MSC/NASTRAN CONFIGURATION AND OPERATIONS GUIDE

UNIX EDITION

Version 70.5

The MacNeal-Schwendler Corporation

Corporate Headquarters
The MacNeal-Schwendler Corporation
815 Colorado Boulevard
Los Angeles, CA 90041-1777
Tel: (213) 258-9111 or (800) 336-4858
FAX: (213) 259-3838

Headquarters, European Operations
MacNeal-Schwendler GmbH
Innsbrucker Ring 15
Postfach 80 12 40
81612 München, GERMANY
Tel: (89) 431 9870
FAX: (89) 436 1716

Headquarters, Far East Operations
MSC Japan Ltd.
Entsuji-Gadelius Building
2-39, Akasaka 5-chome
Minato-ku, Tokyo 107, JAPAN
Tel: (03) 3505-0266
FAX: (03) 3505-0914

DISCLAIMER

The concepts, methods, and examples presented in this text are for illustrative and educational purposes only, and are not intended to be exhaustive or to apply to any particular engineering problem or design. The MacNeal-Schwendler Corporation assumes no liability or responsibility to any person or company for direct or indirect damages resulting from the use of any information contained herein.

©1972, 1997, 1998 by The MacNeal-Schwendler Corporation
1st Printing April 1998
All rights reserved.

MSC, MSC/, MSC/PATRAN and MSC/MVISION are registered trademarks and service marks of The MacNeal-Schwendler Corporation. NASTRAN is a registered trademark of the National Aeronautics and Space Administration. MSC/NASTRAN is an enhanced, proprietary version developed and maintained by The MacNeal-Schwendler Corporation. I-DEAS is a trademark of Structural Dynamics Research Corporation. ADAMS is a registered trademark of Mechanical Dynamics, Inc. The installation procedure uses the GZIP package from the Free Software Foundation. GZIP can be obtained by anonymous ftp at prep.ai.mit.edu or by contacting the Free Software Foundation at 675 Mass Ave., Cambridge, MA 02139 U.S.A. Other product names and trademarks are the property of their respective owners.

NA * V70.5 * Z * Z * Z * DC-OPS

TABLE OF CONTENTS

C H A P T E R

1 INTRODUCTION

1.1 The Scope of This Document

The *MSC/NASTRAN Configuration and Operations Guide* provides instructions on how to install, customize, and use MSC/NASTRAN Version 70.5.

Note: This document provides information for systems that are not yet supported by MSC/NASTRAN Version 70.5. MSC does not guarantee that these systems will be supported at a later date.

1.1.1 Key for Readers

As an aid to clarity, this document uses several visual conventions to indicate the action of the MSC/NASTRAN user. These conventions are described as follows:

Italics	Indicate a place holder for a variable value that must be inserted. Example: The system RC file is *install_dir*/conf/nast705rc.
`Courier Font`	Indicates input to the system or output from the system. Example: `$ install_dir/bin/mscid`
“Quote marks”	Used to indicate other items (such as lowercase keywords, commands, variables, DBsets, or file suffixes) that might not be otherwise distinguishable from the descriptive text surrounding them. Example: If “out” is not specified, the output files are saved using the basename of the input data file as a prefix.

1.2 The Structure of This Document

This document contains four major parts:

- Chapters 2 and 3 – Installation and configuration.
- Chapters 4 and 5 – Basic and advanced use of MSC/NASTRAN.
- Chapters 6 and 7 – Utility and sample programs including MSC/ACCESS and the beam server.
- Appendixes – Glossary, system descriptions, etc.

1.2.1 Installation and Configuration

Chapter 2 shows how to use the MSC/NASTRAN interactive installation script. Chapter 3 demonstrates how to customize MSC/NASTRAN for your computing environment.

1.2.2 Basic and Advanced Use

There are two chapters containing information on MSC/NASTRAN usage. Chapter 4 presents the basic functions of the nastran command and provides some details on file and database usage. Chapter 5 explains how to use the advanced features of the nastran command and includes information on computer resource management.

1.2.3 Utility Programs

There are two chapters that provide information on the utility and sample programs. Chapter 6 describes how to use and customize the utility programs. Chapter 7 explains how to build and use the sample programs.

1.3 Changes for MSC/NASTRAN Version 70.5

1.3.1 Changes to MSC/NASTRAN Capabilities

IBM RISC System/6000

Database files larger than 2GB are now supported if you are running MSC/NASTRAN on AIX 4.2 or later. The filesystem containing the file must also support large files. See your system administrator to determine which filesystems, if any, support large files.

Sun SPARC Solaris

Database files larger than 2GB are now supported if you are running MSC/NASTRAN on Solaris 2.6 or later. The filesystem containing the file must also support large files. See your system administrator to determine which filesystems, if any, support large files.

1.3.2 The nastran Command

New Keywords

The following keywords have been added. See Appendix B for additional details.

pause — The “pause” keyword stops the nastran command before it exits, and waits for the user to type either the “Enter” or “Return” key. This can be useful when the nastran command is embedded in another program. See Appendix B for further details.

whence — The “whence” keyword displays the value of one or more keywords after the command line and all RC files have been processed. This can be useful to determine where a keyword is being assigned.

Modified Keywords

The following keywords have been changed. See Appendix B for additional details.

authorize — The “authorize” keyword will now check for the existence of the license server node if one is identified using the “@*node*” syntax. If the server does not exist, a USER FATAL MESSAGE will be issued.

batch — The “batch” keyword will now run a job under nice(1) when “batch=yes” is specified.

cpu, ppcdelta — The “cpu” and “ppcdelta” keywords will now accept a value in either decimal seconds or hours:minutes:seconds format. The value will always be converted to decimal seconds. For example, all of the following values specify one hour and fifteen minutes: “1:15:0”, “75:0”, “::4500”, or “4500”.

help — The “help” keyword now provides keyword “categories”. In addition, the simple request “msc705 nastran help” will now display a summary of help commands.

memory — The “memory” keyword default is no longer 4MW. If a value has not been assigned to the “memory” keyword, the nastran command will assume “memory=estimate”. If an explicit null value has been set, the nastran command will not assume a default value and the run will fail.

This allows your site to establish a default policy for memory as follows:

- If your site sets the memory keyword to a non-null value, you can choose to override this default by explicitly setting the memory keyword on the command line or in an RC file. You can accept the default by not setting a new value.
- If your site sets the memory keyword to a null value, i.e., “memory=”, in an RC file, users must set the “memory” keyword to a non-null value in one of their RC files or on the command line. If they do not set a non-null value, i.e., they leave the null value, the nastran command will report a fatal error.
- If no value for the “memory” keyword in has been set in any RC file or on the command line, “memory=estimate” will be assumed.

msgcat — The “msgcat” keyword will now validate the binary message catalog before starting the MSC/NASTRAN analysis.

node — The “node” keyword will now check for the existence of the specified node. If the node does not exist, a USER FATAL MESSAGE will be issued.

symbol — Symbols defined in the system and home RC files can now be used on the command line. Symbols defined in an RC file explicitly specified on the command line can also be used. This can be useful for specifying private or site libraries that are easily referenced using symbols.

Other Changes to the nastran Command

The following additional changes have been made to the nastran command.

- The template files now generate Korn shell scripts (they previously generated Bourne shell scripts).
- The “acid” and “acdata” keywords are now always sent to the accounting data logging program, acct, if accounting has been enabled using the “acct=yes” keyword. Sites no longer have to provide this customization.
- The accounting data logger will now only modify or create files with standard MSC/NASTRAN filenames, i.e., msc*yymm*.acc where *yy* is the last two digits of the year and *mm* is the month number. The argument specifying the filename is now only used to specify the directory.
- RC file lines can now be split using a backslash, “\”, as the last character on the line to be continued. For example, the following lines are now valid in an RC file:

```
submit=medium,medium2,long,batch=\
qsub –q %queue% –x –eo \
–lf 9000Mb –lF 9000 MB\
%ppc:–lt {}% %cpu:–lT {}% –lm %ppm% –lM %prm%\
–s /bin/ksh job
```

- The INI and template files are now linked from the architecture directory, e.g., *install_dir*/msc705/arch, to the bin directory *install_dir*/bin.

1.3.3 Modified Utilities

The following utilities have been changed. See Chapter 6 for additional details.

BMSRV

The Beam Server is now available on all Cray systems.

ESTIMATE

- The ESTIMATE utility will now read the standard MSC/NASTRAN RC files. This is accomplished by invoking the nastran command as a subprocess, therefore the nastran command must be fully functional for this feature to work. If you are using the ESTIMATE utility on a system that does not have a functional MSC/NASTRAN installation, you must continue to use the standard ESTIMATE RC files, i.e., "$HOME/.estimaterc" and "*data-file-directory*/.estimaterc". See the "nastrc" keyword for additional details.

- An error in CTRIAX6 processing has been corrected. ESTIMATE now correctly assigns the "1" and "3" DOF.

- Setting a value on the command line, or in an RC file will now automatically suppress any rule that would estimate the value. For example, setting "buffsize" in an RC file will now suppress rule 1, the rule that would calculate a new BUFFSIZE value.

New Rules

Rule 12 — The existing rule 4, which deleted the HICORE and REAL entries, has been split into rule 4 (delete HICORE) and rule 12 (delete REAL).

Rule 13 — This rule will suppress use of the undocumented SuperModule feature.

Rule 14 — This rule will suppress use of the Parallel Lanczos capability of MSC/NASTRAN. This capability was removed in V70.5.

Modified Keywords

buffsize — The "buffsize" keyword now accepts the value "estimate". This can be used to override a previous RC file or command line entry that set a value for BUFFSIZE. This keyword cannot appear in a ESTIMATE RC file if "nastrc=yes" was specified.

bpool — This keyword cannot appear in an ESTIMATE RC file if "nastrc=yes" was specified.

memory — The "memory" keyword now accepts the value "estimate". This can be used to override a previous RC file or command line entry that set a value for memory. This keyword cannot appear in an ESTIMATE RC file if "nastrc=yes" was specified.

smemory — This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” was specified.

version — This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” was specified.

New Keywords

enable — The “enable” keyword can be used to explicitly enable rules. This may be useful to enable a rule that was automatically suppressed when a value was assigned. For example, the following command will now calculate the estimated memory requirements for a job even though a value for memory was specified on the command line:

```
msc705 estimate myjob memory=5mb enable=10
```

nastrc — The “nastrc” keyword allows you to select the type of RC file processing invoked by the ESTIMATE utility. Setting “nastrc=yes”, the default, will process the standard MSC/NASTRAN RC files before the standard ESTIMATE RC files, i.e., $HOME/.estimaterc and “*data-file-directory*/.estimaterc”, are processed. Setting “nastrc=no” will only process the standard ESTIMATE RC files.

