting rooms.
PHONE	Invokes the program from the TODO facility that maintains a log of telephone conversations.
PIPE	Invokes a device-independent pipe-line interface under MUSIC.
POST	Submits an article to one or more newsgroups of USENET.
RN	Invokes the News Reader for exchanging messages on USENET.
ROUTE	Displays or changes the current route destination.
SCHED	Invokes the TODO scheduling program for processing personal calendars, rooms, or equipment.
SHOPAN	Displays a panel file without changing it.
TUT	Invokes the Tutorials facility for learning programming languages.

MUSIC Commands Equivalent to Other Systems

MUSIC supports some commands from other systems (UNIX, DOS, or CMS). If the command you try is identical to a MUSIC command then that command is invoked. If MUSIC has a similar command to the one you enter then you will receive a message suggesting the appropriate MUSIC command.

Below is a short list of commands for UNIX that are similar to MUSIC commands. All the commands below can be typed an the MUSIC prompt (*Go) or on the command line of most applications.

Do not forget that MUSIC offers a Full Screen Interface (FSI) facility where the following commands (as well as many other MUSIC commands and utilities) can be easily accessed through menus. To invoke FSI, type "FSI" or "MENU" at the *Go prompt.

UNIX	*MUSIC*	*Comments*
help	help	
man	man	this command accesses MUSIC manuals for text searching. It is organized by publication and not sections.
ls ls - a	lib * p	MUSIC does not have hidden files. It creates and uses several files that start with the "@" character.

dir	dir or lib	FLIB is a full screen library directory.
vi	edit	The MUSIC editor is similar to another Unix editor called pico.
cd	cd	
mkdir	md	
rmdir	rd	
passwd	newpw	
cp	copy	
rm	delete or purge	
mv	rename	
lp / lpr	print	
pwd	<ENTER>	
elm	mail	
grep	findtext	
cmp	compare	
cat	list	
logout/<CTRL-D>	off	

File Attributes

The following shows MUSIC equivalents to UNIX for accessing files:

UNIX	*MUSIC*	*Effect* chmod
ugo+/-[rwx]	makpriv*	u+[rwx] g-[rwx] o-[rwx]
	makshr **	u+[rwx] g+[rwx] o+[rx]
	makpubl	u+[rwx] g+[rx] o+[rx]
	makxo	u+[rwx] g+[x] o+[x]

* all files are PRIV (private) as default.
** to rwx, g and o must know full filename (e.g. Owner:FileName)

MUSIC Command Descriptions

Command Presentation

The remainder of this chapter gives a description of all commands in alphabetic order.

Job Control Statements (sometimes referred to as *deferred* commands) can be found in the next Chapter.

Conventions

The following conventions are used in this manual to describe the commands:

1. Upper case letters and punctuation marks represent information that must be coded exactly as shown.

2. Lower case letters and words are generic terms representing information that must be supplied. That is, a substitution must be made when coding a parameter or option so represented.

3. Information within square brackets [ ] represents an option or choice of options that may be included or omitted, depending on the requirements.

4. In the examples, information typed by the user is shown in a larger and bolder print than the information shown by the computer.

5. Underlined parameters are the default values.

6. In most cases and for most commands the preceding / is optional. The / is shown in the command description if it is necessary. For example, if a command is valid during Break mode and Command mode, the / is shown even though it is not required during Command mode. During Break mode the / is always required. The /CANCEL command requires the / at all times.

   A slash preceding a command can be used to distinguish MUSIC commands from commands for the current program. For example, if you wanted to use MUSIC's COPY command while in the Editor program, type "/COPY" (otherwise you will get the Editor's COPY command).

7. Many MUSIC commands are used with a file name. For example, "EDIT filename'. If the file name is not specified the *Input File* (@INPUT) for your userid is assumed.

Abbreviations

Most commands have abbreviations which may be used only from workstations. These abbreviations, shown in the description of each command, represent the shortest form of the command. Any abbreviation between the shortest form and the full form is valid.

ACCESS

This facility allows you to access systems outside of MUSIC such as CMS, VSE, MVS, and other MUSIC systems using the MUSIC Passthru interface.

Syntax:

```
ACCESS accessname [parameter]
ACCESS ?
ACCESS
```

Parameters:

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.

Note: Your site may not allow/support this facility.

Examples:

```
access telnet nic.ddn.mil
```

logs in to foreign host nic.ddn.mil using the VM/SP telnet program, a TCP/IP protocol.

```
access sql
```

accesses sql on the local VM/SP system.

ADD

This command is used to create extra MUSIC sessions. New sessions are added in a chain that can be accessed using the PREVIOUS (F7) and NEXT (F8) commands. Sessions are deleted with the DELETE (F5) command.

By default the F4 key has the ADD command as its definition for *Go mode. To add another session while using a full screen program like the Editor, press PA2 before the F4 key. See *Chapter 2. Workstations* under the heading "Multi-Session Support" for more information.

Syntax:

```
/ADD
```

AMS

This command invokes the MUSIC/SP Access Method Services (AMS) utility program. This program supports VSAM (Virtual Storage Access Method). AMS allocates, manipulates, and generally maintains VSAM datasets. See *Chapter 10. Utilities* for more complete information.

Syntax:

```
AMS
```

ATTRIB

This command displays the attributes of a specified file. These include the creation date, date last read, the number of lines, and its record length.

Syntax:

```
ATTRIB  filename
AT
```

Parameters:

filename is the name of the file.

Example:

```
*Go
attrib work1
*In progress
NAME=CCFP:WORK1           PRIV
TAG=
LRECL=80   RECFM=0200 (FC)  ACCESS CONTROL=00 00 C0 C0
SPACE (K):  PRIMARY=2  SECONDARY=0   MAXIMUM=-1
LINES=14    HIGH BLK=1  (MAX=3)   EOF DISPL=268
BACKUP NUMBER=147     USAGE COUNT=0    EXTENTS=1
CREATED 12MAR85  LAST READ 000000  LAST WRITTEN 12MAR85
CREATOR: CCFP000    LAST WRITER: CCFP000
*End
*Go
```

BASIC

This command invokes the BASIC Editor. This Editor is used to edit a VS BASIC programs. Use the special name of NEW to tell the Editor that you wish to start entering a new program. The commands that you use with this Editor and examples of its use are located in *Chapter 8. Processors* of this guide.

Syntax:

```
BASIC  [filename]
BA
```

Parameters:

filename is the name of the file. If a name is not specified then the Input File is edited.

BIGEDIT

This command is the same as the EDIT command, except that is uses a larger editor work file. This allows a larger file to be edited, up to about 12000 80-byte records.

BROWSE

This command allows you to browse the contents of a file. BROWSE is equivalent to the EDIT command with the RO option and the CASE IGNORE Editor command in effect. For example, "BROWSE filename" is the same as "EDIT filename RO;CASE I". See the EDIT command for more information.

Syntax:

```
BROWSE [filename] [LRECL(n)] [UIO] [MAX(n)]
B                 [n       ]
```

Parameters:

filename is the name of the file.

LRECL(n)
n is the record length to be used during the session.

UIO causes the Editor to read the file by 512-byte records.

MAX(n) specifies the maximum number of records the Editor is to read from the file.

/CANCEL

This command instructs the system to terminate whatever activity is in progress as well as stop all output associated with it. The workstation returns to *Go mode. See the notes below. If your workstation is in execution mode you must first press the ATTN or BREAK key to force the workstation to break mode before typing in this command.

You may type this command as response to any read that your job may have done.

Syntax:

```
/CANCEL [ALL]
/CA
```

Parameters:

ALL indicates that any always-program is also to be cancelled and could be used, for example, to terminate a chain of programs scheduled from REXX.

Example:

```
*Go
exec hello
*In progress
          (ATTN or BREAK key pressed)
/cancel
*Terminated
*Go
```

Notes:

1. When you are using a menu driven program such as TODO, entering the /CANCEL command to a conversational read returns you to a previous menu. /CANCEL ALL returns you directly to MUSIC command (*Go) mode.

2. Some programs may be set up as non-cancellable in which case the /CANCEL command will be rejected. This non-cancellable feature is in effect if your userid is set up to automatically run a job when you sign on.

CD

This command is used to change your current directory to the one specified in the *dirname* parameter.

Syntax:

```
CD [dirname]
CD..
CD\
```

If *dirname* is not specified, then the name of your current directory is displayed.

Directories contain names of files that you want to keep together as a group. Directories are created by specifying their name on a MD (Make Directory) command. The CD command is used to set the name of your current directory. New files that you create will be stored by default in your current directory. The system will search the current directory first when it looks for an existing file.

When you first connect to MUSIC, you are in the "root directory". You can return to the root directory by

issuing the "CD \" command.

Directories can be established within directories. For example, you can have a directory called PARTS within a directory called CAR. You can use the command "CD \CAR\PARTS" to set the current directory to the one for PARTS that is under the CAR directory. Alternately you can issue the two commands "CD \CAR" followed by "CD PARTS" to do the same thing. Notice that "CD PARTS" was not written "CD \PARTS". That would have tried to find a directory called PARTS in the root directory. But since PARTS was in the CAR directory it would not have found it.

When you are at the root directory, specifying a command like "CD CAR" or "CD \CAR" have the same affect. When you are not at the root directory, the two commands will act differently as explained above.

Use the TREE command to visually show the names of your directories and how they relate to each other.

You can add the characters "..\" in front of parm to mean the same as the name of the "parent" directory. Suppose you are in the "\CAR\PARTS" directory and want to get to the "\CAR\SALES" directory. You can do this by using the command "CD ..\SALES" command.

Examples:

```
*Go
md car

*Go
cd car

*Go \CAR>
md parts

*Go \CAR>
cd parts

*Go \CAR\PARTS>
```

CHAT

The CHAT command brings up the CHAT screen. A chat signal "*CHAT" is sent, via the TELL command, to each user you want to chat with. Anything you type in the command area is sent to the specified users (if any), via the TELL command, and appears on the recipients' CHAT screens, if they have started CHAT (if not, they get the message in the usual box at the top of the screen). Exceptions: (1) Text in the command area starting with / is assumed to be a MUSIC command and is executed. (2) A CHAT command ("chat userid") can be entered in the command area to add a user to the list of people you want to chat with; this form of CHAT command takes only 1 userid.

When someone exits from CHAT, an end signal ("*END") is sent to you. That removes the user from your list.

Syntax:

```
CHAT [userid1] [userid2] ... [-NOLOG]
```

Outgoing and incoming messages, prefixed by the sender's userid, are appended to log file *USR:@CHAT.LOG, unless -NOLOG option is used.

CHMOD

This command changes the file attributes for a single file or a list of files. When a MUSIC file is created, it is given various attributes that control how the file can be accessed by you (the owner) and other users. The commands CHMOD ("change mode") and MAKxxxx (where xxxx has various values) are used to change the access attributes of an existing file.

You can use the ATTRIB command to display information about a file, including access, tag, count, and dates. E.g. attrib myfile For info about file usage counts, see the COUNT command.

Normally the owner of a file has unlimited access (unless the file is execute-only), while other users have limited access. The following keywords are used to refer to access attributes:

PRIV The file is private. Only the owner can access it.

PUBL The file is public. Non-owners can read the file, but not change it. The file name is in the common index, so that non-owners can refer to it without specifying the owner's userid.

SHR The file is "sharable". Non-owners can read the file, but not change it. The file name is NOT in the common index; a non-owner must specify the owner's userid to access it. For example, if user ABCD creates file DATA1 as SHR, then user EFGH must use the full file name ABCD:DATA1 to read the file. SHR is indicated in the output of the ATTRIB command by "RD NOWR" (read, no write).

COM The file name is in the common index (see PUBL). If COM is used alone, the file is normally private.

XO The file is execute-only. It can be executed as a command, but not otherwise read or changed.

APPO The file is append-only. Data records can be added to the end of the file. This type of access is usually used for log files.

LOG Same as APPO.

WRITE The file is writable by non-owners.

CNT The system maintains a usage counter for the file.

Syntax:

```
CHMOD filespec attrib     or    CHMOD <file attrib
```

Parameters:

filespec is a file name specification that may include wildcard characters

<file specifies a file containing a list of file names (one file name per record, with the name starting in column 1). Each of the files in the list will be changed to the specified

attribute(s).

attrib	PRIV or PRIV,XO SHR or SHR,XO COM or COM,XO PUBL or PUBL,XO LOG or COM,LOG

Example:

```
chmod myfile shr
```

See also: MAKxxxx commands.

CI

This command invokes the Class Information facility for students. This menu program works in conjunction with CM (Course Management facility) for instructors. Help is available once the program is invoked.

Syntax:

```
CI
```

CICS

This command is used to invoke the optional IBM CICS product. For details see *Chapter 8. Processors*.

Syntax:

```
CICS
```

CM

This command invokes the Course Management facility for teachers. For more information refer to the *MUSIC/SP Teacher's Guide* or type "HELP CM".

Syntax:

```
CM
```

COBTEST

This command is used to invoke the interactive debugger for the optional IBM VS COBOL compiler. For details see *Chapter 8. Processors*.

Syntax:

```
COBTEST
```

COMPARE

The COMPARE command compares the records in two sequential files and reports records which are different and records which are in one file but not the other.

Syntax:

```
COMPARE filename1 filename2 <n-m> <ENDCHK> <TBL(k)>
COMP                        <m  >
```

Parameters:

filename1 The name of the first file. This file is called the "A" file.

filename2 The name of the second file. This file is called the "B" file.

n-m Starting (n) and ending (m) column numbers of the area to be compared in each record. The default is *n* = 1 and *m* = the larger of the two record lengths, to a maximum of 512 (i.e. for files with record length 512 or less, all characters of each record are compared). For purposes of comparison, a shorter record is considered to be extended with blanks.

When only one column number is specified, it is the ending column, and the starting column is assumed to be 1.

ENDCHK (Optional) This option causes object module END records to be compared like any other records. Without this option, object END records are compared only on columns 1-16 (provided the compare area includes 1-16).

TBL(k) *k* is a number from 2 to 120. This option specifies how many records to search ahead when unequal records are found. The command searches forward in each file for a matching record; if found, the intervening records are reported as records in one file but not the other. The default is TBL(120). You can specify a smaller number to restrict the look-ahead.

Output: Unequal and mismatched records are displayed with their line numbers and a flag "A" (records in file 1 but not in file 2), "B" (in file 2 but not in file 1), or "AB" (unequal records).

Return Codes:

0	Files compare equal in the specified column range.
1	Files compare unequal.
8	An error occurred opening one of the files.
12	An error occurred reading one of the files.
16	An invalid command option was specified.

Examples:

```
comp data1 data2
```
Compares files DATA1 and DATA2 (all of each record, to a maximum of 512 bytes per record).

```
comp data1 data2 11-30
```
Compares columns 11 through 30 of files DATA1 and DATA2.

```
comp progx.obj progy.obj 72 endchk tbl(10)
```
Compare columns 1-72 of two object modules, with full compare of object END records, and look-ahead limited to 10 records.

/COMPRESS

This command, when used in break mode, causes all succeeding output lines to be compressed. All multiple blanks appearing in each line are compressed to one blank. All leading blanks are also deleted. This option is reset the next time your workstation enters command mode.

Syntax:

```
/COMPRESS [ON ]
/COM      [OFF]
```

Parameters:

ON is the default parameter to turn on compression.

OFF cancels the effect of this command.

Example:

```
*Go
list sample
*In progress
1      2      3      4      5
1      2      3      4(ATTN or BREAK key pressed)
/compress
*OK
6 7 8 9 10
11 12 13 14 15
*End
*Go
```

Note: This function reduces output time, and can make output more readable, particularly if lines longer

than 72 characters in length are to be output on a TTY terminal.

CONF

Invokes the MUSIC conference facility for discussing topics of interest. For more information refer to the *MUSIC/SP Communications Guide* or type "HELP CONF".

Syntax:

```
CONF confname
```

CONFMAN

Invokes the Conference Manager Utility for setting up and administering a conference. For more information refer to the *MUSIC/SP Communications Guide* or type "HELP CONFMAN".

Syntax:

```
CONFMAN
```

COPY

This command is used to create a copy of a file. The contents and attributes of the original file remain as before.

Use * ("wild" character file name pattern) to specify a group of files. See the examples below.

Syntax:

```
COPY oldfile newfile <-REPL> <-SUB>
```

Parameters:

oldfile — The name of the file you wish to copy. To copy a group of files, this can be a file name pattern, containing the character "*". The * matches any string of 0 or more characters. The general form is A*B, where A and B are each 0 or more characters. A*B matches any name that starts with A and ends with B.

newfile — The name of the new file. It will be created as an exact copy of the first file, with the same attributes and data contents. If oldfile is a pattern (contains *), then newfile should also be a pattern.

-REPL This optional parameter causes the target file (newfile) to be replaced, without any prompting, if it already exists. Without this option, you are prompted whether to replace the file; reply "y" or "yes" to replace it, "n" or "no" to not do the copy. You can also enter "yes all" (to copy the file and all further files for this command, without more prompting) or "no all" (to not copy the file, or any further files for this command which already exist, without more prompting). To cancel the command, enter "/cancel".

-SUB This optional parameter, when used with a file name containing * (a name pattern), indicates that the matching names in the current (or specified) directory and all lower subdirectories, should be copied. Without this option, the copy operation applies only to files in the current (or specified) directory. The -SUB option should be used when copying an entire directory, or an entire userid. See the examples below.

Examples:

`copy work1 work2` Copies the file WORK1 to a new file WORK2. WORK1 still exists after the copy operation.

`copy work* try*` Copies all file names that begin with "work" to the corresponding names beginning with "try". For example, WORK23 is copied to TRY23, WORK.ABC to TRY.ABC, WORK to TRY, etc. Only files in the current directory are copied.

`copy x\abc y\defg` Copies ABC in directory X to a new file DEFG in directory Y. Directory Y must already exist.

`copy ab\* xyz\* -sub`

Copies all files in directory AB to directory XYZ, including all files in lower subdirectories. Target subdirectories are created as needed (including XYZ).

`copy fred:* susan:* -sub`

Copies all files in userid FRED to userid SUSAN. (This form of the command is used only by system administrators.)