Note: If “nastrc=yes” has been specified, the following keywords cannot appear in the ESTIMATE RC files:

buffsize, bpool, memory, real, smemory, version.

pause — The “pause” keyword stops ESTIMATE before it exits, and waits for the user to type either the “Enter” or “Return” key. This can be useful when ESTIMATE is embedded in another program. See Chapter 6 for further details.

real — The “real” keyword functions identically to the nastran command’s “real” keyword.

MSCACT

- The “acid” and “acdata” keywords are now always processed. This change has been made such that any accounting file generated by any previous version of MSC/NASTRAN is still fully compatible.

- The filename can now be specified as "*yymm*" where *yy* is the the last two digits of the year and *mm* is the month number. For example,

  ```
  msc705 mscact 9706
  ```

  will generate the standard usage report for June 1997. The accounting files must use the standard naming conventions and must be stored in *install-dir*/acct.

- If a file suffix is not given, the standard ".acc" suffix is assumed.

- A header has been added to the MSCACT report. The report's detail lines are now sorted according to the "sortby" keyword.

New Keywords

sortby — The "sortby" keyword allows you to specify the order of the report's detail lines. Valid values are "none", "name", "cpu" and "count" to suppress sorting, or sort by name (the default), cpu time, or count of items respectively. For example,

```
msc705 mscact yymm sortby=none
```

will produce a report very similar to the previous versions of this utility.

MSGCMP

- The MSGCMP utility will now validate a binary message catalog before attempting to convert it to text form.

1.3.4 MSC/ACCESS

The MSC/ACCESS libraries on Hewlett-Packard systems now export names with a trailing underscore character. This change brings MSC/ACCESS on HP systems in line with "standard" UNIX practice for inter-language calling conventions.

The old name formats, i.e., without the trailing underscores, are available in

install-dir/msc705/hpux/libdbio.old.a

and

install-dir/msc705/sppux/libdbio.old.a

You may need these libraries if you have programs that must be linked using the old naming conventions. These programs should be converted to use the trailing-underscore naming convention.

1.4 The Directory Structure

The installation directory structure provides the following capabilities:

- Multiple versions of MSC products, such as the current and prior versions of MSC/NASTRAN.
- Multiple computer architectures, such as Cray UNICOS, SUN SPARC Solaris, etc.

Figure 1-1 shows the directory structure in the *install_dir* directory, which is named during installation.

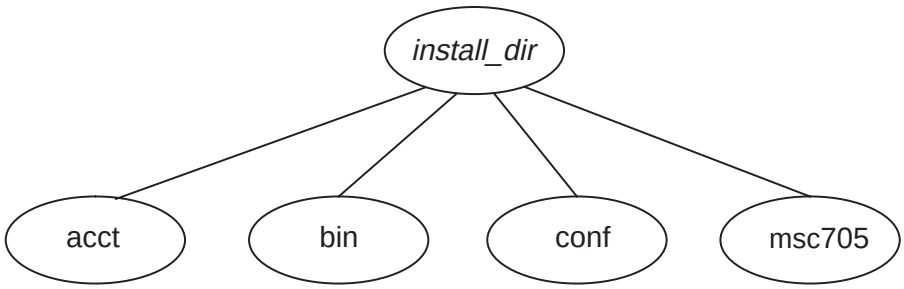


Figure 1-1. Directory for *install_dir*.

1.4.1 Multiple Products Support

The MSC/NASTRAN installation directory structure supports multiple products by using product-dependent and architecture-independent directories and files. For example, Figure 1-2 shows that the *install_dir*/msc705/nast directory contains the product-dependent files for MSC/NASTRAN Version 70.5 while the util and access directories contain the product-independent files for the various utilities and MSC/ACCESS.

1.4.2 Multiple Computer Architecture Support

The MSC/NASTRAN installation directory structure also supports multiple computer architectures by using architecture-dependent directories and files. All files that are dependent upon a computer architecture are isolated in a single architecture directory *install_dir*/msc705/*arch*, where *arch* is the name of the architecture, e.g., aix, alpha, hpux (see Table 3-1).

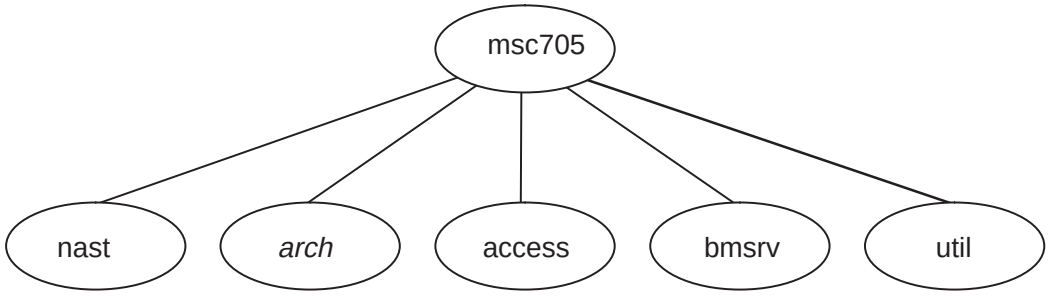


Figure 1-2. Directory for msc705.

The *install_dir*/msc705/nast directory contains news, documentation, and sample problems for MSC/NASTRAN. None of these files are architecture dependent.

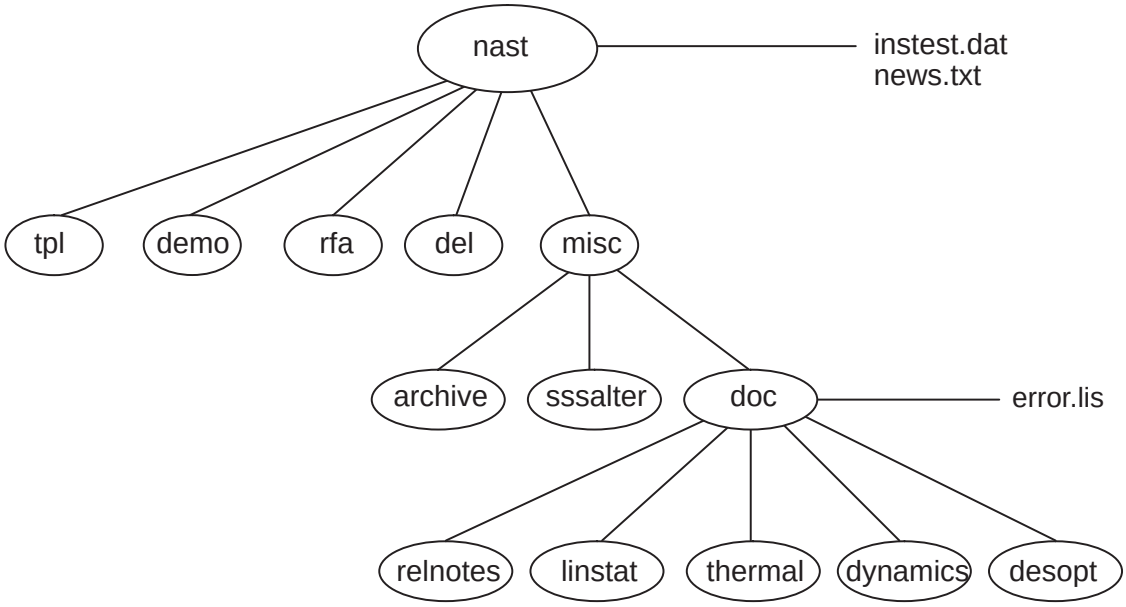


Figure 1-3. Directory for nast.

The MSC/ACCESS directory (*install_dir*/msc705/access) contains source and make files (see Figure 1-4) for the MSC/ACCESS sample programs. None of these files are architecture dependent. The DBIO library, which is architecture dependent, is located in the architecture directory, i.e., *install_dir*/msc705/*arch*/libdbio.a.

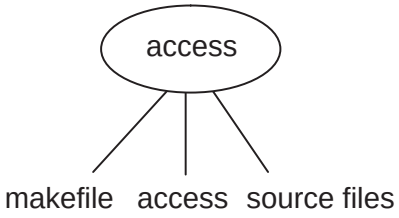


Figure 1-4. Directory for access.

The beam server directory (*install_dir*/msc705/bmsrv) contains source and make files (see Figure 1-5) for the beam server sample programs. None of these files are architecture dependent. The beam server library, which is architecture dependent, is located in the architecture directory, i.e., *install_dir*/msc705/*arch*/libbmsrv.a.

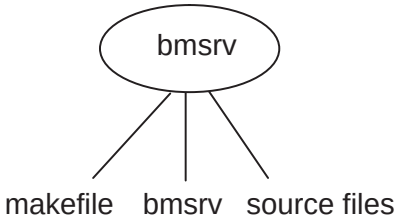


Figure 1-5. Directory for bmsrv.

The utility programs directory (*install_dir*/msc705/util) contains source and make files (see Figure 1-6) for the utilities that are also delivered in source form. None of these files are architecture dependent.

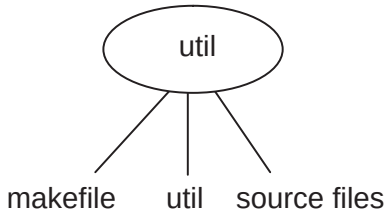


Figure 1-6. Directory for util.

CHAPTER 2

HOW TO INSTALL MSC/NASTRAN

The procedures to install MSC/NASTRAN are discussed in the following sections.

Note: This chapter provides information for systems and media that are not yet supported by MSC/NASTRAN Version 70.5. MSC does not guarantee that these systems or media will be supported at a later date.

2.1 Before You Begin

Please check the System Descriptions in Appendix C to verify that your system's operating system is compatible with this version of MSC/NASTRAN.

2.2 Installing MSC/NASTRAN

2.2.1 Installing from a Local CD-ROM

Note: The CD contains the "INSTALL.htm" and "INSTALL.txt" files which are present on the CD when the CD is mounted in step 3 of the following procedure. These files provide detailed information on the CD contents and space requirements.

Procedure:

1. Log on to the system as Superuser (root).
2. If necessary, create a directory to use as the CD-ROM mount point:

   ```
   mkdir /CDROM
   ```

3. Insert the CD-ROM and mount the CD-ROM filesystem. The device names in the following commands are examples, the actual device name on your system may differ.

   Digital

   ```
   mount –rt cdfs –o noversion,rrip /dev/rz4c /CDROM
   ```

   HP 9000, HP Exemplar SPP-UX 5

   ```
   /usr/sbin/mount –rF cdfs /dev/dsk/c1t2d0 /CDROM
   ```

   HP Exemplar SPP-UX 3, 4

   ```
   mount –rt cdfs /dev/dsk/c201d4s0 /CDROM
   ```

   IBM

   ```
   mount –prv cdrfs /dev/cd0 /CDROM
   ```

   Silicon Graphics, Sun

   Mounts automatically.