COUNT

The COUNT command displays the usage count and creation date for specified files. Note that counts are maintained only for files that have the count attribute set. The usage count is the number of times the file has been opened since the creation date.

Syntax:

```
COUNT  {filename} {pattern} {<listname} {*}
CNT
```

Parameters:

filename is the name of the file.

pattern is the file specification including wild characters * and/or ?. Note that a pattern only applies to files in the current directory.

<listname is the name of the file containing a list of file names.

* give the count for all your files that have the CNT attribute.

DEBUG

This command is used to invoke the DEBUG utility for debugging programs at the machine level. For details see *Chapter 10. Utilities.*

Syntax:

```
DEBUG [parameters]
```

Note: The DBG command is faster than the DEBUG command and can be used when no parameters are needed.

DECRYPT

The DECRYPT program is used to restore a file that has been previously "encrypted". ENCRYPT and DECRYPT are two programs that respectively encrypt (code) and decrypt (de-code) files to provide added security. See *Chapter 10. Utilities* for a complete description.

Syntax:

```
DECRYPT infile outfile PW(password) REPL(ON|OFF)
```

/DEFINE

This command defines specified program function (PF) key (on a 3270-type workstation) to represent a string of characters. The subsequent pressing of that key causes the string to be executed or optionally written to the input area to allow for modification of the string before it is executed. It can also be used to redefine the full screen escape key (PA2 by default).

Your function key definitions are active for the current session except when a program using the full screen interface such as the Editor or Mail is running.

Syntax:

```
/DEFINE  PFn  [string ]
/DEF     PFn  [=string]
/DEF     Fn   [string]
/DEF     Fn  [=string]
/DEF     ESC  PFn
/DEF     ESC  PAn
/DEF     ESC  TEST
/DEF     ESC  NULL
```

Parameters:

PFn
Fn — indicates which Program Function Key to define. *n* can be any number from 1 to 12.

Some models of 3270s have 24 function keys while others have 12. In order to maintain compatibility between models, only PF1 through 12 may be defined. For ease of use, the system automatically maps PF13 through 24 into PF1 through 12.

The function keys are also used to activate functions associated with the Multi-Session and Retrieve support. Specific function keys are allocated for this purpose by default. These default definitions are as follows:

PF Key	Function	Description
PF1	HELP	Access help facility
PF4	ADD	Create a new session
PF5	DELETE	Delete a session
PF7	PREVIOUS	Activate the previous session
PF8	NEXT	Activate the next session
PF12	RETRIEVE	Retrieve the last input entered

For more information about F4, 5, 7 and 8, see *Chapter 2. Workstations* of this guide.

RETRIEVE (F12)

As commands are entered they are saved in a buffer (except in full screen mode). The RETRIEVE function allows the user to scan back through these commands. Each time F12 (Retrieve) is pressed the previous command is displayed in the input area. The user may enter the command as is, modify it, or continue scanning back by pressing the function key. If there are no more commands in the buffer, the system automatically loops back to display the most recent command.

string — is the *string* of characters to be associated with the specified key. When the key is pressed the *string* is immediately executed.

=string — same as above except the *string* is written to the input area on the 3270 screen. This allows the user to modify or add to the string before pressing the Enter key.

ESC — This keyword is used to redefine the full screen escape key. This is usually PA2 by default and is used during fullscreen applications to access the multi-session function keys. See Note 1. In addition to function and PA keys, the escape key can be defined as TEST or

NULL. TEST indicates that the TEST-REQUEST or SYS-REQUEST key is to be used as the escape. NULL means that no escape key is recognized.

Notes:

1. The Multi-Session functions assigned to F4 (ADD), F5 (DELETE), F7 (PREVIOUS), and F8 (NEXT) can be accessed during full screen programs such as the Editor and MAIL. Press the full screen escape key (PA2) before pressing the function key. Any other definitions assigned to function keys do not work during these types of programs.

2. In the full screen applications, especially the Editor, it is important to note that changes to the current screen are not recorded when you press the full screen escape key (PA2). Make sure that you have pressed an action key (i.e. ENTER) before switching sessions.

3. In *Go mode, the program SHOWPFK can be executed to display your current function key definitions.

4. A DEFINE utility program is available for defining a number of function keys at one time. This utility could, for example be set up as an AUTOPROGRAM to assign non-standard default definitions by creating a file, which includes the DEFINE utility followed by the required /DEFINE commands. Whenever that file is executed, the definitions are made. The following is an example of such a file.

```
/INC DEFINE
/DEF PF1 HELP
/DEF PF2 =EDIT
/DEF PF3 OFF
/DEF PF9 MAIL 1
```

5. From within a program, the NXTCMD subroutine can be used to define function keys for the remainder of the session. For example:

```
CALL NXTCMD('/DEFINE PF1 HELP',16)
```

   The second argument is the length of the command string in the first argument.

Example:

```
*Go
def pf1 help
*OK
def pf2 =edit
*OK
```

DELETE

This command is used to delete an extra MUSIC session on a 3270-type workstation. (A session that was previously added with the ADD command.) By default the F5 key has this definition. The DELETE command can only be used in *Go mode for deleting sessions. You cannot use this command if it is the last session or while using a full screen program like the Editor. See *Chapter 2. Workstations* under the heading "Multi-Session Support" for more information.

Syntax:

```
DELETE
DEL
```

Note: If a file name parameter is added, then this command works like the PURGE command and deletes files. See the PURGE command description for more information.

DIR

This command is used to obtain the names of the files in your current directory, or some other directory that you specify. The list is produced in alphabetical order. You can use the F or X option to display attributes of the files, in addition to their names. Use the LIBRARY command to list the file names in your entire library, rather than in a single directory.

Syntax:

```
DIR searchspec [parameters]
```

See the LIBRARY command for a description of the parameters.

"searchspec" specifies either a directory name, or a file name pattern indicating which names should be listed. A pattern contains 1 or more "wild" characters * and ?.

In the listing, directory names are indicated by "<DIR>".

Additional parameters for the DIR command:

SUB	Causes all subdirectories to be included in the search. This lists files in the entire subtree. For example, to find all occurrences of the file ABC in the current directory and all subdirectories, type: dir *abc sub
FNAME	Causes full file names (including userid and directory path) to be put out.
NODIRN	Causes the names of subdirectories to be omitted from the listing.
/W	This option provides compatibility with DOS. It lists the file names in packed form, several per line. It is equivalent to the PACK option.
/P	This option provides compatibility with the DOS option for pausing after each screen of output. It is ignored by MUSIC, since screen pausing is automatic.

Usage on Batch:

To run the DIR command in a batch job, use the following control statements:

```
/PARM searchspec <options>
/INC *COM:DIR
```

Examples:

`dir`	Lists the names of files in your current directory.
`dir /w`	Lists the files in packed form, several per line.
`dir \`	Lists the files in your root directory.
`dir\*.doc`	Lists files in your root directory whose names end in ".DOC".
`dir abc* x`	Lists names in your current directory that start with ABC, and requests additional information about each file.
`dir test`	Lists the names of files in the TEST directory (which is assumed to be a subdirectory of the current directory)
`dir \mydir p`	Lists the names of files in the MYDIR directory, in packed form (several names per line)
`dir *\`	Lists the names of all subdirectories in the current directory.
`dir * sub`	Lists the names of all files in the current directory and all subdirectories of it.
`dir *abc sub`	Lists all occurrences of the file ABC in this directory and all subdirectories.

/DISCON

This command disconnects your workstation from the active session while leaving a program running. The program continues to run until one of the following events occur: 1) the program requests input from the workstation; 2) the program's output exhausts the supply of output buffers or; When one of the above events occur, the program stops execution until the workstation is reconnected.

To reconnect to the disconnected session, use a /ID command specifying the original userid. The reconnection must be made from the same type of workstation. For example, if the user was on a 3270-type workstation, then the reconnection must also be made from a 3270-type workstation, not an ASCII one.

Syntax:

```
/DISCON
/DISC
```

Notes:

1. This command is not valid in command mode.

2. If a userid has been restricted from using the Multi-Session feature then this command is not allowed.

3. Each installation defines a limit to the number of simultaneous users who can use the disconnect and

Multi-Session features. When this limit is exceeded then the /DISCON command is rejected.

DISPLAY

This command causes all or part of a file to be displayed on the workstation. Each line displayed is preceded by a 4 digit line number. This line number is not stored as part of your file, nor does it have any special significance except to show the order that they are stored in the file. Line numbers beyond 9999 will display only the last 4 digits. Files can be listed without numbers by the use of the LIST command. You cannot use this command from batch, though you can obtain listings with or without line numbers by using the UTIL program described in *Chapter 10. Utilities* of this guide.

Syntax:

```
DISPLAY  [filename] [x     ] [y     ]
D                  [LAST  ] [LAST  ]
                   [LAST-n] [LAST-n]
```

Parameters:

filename	the name of the file to be displayed. If not specified, the Input file is displayed.
x	is the line number (or the first line number of a group of line numbers) of the file to be displayed.
y	is the last line number of a group of line numbers to be displayed. If *y* is larger than the number of lines in the file, the display continues to the end of the file. If *x* or *x* to *y* is not specified, the entire file is displayed starting at the beginning.
LAST	is used in place of *x* and/or *y* refers to the last line of the file. LAST-1 refers to the second to last line, etc.

Examples:

```
*Go
display sample
*In progress
0001 LINE 1
0002 LINE 2
0003 LINE 3
*End
*Go

display sample 2 3
*In progress
0002 LINE2
0003 LINE3
*end
*Go
```

Note: The parameters for this command may appear in any order.

Messages:

`REQUEST OUT OF RANGE`
The file to be displayed is a null file or the specified line number is out of the range in the file.

DW370

This command is used to invoke the optional IBM DisplayWrite/370 word processing facility. For details see the *MUSIC/SP Mail and Office Applications Guide*.

Syntax:

```
DW370 filename [options]
```

EDIT

The EDIT command is used to invoke the Editor program to edit a file. For a description of various Editor commands, enter HELP or "HELP cmd" (*cmd* is the name of the Editor command) when you are in edit mode of the Editor (not MUSIC command mode). For more information about the Editor and this EDIT command refer *Chapter 7. Using the Editor* of this guide. See also TEDIT and BIGEDIT.

Syntax:

```
EDIT [filename] [NEW] [LRECL(n)] [fmt] [RO] [NOLOG] [UIO] [MAX(n)]
E                    [n       ]
```

Parameters:

filename — is the name of the file to be edited. If *filename* is omitted, the Input file is assumed and other parameters must be omitted.

NEW — If NEW is specified after *filename* then the Editor assumes the file being edited is a new file and goes to input mode directly. If not specified, OLD is assumed. If *filename* is excluded and NEW is specified, then the Editor goes directly to input mode. (This file will not have a name until you specify one with either the NAME, SAVE, FILE, or EXEC Editor commands.)

LRECL(n) — is the record length to be used during edit. It must be a number from 1 to 8192. If not specified, the record length of the file being edited is used, or 80 for a new file. LRECL(*n*) can be abbreviated to simply *n*, if *n* has two or more digits.

fmt — is the record format of the file to be edited. It can be F (fixed), FC (fixed compressed), V (variable), and VC (variable compressed). If not specified, the original record format of the named file (if the file is old), or FC (if the file is new) is used.

RO — Causes the Editor to go into Read Only (RO) mode. This is equivalent to the BROWSE

command. In full-screen mode, the text on the screen is protected (non-modifiable). Note that RO also implies NOLOG.

NOLOG — Suppresses the Editor restart feature by not looking for a log file or creating a new one.

UIO — Causes the Editor to read the file by 512-byte blocks, using MFIO UIO requests. Each block becomes a record for the edit. This lets you edit (but not change) record format U files, or files that cannot be read sequentially. The record length for the edit is automatically set to 512. UIO implies RO and NOLOG. Do not use the FILE or SAVE Editor commands to write to the file being edited, since UIO is not used for the writes.

MAX(n) — Specifies the maximum number of records the Editor is to read from the file being edited. This option implies RO and NOLOG. It is useful for looking at the first part of a large file. For example: EDIT BIGFILE MAX(500).

Examples:

`EDIT MYPROG` edits the file called MYPROG.

`E X NEW 100` edits a new file called X. The record length of the file is 100.

Messages:

`*** FILE NOT FOUND`
The file to be edited cannot be found in the Library. Make sure that the file name specified on the EDIT command is correct.

ENCRYPT

The ENCRYPT program is used to code a file for added security. ENCRYPT and DECRYPT are two programs that respectively encrypt (code) and decrypt (de-code) files. CAUTION: If the encryption password is forgotten, there is no way to restore the file to its original form. See *Chapter 10. Utilities* for a complete description.

Syntax:

```
ENCRYPT infile [outfile] [PW(password)] [REPL(ON|OFF)]
```

ERASE

Same as the PURGE command. See the description for the PURGE command for information.

EVIEW

Invokes the VIEW program for viewing encrypted files.

Syntax:

```
EVIEW [filename] [password]
```

EXECUTE

This command is used to run a program from a file (execute the file). This file should contain the appropriate job control statements control program execution. See *Chapter 6. MUSIC Job Control Statements* for details. The command scanner defaults to EXECUTE if it does not recognize the input as a valid MUSIC command. So the command verb (EXECUTE), can be omitted from in front of the filename. Unless the filename happens to match a MUSIC command the file can be executed simply by typing its name.

Syntax:

```
EXECUTE  filename  [ppppp]
EX

filename  [ppppp]
```

Parameters:

filename — The name of a file which contains job control statements to run the program.

ppppp — If specified, it is the parameter string which is to be passed to the program. The use of "EXEC filename ppppp" is logically equivalent to "EXEC SAMPLE" where SAMPLE is the name of a file containing the following statements:

```
/PARM ppppp
/INCLUDE filename
```

A /PARM statement is only generated if the *ppppp* field is specified.

Example:

```
*Go
exec hello
*In progress
      (output)
*END
*Go

hello                (same as "exec hello")
*IN PROGRESS
      (output)
```

Note: If *filename* does not correspond to any MUSIC command or command abbreviation, the EXEC part can be omitted (unless the user's profile disallows this). This usage is called the implied EXEC command.

Messages:

`XXX ERR11 FILE NOT ACCESSIBLE`
The file XXX cannot be found or the user is not allowed to access the file. Make sure that the file name specified on the EXECUTE command is correct.

FINDTEXT

FINDTEXT is used to search through some or all of the files in your library for a text string. It produces a list of text lines, with line numbers, when the text is found. The file name is also reported.

FINDTEXT supports both full screen and line mode usage. If you want to enter all parameters on screen fields, enter FINDTEXT or FT without parameters. Otherwise a screen is only provide, to assist you in correcting parameters that are in error.

Syntax:

```
FINDTEXT 'text'[FILE(spec)] [FROMLINE(n)] [TOLINE(n|END)]
FT             [FROMCOLUMN(n|END)] [TOCOLUMN(n)] [FIRST(YES|NO)]
               [FINDS(n|ALL)] [CASE(I|R)] [OUTPUT(filename)]
```

Parameters:

'text' text is a character string that is to be searched for in the list of files defined by FILE. If invoked from *Go, the quotes are required when any of the options below are also specified.

File(spec) *spec* can be one of:
a) a library pattern such as "*.s", "*", or "*work.*.?" etc. All file names in your library that match the specified pattern will be searched for "text".
b) a file that contains a list of files to be used in the search of "text", specified as "<file-name" where filename is an existing file. If this parameter is not specified then all your files are checked.

FROMLine(n|END)
n is an integer greater than 0, that specifies the starting line within each file that the search is to begin at. The default is 1. (Abbreviations: FROML and FL.)

TOLine(n|END) *n* is an integer greater than 0, that specifies the last line within each file that the search is to stop at. Keywords "all", "max", and "end" are used to indicate the entire file. The default is end. (Abbreviations: TOL and TL.)

FROMColumn(n) *n* is an integer greater than 0, that specifies the starting column within each line of the file that the search is to begin at. The default is 1. (Abbreviations: FROMC and FC.)

TOColumn(n) *n* is an integer greater than 0, that specifies the last column within each line of the file that the search is to stop at. Keywords "all", "max", and "end" are used to indicate the entire file. The default is end. (Abbreviations: TOC and TC.)

FIRst(YES|NO) When set to "yes", causes the search to stop at the very first match. The default is no.

FINds(n|ALL) specifies the number of times to search within each file or all lines. After *n* matches in a

file, searching is halted in that file. The default is 1.

Case(I|R) When set to I (ignore) the matching is done as if all characters in "text" and the file were in the exact same case. So that "Case" will match with the string "case". When R (respect) is used "Case" will not match "case". The default is ignore.

Output(filename) This option defines where the output of the search will be placed. You can enter here any file name. The default is "*terminal" to display output at your workstation.

Examples:

1. In this example the string "call tstime(" will be searched for in the library files that end in ".s" .

```
FT 'call tstime(' f(*.s)
```

2. This example searches for the string 'montreal' in the files that end in ".doc" and stores the output of the search in file LIST. Since we want to find only lower case "montreal", we will set case to respect.

```
ft 'montreal' f(*.doc) c(r) o(list)
```

3. In this example we will locate and display only those files where "/inc gork" occurs on line 5 of the file.

```
ft '/inc gork' froml(5) tol(5)
```

FINGER

The FINGER command allows you to send a one-line query to a remote Internet site, and receive back information on who is logged into that remote site.

Note that you can create a "plan" file, using the PLAN command. When remote users query your userid on MUSIC, the contents of this file is transmitted to them.

Syntax:

```
FINGER [/W ][user-name]@site-name[@site-name...]
```

Parameters:

user-name is the name of the login ID that you are querying. If this is omitted, then information about all users logged in is displayed. Note that without a '@' present, a local user-name is assumed.

/W is an option sent to the remote host requesting more detail in the displayed information. (Note: the remote host may ignore this request.) A blank **must** follow this option.

@site-name is the name of the remote site to send the query to. Note that you may have a number of sitenames strung together. In this case, FINGER sends all of the text to the right of the last

'@' character to the remote host - it will do the same - and will then display the information that it receives from the remote host. Note that you should fully qualify all of leftmost addresses, as other systems may not extend the address with the local site's domain name as MUSIC does. (Note that site-name may be set to '*' - this indicates the local site (only valid for the right-most address.)