4. Change the working directory to the CD-ROM filesystem:

   Sun

   ```
   cd /cdrom/cdrom0
   ```

   All Others

   ```
   cd /CDROM
   ```

5. Start the installation script:

   HP

   ```
   ksh ”./MSCSETUP.;1”
   ```

   All Others

   ```
   ksh ./mscsetup
   ```

6. Choose option 1: *Install from a local CD*.
7. Follow the prompts to complete the installation. See Section 2.3 for installation notes.

8. Unmount the CD-ROM.

   Silicon Graphics

   ```
   eject /CDROM
   ```

   Sun

   ```
   eject cdrom0
   ```

   All Others

   ```
   umount /CDROM
   ```

2.2.2 Installing from a Remote CD-ROM

Notes:
1. The “/etc/hosts.equiv” and your “.rhosts” files on the remote system must allow access from the local system.
2. The CD contains the “INSTALL.htm” and “INSTALL.txt” files (loaded in step 4 of the following procedure). These files provide detailed information on the CD contents and space requirements.

Procedure:

1. Mount the CD-ROM filesystem on the remote system as described in steps 1 through 3 of Section 2.2.1.
2. Log on to the local system. Some of the following operations may require Superuser (root) privileges.
3. Change the working directory to /tmp or some other scratch directory with about 350 Kb of free space.

   ```
   cd /tmp
   ```

4. Copy the installation files from the remote CD:

   Cray, HP

   ```
   remsh node [ –l user] dd if=file_set bs=10240 | tar xvfo –
   ```

   Digital, IBM

   ```
   rsh node dd [ –l user] if=file_set bs=10240 | tar xvfB –
   ```

Fujitsu

```
rsh node dd [ –l user] if=file_set bs=10240 | tar xvfo –
```

Hitachi

```
rsh node dd [ –l user] if=file_set bs=10240 | tar xvfpB –
```

NEC

```
/usr/ucb/rsh node [ –l user] dd if=file_set bs=10240 | tar
xvfop –
```

All others

```
rsh node dd [ –l user] if=file_set bs=10240 | tar xvfoB –
```

where *node* is the network name of the remote node, *user* is an alternate user if the current user does not have remote shell privileges on *node*, and *file_set* is based on the remote system as follows:

HP

```
"'/CDROM/MSCSETUP.TAR;1'"
```

Sun

```
/cdrom/cdrom0/mscsetup.tar
```

All others

```
/CDROM/mscsetup.tar
```

5. Start the interactive installation script:

```
ksh ./mscsetup
```

6. Choose option 2: *Install from a remote CD*.

7. Follow the prompts to complete the installation. See Section 2.3 for installation notes.

8. Unmount the CD-ROM filesystem on the remote system as described in step 8 of Section 2.2.1.

2.2.3 Installing from a Local Tape

Notes:
1. You must have a tape device as described in Table 2-1 to read the tape.
2. The tape contains the “INSTALL.htm” and “INSTALL.txt” files in the first file set (loaded in step 5 of the following procedure). These files provide detailed information on the tape contents and space requirements.
3. Cray only
   The method used to mount and read a local tape depends on whether the tape mount daemon is active. The tape mount daemon is always active on mainframe systems; the daemon may be active on J90 and EL systems.

   The following command may be used to determine if the tape mount daemon is active:

   ```
   ps –e | grep tpdaemon
   ```

   If any lines, other than the grep command are displayed, the daemon is active.
4. Cray only
   The installation requires the ability to read unlabeled tapes. This privilege is granted by the command

   ```
   /etc/udbgen –c update:uid:permbits:wrunlab
   ```

Procedure:

1. Log on to the local system. Some of the following operations may require Superuser (root) privileges.
2. Verify that the tape is write protected (i.e., the write-enable ring is removed on an open reel tape; in the “SAFE” position on a quarter-inch cartridge; the tab is closed on an 8 mm tape; the tab is open on a 4 mm tape; the white dot on the write protect switch is visible on a 3480 cartridge).
3. Mount the delivery tape.

   Cray - tape mount daemon active

   ```
   rsv TAPE
   tpmnt –l nl –v MSC705 –p /tmp/msc705_1 –q 1 –b 10240 –g
   TAPE –r out
   ```

```
tpmnt –l nl –v MSC705 –p /tmp/msc705_2 –q 2 –b 10240 –g
TAPE –r out –w
```

where *TAPE* is a device group name to refer to the media you are mounting. This tape must be valid on your system; *MSC705* is a volume ID used to identify your tape to the operator; any one-to-six character alphanumeric string may be used. The pathnames */tmp/msc705_1* and */tmp/msc705_2* are used to reference the two files on the tape; any two path names may be used.

Fujitsu

```
drvrsv –d DAT dat
medmnt –o ro MSC705 dat
```

where *dat* is the name that will refer to the tape and *MSC705* is the tape volume ID used to identify your tape to the operator.

All others

No command required.

4. Change directory to /tmp or some other scratch directory with about 350 kilobytes of free space.

```
cd /tmp
```

5. Copy the installation files from the local tape:

Digital, IBM

```
tar xvf tapename
```

Hitachi

```
tar xvfp tapename
```

NEC

```
tar xvfop tapename
```

All others

```
tar xvfo tapename
```

where *tapename* is the name of the tape device. Typical device names are shown in Table 2-1.

Table 2-1. Tape Device Names.

Computer	Tape Device Characteristics	Typical *tapename*
Cray (tape daemon)		See first “tpmnt” in step 3.
Cray (no tape daemon)	no-rewind	/dev/*nrpd03*
Digital	no-rewind	/dev/*nrmt0a*
Fujitsu	no-rewind	/dev/media/*dat*
Hitachi	no-rewind	/dev/*rmt0n*
HP	no-rewind, BSD-style	/dev/rmt/*0mn*
IBM	no-rewind, block_size=1024	/dev/*rmt0.1*
NEC	no-rewind	/dev/*mt06d62nr*
Silicon Graphics	no-rewind, no-swap	/dev/rmt/*tps0d5nrns*
Sun	no-rewind	/dev/rmt/*0mn*

6. Start the interactive installation script:

```
ksh ./mscsetup
```

7. Choose option 1: *Install from a local tape*.

   Cray - tape daemon active

   Use the name defined by the second “tpmnt” command in step 3

   All others

   Use the *tapename* selected in step 5

8. Follow the prompts to complete the installation. See Section 2.3 for installation notes.

9. Unmount the tape:

   Cray - tape mount daemon active

```
rls –a
```

   Fujitsu

```
medumnt dat
drvrel dat
```

   where *dat* is the name used in step 3.

All others

No command required.

2.2.4 Installing from a Remote Tape

Notes:
1. You must have a tape device as described in Table 2-1 to read the tape.
2. The “/etc/hosts.equiv” and your “.rhosts” files on the remote system must allow access from the local system.
3. The tape contains the “INSTALL.htm” and “INSTALL.txt” files in the first file set (loaded in step 4 of the following procedure). These files provide detailed information on the tape contents and space requirements.

Procedure:

1. Mount the tape on the remote system as described in steps 1 through 3 of Section 2.2.3.
2. Log on to the local system. Some of the following operations may require Superuser (root) privileges.
3. Change directory to /tmp or some other scratch directory with about 350 Kb of free space.

   ```
   cd /tmp
   ```

4. Copy the installation files from the remote tape:

   Cray, HP

   ```
   remsh node [–l user] dd if=tapename bs=10240 | tar xvfo –
   ```

   Digital, and IBM

   ```
   rsh node [–l user] dd if=tapename bs=10240 | tar xvfB –
   ```

   Fujitsu

   ```
   rsh node [–l user] dd if=tapename bs=10240 | tar xvfo –
   ```

   Hitachi

   ```
   rsh node [–l user] dd if=tapename bs=10240 | tar xvfpB –
   ```

NEC

```
/usr/ucb/rsh node [–l user] dd if=tapename bs=10240 | tar
xvfop –
```

All others

```
rsh node [–l user] dd if=tapename bs=10240 | tar xvfoB –
```

where *node* is the network name of the remote node, *user* is an alternate user if the current user does not have remote shell privileges on *node*, and *tapename* is based on the remote system as shown in Table 2-1.

5. Start the interactive installation script:

```
ksh ./mscsetup
```

6. Choose option 2: *Install from a remote tape*. Use the *tapename* selected in step 4.

7. Follow the prompts to complete the installation. See Section 2.3 for installation notes.

8. Unmount the tape on the remote system as described in step 9 of Section 2.2.3.

2.3 Installation Notes

- The following environment variables will affect mscsetup:

  MSC_ARCH, *MSC_BASE*, *MSC_SETUP*, *TMPDIR*.

- The installation script, mscsetup, is a fairly complicated shell script. If too many processes are running when mscsetup runs, the script may hang or generate utility errors. If this occurs, try closing unnecessary windows.

- The installation script identifies the system type in the first screen. If this identification is incorrect, e.g., a new computer model or a new operating system is detected, exit the script. Before restarting the script, set the environment variable *MSC_ARCH* to the correct architecture name as shown in Table 3-1.

- Use this option to install multiple architectures on an application server that is NFS mounted by other systems.

- The disk space requirements shown by mscsetup does not include the scratch space needed to decompress the installation files. Depending upon the particular installation, up to 45 MB of additional space may be needed in the installation file system or the temporary file system. The temporary file system is defined by the TMPDIR environment variable, or “/var/tmp” on Silicon Graphics, or “/tmp” on all others.

- An alternate temporary file system can be assigned using the “-t” option. For example,

  ```
  ./mscsetup –t alternate_temporary_directory
  ```

- During installation, you can generate an “MSC Contract Amendment” form that you can FAX or mail to MSC. Instructions for completing the form and submitting it to MSC are on the form.

- If you need a node-locked authorization code or a license.dat file, you must do one of the following:

  - Perform the installation and generate the “MSC Contract Amendment” on the machine that will run MSC/NASTRAN (in the case of a nodelock authorization code) or the FLEXlm license server (in the case of a license.dat file).

  - Manually change the hostname and ID information in the “MSC Contract Amendment” to correctly identify the machine that will run the software.

- The installation procedure uses the GZIP package from the Free Software Foundation. GZIP can be obtained by anonymous ftp at ftp://prep.ai.mit.edu/pub/gnu or by contacting the Free Software Foundation at 675 Massachusetts Ave., Cambridge, MA 02139 USA.

MSC/NASTRAN Version 70.5

- FLEXlm licensing is only available for the following MSC/NASTRAN V70.5 clients:

  - Cray J90/Y-MP

  - Digital

  - HP 9000

  - IBM

  - SGI

  - Sun

- Use authorize=demo in an RC file or on the command line to invoke the available FLEXlm-based demo license.

Note: The demo license expires on June 30, 1998.

The demo license does not require the FLEXlm server to be installed or running.

If you have a FLEXlm network or counted node-lock license file, identify the name of the FLEXlm license server using option 3 in the “Authorization Information” menu.

If you have a FLEXlm uncounted node-lock license file, identify the pathname of the license.dat file using option 1 in the “Authorization Information” menu; the file will be copied to *install-dir*/flexlm/licenses/license.dat.