Note: Note that the @ character must be specified in order to indicate an address; otherwise, the local site is assumed

Examples:

```
all users at local site:    finger *

user xx00 at local site:    finger xx00

user fred at remote site    finger fred@remote.site.edu

user fred at remote site,
extended info requested:    finger /W fred @remote.site1.edu

user fred at site 1 via     finger /W fred @rmt.site1.edu@rmt.site2.edu
site2:                      finger /W fred @rmt.site1.edu@rmt.site2.edu
```

FLIB

This command lists the names for your files and directories, and allows selection of these names for browsing, editing, renaming, copying, etc.

Syntax:

```
FLIB filespec
```

You can invoke this program by using the FLIB command, selecting the file management item on the FSI main menu, or invoking FSI with a file specification. Include the *filespec* parameter to specify which group of files you wish to access; otherwise the same *filespec* from the last FLIB or FSI command is used again. Help is provided once this program is invoked.

FSI

This command is used to invoke the FSI (Full Screen Interface) subsystem. This facility allows you access to various components of MUSIC system through a series of selection menus. For details see *Chapter 1. Introduction*.

Syntax:

```
FSI [filespec|n]
```

FTP

This command is used to invoke the File Transfer Protocol program for transferring files between sites that are connected via TCP/IP to the Internet. A list of Internet addresses that allow anonymous access is supplied. A brief description of FTP can be found in *Chapter 10 - Utilities* under "FTP and TELNET from MUSIC". For full details refer to the *MUSIC/SP Communications Guide*.

Syntax:

```
FTP [internet address]
```

Help is provided once this program is invoked.

GDDM

This command is used to invoke the optional IBM GDDM graphics utilities. For details see *Chapter 8. Processors*.

Syntax:

```
GDDM
```

GETMAIL

This command invokes the GETMAIL utility program to read your incoming mail and store the text in a file.

Syntax:

```
GETMAIL filename options
```

filename	is the name of the file for storing the text of all of the incoming mail items. A file name must be specified with this command.
options	A number of options can be used with this command and are specified after the file name separated by blanks. If an option is repeated the last option specified takes precedence.

Options:

APPEND	store the mail text at the end of the file.
REPLACE	store the mail text in a new file.
DELETE	delete the mail item after it has been processed.
KEEP	do not delete the mail item after it has been processed.

N	N is an integer number which represents the incoming mail item to get. (if ALL is used before N, then N is used.)
ALL	process all incoming mail items in the mailbox. (if N is used before ALL, then ALL is used.)
MSGSON	display error messages.
MSGSOFF	suppress the display of error messages.
DISCARD	discard the list of information that is supposed to be written to the file.
SHRINK	shrink the mailbox to its absolute minimum file size.
SELECT	select incoming mail items in the mailbox via a criterion.
SELECTN	select incoming mail items in the mailbox via a criterion.
FOR(name)	allows you to list the mail information for *name's* mailbox provided you are allowed as a surrogate for *name's* mailbox. This parameter is similar to the MAIL program command FOR.
UIDL(n)	get the incoming mail item designated by n, where n represents the unique-id listing, UIDL, for the mail item. (if UIDL(n) is given, it always takes precedence over N and ALL.)

For more information type "/help getmail"

GETMINFO

This command invokes the GETMINFO utility program. It gets a list of information about your incoming mail items and stores this in a file.

Syntax:

```
GETMINFO filename options
```

Parameters:

filename	is the name of the file for storing the list of incoming mail. A file name must be specified with this command.
options	A number of options can be used with this command and are specified after the file name separated by blanks. If an option is repeated the last option specified takes precedence.

Options:

APPEND	store the mail information at the end of the file.
REPLACE	store the mail information in a new file.
DELETE	delete the mail item after it has been processed.
KEEP	do not delete the mail item after it has been processed.
N	N is an integer number which represents the incoming mail item to get. (if ALL is used before N, then N is used.)
ALL	process all incoming mail items in the mailbox. (if N is used before ALL, then ALL is used.)
MSGSON	display error messages.
MSGSOFF	suppress the display of error messages.
DISCARD	discard the list of information that is supposed to be written to the file.
SHRINK	shrink the mailbox to its absolute minimum file size.
SELECT	select incoming mail items in the mailbox via a criterion.
SELECTN	select incoming mail items in the mailbox via a criterion.

FOR(name) allows you to list the mail information for *name's* mailbox provided you are allowed as a surrogate for *name's* mailbox. This parameter is similar to the MAIL program command FOR.

UIDLS displays a unique-id listing, UIDL, for each of the selected incoming mail items. Since the UIDL associated with a specific mail item is never reused for the life of the item, using the UIDL is a more exact method to specify a particular item. GETMAIL with the UIDL(n) parameter can be used to get individual mail items.

Here are some examples of GETMINFO commands:

```
GETMINFO NEWINFO APPEND ALL SELECT TYPE NEW

GETMINFO NEWINFO APPEND 1 KEEP

/INC *COM:GETMINFO
NEWINFO APPEND ALL KEEP -
SELECT SUBJECT THIS IS A VERY LONG SUBJECT THAT GOES ON AN ON

/INC *COM:GETMINFO
NEWINFO APPEND ALL KEEP -
SELECT (SUBJECT THIS IS A VERY LONG)&(SEND 11JUL91)

GETMINFO NEWINFO ALL DELETE DISCARD SELECT BEFORE 01APR93

GETMINFO NEWINFO UIDLS KEEP
```

For more information type "/help getminfo"

GOPHER

This command invokes the Gopher server (provided your site has TCP/IP connections). Gopher is a document search and retrieval system running on the Internet.

Syntax:

```
GOPHER [site_address]
```

Help is provided once the program is invoked. For more information, refer to the *MUSIC/SP Communications Guide*.

HELP

This command invokes the MUSIC HELP facility for accessing information about a wide variety of topics. Most of this guide can be found online through this facility. For example, you can enquire about how to use a particular MUSIC command or a utility program. If the information about a particular item is not available, the item will be recorded in a system log file which will be reviewed by the MUSIC administrator.

Workstations with full-screen display enables you to easily browse the HELP facility through menus and text

screens. You are able to page forward and backward. You can place your cursor on any highlighted topic names to jump from one topic to another.

The HELP command can be used when you are in command mode. If just HELP is entered without a topic, general information about MUSIC is given and a list of general topics is displayed. You can then go into more detail on any of the topics available.

Syntax:

```
HELP [topicname n]
```

topicname — is the name of the item on which you want information. *topicname* can be the name of an individual item or the name of a menu of items.

n — *n* is the item selection code (usually a number) from a help menu. *(n* is only used if you know the item selection code in advance.)

Examples:

`HELP HOURS` gets the hours of operation of MUSIC.

`HELP TOPICS` displays a list of the available topics under this facility.

Notes:

1. The HELP command used from command mode accesses MUSIC's general help facility. Other help facilities are provided with a variety of programs on MUSIC. For example, when you are using FSI (Full Screen Interface), the command "HELP" (or F1) gives you the FSI help facility. If you wanted MUSIC's general help facility while you are within FSI, use the command "/HELP". The slash is necessary to distinguish MUSIC commands from FSI commands.

2. Systems administrators should refer to the *MUSIC/SP Administrator's Reference* for information about updating existing help facilities and creating new ones.

/ID

This command is used to identify and validate a user signing on the system. It performs the following functions:

- identifies you to the system (by userid).
- asks for the password assigned to your userid. (The area where the password is typed is blacked out by the system.)
- displays system information messages.
- sets up default tabs, etc., if any are defined in your *profile*. (See the PROFILE utility for more information.)
- schedules your AUTO/ALWAYS program (if any exists).

A job control statement form of the /ID command is used when submitting batch jobs. See *Chapter 3. Using Batch* of this guide.

If your userid has a fund restriction, then the funds remaining for the userid as of the last accounting are displayed at sign-on time. When the funds remaining is less than 10% of the total allocation, or $10, whichever is less, a warning message is issued. If the allocation of funds for MUSIC usage have been used up, the message `*USERID OUT OF FUNDS` appears and the user must contact the MUSIC System Administrator for additional funds.

Messages of general interest may be received while signing on. These messages are sent to all workstations and may contain news items, etc, or may instruct you to type the NEWS command to receive more information.

MUSIC automatically requests a /ID command be entered when a workstation first connects to the system. The /ID command can also be entered from command (*Go) mode, should the user wish to change the current userid in mid-session.

Syntax:

```
/ID   [tn,]userid[,ident][;trmcls]
```

Parameters:

tn — is the terminal (workstation) identification number. This optional parameter may be used to distinguish one session from another for accounting purposes. If specified, this parameter must be a decimal number between 0 and 99 and must be separated from the /ID by at least one blank.

userid — is your 1-16 character userid assigned to the MUSIC user by the MUSIC System Administrator.

ident — is an identification field of up to eight characters. This field is optional unless your userid has been set up to require it.

trmcls — is the terminal (workstation) class parameter which provides MUSIC with additional information about the physical characteristics of your workstation. Rather than always entering this parameter on the /ID command, a default terminal (workstation) class can be set in the user profile. (See TERM parameter of the PROFIL utility). The following lists the workstation classes that are available on the base MUSIC system. Your installation may define other workstation class names that you could use in this field. Those flagged with the * are the defaults for the particular type of workstation and need not be specified on the /ID command.

trmcls	Terminal Model
3270	*Any of the 3270 family of terminals
3270A	3277 with DAF/APL
3270B	3270 APL/TEXT feature
ASCII	*Any ASCII printer or ASCII video display terminal
3101	3101 terminal or PC running 3101 emulator
PCWS	PC running MUSIC's PC Workstation Software
IBMPC	PC running Async Communications Software
2741	*Any of the 2741 family of terminals

Note: For users with 3270-type workstations, it is important for the Editor to know the actual number of function keys on the workstation being used. On the sign-on screen, if the /ID command and password are entered by pressing the ENTER key, MUSIC assumes 24 function keys for IBM 3178, 3278 or 3279 terminals. Otherwise, MUSIC assumes 12 function keys (e.g., IBM 3277). You can override these assumptions by pressing the highest numbered function key instead of the ENTER key when signing on. (To receive the sign-on screen when you are in *Go mode type /ID.) Refer to the "Editor Full Screen Mode" in *Chapter 7 - Using the Editor* of this guide.

Examples:

Example 1 - Sign-on Screen on a 3270-type workstation

```
*MUSIC/SP -- PLEASE SIGN ON

ID Command: /ID _              <-- The cursor is positioned
                                   on this line for typing
  Password:                        your userid.  Use the NEW
                                   LINE or TAB key to skip to
                                   the password field, type
                                   your password, press ENTER
                                   (or F12 or F24).
F1/13 - HELP    F3/15 - /OFF
------------------------------------------------------------

*Userid last signed on 17:10 1993/05/14
*Sign-on 1993/05/14, Time=09:10, Port=08E, TCB=104
*Funds Remaining as of Last Accounting..$168.17
*Go
```

Example 2 - Signing on an ASCII terminal

```
*MUSIC/SP  --  SIGN ON
/id user
*Password?
XXXXXXXX
*Userid last signed on 08:26 1993/01/14
*Sign-on 1993/05/14, Time=10:45, Port=0B0, TCB=019
*Funds Remaining as of Last Accounting..$ 184.52
*Go
```

IDP

This command invokes the Information Display Program (IDP). This program is used to create help facilities and bulletin boards.

Syntax:

```
IDP
```

Help is provided once the program is invoked. For more information, refer to the *MUSIC/SP Campus-Wide*

Information Systems (*CWIS*) *Guide*.

IRC

Initiates the Internet Relay Chat program for communicating with others connected to the Internet.

Syntax:

```
IRC
```

Help is provided once the program is invoked. For more information, refer to the *MUSIC/SP Communications Guide*.

LANG

The LANG command allows you to display or change the default language setting 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. Enter "LANG ?" to get a list of the languages supported at your site.

Syntax:

```
LANGUAGE [language] [?]
LANG
```

Parameters:

language specifies the language of your choice.

? lists the languages available at your site.

LIBRARY

This command is used to obtain a list of file names saved in the under your userid. The list is produced in alphabetical order. The abbreviation of each parameter is shown under its full form. Use the DIR command to list the file names in your current directory.

Syntax:

```
LIBRARY searchspec [FULL][TAG][VSAM][PACK][NOSORT][SAVE(name)]
LIB                [F   ][T  ][V   ][P   ]        [S(name)   ]

                   [COM][X][SPACE(n)][FNAME][SORT(type)][APPEND]
                                           [SO(type)  ][AP    ]
```

The LIBRARY command can be entered without specifying *searchspec* or parameters. If you wish to add parameters to this command then you must specify *searchspec*. Parameters after *searchspec* can be in any order. They must be separated by 1 or more blanks.

Parameters:

searchspec — (search specification) specifies which file names, belonging to the user, are to be searched for in the Save Library index. It may be an actual file name, in which case only that file is listed. Or, the string may contain one or more wild characters ? and *, in which case all file names matching the pattern are listed. A ? matches any single character in the corresponding position of a file name. A * matches any group of 0 or more characters. If the *searchspec* parameter is not specified, all the file names belonging to the user are listed.

To list all your filenames you can enter LIBRARY or if you wish to specify parameters then enter "LIBRARY * parameters". The * is your search specification indicating all files on your userid.

FULL — indicates that for each file listed, its corresponding attributes are also given. See the discussion below on file attributes about the information provided.

X — This option is similar to FULL, but the output is in a slightly different format and includes time of last open for write, userid of last writer, and number of records. An asterisk (*) appears after the file size if the file has releasable unused space.

TAG — is the same as specifying FULL except that it also displays the tag information for each file. TAG implies FULL.

VSAM — lists only VSAM (Virtual Storage Access Method) files. VSAM implies FULL.

COM — lists only files in the common index.

PACK — Normally only one file name is displayed per line for the LIBRARY command. Specifying PACK indicates that several file names may be combined on one line. PACK cannot be specified if a FULL, TAG or VSAM parameter is used. Alternate forms: P, WIDE, W, /W

NOSORT — causes the file names to be put out in unsorted order. This causes output to appear immediately. When the library listing is sorted then there is a short delay.

SORT(type) — specifies how the listing is to be sorted. Type is NAME (sort by file name), SIZE (sort by file size in K), RDATE (sort by date last read), WDATE (sort by date last written), or UDATE (sort by reference date, which is the higher of the read and write dates). Abbreviations are N, S, R, W, U. The default is SORT(NAME). A minus sign (-) can be placed before the type to sort in descending order. If the SORT option specifies a type other than NAME, and neither X nor FULL is used, then the X option is automatically used. Examples: SORT(WDATE), SORT(-S).

SAVE(name) indicates that the output of the LIBRARY command is to be saved in a file instead of displayed on the workstation (unit 6). If *(name)* is specified with the SAVE parameter, the output is written to the file called *name*. If *name* is omitted, the name @LIB is used. The original contents in the file is overwritten if the file already exists. See also the SPACE(n) and APPEND parameters below.

SPACE(n) specifies the initial space (in K) to be allocated for the new file specified by the SAVE parameter. The default is SPACE(32), meaning 32K.

APPEND specifies that the output should be written to the end of the file given by the SAVE parameter, after any existing data. This is useful for accumulating the output of several LIB or DIR commands into a single file. If the output file does not already exist, a new one is created. In all cases when the APPEND option is specified, unused space in the output file is NOT freed at the end of the command; this is in anticipation of further appends to the file. Without APPEND, unused space is freed.

FNAME Causes full file names (including userid and directory path) to be displayed.

File Attributes

When FULL is specified on the LIBRARY command, extra information about each listed file is also displayed (See examples below). This information consists of:

name indicates the name of the file.

Rsiz indicates the logical record length of the file.

Rfm indicates the record format of the file. The possible record formats are F (fixed length), FC (fixed compressed), V (variable length), VC (variable compressed), and U (undefined).

Size indicates the size of the file in number of K (1024) bytes incremented in 2K bytes. The smallest size for a file 2K.

Used indicates the percentage of the file space that is used.

Ext indicates the number of extents of disk space that are used by the file.

Ref indicates the date the file was last opened for reading only. 0000000 means that the file has not been referenced since it was last written on.

Write indicates the date the file was last opened for writing.

T indicates the type of file. The file is in the common library (public) when the letter C appears in this column. If a V appears, the file is a VSAM file.

Own indicates the access control of the file for the owner. R means read access is allowed. W means write access is allowed. X means only read access for execute-only is allowed. A means only write access for append is allowed.

Other indicates the access control of the file for non-owners. The meaning of various letters is the same as listed above.

Usage on Batch:

To run the LIBRARY command in a batch job, use the following control statements:

```
/PARM searchspec <options>
/INC *COM:LIB
```

Examples:

```
*Go
library
*In progress

Files - CCXA 21OCT91    11 Files
   1 @ELOG.000
   2 ALICE
   3 BOB
   4 CAROL
   5 DATAXX
   6 FIL18
   7 KETTLE.S
   8 MISC\
   9 MISC\FILE1
  10 MISC\FILE2
  11 ROY
*End
*Go

lib * s(mylib)
*In progress

11 filenames saved to file MYLIB
*End
*Go

lib * full
*In progress

Files - CCXA 21OCT91    12 files
    File Name        Rsiz Rfm  Size Used Ext  Ref    Write T  Own Other
1 @ELOG.000            72 VC     2K   0%  1 21OCT91 21OCT91   RW
2 ALICE                80 FC     2K  33%  1 15OCT91 12OCT91 C R
 etc.
Total size of the listed files is  68K
```

Note: Files that begin with @ are often generated and regenerated by the programs that you use. For example, @ELOG.000 is the Edit log file created by the Editor program.

LIST

This command displays all or part of the contents of a file. This command is similar to DISPLAY except no line numbers are shown. This command cannot be used in a job run from batch. Instead you may use the UTIL program from batch to accomplish a similar function. The UTIL program is documented in *Chapter 10. Utilities* of this guide.

Syntax:

```
LIST  [filename] [x     ] [y     ]
L               [LAST  ] [LAST  ]
                [LAST-n] [LAST-n]
```

Parameters:

filename	the name of the file to be listed. If not specified, the Input file is listed.
x	is the line number (or the first line number of a group of lines) of the file to be listed.
y	is the last line number of a group of line numbers to be listed. If *y* is larger than the number of lines in the file, the list continues to the end of the file. If *x* or *x* to *y* is not specified, the entire file is listed starting at the beginning.
LAST	is used in place of *x* and/or *y* and refers to the last line of the file. LAST-1 refers to the second to last line, etc.

Example:

```
*Go
list sample
*In progress
LINE 1
LINE 2
LINE 3
*End
*Go
list sample last
*In Progress
LINE 3
*END
*Go
```

Note: The parameters for this command may appear in any order.

Messages:

`REQUEST OUT OF RANGE`
The file which is to be listed is a null file or the specified line number is out of the range of the file.

LM

This command invokes the List Manager facility for managing your subscriptions to discussion lists.

Syntax:

```
LM
```

Help is provided once this program is invoked.

MAIL

This command invokes the Electronic Mail Facility for sending and receiving mail. Help is available once the facility is invoked. For complete information see the *MUSIC/SP Mail and Office Applications Guide*.

Syntax:

```
MAIL [n]
```

MAKxxxx

The MAKxxxx commands allows you to change the attribute of a file.

Syntax:

```
MAKAPPO filename
MAKCOM filename
MAKPRIV filename
MAKPUBL filename
MAKSHR filename
```

Note: Any of the MAKxxxx commands can be typed in the margin area on the FSI file management screen, to change the attributes of the corresponding file.

Example:

```
makshr myfile
```

See the CHMOD command for details about file attributes.

MAN

This command invokes the word search facility for displaying MUSIC manuals. Help is provided once the program is invoked.

Syntax:

```
MAN
```

MD

This command makes a new directory.

Syntax:

```
MD dirname
```

The *dirname* parameter specifies the file name for the directory. Directories contain names of files that you want to keep together as a group. The CD command is used to specify the name of your current directory.

Notice that you must issue the CD command to actually use the directory that you create.

Directories can be established within directories. For example, you can have a directory called PARTS within a directory called CAR. You can use the command "MD PARTS" to make a directory called PARTS under your current directory.

Use the RD command to remove directories that you have created.

Use the TREE command to visually show the names of your directories and how they relate to each other.

Examples:

```
*Go
md car

*Go
cd car

*Go \CAR>
md parts

*Go \CAR>
cd parts

*Go \CAR\PARTS>
```

MEET

Invokes the TODO program called MEET. It works in conjunction with the SCHEDULE program to help schedule a meeting by: finding free time in the schedules of available rooms and attendees; adding the meeting to these schedules; and sending mail. See the *MUSIC/SP Mail and Office Applications Guide* for details.

Syntax:

```
MEET        or        TODO M
```

MESSAGES

This command controls whether you are receiving messages or not. By default messages from other users and programs such a MAIL are displayed in a pop up window as soon as they arrive. These can be suppressed by specifying the OFF option. Messages from the system operator cannot be suppressed.

Syntax:

```
MESSAGES [ON]
M        [OFF]
```

MNSORT

This command is used to invoke the MNSORT utility program for sorting data on disk or tape. (See the SORT command for sorting data in a file.) For details about MNSORT see *Chapter 10. Utilities* under the topic "Sorting".

Syntax:

```
MNSORT
```

MS

This command is used to display information about the MUSIC sessions that you are currently signed onto. The terminal (workstation) ID, your userid, and the last command entered for each session are listed.

Syntax:

```
MS
```

NET

This command displays a list of network nodes and allows you to make a connection using the FTP, GOPHER, or TELNET commands. For more details see *Chapter 10 - Utilities* under "FTP and TELNET from MUSIC". For complete documentation see the *MUSIC/SP Communications Guide*.

Syntax:

```
NET
```

NEWPW

This command is used to change your MUSIC sign-on password.

Syntax:

```
NEWPW
```

NEWS

This command is used to list current news items. This news facility is used to communicate items of interest to MUSIC users such as hours of operations, new programs available, etc. Some installations may choose not to maintain this list.

The items in this news file are arranged so that the newest item is displayed first.

Syntax:

```
NEWS
NEW
```

Example:

```
*Go
news
*In progress

CURRENT NEWS - MAY 14, 1975     12.25.05

......
......
```

/NEXT

This command is used to go to the next session in the extra session chain. (Extra sessions are added with the ADD command.) By default the F8 key has this definition in command mode. To access the next session while using a full screen program like the Editor, press the full screen escape (PA2) before the F8 key. See *Chapter 2. Workstations* under the heading "Multi-Session Support" for more information.

Syntax:

```
/NEXT
```

OFF

This command is used to terminate a workstation session and to close off accounting for that session. Once the OFF command is entered the system disconnects the hook up (ie. telephone line) to the computer. The connect time and service units used are displayed giving a rough indication of the cost for the session. The charge for I/O to the workstation and the surcharge for larger user regions are not included in the number of service units used.

Syntax:

```
OFF  [HOLD]
```

Parameters:

HOLD — If OFF HOLD is specified, the workstation session is terminated and workstation specifications such as input/output tab settings, tab and backspace characters, are all reset. The workstation remains connected to MUSIC for a short period of time so a new /ID command can be entered to begin another workstation session. In other words, you do not have to dial MUSIC before signing on.

Example:

```
*Go
off
*Good-bye. Connect =  01:25, S.U. =    24
```

OUTPUT

This command is used to invoke the OUTPUT program to inspect output sent back to MUSIC from batch jobs. A full description of the OUTPUT program is given in *Chapter 3. Using Batch* of this guide. Online help is available once the program has been invoked.

Syntax:

```
OUTPUT
```

PCEXEC

Executes the specified DOS command or PC program as though it had been entered at the DOS prompt. For complete details, refer to the *MUSIC/SP Personal Computer WorkStation User's Guide*, or type "HELP PCEXEC".

Syntax:

```
PCEXEC command [-Hold] [-Direct]
               [-H   ] [-D     ]
```

PHONE

Invokes the TODO program called PHONE. It keeps a log of your telephone conversations. See the *MUSIC/SP Mail and Office Applications Guide* for details.

Syntax:

```
PHONE        or         TODO 3
```

PING

The PING (Packet Internet Groper) command measures round-trip-times to Internet sites. For a brief explanation about TCP/IP and Internet see *Chapter 10 - Utilities* under "FTP and TELNET from MUSIC". For full details refer to the *MUSIC/SP Communications Guide*.

Syntax:

```
   PING site-name #packets <-V -D>
```

Parameters:

-v (verbose) describes, in more detail, the operation of the program.

-d (dump) provides -v output as well as dumping extra information.

PIPE

This command provides access to pipe-lines.

Syntax:

```
PIPE xxxxxxxx
```

See the section in *Chapter 8 - Processors* or type "HELP" for more information.

PLAN

Invokes the Editor for updating your @PLAN file. This file is used if someone sends the FINGER command to your site about your userid.

Syntax:

```
PLAN
```

For more information, refer to the *MUSIC/SP Communications Guide* or type "HELP FINGER" in *Go mode.

POLYSOLVE

This command invokes the MUSIC/SP POLYSOLVE program for solving calculations and equations. For more information see *Chapter 10. Utilities*.

Syntax:

```
POLYSOLVE
POLYSO
```

POST

The POST command allows you to submit an article to one or more newsgroups. Fill in the appropriate fields on the screen. Help is provided once the POST command is used.

Syntax:

```
POST
```

PQ

This command is used to find out what is queued to print on various printers.

Syntax:

```
PQ
```

/PREVIOUS

This command is used to go to the previous MUSIC session on a 3270-type workstation. (Extra sessions are added with the ADD command.) By default the F7 key has this definition in command mode. To access the previous session while using a full screen program like the Editor, press the fullscreen escape (PA2) before the F7 key. See *Chapter 2. Workstations* under the heading "Multi-Session Support" for more information.

Syntax:

```
/PREVIOUS
/PREV
```

PRINT

This command schedules the printing of a file to a specified printer. (More information about PRINT is given in *Chapter 3 - Using Batch*.) The file name must be the first parameter. The other parameters are optional and may appear in any order. Carriage control is added to skip to a new page every sixty lines unless CC is specified or the file's record length is 121 or 133.

The PRINT command does not send your file for execution. It prints the contents of the file. The Editor also has a similar PRINT command.

Syntax:

```
PRINT filename [ROUTE(printername)] [FORMS(x)] [COPIES(n)] [CC  ]
                R                    F          C          [NOCC]

               [PAGELEN(m)]
                P
```

Parameters:

filename — The name of the file to be printed. Under the Editor, the special name **CUR* indicates the current contents of the file being edited, and the special name . indicates marked lines.

printername — The name of the printer where the file is to be printed. This may be an actual printer name or a printer location. It is 1 to 8 characters long. Some documentation refers to this as a "route name" or "routing name". The names are assigned by your system administrator.

If you do not specify a printer name, a default name is used. If you have used the command "ROUTE printername" previously in this MUSIC session, that name is the default. Otherwise, the name defined by ROUTE(name) in your User Profile (the PROFILE command) is used, if any. Otherwise, a default name based on your workstation location may be used If none of the above cases apply, the name SYSTEM is used.

The following names are valid:

SYSTEM — Sends the output to the standard system printer.

MUSIC — Sends the output to the MUSIC Output Queue (the OUTPUT Facility).

DUMMY — Discards the output. Nothing is printed.

rscsname — The name of an RSCS printer. For example: R(PRINTER3) means the output is sent to RSCS and queued for printing on linkid PRINTER3.

prtname — The name of a MUSIC-controlled ASCII or 3270 printer as defined by your installation. Consult your installation for a list of valid names.

PC1 — A printer name such as PC1 may be defined by your installation. It prints the file on your PC printer (using DOS device LPT1), provided your PC is connected to MUSIC via NET3270 or PCWS. If the connection does not support PC printing, the data is sent to the MUSIC Output Queue (the OUTPUT Facility). Similarly, PC2 uses device LPT2 and PC3 uses device LPT3. When printing to a PC printer, some PRINT parameters such as COPIES, FORMS and PAGELEN may be ignored.

n — The number of copies that should be printed. The default is 1 copy.

x — A one to 8 character string indicating which forms are to be used when printing the file.

m — The number of lines per page. The default is 60 lines. This parameter is used only when the NOCC parameter is in effect.

CC — If specified, it indicates that the file to be printed already contains a carriage control charac-

ter on the first character of each line in the file. The PRINT command attempts to honour these control characters.

NOCC Indicates that the file does not contain printer control characters. The file is printed single spaced, with a skip to a new page after each m lines of output. NOCC is the default, except when the record length of the file is 121 or 133, in which case CC is assumed.

Example:

```
*Go
print file1
*In Progress
15 records scheduled to print, route SYSTEM
*End
*Go
```

PROFILE

This command invokes the User Profile Program for changing such things as your sign-on password, default tab characters, job time limits, etc. For complete information see *Chapter 10. Utilities*.

Syntax:

```
PROFILE
PROFIL
```

PROG

This command is used to invoke the PROG (Programmer's Menu) subsystem. This facility allows you access to various components of MUSIC system through a series of selection menus. For details see *Chapter 1. Introduction*.

Syntax:

```
PROG
```

PURGE

This command permanently removes (deletes) one or more files. Files are protected by the first 4 characters of your userid. (You can only purge files that you own; files created by your userid.) The PURGE command does not work with files owned by another userid. The ERASE and DELETE commands (with a file name as a parameter) do the same as PURGE.

The parameters (1 or more) indicate which files to delete and any special options. The Editor also has a

similar command.

Syntax:

```
PURGE  filespec <filespec>... <-NOPROMPT><-NOLIST><-SUB>
PUR                           <-NOP     ><-NOL   ><    >
```

Parameters:

filespec Each filespec can be one of the following 3 types:

1. The name of a file. The special names /INPUT, /REC and /HOLD may also be used.

2. A file name preceded by the character <, for example, "PURGE <fileabc". Where fileabc is the name of a file that contains a list of file names to be deleted. Each name must start in column 1 and be followed by at least blank. Remaining characters in the line are ignored. Note that the LIBRARY command, with the SAVE(filename) option, can be used to create this list file.

3. A file name pattern (also called a generic file name). The purge command searches the current (or specified) directory for all file names which match the specified pattern and purges them. A pattern is similar to a normal file name, except that it contains one or more of the special wild characters ? and *. A ? in the pattern is considered to match any single character in the corresponding position of a file name. A * in the pattern matches any group of 0 or more characters in a file name. Some examples are shown below.

   Before the files are purged, the matching file names are displayed and you are prompted for permission to proceed. You may respond by typing PURGE (or PUR or PU) to allow the purges, or HELP. Any other response, such as a blank line or /CANCEL, cancels the request and the files are not deleted. The names are displayed in alphabetical order.

-NOPROMPT (abbreviation -NOP or -NOPR) Causes files to be purged immediately, without prompting, when a pattern is specified. Note: for a PURGE command in a batch job, -NOPROMPT is always assumed.

-NOLIST (abbreviation -NOL) Suppresses the message which is normally given for each file purged. However, a message is always displayed if a file cannot be purged (e.g. because it does not exist or is in use).

-SUB This option, used with a filespec that is a pattern, causes the search to include all sub-directories below the current (or specified) directory, in addition to the current (or specified) directory. It can be used to delete an entire subtree. Without this option, deletes are limited to the current (or specified) directory.

Examples:

purge myprog.obj	Deletes the single file MYPROG.OBJ.
pur fila filb filc	Purges 3 files.
purge <mylist	Purges the files whose names are in the file MYLIST.
purge abc??	Purges all files whose names are 5 characters long and start with ABC.

purge prog1.* prog1	Purges all files whose names start with PROG1. and also the file PROG1 itself.
purge *old*	Purges all files that contain the string OLD anywhere in the name.

QFTP

This command invokes the Quick FTP menu for transferring files. It is a fast method of using FTP when you know the names of the files you want to get.

Syntax:

```
QFTP
```

RD

This command removes directories.

Syntax:

```
RD dirname
```

The *dirname* parameter indicates which directory to remove.

Directories contain names of files that you want to keep together as a group. The MD command is used to make directories.

Notice that you cannot remove a directory that has any files in it. You can use the command DIR to show you the names of the files in the directory.

You cannot remove the name of a directory that is part of your current directory. For example if you are in the directory "\CAR\PARTS" then you cannot remove either the CAR or the PARTS directory.

Use the TREE command to visually show the names of your directories and how they relate to each other.

Examples:

```
*Go
md car

*Go
cd car

*Go \CAR>
cd \
```

```
*Go
rd car

*Go
```

/RECORD

This command is used to control writing to the session recording file. The name of this file is @REC (or @REC.sub where *sub* is the subcode for your userid). It can also be accessed via the name "/REC". When recording is set on, all workstation input and output associated with application programs is added to the file except for I/O associated with full screen programs such as the Editor. System messages resulting from STATUS commands and messages from other users or the system operators console are not recorded.

Recording can be turned on and off as required. Subsequent recorded information is appended to the end of the /REC file unless the NEW option is specified.

Each output record on the recording file starts with a carriage control character. Input records begin with a greater than sign > to distinguish them from the output records.

The recording file is subject to the user's file space limits. Exceeding these file limits causes the system to issue a message and turn off the recording without affecting the execution of the program.

Syntax:

```
RECORD  [ON ]
REC     [OFF]
        [NEW]
```

Parameters:

ON Turn recording on. If there is text in the file /REC from a previous recording session, then new information is appended to this file.

OFF Turn off recording (stop writing to the /REC file).

NEW Turn recording on. Any text from a previous recording session is deleted and a new /REC file is created.

Messages:

`RECORDING TERMINATED.`
This message appears if you run out of space.

Note: You can use the PRINT command with the CC option to print the recording file. For example,

```
PRINT /REC CC
```

RENAME

This command causes a file to be renamed to a specified new name.

Syntax:

```
RENAME   oldname newname [-REPL] [-SUB]
REN
```

You may only change the names of the files which you own (files created by your userid). Use * to specify file groups.

Parameters:

oldname — the name of the previously existing file to be renamed.

newname — the new name to be assigned to the file. This optional parameter, when used with a file name containing * (a name pattern), indicates that the matching names in the current (or specified) directory and all lower subdirectories, should be renamed. Without this option, the rename applies only to files in the current (or specified) directory.

-REPL — This optional parameter causes the target file (newfile) to be replaced, without any prompting, if it already exists. Without this option, you are prompted whether to replace the file; reply "y" or "yes" to replace it, "n" or "no" to not do the rename. You can also enter "yes all" (to replace the file and all further files for this command, without more prompting) or "no all" (to not rename the file, or any further files for this command which already exist, without more prompting). To cancel the command, enter "/cancel".

-SUB — The -SUB option should be used when renaming an entire directory, or an entire userid. See the examples below.

Examples:

```
ren work1 work2
```
Renames the file work1 to work2.

```
ren work* try*
```
Renames all file names that begin with "work" to the corresponding names beginning with "try". For example, WORK23 is renamed to TRY23, WORK.ABC to TRY.ABC, WORK to TRY, etc. Only files in the current directory are affected.

```
rename abc\* defg\* -sub
```
Renames subdirectory ABC to DEFG, including all files in lower subdirectories.

```
ren fred:* susan:* -sub
```
Renames all files in userid FRED to userid SUSAN. (This form of the command is used only by system administrators.)

/REQUEST

This command is used to send a one-line message to the console operator. As the operator may be busy with other activities, please be patient if the operator does not respond immediately. An installation may limit who can use this command. To send messages to other users, use the TELL command or the MAIL Facility.

Syntax:

```
/REQUEST    text of message...
/REQ
```

Example:

```
*Go
/request this is a sample message
*OK
```

RN

The RN command invokes the News Reader containing USENET information. USENET is a network available to users for exchanging messages and information of any kind. The news articles are grouped into categories called *newsgroups*. Newsgroups are devoted to a wide variety of subjects. These include politics, programming languages, science, recreational activities, and many, many more.

The RN command provides access to Network News services. Through this interface you can read incoming news from a variety of news groups, post your own news items, follow-up on existing items and send electronic mail directly to the news contributors.

Help is provided once the program is invoked. For more information refer to the *MUSIC/SP Communications Guide*.

Syntax:

```
RN [newsgroup]
```

Parameters:

newsgroup The name of the news group that you wish to see. If this is omitted a list of news groups is presented.