If you have a node-lock authorization code file, identify the pathname of the file using option 1 in the “Authorization Information” menu; the file will be appended to *install-dir*/conf/authorize.dat.

If you have a node-lock authorization code, enter the code using option option 2 in the “Authorization Information” menu; the code will be appended to *install-dir*/conf/authorize.dat.

- Any run time libraries needed by MSC/NASTRAN are included in this distribution.
- The installation test option will only be performed on the current architecture.
- You must install the MSC/NASTRAN V70.5 Utility program source option if you want to customize the accounting procedures for your site.
- If you install MSC/NASTRAN V70.5 in an installation base directory containing previous versions of MSC/NASTRAN, your current settings for the “authorize”, “sdirectory”, “buffsize”, and “memory” keywords will be used as defaults.
- HP 9000 running HP-UX 10 and all PA-RISC 1.1. The maximum allocatable memory is controlled by the the *shmmax* and *maxdsiz* kernel parameters. They must be large enough to accommodate the memory requests of each MSC/NASTRAN job.
- If these values are not large enough, MSC/NASTRAN will not be able to allocate open core memory and will terminate with the following message in the LOG file:

  `memory allocation error: unable to allocate` *`mem`* `words`

  where *mem* is the memory allocation request.

  These limits can be increased using the sam(1M) utility. The values are found in “Configurable Parameters” under “Kernel Parameters.”

FLEXlm License Server Version 5.12

- You do not have to install the FLEXlm license server if you are using the FLEXlm-based demo license or an uncounted node-lock license.

- The default port number for the FLEXlm license server is 1700. You must select an alternate port number if this port is already in use.
- If you want the FLEXlm license server to be automatically started at system boot time, you must run mscsetup as root. mscsetup will then be able to add an entry to your /etc/inittab file to start lmgrd at system boot time.
- The FLEXlm license server can be manually started with the command

  ```
  install–dir/bin/flexlm lmgrd
  ```

  where the default license and log files are

  ```
  install–dir/flexlm/licenses/license.dat
  install–dir/flexlm/lmgrd.log
  ```

  An alternate license file can be specified with the “-c” option, e.g.,

  ```
  install–dir/bin/flexlm lmgrd –c license_file
  ```

  An alternate log file can be specified with the “-l” option, e.g.,

  ```
  install–dir/bin/flexlm lmgrd –l log_file
  ```

- Do not shut down the FLEXlm license server using the kill(1) command. Use the following command to shut down the license server.

  ```
  install–dir/bin/flexlm lmdown
  ```

  or

  ```
  install–dir/bin/flexlm lmdown –c license–file
  ```

  It may take a few minutes for the shut down to complete.
- If you have a network or counted node-lock license file from MSC, install this file using option 1 of the “Authorization Information” menu.
- You can install a network license file at any time using the command

  ```
  install–dir/bin/msc705 flexlm new–license.dat–file
  ```

  where new-license.dat-file is the new license file to be installed.
- FLEXlm on-line documentation is available from Globetrotter, see the URL

  http://www.globetrotter.com

- After the installation is completed, see the URL

  file:*install-dir*/flexlm/htmlman/flexframe.html

  for information on configuring and using FLEXlm with MSC products. This file is part of the FLEXlm “HTML Documentation File” option.

2.4 Repeating an Installation

You can repeat the installation process using the playback file generated during every installation. This capability reinstalls MSC/NASTRAN on the same or another computer. The following command is used:

```
./mscsetup playback–file
```

where *playback-file* is the playback file generated during a previous installation (the default playback file is install-dir/mscsetup.pbk).

When a playback file is used, note that:

- The following environment variables will affect mscsetup: *MSC_ARCH*, *MSC_BASE*, *MSC_SETUP*, *TMPDIR*.
- The architectures of every computer using the playback file must be the same as the architecture of the computer that generated the playback file.
- You cannot change the installation base directory.
- You cannot change the installation types and user customizations.
- You cannot install node-lock authorization codes using option 2 of the "Authorization Information" menu. If you use node-lock authorization codes, you must enter the codes using one of the following methods:

1. Place the authorization codes in a file and select option 1 of the "Authorization Information" menu during the installation generating the playback file. This same file must be present during every installation using the playback file.
2. Select option D of the "Authorization Information" menu during the installation generating the playback file and manually edit the authorization code file, *install-dir*/conf/authorize.dat, after every installation is complete.

- If the installation generating the playback file was a remote installation, and you want to mount the CD-ROM in a different system when using the playback file, the *node* and *user* can be changed with the "-r" option. For example

  ```
  ./mscsetup –r node playback–file
  ```

  or

  ```
  ./mscsetup –r user@node playback–file
  ```

CHAPTER 3

HOW TO CONFIGURE MSC/NASTRAN

This chapter shows you how to configure MSC/NASTRAN Version 70.5 for your computer. Authorization must be configured before MSC/NASTRAN will run. Other items that may require configuration include system resource limits, the command initialization file, runtime configuration files, timing blocks, and queue commands.

Two documentation conventions are used throughout the remainder of this document (typically in directory specifications). The string "*install_dir*" indicates the directory where MSC/NASTRAN was installed. The string "*arch*" indicates the architecture or architecture directory for the computer. The architectures are as follows:

Table 3-1. Architecture Names.

Computer	arch
Convex C-Series	convex
Cray UNICOS C90, T90	unicosc90
Cray UNICOS IEEE T90	unicosts
Cray UNICOS J90, Y-MP	unicos
Digital Alpha UNIX	alpha
Fujitsu VX, VPP	uxpv
Hewlett Packard 9000	hpux
Hewlett Packard Exemplar	sppux
Hitachi S-Series HI-OSF/1-MJ	hiosf
IBM RISC System/6000	aix
NEC SX-4	superux
Silicon Graphics R4K R5K	irix

Computer	arch
Silicon Graphics R8K, R10K	irix64
Sun SPARC Solaris	solaris

Architecture names are generally based on the operating system name.

3.1 Using the “msc705” Command

The “msc705” command is shown as a prefix for most of the programs and commands described in this document, for example:

```
msc705 nastran ...
```

By placing the msc705 command in each user’s PATH, e.g., in /usr/bin, all the commands and utilities in this release are uniformly available. The msc705 command also permits version-dependent utilities, such as TRANS, to be easily accessed.

3.2 Configuring a License Manager

MSC/NASTRAN Version 70 reintroduced the FLEXlm license manager as the preferred license manager for node-lock and network licensing. FLEXlm is available on the following platforms:

- Cray Research J90, Y-MP
- Digital Alpha UNIX
- Hewlett Packard 9000
- IBM RISC System/6000
- Silicon Graphics R4K, R5K
- Silicon Graphics R8K, R10K
- Sun SPARC Solaris

MSC/NASTRAN’s implementation of FLEXlm is fully compatible with the FLEXlm implementation within MSC/PATRAN Version 7.0 and later.

If FLEXlm is not supported on your platform, you must use a node-locked authorization code, i.e., the same licensing system available with previous versions of MSC/NASTRAN.

In order to run, MSC/NASTRAN now requires one of the following authorization methods:

- The name of a network license server.
- The pathname of a file containing FLEXlm node-locked licenses.
- The pathname of a file containing one or more node-locked authorization codes.

MSC/NASTRAN will use the first non-null value that it finds in the following hierarchy:

1. The value of the “authorize” keyword in an RC file or on the command line.
2. The value of the MSC_LICENSE_FILE environment variable.
3. The *install_dir*/flexlm/licenses/license.dat file, if it exists.
4. The *install_dir*/conf/authorize.dat file, if it exists.
5. The value of the LM_LICENSE_FILE environment variable. If this environment variable is set, it must point to a valid FLEXlm file.

If a non-null value cannot be found, the following UFM is displayed by the nastran command:

```
*** USER FATAL MESSAGE (nastran.validate_authorize)
    authorize=””     (program default)
      Authorization file pathname, nodename of network license server, or ’demo’
to
      request the demo license.  The environment variable MSC_LICENSE_FILE over-
rides
      the RC files; the command lines overrides the environment variable.

    The keyword shall not be blank or null.
```

If a non-null value is found, but the licensing information is invalid, the following UFM 3060 error message is displayed by MSC/NASTRAN:

```
*** USER FATAL MESSAGE 3060, SUBROUTINE MODEL – OPTION opt NOT IN APPROVED LIST.
     SYSTEM DATE (MM/DD/YY): mm/dd/yy
     SYSTEM MSCID: d (DECIMAL) h (HEXADECIMAL) SYSTEM MODEL NUMBER: m, SYSTEM OS
CODE: c
```

where *opt* is a keyword indicating the specific capability requested. The initial authorization check is for option “NAST”, subsequent checks request specific features as required by your job.

3.2.1 FLEXlm Licensing

> Note: If the FLEXlm HTML documentation has been installed, additional MSC-specific FLEXlm documentation can be viewed using the following URL:
>
> file:*install_dir*/flexlm/htmlman/flexframe.html
>
> Additional FLEXlm documentation can always be found at the following URL:
>
> http://www.globetrotter.com

FLEXlm offers both node-locked licensing and network licensing. With a node-locked license, MSC/NASTRAN can only run on a specified node. With a network license, MSC/NASTRAN can run on any node with a TCP/IP connection to the license server.

FLEXlm offers two types of node-locked licensing: counted and uncounted licenses. An uncounted license does not require a license server, is the easiest to install and maintain, and offers unlimited concurrent MSC/NASTRAN jobs. A counted license requires a license server on the MSC/NASTRAN platform and limits the number of concurrent MSC/NASTRAN jobs. In either case you will need to determine the MSC ID of the system running MSC/NASTRAN.

A FLEXlm network license always requires a license server that can communicate with every computer that will run MSC/NASTRAN.

Installing a FLEXlm “license.dat” File

If you are using a counted node-locked license or a network license, an MSC ID is required for the computer that will run the FLEXlm license server. This ID is obtained with the command:

```
install_dir/bin/msc705 id
```

This command will output a line similar to

```
Please wait...
MSC ID: n
```

where *n* is a hexadecimal number.

A FLEXlm license can be installed during the initial installation or any time thereafter. The following command is used to install a “license.dat” file after installation:

```
install_dir/bin/msc705 flex license.dat
```

where *license.dat* is the new license file. This file may be an email message that has been saved to disk but still contains the email headers. If an existing license file is found at

```
install_dir/flexlm/licenses/license.dat
```

the file will be versioned. In addition, alternate port number and options information from the “SERVER” and “DAEMON” lines will be copied to the new file.

Using FLEXlm Licensing

The “authorize” keyword is used to indicate the authorization source. The value can be any of the following:

Value	Comments
@node	The specified node is the license server using the default port number 1700.
port@node	The specified node is the license server using an alternate port number.
filename	The specified file is used for authorization. This file may contain FLEXlm licensing information for either a node-locked or network license.
value:value:...	A list of alternate FLEXlm licensing files or license server nodes. Note, a port number must be specified if a license server is identified in a list.