ROUTE

The ROUTE command displays or changes the default route destination for the current MUSIC session. By default, the route destination specified in the userid PROFILE is used automatically. If this does not exist then the system default is used. If the destination is omitted from the command, the current destination is

displayed.

Syntax:

```
ROUTE destination
```

SCHED

Invokes the TODO program called SCHED. It allows you to schedule your personal calendar, meeting room, or equipment. See the *MUSIC/SP Mail and Office Applications Guide* for details.

Syntax:

```
SCHED        or         TODO 1
```

SENDFILE

SENDFILE is used to send a copy of a file to another MUSIC user or CMS user on computers that are connected via RSCS (Remote Spooling Communications Subsystem). Unlike MAIL, SENDFILE sends the file free of mail headers and allows record lengths greater than 80. SENDFILE does not currently use the nicknames file (mail directory), nor does it support email domain names. Therefore, you must specify the exact userid and node (system name) of the user to whom you are sending the file. The MAIL program is used to receive send files.

Syntax:

```
SENDFILE filename (TO) userid (AT node)
SF
```

Parameters:

filename — is any MUSIC file. It can also be a file name pattern identical to that used with the library command.

userid — is either a MUSIC userid or a 1 to 8 character VM userid.

node — is the system name where *userid* is located. This is not a mail domain name, but the true system node name. If you are sending the file to a user on your system, you don't have to specify the *node*. A node name of "*" can and is used to represent the name of your system.

SENDFILEs you receive on your MUSIC system userid are deposited in your mail box. These are easily identified by the subject line which is in the form "Sendfile: filename". Such files are unaltered by the mail facility. You can then copy the file to any MUSIC file you desire.

Note: The TO and AT keywords are not required.

Examples:

1. This example sends the file "work" to userid ccfp at node mcgillm.

```
SF work to ccfp at mcgillm
```

2. This example sends all files on my userid that begin with the characters "work" to userid ccfp at node mcgillm.

```
SF work* to ccfp at mcgillm
```

SENDMAIL

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

Syntax:

```
SENDMAIL TO(user1) SUBJ(subject) FILE(filename)
```

The keywords available for use with this command are identical to those keywords used for the MAIL program SEND command. Here are some examples of SENDMAIL commands:

```
 SENDMAIL TO(BOSS) SUBJ(MEETING MONDAY) FILE(MEETING.1)
 SENDMAIL TO(JOE) CC(KATHY) SUBJ(VACATION) FILE(BERMUDA) NOACK
 SENDMAIL TO(JOE,KATHY) SUBJ(CONTRACT) FILE(LAW1)
 SENDMAIL TO(/GROUP) SUBJ(COURSES) FILE(MATH101)
or
 /INC *COM:SENDMAIL
 TO(BOSS) SUBJ(THIS IS A VERY LONG SUBJECT THAT GOES ON AND ON) -
 FILE(MEETING.1)
```

For more information type "/help sendmail".

SHOPAN

This command displays a panel file without changing it. Each accessible field is numbered and filled with a ruler.

Syntax:

```
SHOPAN filename
```

SHOWPFK

This command lists the program function key definitions for command mode. By default the definitions are as follows:

F1 HELP - invokes the help facility
F4 ADD - adds a new session
F5 DELETE - deletes an extra sessions
F7 PREVIOUS - goes to previous session
F8 NEXT - goes to next session
F12 RETRIEVE - echos text from the command area

Syntax:

```
SHOWPFK
```

/SKIP

This command may be used to skip over unwanted lines of output. This command is effective only when the workstation is in break (attention) mode. The ATTN or BREAK key on the workstation can be used to place the workstation in break mode. Use the PA2 key on a 3270-type workstation.

When a 3270-type workstation is in break mode (**Attn** in the lower right corner of the screen), pressing PA1 has the same effect as entering /SKIP ALL. Thus, to skip all remaining output when in MORE status, press PA1 twice (the first one causes break mode and the second one does the skip).

Syntax:

```
/SKIP    [n  ]
/SK      [ALL]
```

Parameters:

n is the number of lines of output to be skipped. If *n* is larger than the number of lines of output up to a conversational read or the end of the output, the system skips all lines up to that point. *n* must be positive. If *n* is not specified, no lines are skipped and output resumes. The value for *n* must be 9999 or less (i.e. 1 to 4 digits).

ALL all output lines up to the next conversational read or end-of-job are skipped. A call to system subroutine STOPSK will have no effect.

Example:

```
*Go
exec hello
*In progress
```

```
MUSIC

(Multi-User System for Interactive Computing)

(ATTN or BREAK key pressed)
/skip 4
from your workstation.(ATTN or BREAK key pressed)
/skip 100
*End
*Go
```

SORT

This command is used to sort the records in a file according to a single control field within each record. The system subroutine DSORT is used. The sorted data replaces the original file or a second specified file. The parameters on the command are separated by blanks or commas. (See the topic "Sorting" in *Chapter 10. Utilities* for other sorting programs and subroutines.)

The SORT command is equivalent to running the following job:

```
/PARM filnm1 etc.
/INCLUDE SORT
```

Units 3 and 4 are sort work files, 300K each, which are needed by the DSORT subroutine. The size of the work files may be increased if necessary, by adding overriding /FILE statements to the above job:

```
/FILE 3 NAME(&&TEMP) NEW DELETE SPACE(nnnn)
/FILE 4 NAME(&&TEMP) NEW DELETE SPACE(nnnn)
```

Execution mode, then command mode.

Syntax:

```
SORT filenm1 [filenm2][-REPLACE][-NOMSG][n-m][-A][-CH][-DELDUPS]
                      [-R      ][-NOM  ][n  ][-D][-BI]
                                             [-FI]
                                             [-FL]
                                             [-ZD]
                                             [-PD]
                                             [-DA]
                                             [-CI]
```

Parameters:

filenm1 The name of the file containing the data to be sorted.

filenm2 The name of the file where the sorted data is to be stored. If *filenm2* is omitted, the sorted data replaces the original file *(filenm1).* If the target file already exists, you are prompted for permission to replace it (unless the -REPLACE option is used). Answer Y or YES to replace the existing file. Any response not starting with "y" stops the program without storing the sorted data.

-REPLACE Causes the target file to be replaced without prompting, if it already exists. Prompting is never done in a batch job. Abbreviations: -R, -RE, -REP, -REPL, etc.

-NOMSG Suppresses information messages during the sort. Error messages are still put out. Abbreviation: -NOM

n-m Defines the starting column number (n) and ending column number (m) of the sort control field within each data record. This is the field on which the records are sorted. n must be from 1 to 4096. If the field is longer than 256 characters, only the first 256 characters are compared. The default sort control field is the entire record. When -*m* is not specified, *n* defines the starting column number of the sort control field, and it is assumed to extend to the end of the record (to a maximum of 256 characters).

-A Records are sorted into ascending (increasing) order. This is the default.

-D Records are sorted into descending order.

-xx Specifies the type of the sort control field. The default type is -CH (character). The possible types are:

- -CH Character.
- -BI Binary (same as -CH).
- -FI Fixed point.
- -FL Normalized floating point.
- -ZD Zoned decimal. This can be used to sort on a field containing a right-justified unsigned decimal number with leading blanks (e.g. as produced by Fortran I format).
- -PD Packed decimal.
- -DA specifies the sort field to be a 7-character date field of the form DDMMMYY (i.e. 01JAN90).
- -CI Case-ignore character field. This is similar to -CH, except upper and lower case characters are considered the same when comparing. For example, a field containing "Fred" is considered equal to a field containing "FRED".

-DELDUPS This option deletes output records that have exactly the same sort control field as the previous output record.

Examples:

```
sort master.file new.master

sort my.data -r -nomsg
```

The following commands produce a list of your files, in decreasing order by file size (columns 33 to 37 of the LIBRARY output):

```
library * f s
sort  @lib -r 33-37 -d -zd
list  @lib
*Go
```

```
sort @lib -r 33-37 -d -zd
*In progress
SORTING COLUMNS   33 THRU   37
SRT000  BEGIN SORT.  RECORD LENGTH = 80, AREA =64000
SRT000  RECORD COUNT =     421
SRT000  NORMAL END OF SORT
*End
*Go
```

Example using -DELDUPS option:

Input data (FILE1):
```
ABCX
ABCD
ABCE
AAAA
ABXX
ABXY
```

Command: **`sort file1 file2 1-3 -deldups`**

Sorted data:
```
AAAA
ABCX
ABCD
ABCE
ABXX
ABXY
```

Output data (FILE2):
```
AAAA
ABCX
ABXX
```

/STATUS

This command is used to get information about the user's workstation and the system. See the example below for details. (The service unit number quoted does not include the charge for I/O to the workstation and the surcharge for regions over 108K.)

Syntax:

```
/STATUS
/ST
```

Example:

```
*Go
status

1993/04/29       21:37
Userid         USER000
Tnum/Port       66/0A1
Connect time     01:38
Service Units       26
116 Users on MUSIC
```

SUBMIT

Submit a file or group of files to batch. A complete description of the SUBMIT command is given in *Chapter 3. Using Batch* of this guide.

Syntax:

```
SUBMIT filename1 [filename2] ... [filenamen] [parameters]
SUB
```

Example:

```
*Go
submit file1
*In Progress
15 RECORDS SUBMITTED.
*End
*Go
```

SUMMARY

This command can be used to produce a brief summary of the contents of a specified file.

All lines in the file which begin with a slash (/) are displayed. All lines which begin with a 12-2-9 punch (hexadecimal 02) are identified as object decks, and the total number of records in the file is displayed.

Syntax:

```
SUMMARY  filename
SUMRY
SUM
```

Parameters:

filename is the name of the file.

Example:

```
*Go
sumry hello
*In progress

LINE    1 /FILE 9 RDR
        2 /INC SCRIPT
       35 /CANCEL

       89 records counted.

*End
*Go
```

SYSDATE

This command displays the current date and system level.

Syntax:

```
SYSDATE
```

Example:

```
*Go
sysdate
*In progress
  FRI MAY 14, 1993     1993/134     18:25:44.19
  MUSIC IPL'd at  7:50 today  (14 May 1993)
  Nucleus level: V24 13MAY93
  MUSIC level:  Version 2   Release  4   Service  0   0
```

TAG

This command is used to display the tag information associated with a file or to assign new tag information to a file.

Syntax:

```
TAG  filename [ tag info ]
```

Parameters:

filename — the name of the file whose tag is to be displayed or assigned.

tag info — optional 64 bytes of tag information to be assigned to the file. If omitted, the file's existing tag is displayed.

Example:

```
*Go
tag work1 sample file to show the use of tag
*In progress
TAGGED
*End
*Go

tag work1
*In progress
TAG IS: SAMPLE FILE TO SHOW THE USE OF TAG
*End
*Go
```

TEDIT

This command is used to invoke the Editor program to edit a file. This command differs from the EDIT command only in that a TEXT SCRIPT Editor command is implied and tab characters are not translated to an appropriate number of blanks. This means that any lower case letters typed in are NOT be translated to upper case. For details about the Editor refer to *Chapter 7. Using the Editor* of this guide.

Syntax:

```
TEDIT [name] [NEW] [LRECL(n)] [fmt] [RO] [NOLOG] [UIO] [MAX(n)]
TED                [n       ]
```

Parameters:

See EDIT command for parameter descriptions.

TELL

This command is used to send a single line message to another user who is currently signed on to the system. The message text is immediately displayed on the receivers screen. If the user is not signed on or has suppressed messages (/MESSAGES OFF) nothing is sent. The MAIL Facility can be used to send longer messages and does not require the receiver to be signed on when the message is sent.

Intersystem tell messages are allowed if your site is connected to BITNET and your system supports this feature.

Syntax:

```
TELL userid [message-text]
TELL userid@systemid [message-text]
```

Parameters:

userid The userid of the person who is to receive the message. The message is sent to each active session using that userid.

userid@systemid
The BITNET address of the person who is to receive the message. The systemid must be a valid BITNET node name. Check with your installation to see if you are connected to BITNET and whether the intersystem tell is supported.

message-text The message test to be displayed on the screen. The case of the text is preserved. A prefix is added to the message indicating who the sender is.

Example:

```
*Go
tell ax01 Hi there, What about lunch.
*In progress
*End
*Go
```

TELNET

This command is used to invoke access other computers connected via TCP/IP to the Internet. A list of Internet addresses that allow anonymous access is supplied. For a brief description of TELNET see *Chapter 10 - Utilities* under "FTP and TELNET from MUSIC". For full details refer to the *MUSIC/SP Communications Guide*.

Syntax:

```
TELNET [internet address]
```

/TIME

This command may be used to determine the time of day and how much computer time has been used up to the present time in the job which is presently running. It is particularly useful if it is suspected that a program may be in a loop. The command may be entered after attention or break has been pressed to put the workstation in break mode. In this case the execution time will represent that used so far by the current job. If the command is entered from command mode the execution time is the total used by the last job to run (the ? abbreviation cannot be used in this case). The command cannot be used when the workstation is in reading mode.

The execution time is displayed in service units. If an asterisk (*) follows the time it indicates that your job was getting service at the instant the command was processed.

Use from break mode does not effect the execution of the program.

Syntax:

```
/TIME
/T
?
```

Example:

```
*Go
list sample
*In progress
      READ(9,*) A
      B=SQRT(A)
      WRITE(6,*) A,B
      CALL EXIT
      END
*Go
sample
*In progress
(ATTN or BREAK key pressed)

/time
21:54   Job time 0.2  SU
```

TODO

This command invokes the Time, Office, and Documentation Organizer (TODO) facility. This facility allows you to access various components of the MUSIC system through a selection menu. This menu consolidates the most frequently used programs for an office environment. For example, access to SCRIPT (word processing program), MAIL (electronic mail), SPELL (spell checking), etc. Help is provided once the facility is invoked. For complete information refer to the *MUSIC/SP Mail and Office Applications Guide*.

Syntax:

```
TODO [n]
```

TREE

Displays your directories graphically and allows you to select directories by tabbing to each name.

Syntax:

```
TREE
```

Help is available once the command is entered.

TUT

This command invokes the TUTORIAL facility for learning programming languages.

Syntax:

```
TUT
```

Help is available once the command is entered.

/USERS

This command is used to find out how many users are active (signed on the system) at the present time.

Syntax:

```
/USERS
/U
```

Example:

```
*Go
users

062 Users on MUSIC
```

VER

Displays the current MUSIC version and release numbers.

Syntax:

```
VER
```

VIEW

This command invokes the VIEW utility program for viewing files of any record length, type, or size on a full-screen workstation. Files can be displayed in hexadecimal. ASCII mode is available to view ASCII files on a 3270-type workstation. In this mode printable ASCII characters are translated to their EBCDIC equivalents.

For full information including command descriptions, refer to "VIEW" in *Chapter 10. Utilities.*

Syntax:

```
VIEW filename
```

See also EVIEW.

VM

The MUSIC/SP Passthru Facility allows you to create a VM session directly from your MUSIC session. The VM command creates the VM session and allows you to connect to CMS, TSO, VSE or another MUSIC machine. A simple escape sequence allows you to issue commands directly to the local session. The facility also supports file transfer between the local session and the remote session using the standard IBM 3270 file transfer protocol.

Creating a VM session

When the command VM is issued a VM session will be created. You must now enter the appropriate commands to connect to CMS or whatever application is required. You cannot dial back into the local MUSIC system.

MUSIC still controls your workstation. Messages from other users or the operators will still be displayed. Because of conflicts with CMS and TSO the full screen escape key (PA2) is disabled by the VM command, but multi-session commands can still be issued if prefixed by a percent sign (%).

Issuing Local Commands

The percent sign has been defined as an escape sequence that causes the command to be executed on the local system, rather than on the session provided by Passthru. The following commands are not passed to MUSIC but are processed directly by the VM program itself.

```
%RECEIVE      - Receive a file from a remote session
```

```
%SEND         - Send a file to a remote session
%END          - Terminate remote session.
%QUIT         - Terminate remote session.
```

Care must be taken when entering "local commands" (%RECEIVE, %SEND, %END, %QUIT). MUSIC looks for the local command at the start of the 3270 data stream, and expects a blank or no data to follow the command keyword. If other screen fields are modified, or marked by the system as modified (as in the VM logon screen), MUSIC may not recognize the local command. The command is recognized in most cases if you follow these rules:

1. Type the local command in the FIRST modifiable field.
2. Type a blank after the local command. (This blank is not always needed, but it is needed on the VM logon screen.)

File Transfer

The SEND and RECEIVE commands are used for file transfer. File transfer is only possible if the system running the remote session has the IBM 3270 File Transfer program installed. Usually this program is used to transfer files between a mainframe and a PC. The local session running VM command plays the role of the PC in this file transfer operation, however since it is running on an IBM mainframe no EBCDIC conversion is required. Both commands have the same format.

Syntax:

```
SEND    local-file remote-file [options]
SEND    >list-file
RECEIVE local-file remote-file [options]
RECEIVE >list-file
```

File Transfer Parameters

local-file	The name of the local file involved.
remote-file	The name of the remote file.
options	Optional parameters. If the remote session is on CMS these must be preceded by an open brace "(". The options are discussed below.
list-file	The file name of a local file containing a list of file transfer parameters. This is useful if a group of files is to be transferred at the same time.

Options

APPEND	Append to the file on the remote session.
ASCII	Convert file to ASCII. This is not useful in host/host file transfer.
CRLF	Use carriage return/line feed characters as record separators. This should always be specified if you want the record oriented characteristics of the file preserved. It should never be used when transferring binary data that may already contain such sequences.

There are other options that are dependent on which remote system you are sending to. Consult documentation on the IBM 3270 File Transfer Program for details.

File Transfer Notes

The standard procedure for the 3270 File Transfer program is to replace existing files without prompting the user. This is also true in this implementation.

To transfer binary data that also has logical records (object modules), pre-allocate a file of the appropriate record length on the target system and then do the file transfer. Do not specify the CRLF option.

The current implementation supports record lengths of up to 2048 bytes.

WEB

This commands invokes the Web line-mode browser for accessing Internet Web sites. Help is available once the program is invoked.