Examples are:

```
auth=install_dir/flexlm/licenses/license.dat
```

The FLEXlm license file will be used. If this license file contains a “SERVER” line, the specified server node will be used. If not, the file will be treated as a FLEXlm node-lock license file.

```
auth=@troll
```

Node “troll” is a FLEXlm license server using the default port number.

```
auth=1700@troll
```

Node “troll” is a FLEXlm license server using the specified port number.

```
auth=1700@banana1:1700@banana2
```

Two alternate network license servers, “banana1” and “banana2”, will be used to provide network licensing services.

3.2.2 Node-locked Authorization Codes

The node-locked authorization system in MSC/NASTRAN Version 70.5 is unchanged from earlier versions.

Number of Users Limit (All Systems but Cray and NEC)

Node-locked licensing for MSC/NASTRAN now enforces a limit on the number of users (number of seats) concurrently running MSC/NASTRAN on a single computer. This limit is defined by your contract with MSC and is encoded in the node-lock authorization code. If the maximum authorized number of jobs is already executing when a job starts, the job can wait until a seat becomes available. This wait is controlled by the “authqueue” keyword (see Section B.1 in Appendix B). The default is 20, i.e., a job will wait up to twenty minutes for a seat to become available.

If a seat does not become available within the wait time, the job will terminate with the following message in the LOG file:

```
NUSR: Limit of n concurrent jobs has been reached
      and queue wait period of authqueue minutes has expired.
      The following jobs are currently active:
      No. Username  Status     PID   Start
      --- --------  --------  -----  ------------------------
      1.  user      active    pid    start_time
      .
      .
      .
      n   usern     queued    pid    start_time
```

where *n* is the maximum authorized number of concurrent jobs; *authqueue* is the wait time set by the “authqueue” keyword; *user*, *pid*, and *start_time* are the user names, process IDs, and starting times, respectively, of all MSC/NASTRAN jobs currently running or waiting to run on this computer.

> Note: When a job is waiting for a seat to become available, the job is consuming computer resources such as memory, swap file space, disk space, etc. Too many jobs waiting for seats could have a severe impact on the system.

Installing a Node-locked Authorization Code

An MSC ID is required for the computer that will run MSC/NASTRAN. This ID is obtained with the command:

```
install_dir/bin/msc705 id
```

This command will output a line similar to

```
Please wait...
MSC ID: n
```

where *n* is a hexadecimal number.

A node-locked authorization code is installed using a text editor. Any number of authorization codes for any number of computers can be present in one file. The standard node-locked authorization code file is

```
install_dir/conf/authorize.dat
```

3.3 Activating MSC Accounting

MSC provides a simple accounting package that collects usage information from each job and saves a summary of the job in the accounting directory, i.e., *install_dir*/acct. To activate MSC accounting, use the keyword “acct=yes” in any RC file or on the command line. Placing the keyword in the system wide RC file, *install_dir*/conf/nast705rc, will enable accounting for all jobs.

Note: Users must have read, write, and execute privileges to *install_dir*/acct.

Instructions for generating usage summaries from the MSC accounting data are provided in Section 6.3. Contact your MSC representative to determine if any usage data must be reported to MSC.

3.3.1 Enabling Account ID and Accounting Data

The “acid” and “acdata” keywords are supported by the nastran command to provide hooks for a site to track additional accounting data. The “acid” keyword may be used to specify an account ID. The “acdata” keyword may be used to specify any additional accounting data needed by a site.

These keywords are activated as follows:

1. Activate accounting by putting the line “acct=yes” in the command initialization file or a system RC file.

2. The account validation keyword, “acvalid”, can be used to validate the “acid” keyword. If “acvalid” is not defined in the command initialization file, MSC/NASTRAN will not require the “acid” keyword. If the “acvalid” keyword is defined, MSC/NASTRAN will require a valid “acid”. See Section 3.3.2 for a complete description of this capability.

3.3.2 Enabling Account ID Validation

Account ID validation is enabled by defining a non-null value for the “acvalid” keyword in the command initialization file “install_dir/bin/nast705.ini” (see Section 3.5 for additional information on the command initialization file). There are two types of account ID validation available. The nastran command’s built-in regular expression facility can be used if the account ID can be described by a regular expression (see Section 3.13). Otherwise an external program can be used.

Validating an Account ID with a Regular Expression

To use a regular expression, the first character of the “acvalid” value must be an “f” or a “w” and the remainder of the value is the regular expression. The “f” indicates that an “acid” value that is not matched by the regular expression is a fatal error, while “w” indicates that an unmatched value is only a warning. Note, the regular expression is always constrained to match the entire account ID string.

For the following examples, assume “acvalid=F” was set in the initialization file, “install_dir/bin/nast705.ini”, and an account ID is not set in an RC file.

```
msc705 nastran example
```

This job will fail with a message indicating an account ID is required.

```
msc705 nastran example acid=123
```

This job will be permitted to start. Since a regular expression was not defined, any non-null acount ID is valid.

For the following examples, assume “acvalid=W” is set in the initialization file and an account ID is not set in an RC file.

```
msc705 nastran example
```

A warning message will be issued indicating an account ID is required and the job will be permitted to start.

```
msc705 nastran example acid=123
```

This job will be permitted to start. Since a regular expression was not defined, any non-null account ID is valid.

For the following examples, assume the following line is set in the command initialization file and an account ID is not set in an RC file:

```
acvalid=f[A–Za–z][0–9]\{6\}
```

This regular expression requires the account ID to be composed of a single letter followed by six digits.

```
msc705 nastran example
```

This job will fail with a message indicating an account ID is required.

```
msc705 nastran example acid=123
```

This job will fail with a message indicating the account ID is not valid.

```
msc705 nastran example acid=Z123456
```

This job will be permitted to start.

Validating an Account ID with an External Program

To use an external program, the first character of the “acvalid” value must be a left quote, “‘” and the remainder of the value is a simple UNIX command to execute the external program. The command may include keyword references (see Section 3.12) but must not include pipes or conditional execution tokens. The program must examine the account ID and write zero or more lines to stdout indicating the result of the examination. A null stdout indicates a valid account ID.

The non-null stdout is composed of two optional parts. The first part is indicated by an equal sign “=” as the first non-blank character. If this is found, the next token is taken as a replacement account ID. With this, the external program can replace the user’s account ID. The second part is indicated by an “f” or “w” character. If either of these two characters is present, the remainder of the line and all remaining lines of stdout are taken as the body of an error message to be issued to the user. If no message text is provided, a generic message is written.

Before we discuss the external program, let’s first consider some examples of the external program’s stdout.

```
=Z123456
```

This job will be permitted to start after the account ID is replaced with “Z123456”.

```
f
The account ID is not valid.
See your Program Manager for a valid account ID.
```

This job will fail with the above message.

```
= Z123456
w
The account ID is not valid, it has been replaced by the
standard overhead charge.  See your Program Manager for a
valid account ID.
```

This job will be permitted to start after the account ID is replaced with “Z123456” and the above message is issued.

Now, let’s examine a sample external program.

```
#!/bin/ksh
#
# Sample site–defined account validation program.
#
# usage: /usr/local/bin/checkac _account_id_
#
# If no argument is specified, issue a warning and use the default
# account ID of 123456.  Note the two echo statements. The first
# supplies a warning message, the second supplies a replacement value.
#
if test –z ”$1”; then
    echo ”= Z123456”
    echo ”w”
    echo ”The account ID is not valid, it has been replaced by the
standard”
    echo ”overhead charge.  See your Program Manager for a valid account
ID.”
#
# If the file containing the list of valid account ID’s is missing,
# report a fatal error.
#
elif test ! –s /usr/local/data/account.data ; then
    echo ”f”
    echo ”Accounting data file is missing.  See SYSADMIN”
#
# The file is organized with one account ID per line.
# Make sure the account ID is in the list.
#
elif fgrep –ix $1 /home/dnl/n705/account.data > /dev/null 2>&1 ; then
    echo ”= $1”
#
# If we get here, the account is invalid.
#
else
    echo ”f”
    echo ”The account ID is not valid.”
    echo ”See your Program Manager for a valid account ID.”
fi
```

Finally, an “acvalid” value that will activate the program is

```
acvalid=‘/usr/local/bin/checkac %acid%‘
```

This keyword may appear in the command initialization file, "*install_dir*/bin/nast705.ini", or any RC rile.

3.3.3 Securing the Accounting Files

Some sites may need to secure the accounting files to prevent unauthorized modification or inspection of the accounting data. This can be done by making the accounting logging program, *install_dir*/msc705/*arch*/acct, a "set uid" program. The following commands may be executed (as root):

```
chown secure-user install_dir/msc705/arch/acct
chgrp secure-group install_dir/msc705/arch/acct
chmod ug+s install_dir/msc705/*/acct
chmod o= install_dir/acct
chmod o= install_dir/acct/*
```

where *secure-user* is the userid that will own the files and *secure-group* is the groupid of the group that will own the files.

3.4 Determining System Limits

UNIX resource limits can have a profound impact on the type and size of analyses that can be performed with MSC/NASTRAN. Resource limits that are too low can result in excessive time to complete a job or even cause a fatal error. The current resource limits on the local computer are obtained with the following command:

```
msc705 nastran limits
```

The resource limits on a remote computer that has MSC/NASTRAN installed are obtained with:

```
msc705 nastran limits node=remote_computer
```

You should also lock the accounting keywords with the following lines in the system RC file *install-dir*/conf/nast705rc

```
lock=acct
lock=accmd
lock=acvalid
```

Notes: 1. The limits can vary among users and computers. If a queuing system such as NQS or NQE is installed, different limits may also be found on the various queues.

2. The output from the limits special function may specify “unlimited.” In this context, “unlimited” means there is no limit on your use of a resource that is less than those architectural limits imposed by the processor or the operating system.

   a. For example, an unlimited real address space on a Cray system is always limited by the physical memory on the machine; on an IBM RISC System/6000, an unlimited virtual memory address space is limited by the smaller of the 2 gigabyte address space or the swap space configured in the operating system; on a Digital Alpha, an unlimited virtual memory address space is only limited by the swap space configured in the operating system, i.e., it may exceed 2 gigabytes.

   b. A more important interpretation of unlimited occurs when describing file size limitations. Table 4-5 lists those systems that support large files, i.e., in excess of 2 gigabytes. In this case, unlimited can mean 2**32-1 (4 294 967 295) bytes if large files are not supported, or upwards of 2**64-1 (18 446 744 073 709 551 615) bytes if large files are supported.

Sample output from this command for the various computers used to port MSC/NASTRAN is shown below:

3.4.1 Cray C90, T90

```
Current resource limits:
 CPU time:                    1000 seconds
 Real address space:           128 megabytes
 Processes:             35184372088831
 Number of open files:          64
 SDS size:                    4096 megabytes
 Filesystem space:     137438953471 megabytes
 Core dump file size:           78 megabytes
```

3.4.2 Cray IEEE T90

```
Current resource limits:
 CPU time:                    1000 seconds
 Real address space:           128 megabytes
 Processes:             35184372088831
 Number of open files:          64
 SDS size:                    4096 megabytes
 Filesystem space:     137438953471 megabytes
 Core dump file size:           78 megabytes
```

3.4.3 Cray J90, Y-MP

```
Current resource limits:
 CPU time:               unlimited
 Real address space:     unlimited
 Processes:                    100
 Number of open files:          64
 SDS size:                       0 megabytes
 Filesystem space:      unlimited
 Core dump file size:    unlimited
```

3.4.4 Digital Alpha UNIX

```
Current resource limits:
 CPU time:               unlimited
 Virtual address space:       1024 megabytes
 Working set size:          122072 kilobytes
 Data segment size:         131072 kilobytes
 Stack size:                  2048 kilobytes
 Number of open files:        4096
 File size:              unlimited
 Core dump file size:    unlimited
```

3.4.5 Fujitsu

```
Current resource limits:
 CPU time:               unlimited
 Virtual address space:  unlimited
 Data segment size:        2097151 kilobytes
 Stack size:                  8192 kilobytes
 Number of open files:         512
 File size:              unlimited
 Core dump file size:    unlimited
```

3.4.6 HP 9000

```
Current resource limits:
 CPU time:               unlimited
 Virtual address space:  unlimited
 Working set size:       unlimited
 Data segment size:        1048576 kilobytes
 Stack size:                  8192 kilobytes
 Number of open files:          60
 File size:              unlimited
 Core dump file size:         2047 megabytes
```

3.4.7 HP Exemplar

```
Current resource limits:
 Virtual address space:       1536 megabytes
 Number of open files:         256
 File size:                   2047 megabytes
```

3.4.8 IBM

```
Current resource limits:
 CPU time:               unlimited
 Working set size:           32768 kilobytes
 Data segment size:         131072 kilobytes
 Stack size:                 32768 kilobytes
 Number of open files:        2000
 File size:                   1023 megabytes
 Core dump file size:    unlimited
```

3.4.9 NEC

```
Current resource limits:
 CPU time:               unlimited
 Tasks:                         16
 Virtual address space:       1900 megabytes
 Data segment size:        1945600 kilobytes
 Stack size:               1945600 kilobytes
 Number of open files:         256
 File size:                   8192 megabytes
 File system space:      unlimited
 Temporary file space:           0 megabytes
 Core dump file size:    unlimited
```

3.4.10Silicon Graphics R4K, R5K

```
Current resource limits:
 CPU time:               unlimited
 Virtual address space:        512 megabytes
 Working set size:          122948 kilobytes
 Data segment size:         524288 kilobytes
 Stack size:                 65536 kilobytes
 Number of open files:         200
 File size:              unlimited
 Core dump file size:    unlimited
```

3.4.11 Silicon Graphics R8K, R10K

```
Current resource limits:
 CPU time:               unlimited
 Virtual address space:  unlimited
 Working set size:          508016 kilobytes
 Data segment size:      unlimited
 Stack size:                65536 kilobytes
 Number of open files:        200
 File size:              unlimited
 Core dump file size:    unlimited
```

3.4.12 Sun

```
Current resource limits:
 CPU time:               unlimited
 Virtual address space:  unlimited
 Data segment size:        2097148 kilobytes
 Stack size:                  8192 kilobytes
 Number of open files:          64
 File size:              unlimited
 Core dump file size:    unlimited
```

3.5 Customizing the Command Initialization File

The command initialization file, *install_dir*/bin/nast705.ini, is used to define keywords that are to be set whenever the nastran command is executed. Typical keywords defined in this file include the installation base directory and the version of MSC/NASTRAN.

3.5.1 Setting Command Initialization File Keywords

The following table lists the keywords that are generally set in the command initialization file.

Keyword	Purpose
acvalid	Activates account ID validation (see Section 3.3.1).
MSC_BASE	Defines the installation base directory. Normally this is defined as an environment variable by the *install_dir*/bin/msc705 architecture selection script.
version	Specifies the version of MSC/NASTRAN to be run.

3.6 Customizing the Runtime Configuration Files

MSC/NASTRAN keywords (in Appendix B) and NASTRAN statements (in Section 5.2) can be placed in a runtime configuration file (RC file). MSC/NASTRAN uses the following files:

- The system RC file is *install_dir*/conf/nast705rc. This file should be used to define parameters that are applied to all MSC/NASTRAN jobs using this installation structure.
- The architecture RC file is *install_dir*/conf/*arch*/nast705rc. This file should be used to define parameters that are applied to all MSC/NASTRAN jobs using this architecture.
- The node RC file is *install_dir*/conf/net/*nodename*/nast705rc. This file should be used to define parameters that are applied to all MSC/NASTRAN jobs running on this node.
- The user RC file is ${HOME}/.nast705rc. This file should be used to define parameters that are applied to all MSC/NASTRAN jobs run by an individual user.
- The local RC file is .nast705rc, in the same directory as the input data file. If the “rcf” keyword is used, this local RC file is ignored. This file should be used to define parameters that are applied to all MSC/NASTRAN jobs run in the input data file directory.

Notes: 1. The tilde “~” character is not recognized in RC files.

2. Environment variables are only recognized in RC files within the context of a logical symbol (see Section 4.1.2).

The order of precedence for duplicated entries is as follows (with number 1 representing the highest precedence):

1. NASTRAN statements in the input file.
2. Keywords on the command line.
3. Local RC file.
4. User RC file.
5. Node RC file.
6. Architecture RC file.
7. System RC file.

An example of an RC file is shown below:

```
NASTRAN SYSTEM(20)=0     $ ALWAYS PRINT BEGIN,END
NASTRAN BUFFSIZE=8193    $ CHANGE DEFAULT BUFFSIZE
mem=3m                   $ run with 3 145 728 words
```

3.6.1 Setting RC File Keywords

Any of the command line keywords can be set in any of the RC files. The following lists those keywords that are generally set in either the system, architecture, or node RC files:

Keyword	Preferred RC file	Purpose
accmd	System	Command line to invoke accounting logger program.
acct	System	Indicates solution accounting is to be performed.
acvalid	System	Enables account ID (acid) validation.
authorize	System	Defines the node-lock authorization code file or enables network licensing.
lock	Any	Allows a site to prevent further changes to a keyword value.
memory	Architecture	Provides a default memory allocation.
ncmd	Architecture	Specifies the notify command when “notify=yes” is set.
news	System	Controls the display of the news file at the beginning of the F06. The news file is *install_dir*/msc705/nast/news.txt.
pcmd	Architecture	Specifies the print command when “prt=yes” is set.
post	Architecture	Specifies UNIX commands to be run after each job is completed.
ppcdelta	Architecture	Sets the value that is subtracted from the “CPU” keyword value to determine the NQS per-process CPU time limit.
ppmdelta	Architecture	Sets the value that is added to the “memory” keyword value to determine the NQS per-process memory limit.
pre	Architecture	Specifies UNIX commands to be run before each jobs begins.
prmdelta	Architecture	Sets the value that is added to the “ppm” value to determine the NQS per-request (per-job) memory limit.

Keyword	Preferred RC file	Purpose
qoption	Architecture	Defines a string of additional queuing options to be set in the queue submittal command.
real	Node	Sets “REAL” parameter to limit memory usage.
scratch	Any	Sets the default job status as scratch or permanent. This keyword is overridden by the “dbs” keyword.
sdirectory	Node	Sets a default scratch directory.
submit	Architecture	Defines queues and their associated submittal commands.
sys*x*	Architecture	Sets system cells.

3.7 Customizing the News File

MSC delivers a news file (*install_dir*/msc705/nast/news.txt) that briefly describes important new features of the release. This news file can also be used by a site to distribute information to the users of MSC/NASTRAN.

There are two ways the news file can be viewed. The most common way is by specifying “news=yes” or “news=auto” on the command line or in an RC file. This specification will cause the news file to be printed in the F06 file just after the title page block. The other method is by using the news special function

```
msc705 nastran news
```

This will cause the news files to be displayed on the screen.

3.8 Customizing the Message Catalog

MSC/NASTRAN uses a message catalog for many messages displayed in the F06 file. The standard message catalog source file is

```
install_dir/msc705/util/analysis.txt
```

This file may be modified to meet the needs of a site or a user.

Once the changes have been made, a message catalog is generated using the command

```
msc705 msgcmp myfile
```

where "*myfile*.txt" is the message catalog source file. This command will generate a message catalog in the current directory with the name "*myfile*.msg". The message catalog can be tested using the command

```
msc705 nastran msgcat=myfile.msg other_nastran_keywords
```

Once the message catalog has been validated, it may be installed with the command

```
cp myfile.msg install_dir/msc705/arch/analysis.msg
```

where *install_dir* is the installation base directory and *arch* is the architecture of the system using the message catalog. You will generally need to be root to install this file.

Note: Message catalogs are machine dependent. Table 6-1, "Binary File Compatibility" identifies the systems that are binary compatible; binary compatible systems can use multiple copies of the same message file.

3.9 Defining a Computer Model Name and CONFIG Number

If the nastran command cannot identify a computer, the following message will be written to the terminal before the MSC/NASTRAN job begins:

```
*** SYSTEM WARNING MESSAGE (validate_local_keywords)
    s.config=0     (program default)
      Default CONFIG value.

    A config number for this computer could not be
determined.  Defining
    this computer in the model file,
    install_dir/conf/arch/model.dat, using rawid=raw_id; or
    defining <config> in an RC file may correct this
problem.
```

There are two possible solutions. The preferred solution is to create the file *install_dir*/conf/*arch*/model.dat with the model name and configuration number of the computer. This file contains lines of the form:

```
model_name, processor_type, raw_id, config_number
```

where:

model_name	The name of the computer. This string should be enclosed in quote marks if it contains spaces or commas.
proc	The filename suffix of the alternate executable. This value is set to null to select the standard executable. The “system” special function reports this name.
raw_id	The “raw_id” value reported in the above message text or by the “system” special function.
config_number	The CONFIG number used to select the timing constants. If this value is null, the raw id is used as the CONFIG number.

Note: In prior releases, a script was needed on IBM systems to return this information. That script has been replaced by this file.

Any values in this table will override the default values built into the nastran command.

An alternative solution to creating this file is to set the “config” keyword in the node RC file, i.e., “*install_dir*/conf/net/*node*/nast705rc”. Note, however, this will not set a model name.