Syntax:

```
WEB {url address}
```

For example,

```
WEB http://musicm.mcgill.ca
```

WHOAMI

This command displays userid information and TCB number for the current session.

Syntax:

```
WHOAMI
```

Example:

```
*Go
whoami
*In progress
You are userid CCGW000 on virtual machine MUSIC
Session id (TCB number) is 17
File ownership id is CCGW
Subcode is 000
```

/WINDOW

This command instructs MUSIC to display only certain portions of each line directed to the workstation. For example, "/WINDOW 5,50" displays only the portion of each line that normally displayed in columns 5 through 50. The characters are displayed starting in the first position of the output line.

This command is particularly useful when displaying long lines of output on an IBM 3270 terminal or other workstation which has a short line length.

The effect of this command is removed when the workstation next enters command (*Go) mode.

If the column numbers are not respecified, they will default to the ones last given.

Syntax:

```
/WINDOW  [m,n] [,OFF ]
/WI
```

Parameters:

m,n — Specifies the start and ending column numbers of the window. The first output position is called number 1. (Normally when your program displays on the workstation, the first character position is used for a carriage control character that is not displayed and it is therefore not counted as output position 1.)

OFF — Turns off the window feature.

Example:

```
*Go
list sample
*In progress
1234567890ABCDEF
*End
*Go
window 10,12
*OK
list sample
*In progress
 0AB
*End
*Go
```

XTMUS

Transfers PC files to MUSIC. You must be using PCWS on your PC to connect to MUSIC. For complete details, refer to the *MUSIC/SP Personal Computer WorkStation User's Guide*, or type "HELP XTMUS".

Syntax:

```
XTMUS src [dest] [-BIN] [-REPL] [-APP] [-COMP] [-Fx] [-n]
                 [-B  ] [-R   ] [-A  ] [-C   ]

                 [-MON] [-SWITCH] [-NOMSGS]
                 [-M  ] [-S     ] [-N     ]
```

XTPC

Transfers MUSIC files to the PC. You must be using PCWS on your PC to connect to MUSIC. For complete details, refer to the *MUSIC/SP Personal Computer WorkStation User's Guide*, or type "HELP XTPC".

Syntax:

```
XTPC src [dest] [-BIN] [-REPL] [-APP] [-COMP]
                [-B  ] [-R   ] [-A  ] [-C   ]

                [-MON] [-SWITCH] [-NOMSGS]
                [-M  ] [-S     ] [-N     ]
```

ZEROCNT

The ZEROCNT command resets a file's usage count to 0 by archiving it to a temporary file, then restoring it. The file's dates are also reset.

Syntax:

```
ZEROCNT  {filename} {pattern} {<listname}  *
```

Parameters:

	filename is the name of the file.
pattern	is the file specification including wild characters * and/or ?. Note that a pattern only applies to files in the current directory.
<listname	is the name of the file containing a list of file names.
*	give the count for all your files that have the CNT attribute.

Chapter 6. MUSIC Job Control Statements

Job Control Statements - Overview

Job control statements are used to control program execution. They define the environment in terms of time and storage limits, define the files used, define the type of process to be run and set optional parameters. The job control statements are processed only when the file containing them is executed. Errors in statement syntax will only be noted at this time. The system processes all control statements up to and including the /LOAD statement. After that the process that is loaded takes over control of reading the input from the file. For this reason statements that apply to the system (/SYS or /FILE) must come before the /LOAD and those that set options for the loaded process (/OPT or /JOB) must come after the /LOAD. The following outlines the order of statements in a typical job.

```
/PARM
/SYS
/FILE
/LOAD
/JOB
/OPT
. . . . .      (Source Language if applicable)
/DATA
. . . . .      (Data records if applicable)
```

Some statements such as /INCLUDE, which includes statements from another file, can appear anywhere in the sequence. Others such as /INFO and /PASSWORD must appear first if used.

All job control statements begin with a slash (/), followed by the statement verb. One or more blanks separate the statement verb from variable parameters (which must begin before column 16). Variable parameters are separated by blanks or commas.

Conventions

The following conventions are used in this chapter to describe Job Control statements:

1. Upper case letters and punctuation marks represent information that must be coded exactly as shown.

2. Lower case letters and words are generic terms representing information that must be supplied. That is, a substitution must be made when coding a parameter or option so represented.

3. Information within square brackets [ ] represents an option or choice of options that may be included or omitted, depending on requirements.

4. In the examples, information typed by the computer is shown in upper case, and information typed by the user is shown in lower case.

5. Underlined parameters are the default values.

6. The only statement abbreviation allowed from batch is /INC for /INCLUDE.

The remainder of this chapter gives a description of all Job Control Statements.

/COM

This statement is used in a file to indicate a comment. These statements are ignored as long as they appear before a /LOAD statement.

Syntax:

```
/COM text
```

Note: If /COM is used between a /FILE statement and a /ETC statement, the /ETC statement is also ignored.

/DATA

This statement is used as an indication that all lines following are data lines to be processed during program execution, and are not to be processed by a compiler or loader program. It is not required unless data lines are to be read during program execution from the file.

Syntax:

```
/DATA
```

Example:

```
*Go
list dat123
*In progress
25.
36.
49.85
*End
*Go
```

```
list sample
*In progress
     1 READ(5,2,END=10)A
     2 FORMAT(F10.2)
       B=SQRT(A)
       WRITE(6,3)A,B
     3 FORMAT(' THE SQUARE ROOT OF',F6.2,' IS',F5.2)
       GO TO 1
    10 STOP
       END
/DATA
/INCLUDE DAT123
*End
*Go
sample
*In progress
MAIN   = 0001A4
003658 BYTES USED
EXECUTION BEGINS     1.4S
THE SQUARE ROOT OF 25.00 IS 5.00
THE SQUARE ROOT OF 36.00 IS 6.00
THE SQUARE ROOT OF 49.85 IS 7.06
STOP         0
*End
*Go
```

/END

This statement is used as a delimiter to indicate the end of a batch job.

Syntax:

```
/END
```

/ETC

This statement is used as a continuation line for the /ID and the /FILE lines. See the /FILE statement writeup below for its usage with that statement.

The /ETC statement can be used as a continuation line for the batch /ID statement. In this case, it is used solely for comments to the operator in regards to where or how to return your printed output. See *Chapter 3. Using Batch* for more details.

Syntax:

```
/ETC  [information]
```

/FILE (General Overview)

Programs usually refer to files and input/output (I/O) devices through *ddnames* (data definition names) or unit numbers. The /FILE statement provides the connection between the program and the actual file by relating the symbolic ddname or unit number to the physical file or device. /FILE statements are processed at execution time, and may be changed from one run of the program to another. In this way a single program can be used to process data from a variety of sources without having the change the program.

Suppose, for example, a program reads from unit 8 and writes to unit 9. It can perform a number of different functions depending on how units 8 and 9 are defined through /FILE statements. If they are both defined as files, the program is copying one file to another. If unit 8 is defined as a file and unit 9 is a printer, the program is printing a file.

The /FILE statement gives the name of a file and some of its characteristics. The program communicates with this file through a logical unit number or data definition name (ddname). (Fortran programs can use logical unit numbers, COBOL programs can only use ddnames.) When the user program does I/O through the logical unit number or ddname, the actual read and write action will be done to the corresponding I/O device or file.

MUSIC allows a user to define a logical unit number from 1 to 15 inclusive. A single /FILE statement cannot refer to two or more I/O devices or files. No duplicate logical unit number or ddname can be defined within one job. The type of I/O devices or files that can be defined will be detailed later as they are discussed separately under three major categories: files, UDS files, and miscellaneous. Additional parameters such as the record size, blocksize, record format, etc. of the file being defined will also be discussed in detail under each major category.

All jobs have the following default logical unit assignments:

```
/FILE  5  RDR
/FILE  6  PRT
/FILE  7  PUN
/FILE  9  TERM
/FILE 10  HOLD
```

The meaning of the above /FILE statements will be discussed later under the 'miscellaneous' category.

Syntax:

General Description:

```
/FILE  u    type  ...
/FILE  ddn  type  ...
```

The syntax of the /FILE statement requires the logical unit number (u) or the ddname (ddn) to be the first parameter defined and separated by at least one blank from the statement word /FILE. The second parameter that must be defined is the type of I/O device or file. Additional parameters can then be defined in any order. Parameters must be separated by one or more blanks or commas. Blanks will be used to separate parameters in the following discussion. Each parameter is specified in the form of KEYWORD or KEYWORD(SUBPARAMETER), for example, RDR, SPACE(10).

The /FILE statements are placed at the beginning of a job and must occur before any /LOAD statements or they will not be processed. The /FILE control line cannot exceed 80 characters in length. If more space is

required, the /FILE line may be continued on a /ETC control line. The parameters may be split between a /FILE and a /ETC line at any place where a blank or a comma is valid, except that a subparameter must not be split between lines.

A single job is limited to a maximum of 4 UDS-type files (including tape files). Some processors (for example PL/I) allow a maximum of only 3 user-defined UDS files. Up to 14 files can be in use at any one time. Files defined via the /FILE statement are considered in use during the entire job. Files referenced by /INCLUDE statements or opened dynamically during the execution of a job count against this limit only when they are in use (i.e. opened).

/FILE (Files)

Files can be created and/or deleted in one job. They could be created in one job and used in another at some later time. In order to be able to refer to the same file again later, the user must give a name to a newly created file. Besides the name, the user can also specify a number of items such as the record length, record format, or the attributes of the file. These items are assigned to a file at the time it is created and cannot be modified afterwards.

Each userid contains a limit to the amount of space that can be allocated to any one file. This is done to guard against abusive use of disk space and in fairness to the other users who may also want to use some of the same space. Should the space limit pose any difficulties, the user should go to the MUSIC Systems Administrator to request a bigger space limit.

A charge is normally made for the space that you allocate for your Save Library. This charge is based on the amount of space occupied and the length of time the file exists.

MUSIC features a facility to automatically preserve the integrity of your files by allowing them to be private, public *execute-only*, or public *read/write*. Furthermore, the system automatically performs an enqueue operation to ensure that no two jobs are accessing the file in a manner that may affect each other. The various types of enqueue protection will be detailed under the *disp* parameter discussed below. The message `SAVE FILE IS IN USE...TRY AGAIN LATER` will be issued if the enqueue detects disallowed multiple use of the file.

Syntax:

```
Files

/FILE  u    Name(fn1) [disp] [RLSE  ] [LRecl(l)] [RECFm(fm)]
       ddn   PDS(fn2)        [NORLSE] [RSIZe(l)]

             [SPace(ps)] [SECsp(ss)] [MAXSP(ms)] [attrib] [NRec(n)]
```

Parameters:

u — specifies the logical unit number by which the file will be referenced. It must be a number from 1 to 15 inclusive. No duplicate logical unit number can be defined in one job.

ddn — specifies the step name and data definition name (ddname) by which the file will be referenced. If only a ddname is specified, the ddname will be applied to both the COMPILE step and the GO step. If both the step name and ddname are specified, they must be specified in

the form *stepname.ddname*. The valid stepnames for the COMPILE step are COMP, FORT, PLI, COB, and ASM. The only valid stepname for the GO step is GO. For example, FORT.SYSIN, GO.DATA, FILE1. No duplicate ddname can be defined in one job.

Name(fn1) specifies the name of the file involved. Refer to *Chapter 4. File System and I/O Interface* for the naming convention of these files. If NAME(&&TEMP) is specified, the file created will be a temporary one which means that it will be discarded when the job is finished. The parameters NEW DELETE are automatically assumed when NAME(&&TEMP) is used.

PDS(fn2) specifies the name of a group of files. VS Assembler uses this parameter to define macro libraries. COBOL (COPY verb) and PL/I (%INCLUDE statement) also use this parameter to incorporate source statements. A ddname, and not a logical unit number, must be specified if this parameter is used. *fn2* is specified in the form of, for example, ABC.*.X. If a member name, for example, M1 is referenced by the specified ddname in the user program, the resulting file involved will be ABC.M1.X. Furthermore, the PDS parameter allows the specification of multiple groups. For example, if PDS(ABC.*.X,MN.*.YZ) is specified, the file ABC.M1.X would be searched for first and if not found, the file MN.M1.YZ will then be tried.

disp specifies the disposition of the file. It indicates the current status of the file and what MUSIC will do to it after the job ends. Items that can be specified are:

NEW indicates that the current status of the file is a new one and space will be allocated for it. The user can write to the file. An error message will be issued if the named file actually does exist in the Save Library. However, if NEW(REPLACE) is specified, the existing file will be purged (deleted) from the Save Library and a new file will be created under the same name. REPLACE can be abbreviated as REP or REPL. Only the current job is allowed to access the file while it is running.

OLD indicates that the current status of the file is old, meaning that it already exists in the Save Library. The user can write to the file. An error message will be issued if the named file actually does not exist. However, if OLD(CREATE) is specified, a new file will be created if the named file does not exist. CREATE can be abbreviated as CR. Only the current job is allowed to access the file while it is running.

SHR has two meanings. For new files, it means that the file is to be saved so that other users can refer to it by prefixing the creator's userid in front of the file name. For existing (old) files, it indicates that more than one job can access the file concurrently. This also prevents any of the jobs from writing to the file while this job is running. For existing files, SHR is assumed if no disposition is specified.

WSHR indicates that the current status of the file is old and that more than one job can write to it at a time. This feature should be used with caution. Discipline should be provided in the programs to prevent one from destructively interfering with another. While this job is running, other users are allowed, if the file is a public one, to access the file as SHR but not as OLD.

APPend indicates that the current status of the file is old and that when output is written to the file, it will be written starting at the end of the file.

APPONLY (append only) indicates that output will be written to the end of the file, and this is the only activity allowed. No reads may be done. A current status of OLD is assumed. SHR and WSHR may not be used.

KEEP — indicates that when the job is finished, the file is to be kept. KEEP is always assumed unless DELETE is specified or the file name is &&TEMP.

DELETE — indicates that when the job is finished, the file is to be deleted (removed) permanently from the Save Library. The parameter OLD is always assumed when DELETE is used.

DEFault — causes the /FILE statement to have no effect if some previous /FILE statement has been given for the same logical unit number or ddname. This feature is useful when setting up generalized programs particularly when they are to be used by other users who may want to use their own /FILE statements in place of some default ones you provide.

Some typical disposition combinations are shown below:

```
NEW
NEW(REPLACE)
OLD
OLD(CREATE)
NEW DELETE
OLD DELETE
NEW DELETE DEFAULT
NEW DEFAULT
```

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. This is the default unless SHR is specified.

NORLSE — specifies that any unused space at the end of the file is not to be released.

LRecl(l) — specifies the length of each record of the file. The maximum record length is 32760 bytes. The minimum record length is 1 byte. If omitted, a record length of 80 bytes is assumed. This parameter is only used when the file is being created. For FORTRAN sequential I/O, the maximum logical record length is 133 (unless the system subroutine BIGBUF is called).

RSIZe(l) — same as LRECL.

RECFm(fm) — specifies the record format of a file. The valid record formats are F (fixed), FC (fixed compressed), V (variable), VC (variable compressed), and U (undefined). Refer to *Chapter 4. File System and I/O Interface* for an explanation of the various record formats. The default is FC. This parameter is only used when the file is being created.

SPace(ps) — specifies the primary space allocation of the file. ps is specified in units of K (1024) bytes. The *K* should be omitted from the parameter. The default is 32K bytes. This parameter is only used when the file is being created.

SECsp(ss) — specifies the secondary space allocation of the file. ss can be specified in units of K bytes or in units of percentage of the current space allocation. For example, SECSP(10) means that the secondary space allocation is 10K bytes, and SECSP(30%) means that it is 30% of the current space allocation. The default is 50% of the current space allocation. This parameter is only used when the file is being created.

MAXSP(ms) — specifies the maximum space limit of the file in units of K bytes. The default value is the maximum allowable space that can be allocated to the user for each file. This parameter is only used when the file is being created.

attrib — specifies the attributes of the file and is used when the file is being created. The valid attributes are shown below. Some users may be restricted from specifying the SHR or PUBL option.

- PRIV — indicates that the file is *private.* This means that it can only be used by the *owner* (the user who signed on to MUSIC with the particular userid which was used to create the file). This is the default attribute.
- PUBL — indicates that the file is *public.* This means that the file is in the common library and that any user can access it. However, only the *owner* can write to or delete the file.
- SHR — indicates that the file can be read by other users but that its name is not to be stored in the common library. Other users must refer to this file by prefixing the owner's userid in front of the file name as in the example userid:PROG. Only the *owner* can write to or delete the file.
- COM — indicates that the file is *private* and it is stored in the common library index.
- XO — indicates the file has the *execute-only* attribute. A file of this type cannot be listed, displayed, or inserted. For complete details refer to the XO parameter of the MUSIC command /SAVE in *Appendix A. /Input Mode*.

Some typical attribute combinations are shown below:

```
PRIV
PUBL
COM
PUBL XO
```

NRec(n) — specifies the maximum number of records in the file. If the parameter SPACE is not used, the number of records, in conjunction with the specified record length (RSIZE or LRECL), is used to determine the approximate amount of space allocated to the file, by multiplying the two numbers. However, if the SPACE parameter is used, the amount of space allocated will be solely determined by the SPACE parameter and NREC will be ignored. This parameter is only used when the file is being created.

Examples:

```
/FILE  3  N(MYFILE)
/FILE  GO.OUTFIL  NAME(PGM1.OUT)  APPEND
/FILE  FT01F001  N(DATA.ONE)  NEW  SP(10)
/ETC   SEC(5)  RLSE  PUBL  LRECL(80)
/FILE  GO.DATA1  PDS(DATA.*.X)
```

```
*Go
list sample
*In progress
/FILE 10 NAME(XXX) OLD
      DO 50 I=1,100
      J=I**2
      WRITE(10,30)I,J
30    FORMAT(I3,3X,I5)
50    CONTINUE
      .
      .
      .
      ETC.
```

Note: For FORTRAN programs and others using the FORTRAN/MUSIC interface (COBOL, PL/I, and ASM programs do not use this interface), the default maximum record size is 133 bytes for sequential I/O. It can be increased by a call to the subroutine BIGBUF. If FORTRAN unformatted I/O is used, allowance for a four-byte control word must be included.