3.10 Generating a Timing Block for a New Computer

MSC/NASTRAN uses timing constants to determine the fastest algorithm or “method” to perform certain numerically intensive operations. Timing constants are installed by MSC for a variety of computers. If constants are not installed for your particular computer, MSC/NASTRAN will select default timing constants and display the following warning message:

```
*** USER WARNING MESSAGE 6080 (TMALOC)

THE TIMING CONSTANTS DATA BLOCK TIMEBLK NOT FOUND ON THE DELIVERY DATABASE FOR:

MACHINE = 5 CONFIG = 56 OPERASYS = 3 OPERALEV = 7 SUBMODEL = 1
LOADING DEFAULT TIMING CONSTANTS DATA BLOCK FOR:
MACHINE = 5 CONFIG = 56 OPERASYS = 3 OPERALEV = 5 SUBMODEL = 1

MODULE TIMING ESTIMATES INACCURATE AND MAY CAUSE INEFFICIENT JOB EXECUTION
```

Ignoring this message may result in excessive runtimes. Proper timing constants for a specific computer may be generated and installed by performing a computer run that measures the timing constants of the computer and stores them in the delivery database.

The following steps will add timing constants for your computer to the delivery database:

1. Determine the MSC architecture name of your system by consulting Table 3-1 or executing the command

   ```
   msc705 nastran system
   ```

2. Change the working directory to the architecture directory, i.e., *install_dir*/msc705/*arch*, where *arch* is the architecture name of your computer determined in step 1 above.

   ```
   cd install_dir/msc705/arch
   ```

3. The timing data is generated by running *install_dir*/msc705/nast/del/gentim2.dat. The value of the Bulk Data parameter “PARAM” is set to 2. (Note that all the lines in the file are not displayed.)

```
NASTRAN MESH SYSTEM(124)=-1
PROJ LTC LOAD TIMING CONSTANTS
INIT MASTER,LOGICAL=(MASTERA(5000))
INIT SCRATCH(NOMEM)
TIME 2000
SOL GENTIMS
CEND
BEGIN BULK
PARAM,PARAM,2
.
.
.
```

   In general, the larger the value of “PARAM”, the longer the gentim2 job runs and the more accurate the timing results. If gentim2 runs for more than one hour, you may choose to set the value of “PARAM” to 1, this will shorten the elapsed time of the gentim2 job.

4. Copy the Structured Solution Sequence files to be modified by the gentim2 run with the commands:

   ```
   cp SSS.MASTERA gentim2.MASTERA
   cp SSS.MSCSOU  gentim2.MSCSOU
   cp SSS.MSCOBJ  gentim2.MSCOBJ
   ```

5. Issue the command

   ```
   msc705 nastran DELDIR:gentim2 batch=no old=yes scratch=no
   ```

6. If there are no errors, replace the old DBsets with the new DBsets created by the gentim2 run. Do this with the following commands:

```
mv gentim2.MASTERA SSS.MASTERA
mv gentim2.MSCOBJ  SSS.MSCOBJ
mv gentim2.MSCSOU  SSS.MSCSOU
```

3.11 Customizing Queue Commands for NQS or NQE

The nastran command runs an MSC/NASTRAN job by validating the command line and RC files, generating a “job script” that will run the MSC/NASTRAN executable, and running that script. When the “queue” keyword is specified, the corresponding “submit” keyword defines the command used to run the job script. The “submit” keyword, only specified in RC files, consists of an optional queue list, followed by the command definition for the specified queues as shown below:

```
submit=queue_list=command_definition
submit=command_definition
```

When specified, the *queue_list* contains one or more “queue” names separated by commas. If a queue list is not supplied, the *command_definition* applies to all queues.

The command definition section of the “submit” keyword value defines the command used to run a job when a “queue” keyword is supplied that matches a queue name in the queuelist. The command definition section can contain keyword names enclosed in percent “%” signs that are replaced with the value of the keyword before the command is run.

Note: When defining queue commands, it may be useful to build the job script but not actually execute it. Set the MSC_NOEXE environment variable, e.g.,

```
MSC_NOEXE=1; export MSC_NOEXE   # sh, ksh
setenv MSC_NOEXE 1              # csh

msc705 nastran myjob
```

The examples presented below are only intended to illustrate the “submit”, “qopt” and “queue” keywords. The examples may not work with your queuing system.

Consider the following example:

```
submit=small,medium,large=qsub –q %queue% –x –eo –s /bin/sh
%job%
```

In this example, the “qsub” command is used to run a job when “queue=small”, “queue=medium”, or “queue=large” is specified.

Any keyword used by the nastran command is available. The most common keywords used in the “submit” keyword’s command definition are:

Keyword	Value
after	Value specified with the “after” keyword.
cputime	Value specified with the “cputime” keyword.
job	Name of the job script file built by the nastran command.
log	Name of the LOG file.
ppc	Value of “ppc”, i.e, (%cputime% - %ppcdelta%).
ppm	Value of “ppm”, i.e., (%memory% + %ppmdelta%).
prm	Value of “prm”, i.e., (%ppm% + %prmdelta%).
qoption	This can be used to define any option not directly represented by the other variables or not explicitly included in the command definition.
username	User name

Using the previous example, the command

```
msc705 nastran example queue=small
```

runs the job script using the command:

```
qsub –q small –x –eo –s /bin/sh example.J12345
```

The %queue% keyword reference is replaced by the specified queue, and the %job% keyword reference is replaced by the name of the execution script.

Keyword references can also contain conditional text that is included only if the value of the keyword is not null, or matches (does not match) a regular expression (a complete description of the keyword reference syntax is described in Section 3.12.1). To check for a nonnull value, use the form

```
%kwd:condtext%
```

where *kwd* is the name of the keyword and *condtext* is the conditional text to be included. If the value of the keyword is null, the keyword reference is removed from the command. If the value of the keyword is not null, the keyword reference is replaced with the contents of *condtext*. Within *condtext*, the value of the keyword is represented by an open-close brace pair “{ }”.

For example:

```
submit=s=qsub –q %queue% %after:–a {}% –x –s /bin/sh %job%
```

In this example, the “aft” keyword is references with conditional text. Using this example, the command

```
msc705 nastran example queue=s after=10:00
```

runs the job script using the following qsub command:

```
qsub –q s –a 10:00 –x –s /bin/sh example.J12345
```

Using the same “submit” keyword, the command

```
msc705 nastran example queue=s
```

runs the job script using the following command:

```
qsub –q s –x –s /bin/sh example.J12345
```

In this case, the “after” keyword was not specified and the entire contents of the %after% keyword reference was removed from the qsub command line.

As a final example, the following “submit” commands are used by MSC on the Cray J90 porting system:

```
submit=short,short2=qsub –q %queue% –x –eo \
     –lf 5000Mb –lF 5000MB \
     %ppc:–lt {}% %CPU:–lT {}% –lm %ppm% –lM %prm% –s
/bin/sh %job%
submit=medium,medium2,long,batch=qsub –q %queue% –x –eo \
     –lf 9000Mb –lF 9000MB \
     %ppc:–lt {}% %CPU:–lT {}% –lm %ppm% –lM %prm% –s
/bin/sh %job%
```

Note: Although these two entries are shown on several lines, each must be entered as a single lin in the RC file.

3.11.1 Special Queues

When the “queue” keyword is not specified, the following three special queues are used:

Keyword	Queue Name	Command Definition
after	-aft	| at %after%
batch=yes	-bg	%job%
batch=no	-fg	%job%

Notes: 1. If the first character of the command is the UNIX pipe character, “|”, the contents of job script will be piped into the command.

2. The command for the “-bg” queue is always executed in the background; the “-fg” and “-aft” commands are always executed in the foreground.

Changing the command definitions of these queues (using the “submit” keyword) will change the way the nastran command runs a job under the “after” and “batch” keywords.

3.12 Customizing the Script Templates

The nastran command relies on script templates to construct the job script that is built for every MSC/NASTRAN job. Two templates are provided: “*install_dir*/bin/nast705.lcl” is used for jobs run on the local system, and “*install_dir*/bin/nast705.rmt” is used for jobs run on a remote system using the “node” keyword. The templates provided by MSC support all versions of MSC/NASTRAN since Version 68.0 for all UNIX platforms. These templates may be modified to suit your needs.

Note: When customizing the script templates, it may be useful to build the job script but not actually execute it. Set the MSC_NOEXE environment variable, e.g.,

```
MSC_NOEXE=1; export MSC_NOEXE   # sh, ksh
setenv MSC_NOEXE 1              # csh

msc705 nastran myjob
```

3.12.1 Keyword Reference Syntax

The script templates use the keyword reference syntax that was partially introduced in Section 3.11.

Syntax	Value	Side effects
%%	%	
%*keyword*%	*Value of keyword.*	
%*keyword*:*condtext*%	*condtext*	
%*keyword*=*re*%	*Value of the parenthetic expression if specified in the re, otherwise the string matched by the re.*	
%*keyword*=*re*:*condtext*%	*condtext if re is matched.*	
%*keyword*!*re*:*condtext*%	*condtext if re is not matched.*	
%*keyword*:%		Kill remainder of line if *keyword* has null value. In a case construct, the default case.
%*keyword*=*re*:%		Kill remainder of line if *re* matches.
%*keyword*!*re*:%		Kill remainder of line if *re* does not match.
%*keyword*?:%		Start of case construct. See Section 3.12.2.

3.12.2 Keyword Reference Examples

The keyword reference syntax is described using the following examples from *install_dir*/bin/nast705.lcl.

Unconditional Keyword Substitution

```
MSC_BASE=%MSC_BASE%; export MSC_BASE
```

The keyword reference %MSC_BASE% will be replaced by the value of the “MSC_BASE” keyword.

```
DBSDIR=%dbs=\(.*\)/%; export DBSDIR
```

The keyword reference %dbs=\(.*\)/% will be replaced with the value of the parenthetic regular expression. For example, given the keyword value “onedir/anotherdir/myfile”, the parenthetic expression is “onedir/anotherdir”, and the substituted line would read:

```
DBSDIR=onedir/anotherdir; export DBSDIR
```

Conditional Keyword Substitution

```
%sysfield:SYSFIELD={}%
```

The keyword reference %sysfield:SYSFIELD={}% will be replaced by the string “SYSFIELD=*keyword-value*” if and only if the keyword is not null.

```
%dcmd=dbx:run%
```

The keyword reference %dcmd=dbx:run% will be replaced by “run” if and only if “dcmd=dbx” was specified. If the equal sign in the keyword reference was replaced by an exclamation mark, i.e., %dcmd!dbx:run%, then the keyword reference will be replaced by “run” if and only if “dcmd” was set to a nonnull value not equal to “dbx”.

Conditional Inclusion

```
%MSC_ARCH=aix:%startdate=date +%%a %%h %%d %%H:%%M:%%S %%Z
%%Y
%MSC_ARCH!aix:%startdate=date
```

Conditional inclusion is indicated by a null conditional text string; i.e., the colon is immediately followed by a percent sign. This capability is generally used with a regular expression to include the remainder of the line if a keyword value matches or does not match a regular expression. In the first line, the remainder of the line will be included if the “MSC_ARCH” keyword contains the string “aix” while the remainder of the second line will be included if “MSC_ARCH” does not contain the string “aix”. More than one conditional inclusion keyword reference can be used on a line to create more complex tests.

```
%prt=y:%%pdel=y:%/bin/rm %out%.f04 %out%.f06 %out%.log
```

The “rm” command will included if and only if “prt=yes” and “pdel=yes”.