/FILE (Temporary UDS)

Temporary Disk UDS files are those used during a single job and then discarded. The total number of temporary and permanent UDS files defined in one job cannot exceed four. In order to fairly share the common pool of temporary disk space, MUSIC sets a limit to the total amount of temporary disk space any one job can use at any one time through all the user's /FILE statements. This limit is roughly equivalent to the space required to hold 48,000 80-byte or 32,000 128-byte records. If this proves a limitation, then you can use a permanent UDS file and specify the disposition NEW DELETE on it to make it behave like a temporary UDS file.

Some compilers and loaders also require some of this temporary space in order to handle your job. Generally they will try to use what is left over after your /FILE statement definitions, though this may not be sufficient in some unusual conditions.

Syntax:

Temporary Disk User Data Set (UDS) Files

```
/FILE u    UDS(&&TEMP) [NRec(n)] [LRecl(l)]
      ddn                        [RSIZe(l)]
```

Parameters:

u — specifies the logical unit number by which the temporary UDS file will be referenced. It must be a number from 1 to 15 inclusive. No duplicate logical unit number can be defined in one job.

ddn — specifies the step name and data definition name (ddname) by which the temporary UDS file will be referenced. Refer to the writeup of the /FILE statement for the files for complete details.

UDS(&&TEMP)

specifies that the file involved is a temporary UDS file.

NRec(n) specifies the maximum number of records that you will use in the file. If this parameter is omitted, a number of records equivalent to 800 128-byte records will be used.

LRecl(l) specifies the length of each record of the temporary UDS file. This can be any number of bytes between 20 and 512 inclusive. If omitted, a record length of 128 bytes is assumed. For FORTRAN sequential I/O, the maximum logical record length is 133 (unless the system subroutine BIGBUF is called).

RSIZe(l) same as LRECL.

Examples:

```
/FILE  5  UDS(&&TEMP)  NREC(500)
/FILE  XFILE  UDS(&&TEMP)  NREC(1000)  RSIZ(80)
/FILE  15  UDS(&&TEMP)  LRECL(100)
```

Notes:

1. For FORTRAN programs and others using the FORTRAN/MUSIC interface (COBOL, PL/I, and ASM programs do not use this interface), the following comments apply:

   The default maximum record size is 133 bytes for sequential I/O. (Sequential I/O is the normal access technique.) It can be increased by a call to the subroutine BIGBUF. If FORTRAN unformatted I/O is used, allowance for a four-byte control word must be included. If the data set is to be used for FORTRAN direct access, the DEFINE FILE statement in the FORTRAN program may specify any record size from 1 to 512 bytes. In this case the value of LRECL is used only to calculate how much disk space must be allocated.

   This also applies to the permanent UDS files.

2. There is another form of the /FILE statement which can be used to define temporary UDS files. The syntax of this form is:

   /FILE DISK=(u,NREC=n),RSIZ=r

   This form is used by the older versions of MUSIC. Users are NOT encouraged to use this form of the /FILE statement as it is not as powerful as the new form and will not be supported at some later time. Refer to the User's Guide of older versions of MUSIC for a description of the parameters used.

/FILE (Permanent UDS)

Permanent disk UDS files are those that can be created in one job and used in another at some later time. In order to be able to identify the same file again later, you give the file a name called the data set name (dsn). You also choose the disk volume that is to hold your file. This allows you to separate your UDS files on different volumes to optimize job time. MUSIC1 is a valid disk volume name. Check with your installation about other disk volume names available to you.

The parameters NREC and RSIZE (or LRECL) are only given when you create the file. When you use the file later on, the same numbers will apply as when the file was created.

Each userid contains a limit to the amount of space that can be allocated to any one file. This is done in fairness to the other users who may also want to use some of the same space. The amount of space that you may

allocate can be determined by running the MUSIC PROFIL utility program. The MUSIC Systems Administrator can allocate specific larger files for you if you have this requirement.

A charge is normally made for the space that you allocate for your permanent UDS files. This charge is based on the amount of space used and the length of time the file exists.

MUSIC features a facility to automatically preserve the integrity of your UDS files by allowing them to be private, public *read-only*, or public *read/write*. Additionally, the system automatically performs an *enqueue* operation to ensure that no two jobs are accessing the file in a manner that may affect each other. The various types of enqueue protection are detailed under the *disp* parameter discussed below. The message `DATA SET IN USE...TRY AGAIN LATER` will be issued if the enqueue detects disallowed multiple use of the file.

Syntax:

```
Permanent Disk User Data Set (UDS) Files

/FILE u   UDS(dsn) [NRec(n)] [LRecl(l)] VOLume(v) [disp]
      ddn                    [RSIZe(l)]
```

Parameters:

u — specifies the logical unit number by which the permanent UDS file will be referenced. It must be a number from 1 to 15 inclusive. No duplicate logical unit number can be defined in one job.

ddn — specifies the step name and data definition name (ddname) by which the permanent UDS file will be referenced. Refer to the writeup of the /FILE statement for the files for complete details.

UDS(dsn) — specifies the data set name. The dsname field can be up to 22 characters in length. A data set naming convention is established for MUSIC whereby the owner of the file can be identified. There are two types of dsnames:
(1) A dot (.) is used to separate the ownership id from the name. The dsname can be up to 22 characters long. For example: "GEORGEW.TEMP1".
(2) When there is no dot (.) in the dsname then the first 4 characters of the dsname must match your userid. The maximum length of the dsname is 8 characters. For example: "CCGWTMP1". Type 2 is available for compatibility with older versions of MUSIC.

Access to a UDS is governed by the use of an indicator character included in the dsname. This character is in the 1st position after the dot (type 1) or in the 5th position (type 2). The indicator may be one of the following:

$ for a private data set: can be read or written only by the owner. That is, the user code used at sign-on must match the user code in the data set name.

for a writable public access data set: can be written by anyone.

any other letter or numeric digit for a public *read-only* data set: can be read by anyone but can only be modified or deleted by the owner.

NRec(n) — specifies the number of records in the file. This parameter is only used on the /FILE statement when the file is being created. Refer to the writeup about temporary /FILE

statements for more information about this parameter.

LRecl(l) specifies the record length for files that are being created. It must be between 20 and 512 inclusive. If omitted, the record size is assumed to be 128. This parameter cannot be used to change the record size of an existing file. Refer to the LRECL description under the writeup for temporary UDS files for more information about this parameter. For FORTRAN sequential I/O, the maximum logical record length is 133 (unless the system subroutine BIGBUF is called).

RSIZe(l) same as LRECL.

VOLume(v) specifies the disk pack volume name for this data set. MUSIC1 is a valid volume name. Check with your installation about other volume names available to you.

Normally the volume name refers to a disk pack that is already mounted and accessible at the central computer site. Jobs run from batch can request that special volumes be mounted for the duration of the job. If the volume name given by this parameter is not currently mounted, then MUSIC will assume that you are requesting the mount of a special volume. Your MUSIC Systems Administrator may be of assistance in setting up such a volume. You may be required to specify the general type of disk pack that is to be mounted. This is done through the choice of one of the following numbers: 2311, 2314, 3330, 3340, 3350, 2305, 2305.1, 2305.2. This number is specified as a parameter in the /FILE statement. The form of the parameter is DEVice(d) where d is one of the numbers mentioned above.

disp specifies the disposition of the UDS file. It indicates the current status of the file and what MUSIC will do to it after the job ends. Items that can be specified are the same as those in the /FILE statement for the files with the following exceptions:

1. The REPLACE option of the disposition NEW, i.e. NEW(REPLACE), is not allowed.
2. The CREATE option of the disposition OLD, ie. OLD(CREATE), is not allowed.
3. The dispositions APPEND and APPONLY are not allowed.
4. For new UDS files, you can request that you want this file to be flagged as requiring backup. This is specified by the parameter BACKup. The opposite is NOBACKup which is assumed by the system if BACKUP is not specified.

Refer to the writeup of the /FILE statement for the files for a description of each item.

Some typical disposition combinations are shown below:

SHR
NEW
OLD
NEW DELETE
DELETE
NEW DELETE DEFAULT
NEW BACKUP

Examples:

```
/FILE    12  UDS(xxxxABCD)  VOL(MUSIC2)  OLD
/FILE  DATA  UDS(xxxx$PRV)  VOL(MUSIC2)  SHR
/FILE     2  UDS(xxxx#PUB)  VOL(MUSIC3)  NREC(1000)
/ETC  RSIZ(80)  NEW  BACKUP

*Go
list sample
*In progress
/FILE  1  NAME(ABCD)  NREC(35200)  RSIZ(32)
/FILE  2  UDS(USERXXXX)  VOL(MUSIC2)  OLD
       DIMENSION TABLE(8)
       READ(2,10)TABLE
10     FORMAT(8A4)
       WRITE(1,10)TABLE
       .
       .
       .
       ETC.
```

Notes:

1. If your job abnormally terminates, it is possible that the system will disregard the KEEP attribute for a NEW file. (This will happen ONLY for NEW files.) If the entire MUSIC system should fail, before your job finishes, it is possible that a file that you had asked to be deleted at the end of your job will not be deleted. To avoid these cases, it is advisable to create and delete your UDS files separately from any long running program that will use them. The following are suggested techniques for performing these creation and deletion operations:

   Creation of UDS Files:

   ```
   /FILE 1 UDS(...) NEW ...
   /INCLUDE UTIL
   $INFO
   ```

   Deletion of UDS Files:

   ```
   /FILE 1 UDS(...) DELETE
   /LOAD IEFBR
   ```

2. There is another form of the /FILE statement which can be used to define permanent UDS files. The syntax of this form is:

   ```
   /FILE DISK=(u,dsname,NREC=n),RSIZ=r,VOL=v,DISP=d
   ```

   This form is used by the older versions of MUSIC. Users are NOT encouraged to use this form of the /FILE statement as it is not as powerful as the new form and will not be supported at some later time. Refer to the User's Guide of older versions of MUSIC for a description of the parameters used.

/FILE (Tape UDS)

UDS files on magnetic tape are handled in a similar way to those for disk UDS files. Magnetic tape units come in two different versions: 9 and 7 track and each type can record at different densities.

MUSIC automatically handles the blocking of records for disk data sets but for tape you must specify how your records are to be blocked. This allows for compatibility with other operating systems. Normally you would want to use a large blocksize to reduce the amount of tape used as well as reduce the processing time. The block size (BLKSIZE) should be a multiple of the record size (RSIZE). For example, BLKSIZE(800) and RSIZE(80) is permissible.

Magnetic tapes have a special circular ring known as a *write-enable ring*. When this ring is removed from the tape, then no write operations will be allowed on the tape. The ring can be replaced at a later time, should you wish to write on the tape again. MUSIC features an additional write protection through the use of the disposition SHR. In order to write on a tape from MUSIC both the write-enable ring must be in place on the tape and the disposition SHR must NOT be specified.

Some operating systems put special *label* records on the tape. MUSIC does not. Label records, if present, will look to MUSIC just like a file written by a user program. You may skip over the label records if you want by first reading until you get an end-of-file indication. This EOF indication would mean that you are now beyond the label records and that you can start reading the data file that follows. (Some installations use MUSIC to read the label records created by other operating systems when the contents of these records are in doubt.)

Syntax:

Magnetic Tape User Data Set (UDS) Files

```
/FILE u    TAPE [BLksize(s)  ] [LRecl(l)] VOLume(v) [disp]
     ddn        [BLocksize(s)] [RSIZe(l)]

                [DENsity(d)] [9TRk  ] [TRTCH(t)]
                             [9TRack]
                             [7TRk  ]
                             [7TRack]
```

Parameters:

u — specifies the logical unit number by which the tape UDS file will be referenced. It must be a number from 1 to 15 inclusive. No duplicate logical unit number can be defined in one job.

ddn — specifies the step name and data definition name (ddname) by which the tape UDS file will be referenced. Refer to the writeup of the /FILE statement for files for complete details.

TAPE — specifies that the file involved is a tape UDS file.

BLksize(s) — specifies the length of the physical block on tape. It is limited only by the buffer space available. The blocksize should be a multiple of the record size. If not specified, the record size is assumed.

BLocksize(s) — same as BLKSIZE.

LRecl(l) — specifies the size of each record. It must be a value between 20 and 32760. The default record size is 128 bytes. For FORTRAN sequential I/O, the maximum logical record length is 133 (unless the system subroutine BIGBUF is called).

RSIZe(l) — same as LRECL.

VOLume(v) specifies the name of the magnetic tape reel. The name can be from 1 to 6 characters in length and any letter (A-Z) and number (0-9) may be used.

disp specifies the disposition of the tape UDS file. If not specified the system will assume OLD which allows the writing operation on the tape, subject to the write-enable ring on the tape reel. SHR may be specified to allow read operation only.

DENsity(d) specifies the density of the tape. For 7-track magnetic tapes, the valid densities that can be specified are 200, 556 and 800 bits per inch (BPI). For 9-track magnetic tapes, the valid ones are 800, 1600 and 6250 BPI. If a 7-track tape is specified, by specifying 7TRK or 7TRACK, the density will be assumed as 800 BPI if it is not specified. For a 9-track tape (9TRK or 9TRACK), MUSIC will assume a density of 1600 BPI if the density is not specified. You can also specify DEN(LOW) or DEN(HIGH) to use the lowest or highest density available for the tape drive. Note that you do not need to specify the density if you are only reading from the tape (not writing).

9TRack
7TRk
7TRack
9TRk
Specifies whether the tape is to be mounted on a 7 or 9 track tape drive. Your installation may not have both types available for your use. See your installation about the types of tape drives available. MUSIC assumes 9TRK (9TRACK) if the density is not specified or is specified as 800, 1600 or 6250 BPI. 7TRK (7TRACK) will be assumed if the specified density is either 200 or 556 BPI, or if the parameter TRTCH is specified.

TRTCH(t) specifies the tape recording technique. This is valid only for 7-track tapes. The following is a table of a list of tape recording techniques and their corresponding actions.

TRTCH	PARITY	DATA CONVERSION	TRANSLATION
C	ODD	YES	NO
OC	ODD	YES	NO
E	EVEN	NO	NO
ET	EVEN	NO	YES
O	ODD	NO	NO
T	ODD	NO	YES
OT	ODD	NO	YES

The *Data Converter* feature of 7 track tape drives causes 4 six-bit characters (24 bits) to be written on the tape for every 3 eight-bit bytes sent from your program. During a read operation, this feature reverses the process causing 4 six-bit tape characters to be converted to 3 eight-bit bytes. This conversion feature is done by the tape unit itself as it reads the tape. Even though the tape can move at the same speed with this feature active, the effective read/write rate is reduced to about 75%.

The *Translate* feature of 7 track tape drives causes eight-bit EBCDIC code bytes (the normal internal processing unit representation), to be written on the tape as six-bit BCD characters. Conversely, 6-bit BCD characters read from the tape are translated into their eight-bit EBCDIC equivalent. This translation feature of the tape unit takes no additional time to perform this operation.

Notes:

1. Multiple magnetic tapes can be read by reading each file with the END= specification on the READ statement. Multiple magnetic tapes can be written by the use of ENDFILE followed by further WRITEs.

2. Magnetic tapes must be of record format *fixed* or *fixed-blocked*.

3. Magnetic tapes may be used only by jobs run on batch. (Users may submit batch jobs from their work-stations. See the discussion of the SUBMIT program for further details.)

4. There is another form of /FILE statement which can be used to define tape UDS files. The syntax of this form is:

   ```
   /FILE TAPE=(u,X'mn',BLK=s),RSIZ=r,VOL=v,DISP=d
   ```

   This form is used by the older versions of MUSIC. Users are NOT encouraged to use this form of the /FILE statement as it is not as powerful as the new form and will not be supported at some later time. Refer to the User's Guide of older versions of MUSIC for a description of the parameters used.

Examples:

```
/FILE 2 TAPE BLK(800) RSIZ(80) VOL(MYTAPE)
/FILE GO.DATA TAPE LRECL(80) VOL(OLDTAP) SHR
/FILE 11 TAPE RSIZ(80) VOL(ABCDEF) 7TRK TRTCH(OC)

/ID TAPEJOB          UUUU,SSS,003,100,100
/FILE 1 TAPE 9TRK BLK(1600) RSIZ(80) VOL(MYTAPE)
       DIMENSION A(100)
       .
       .
       .
       WRITE(1,10)A
10     FORMAT(10I5)
       STOP
       END
/END
```

Adding data to the end of a tape file:

It is possible for MUSIC FORTRAN, ASSEMBLER, COBOL and PL/I programs to add data records to the end of an existing tape file. This is similar to the DISP=MOD feature of OS. The procedure is:

a. Read the existing records until end of file (EOF) is reached. Physically, this means that a special tape mark record has been read.

b. Backspace one record. This backspaces over the EOF tape mark record, so that the tape is positioned immediately after the last data block.

c. Write the new records.

With OS-mode programs (COBOL and PL/I), the LEAVE option must be specified when the tape file is closed at the end of step (a), to prevent automatic rewinding of the tape. In PL/I, use ENVIRONMENT(LEAVE) on the CLOSE statement. See the example below.

Step (b) can be done in FORTRAN by the BACKSPACE statement. There is no equivalent statement in COBOL and PL/I, but the MUSIC library subroutine BACKSP can be used. The calling sequence is:

```
CALL BACKSP('ddname  ')
```

where the character string argument is the 8-character ddname of the tape file. If the ddname is shorter than 8 characters, it must be padded on the right with blanks. In PL/I, BACKSP must be declared as EXTERNAL ENTRY OPTIONS (ASSEMBLER).

Sample PL/I program:

```
/FILE DD1 TAPE VOL(MYTAPE) LRECL(80) BLK(3200) OLD
/LOAD PLI
SAMPLE: PROC OPTIONS(MAIN);
DCL DD1 FILE RECORD SEQUENTIAL,
    BACKSP EXTERNAL ENTRY OPTIONS(ASSEMBLER INTER),
    REC CHAR(80);
ON ENDFILE(DD1) GO TO EOF;
OPEN FILE(DD1) INPUT;
LOOP: READ FILE(DD1) INTO(REC);
    GO TO LOOP;
EOF: CLOSE FILE(DD1) ENVIRONMENT(LEAVE);
CALL BACKSP('DD1     ');
OPEN FILE(DD1) OUTPUT;
REC='ADDED1'; WRITE FILE(DD1) FROM(REC);
REC='ADDED2'; WRITE FILE(DD1) FROM(REC);
CLOSE FILE(DD1);
END;
```

/FILE (Miscellaneous)

This category includes unit record I/O devices and files that are pre-allocated to the user once signing on to MUSIC is complete. No other parameters are required except those listed above.

Syntax:

```
Miscellaneous

/FILE  u    type
       ddn
```

Parameters:

u — specifies the logical unit number by which the unit record I/O device or file will be referenced. It must be a number from 1 to 15 inclusive. No duplicate logical unit number can be defined in one job.

ddn — specifies the step name and data definition name (ddname) by which the unit record I/O device or file will be referenced. Refer to the writeup of the /FILE statement for files for complete details.

type — specifies the type of unit record I/O device or file involved. The valid types that can be specified are the following:

RDR	this is the same as the default logical unit number 5 which is used to read data from the input stream following the /DATA statement. The maximum record size is 80 bytes. Refer to *Chapter 4. File System and I/O Interface* for more information.
PRT	this is the same as the default logical unit number 6 which is the workstation if the program is being run from a workstation, or the high speed printer, if the program is being run from batch. In addition the PRT parameter can be used to send printed output directly to a system printer. (See following section on /FILE statement for printers). The maximum record size is 133 bytes with the first character as a carriage control character. Refer to *Chapter 4. File System and I/O Interface* for more information.
PUN	this is the same as the default logical unit number 7 which is the card punch. This is mainly used if your job is being run from batch. The maximum record size is 80 bytes. Refer to *Chapter 4. File System and I/O Interface* for more information.
TERM	this is the same as the default logical unit number 9 which is used to read input conversationally from your workstation. Your job will temporarily pause waiting for you to respond to the read. From batch, a read on TERM is taken as if it were one on RDR. The maximum record size is 80 bytes.
HOLD	this is the same as the default logical unit number 10 which is used to temporarily store the program output in the *Holding File* until it can be saved when the job is over. You can save the output in the Holding File by using the SV command when your workstation is next in *Go status. The maximum record size is 80 bytes. Refer to *Chapter 4. File System and I/O Interface* for more details about the Holding File.
DUMmy	this specifies a dummy file. A read from this dummy file will cause an end-of-file condition, while the output written to it will be ignored.
UNDEFined	this specifies an undefined file. A read or write on this file will cause an error message to be issued.

Examples:

```
/FILE  GO.PRINT  PRT
/FILE   5  TERM
/FILE  15  DUM
```

```
*Go
list sample
*In progress
/FILE   7   RDR
/FILE  12   PRT
10     READ(7,*,END=50)A
       B=SQRT(A)
       WRITE(12,30)A,B
30     FORMAT(10X,F10.3,5X,F10.3)
       GO TO 10
50     STOP
       END
/DATA
78.45
27.96
85.66
 .
 .
 .
 ETC.
```

/FILE (Printers)

This form of the /FILE statement is used to send program output directly to one of the system printers. The first character of each output record should be a valid carriage control for the printer in question. The maximum record length is 133. The output is placed on the print queue and scheduled for printing when the printer becomes available.

Syntax:

```
Printers

/FILE  u    PRT(name) [COPIES(num)] [FORMS(forms)] [SPACE(n)]
       ddn
```

Parameters:

u	specifies the logical unit number which the program uses to reference the printer. It must be a number from 1 to 15 inclusive. No duplicate logical unit number can be defined in one job.
ddn	specifies the step name and data definition name (ddname) which the program uses to reference the printer. Refer to the writeup of the /FILE statement for files for complete details.
name	specifies the printer location name of the printer to be used.
num	specifies the number of copies required.
forms	specifies on which type of forms the output is to be printed.
SPACE(n)	*n* specifies the amount space needed. SPACE(50) is the default.

Examples:

```
/FILE  6 PRT(SYSTEM)
/FILE  SYSPRINT PRT(ROOM112) COPIES(10)
```

Note: The valid values for the *name* and *forms* fields are dependent on your local MUSIC configuration.

/ID

This statement is used to identify your batch job. For a full description of this statement refer to the topic "Format of the Batch /ID" in *Chapter 3. Using Batch*.

/INCLUDE

This statement serves as a pointer to a file in the Save Library. It may be placed in the input file as well as in files. You may use many /INCLUDE statements if you wish in one job. When this statement is encountered during job processing, the named file will be located and read instead of the /INCLUDE line itself. When the named file has been read, the reading of the original file will be resumed.

This /INCLUDE feature allows your program or data to be stored in multiple files. For example, each subroutine could be saved as a separate file. The /INCLUDE facility allows the user complete freedom of how and when to split up the user's program and/or data into files.

The /INCLUDE statement is not processed as such during EDIT and SAVE operations. This feature allows you to create and maintain files which contain /INCLUDE statements. The /INCLUDE statements only take effect when your program is executing.

/INCLUDE statements may be *nested* if desired. That is, an included file may itself contain /INCLUDE statements pointing to files which themselves contain /INCLUDE statements, etc. The maximum level of nesting is 5, with the first include statement pointing to the second level.

Include functions can also be controlled during program execution by use of the following subroutines. Refer to *Chapter 9. System Subroutines* of this publication for further details on these routines.

SYSINE, SYSINL, SYSINM, and SYSINR

If a /INCLUDE statement refers to a file saved with the XO (exec-only) attribute, then the only lines that may precede the /INCLUDE statement are /SYS, /FILE, /COM, /INFO, and /ETC. For batch jobs, /ID and /PAUSE may also precede the statement.

If a /INCLUDE statement refers to a file which has a record length of longer than 80 bytes, then only the first 80 bytes of each record will be processed. If the entire record length of the file has to be processed, then the file must be referred to by a /FILE statement as previously described.

Syntax:

```
/INCLUDE    name[([NEST][,EOF][,ERRS][+m-n])]
/INC
```

Parameters:

name — the name of a file saved in the Save Library. In order to access another user's file that was saved with the SHR option, you must prefix the file's name with the owner's userid as in the example userid:ABC.

NEST — /INCLUDE statements found within the named file are to be processed. If NEST is not specified, any /INCLUDE statements will be treated as data lines rather than statements. This option is the default up to the point in time that a line other than a /INCLUDE, /SYS, /FILE, /COM, or /ETC is found in the input job stream. From this time on, /INCLUDE statements found in *included* files will be treated as data lines unless the /INCLUDE statement which *includes* the file specifies the nest option.

EOF — causes an end-of-file condition after reading the last line of the named file. If the NEST option is also in effect then the EOF condition occurs at the end of each nested file. If not specified, the EOF condition occurs only after all lines of all files have been read.

ERRS — causes an end-of-file condition for any error associated with the reading of any included file. Errors that cause this condition include FILE NOT ACCESSIBLE, a bad /INCLUDE statement, etc. See system subroutine SYSINE for further information.

+m-n — specifies which lines in the file to include from line *m* to *n.*

Note: The options specified must be enclosed in parentheses.

Examples:

```
/INCLUDE ABC
/INCLUDE USERID:ABC
/INC ABC(NEST)
/INC ABC(NEST,ERRS)

*Go
list sample
*In progress
/include alan
/include roy
/include wilf(nest)
*End
*Go
sample
*In progress
      (program output)
```

Messages:

`/INCLUDE xxxxxx  ERR02 USER OR PROGRAM ERROR`
No name was specified for a dynamic /INCLUDE requested by a program, or a /INCLUDE was incorrect.

`/INCLUDE xxxxxx  ERR04 FILE NOT ACCESSIBLE`
The specified file name was not found in the Save Library, or is a private file belonging to another user, or is an execute-only file.

`ERROR WHILE READING FILE xxxxxx  ERR03`
Either a system error was encountered while reading the file, or the file's contents does not match its

record format. This could happen if an attempt is made to /INCLUDE a load module file which was created with record format FC (the default). Load modules should be created with record format F.

/INFO

This statement is used to inform SUBMIT about where the job is to be submitted to. If specified, it must be the first statement in the job. The first parameter must be the name of the processor (system) to which the job is to be submitted. If *processor* is specified as *MUSIC*, the job will be submitted to MUSIC batch for processing. Consult your installation for a list of other valid processors. The keyword parameters *(kw1, kw2, ...)* are used to specify values *(value1, value2, ...)* which will be used in the appropriate control statements of the job. These parameters are dependant on the particular processor to which the job is submitted, and are usually used to specify items such as time and page limits, passwords, output destinations, etc.

Refer to the description of "Submitting Jobs to MUSIC Batch and Other Operating Systems" in *Chapter 3. Using Batch* of this guide for full details.

Syntax:

```
/INFO        processor  [kw1(value1)]  [kw2(value2)]  . . .
```

/JOB

This statement is used to specify certain parameters that affect processing of the job. This statement is normally used to pass parameters to the MUSIC loaders or to the Linkage Editor.

One blank must appear between the /JOB and the first parameter. As usual, each parameter is separated from each other by commas. Invalid parameters are sometimes ignored. The parameters which may be used with each processor are described in *Chapter 8. Processors*.

Syntax:

```
/JOB         parms...
```

/LOAD

This statement is used to designate a particular processor to be used to process the lines following. The /LOAD statement usually can be followed by a /OPT statement specifying options for the particular compiler or assembler and by a /JOB statement specifying parameters for the loading step. In all cases the processor loads and executes the object code unless /JOB NOGO has been specified. /LOAD statements following the first one are not allowed.

Syntax:

```
/LOAD       name
```

Parameters:

name — where name is one of the following:

APL — specifies that the APL subsystem is to be invoked. See the discussion of MUSIC/APL in the subsystems chapter.

ASM — specifies that the lines following form a program written for the VS Assembler (and optionally, object modules), and are to be processed by the Assembler and the resulting object program is to be loaded and executed using the OS/MUSIC interface.

ASMLG — specifies that the lines following consist exclusively of object modules to be loaded and executed using the OS/MUSIC interface. Utilization of processing unit time is optimized if this statement is used instead of /LOAD ASM.

COBLG — specifies that the lines following consist exclusively of object modules to be loaded and executed using the OS/MUSIC interface. Utilization of processing unit time is optimized if this statement is used instead of /LOAD COBOL.

COBOL — specifies that the lines following are an VS COBOL program (and optionally, object modules), and are to be processed by the VS COBOL Program Product compiler and the resulting object program is to be loaded and executed using the OS/MUSIC interface.

COBOL2 — specifies that the lines following are an VS COBOL II program (and optionally, object modules), and are to be processed by the VS COBOL II Program Product compiler and the resulting object program is to be loaded and executed using the OS/MUSIC interface.

C370 — specifies that the lines following are a C program (and optionally, object modules), and are to be processed by the C/370 compiler and the resulting object program is to be loaded and executed using the OS/MUSIC interface.

EXEC — specifies that a load module contained in a file is to be executed. The data set must have been defined on a previous /FILE statement.

FORTG1 — specifies that the lines following are a FORTRAN IV (G1) program (and optionally, object modules), and are to be processed by the IBM FORTRAN (G1) Program Product Compiler and the resulting object program is to be loaded and executed.

GPSS — specifies that the lines following are a GPSS V program, and are to be processed by the General Purpose Simulation System program.

IEFBR — specifies that a program that does nothing except return control to MUSIC. This program is useful when you wish to create or delete a UDS file via a /FILE statement. Refer to the /FILE statement writeup for an example of a usage of this program.

LKED — specifies that the lines following consist of object modules and Linkage Editor control lines to be processed by the Linkage Editor.

LOADER — specifies that the lines following consist exclusively of object modules to be loaded and

executed by the MUSIC loader. Utilization of processing unit time is optimized if this statement is used instead of /LOAD BASIC or FORTG1.

VSPASCAL specifies that the lines following consist of source statements to be processed by the IBM VS/PASCAL compiler. Object modules may also be present.

PLI compile and execute a PL/I program. This processor invokes the PL/I Optimizing Compiler, and then the loader. After loading, the program is automatically executed using the OS/MUSIC interface (unless /JOB NOGO is used or the compiler return code is 12 or more). The /OPT and /JOB control statements may be used with this processor. PL/I object modules can optionally be produced and saved, intermixed with source. The object modules are identical with those produced on OS.

PLILG load and execute a PL/I program in object module form. It is more efficient to use this processor than /LOAD PLI when the input consists solely of object modules. The /JOB control statement may be used with this processor to specify LOADER options.

REXX Specifies that the lines following are to be processed by the high level language REXX (Restructured Extended Executor). REXX allows the use of MUSIC statements directly in the REXX program.

RPG specifies that the lines following are an RPG II program and are to be processed by the RPG II and the resulting object program is to be loaded and executed.

RPGAUTO specifies that the lines following are RPG auto report control statements and are to be processed by the RPG auto report processor and the resulting RPG II program is to be compiled and executed.

RPGLG specifies that the lines following are RPG II object modules and are to be loaded and executed.

VSBASIC Specifies that the lines following are to be processed by the VS BASIC compiler. Refer to *Chapter 8. Processors* of this guide for more information about this compiler.

VSFORT specifies that the lines following are a FORTRAN program (and optionally, object modules), and are to be processed by the IBM VS FORTRAN compiler and the resulting object program is to be loaded and executed.

XMON specifies that a COBOL, VS Assembler or VS Fortran program in load module form is to be executed using the OS/MUSIC interface. The load module file must be defined on a previous /FILE statement. The load module is run using the OS/MUSIC interface.

XMPLI execute a PL/I load module created by the MUSIC Linkage Editor. This is similar to /LOAD XMON. It results in faster loading (compared with /LOAD PLILG) and allows overlay structures to be used. Usage is the same as for /LOAD XMON except that the load module file must not be on I/O unit number 4.

XMRPG executes an RPG II load module created by the MUSIC Linkage Editor.

Note: For further information regarding the use of each processor, see the description in *Chapter 8. Processors*.

/OPT

This statement is used to specify parameters for MUSIC language compilers. If used, it must appear following the /LOAD or /JOB statement. See *Chapter 8. Processors* for parameters which may be used with each language compiler. This statement may be omitted if default parameters are to be used.

Syntax:

```
/OPT        parameters
```

/PARM

This statement is used to pass information to a program. A /PARM is automatically generated by some /EXEC and implied EXEC statements. If several /PARM statements are encountered, only the first one is used.

The character string *parameters* (with leading and trailing blanks removed) can be obtained by the program by calling the PARM subroutine described in *Chapter 9. System Subroutines*.

Syntax:

```
/PARM        parameters
```

/PASSWORD

This statement is used in batch to provide the user's batch password. For more information about batch processing, refer to *Chapter 3. Using Batch*.

Syntax:

```
/PASSWORD=xxxxxxxx
```

Parameters:

xxxxxxxx is the 1- to 8-character batch password that must be supplied when submitting batch jobs.

/PAUSE

This statement is used on batch to send messages to the operator. The operator has to respond to this message before your batch job continues. For more information about batch processing, refer to *Chapter 3*.

Using Batch.

Syntax:

```
/PAUSE message
```

Parameters:

message — is the text of your message which is sent along with your batch job. This message might contain information about special handling for your job.

/SYS

This statement is used to specify a limit to the processing unit time to be used by a job. If the time requested on the /SYS statement is greater than the maximum allowed in the user's User Profile, the latter is used. The /SYS statement must appear before any /LOAD statement in order to be correctly handled. /SYS can also specify other options such as job region size.

Syntax:

```
/SYS [NOPRINT][,TIME=nnn][,NOSKIP][,REG=nnn][,NEXT=filenm             ]
             [,TIME=nnS]                     [,NEXT=filenm/parameter  ]
             [,TIME=MAX]                     [,NEXT=!filenm/parameters]

       [,CD=\directory][,DEBUG]
```

Parameters:

NOPRINT — specifies that control statements (such as /FILE) are not to be printed for this job. If this parameter is specified it should be the first parameter, with one blank space between /SYS and NOPRINT.

TIME=nnn — maximum job time (in units of 60 service units) for which the job is to be allowed to run. When this time is reached, the job is automatically terminated. If the form nnS is used, *nn* specifies the maximum time in service units. If MAX is specified, the maximum time limit allowed in the User Profile is used. The TIME parameter is ignored for a job run from batch since the batch time limit is given on the /ID statement. This parameter may be omitted if the default time limit specified in the User Profile is desired.

Some users may be limited to a maximum they can specify. This limit may be different for prime and non-prime time. You can find the maximum time limits allocated for your userid by running the PROFILE program and specifying the option PRINT.

NOSKIP — for batch jobs, causes the system to suppress the automatic skip to a new page at the bottom of every page of printed output on the high speed printer.

REG=nnnn — specifies the region size desired, in units of 1K = 1024 bytes. For example, "/SYS REG=512" results in a region size of 512K. If this parameter is not specified, MUSIC will

usually assume a region size of 256K (some processors may assume a larger size region). The region size specified should be an exact multiple of 4. Thus, valid sizes include 108, 112, 116, etc. If multiple /SYS statements are used in the same job, the largest region size specified is used. This allows a user to increase a default region size specified within a program file. To specify a region smaller than given on a later /SYS statement, use a Y after the size, for example /SYS REGION=200Y. The Y causes subsequent REGION options to be ignored.

NEXT — specifies the next program to execute after the current program terminates. When the file name is followed by a slash, the text following the slash will be passed to the next program as a parameter list. To make the next program non-cancellable, place a ! directly before the file name. The NEXT parameter must be the last parameter on the /SYS line.

CD=\directory — specifies the current directory to be used for the job. "directory" is anything that can be used as a parameter on the CD (Change Directory) command. The specified directory is made the current one when the job starts. Examples:

```
/SYS CD=\MYDIR
/SYS CD=\              <-- root dir
```

DEBUG — invokes the DEBUG utility as the first user program in the user region. Userid privileges are required.

Examples:

```
*Go
list sample
*In progress
/SYS TIME=8S
    1 GO TO 1
      END
*End
*Go
sample
*In progress
MAIN   = 0000FE
003448 BYTES USED
EXECUTION BEGINS
*VSM006 TIME ESTIMATE OF 8 SUs EXCEEDED. JOB TERMINATED
*End
*Go
```