A “case” structure is specified as follows:

```
...%s.model?:%
...%s.model=IP.$:%  SGI_ISA=mips1; export SGI_ISA
...%s.model=IP12:%  SGI_ISA=mips1; export SGI_ISA
...%s.model=IP15:%  SGI_ISA=mips1; export SGI_ISA
...%s.model=:%      SGI_ISA=mips2; export SGI_ISA
```

This sequence will result in the line

```
SGI_ISA=mips1
```

if “s.model” is “IP” followed by a single character (using the second line), or “IP12” (using the third line), or “IP15” (using the fourth line), otherwise

```
SGI_ISA=mips2
```

will be generated using the last line. Case constructs can be nested, but a keyword may only be active in one case at a time.

3.13 Using Regular Expressions

The regular expression syntax supported by the nastran command is compatible with the standard ed(1) regular expression syntax with the exception that only one parenthetic expression is permitted. The syntax follows.

One-character Regular Expressions

- Any character, except for the special characters listed below, is a one-character regular expression that matches itself.
- A backslash, “\”, followed by any special character is a one-character regular expression that matches the special character itself. The special characters are: period, “.”, asterisk, “*”, and backslash “\”, which are always special except when they appear within brackets; circumflex, “^”, which is special at the beginning of a regular expression or when it immediately follows the left bracket of a bracketed expression; and dollar sign “$”, which is special at the end of a regular expression.
- A period, “.”, is a one-character regular expression that matches any character.

- A nonempty string of characters enclosed within brackets, “[” and “]”, is a one-character regular expression that matches one character in that string. If, however, the first character of the string is a circumflex, “^”, the one-character regular expression matches any character except the characters in the string. The circumflex has this special meaning only if it occurs first in the string. The dash, “-”, may be used to indicate a range of consecutive characters. The dash loses this special meaning if it occurs first (after an initial circumflex, if any) or last in the string. The right square bracket, “]”, does not terminate such a string when it is the first character within it (after an initial circumflex, if any).

Regular Expressions

- A one-character regular expression is a regular expression that matches whatever the one-character regular expression matches.
- A one-character regular expression followed by an asterisk, “*”, is a regular expression that matches zero or more occurrences of the one-character regular expression. If there is any choice, the longest leftmost string that permits a match is chosen.
- A one-character regular expression followed by “\{*m*\}”, “\{*m*,\}”, or “\{*m*,*n*\}” is a regular expression that matches a ranges of occurrences of the one-character regular expression. The values of *m* and *n* must satisfy $0 \leq m \leq n \leq 254$; “\{*m*\}” exactly matches *m* occurrences; “\{*m*,\}” matches at least *m* occurrences; “\{*m*,*n*\}” matches any number of occurrences between *m* and *n* inclusive.
- A concatenation of regular expressions is a regular expression that matches the concatenation of the strings matched by each component of the regular expression.
- A regular expression enclosed between the character sequences “\(” and “\)” defines a parenthetic expression that matches whatever the unadorned regular expression matches. Only one parenthetic expression may be specified.
- The expression “\1” matches the same string of characters as was matched by the parenthetic expression earlier in the regular expression.

Constraining Regular Expressions

- A circumflex, “^”, at the beginning of an entire regular expression constrains the regular expression to match an initial segment of a string.

- A dollar sign, “$”, at the end of an entire regular expression constrains the regular expression to match a final segment of a string.
- The construction “^*re*$” constrains the regular expression to match the entire string.
- The construction “^$” matches a null string.

CHAPTER 4

HOW TO USE THE BASIC FUNCTIONS OF MSC/NASTRAN

This chapter discusses the nastran command, filenames and logical symbols, the nastran command special functions, the basic keywords, using the test problem libraries, making file assignments, the basics of using databases, and resolving abnormal terminations.

4.1 Using the nastran Command

MSC/NASTRAN jobs are run using the nastran command. The basic format of this command is

```
msc705 nastran input_data_file keywords
```

where *input_data_file* is the name of the file containing the input data and *keywords* is zero or more optional keyword assignments. For example, to run an MSC/NASTRAN job using the data file example1.dat, enter the following command:

```
msc705 nastran example1
```

Various options to the nastran command are available using keywords described in Appendix B. Keyword assignments consist of a keyword, followed by an equal sign, followed by the keyword value, for example:

```
msc705 nastran example1 batch=no
```

Keyword assignments can be specified as command line arguments and included in RC files.

There are two RC files controlled by each user:

- The user RC file is ${HOME}/.nast705rc. This file should be used to define parameters applicable to all MSC/NASTRAN jobs you run.
- The local RC file is .nast705rc, in the same directory as the input data file. If the “rcf” keyword is used, this local RC file is ignored. This file should be used to define parameters applicable to all MSC/NASTRAN jobs contained in the input data file directory.

Notes: 1. The tilde (~) character is not recognized within RC files.

2. Environment variables are only recognized when used in the context of a logical symbol (see Section 4.1.2).

3. When a keyword is specified on the command line, embedded spaces or special characters that are significant to the shell must be enclosed in quote marks; quotes marks should not be used within RC files.

4.1.1 Using File Suffixes

MSC/NASTRAN input and output files use the following suffixes:

Suffix	Type of File	Description of File
.dat	Input	Input Data File
.f04	Output	Execution Summary File
.f06	Output	Output Data File
.log	Output	System Log Output File
.op2	Output	OUTPUT2 File
.pch	Output	Punch File
.plt	Output	Binary Plot File

Notes: 1. If the input file is specified as “example1” and the files “example1.dat” and “example1” both exist, the file “example1.dat” will be chosen. In fact it is *impossible* to use a file named “example1” as the input data file if a file named “example1.dat” exists.

2. The “j.dat” keyword may be used to specify an alternate default suffix for the input data file. For example, “j.dat=.bdf” will change the default suffix to “.bdf”.

When a job is run more than once from the same directory, the previous output files are versioned, or given indices. The indices are integers appended to the filename; the same integer will designate files for the same job. For example,

v2401.f04	v2401.f04.1	v2401.f04.2	v2401.f04.3
v2401.f06	v2401.f06.1	v2401.f06.2	v2401.f06.3

The files listed (according to time of execution from oldest to newest) are:

v2401.f04.1	v2401.f06.1
v2401.f04.2	v2401.f06.2
v2401.f04.3	v2401.f06.3
v2401.f04	v2401.f06

4.1.2 Using Filenames and Logical Symbols

Many of the parameters used by MSC/NASTRAN, including command line arguments, initialization and RC file commands, and statements within MSC/NASTRAN input files, specify filenames. The filenames must follow UNIX filename conventions with the addition that filenames can include a “logical symbol” component, i.e., the filename can be specified in either of the following forms:

filename

logical–symbol:*filename*

Logical symbols provide you with a way of specifying file locations with a convenient shorthand. This feature also allows input files containing filename specifications to be moved between computers without requiring modifications to the input files. Only the logical symbol definitions that specify actual file locations need to be modified.

Only one logical symbol name may be used in a filename specification. This logical symbol must be the initial component of the filename string, and it must be separated from the filename by a colon “:”. If the symbol has a non-null value, the actual filename is created by replacing the symbol name with its value and replacing the colon with a slash; otherwise, both the symbol name and the colon are left as is.

> Note: The logical symbol capability does not support symbol nesting, i.e., a symbol cannot refer to another symbol.

For example, assume that your home RC file, $HOME/.nast705rc, contains the line

```
SYMBOL=DATADIR=/dbs/data
```

and a job is submitted with the command

```
msc705 DATADIR:nastran example1
```

Since MSC/NASTRAN automatically sets the OUTDIR environment variable to the value of the “out” keyword, the filenames

```
'DATADIR:myfile.dat'
'OUTDIR:testdata.info'
```

will reference the files

```
/dbs/data/myfile.dat
./testdata.info
```

respectively. See the description of the symbol keyword in Section B.1 of Appendix B for more information.

Several other symbols are automatically created by the nastran command. These include DELDIR, DEMODIR, TPLDIR, and SSSALTERDIR to access the delivery database source directory, and DEMO, TPL, and SSSALTER libraries, respectively.

4.1.3 Using the Help Facility and Other Special Functions

Several special functions are supported by reserved input data filenames. If these names are specified as the input data file, the nastran command will execute the special function and exit.

> Note: If you need to use one of these reserved names as an actual input filename, you must either prefix the name with a path, e.g., “./news”, or append a suffix, e.g., “news.dat”.

The special functions are invoked as follows:

```
msc705 nastran help
```

This request will display the basic help output. Additional help capabilities are described in the basic help output.

```
msc705 nastran help keyword1 [ keyword2 ... ]
```

This request will display help for the specific keywords listed on the command line.

```
msc705 nastran limits
```

This request will display the current UNIX resource limits.

```
msc705 nastran news
```

This request will display the news file.

```
msc705 nastran system
```

This request will display system information about the current computer.

These requests can be executed on a remote computer that has MSC/NASTRAN installed by also specifying the keyword “node=*nodename*”, for example:

```
msc705 nastran system node=thatnode
```

4.2 Using the Basic Keywords

The following table is a partial list of the basic keywords that may be used on the command line or placed into RC files as appropriate. More advanced keywords are listed in Section 5.1 and a complete list of all keywords and their syntax is listed in Section B.1 of Appendix B.

4.2.1 All Systems

Keyword	Purpose
after	Hold the job until the specified time.
append	Combines the F06, F04, and LOG files into a single file after the jobs completes.
batch	Runs the job in background or foreground.
dbs	Specifies an alternate name for user database files.
memory	Specifies the amount of memory to be used by the job.
old	Renames existing output files with version numbers or deletes existing output files.
out	Specifies an alternate name for output files.
prt	Prints the F06, F04, and LOG output files when the job completes.
rcf	Specifies an alternate name of the local RC file.
scratch	Indicates databases are to be deleted when job completes; saved for data recovery restarts only; or saved when job completes.
sdirectory	Specifies an alternate scratch file directory.
symbol	Define a symbolic name and value.
xmonast	Runs the Motif-based output file monitor.

4.2.2 Queuing Keywords

Note: These capabilities are dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The keywords may not work on your system.

Keyword	Purpose
cputime	Specifies maximum CPU time to be allowed.
queue	Specifies name of queue where the job will be submitted to.

4.3 Determining Resource Requirements

For most models of moderate size (up to 5000 grid points for static analysis), you need not be concerned with resource requirements since the default MSC/NASTRAN parameters allocate sufficient resources. The analysis of larger models may require you to check the resource requirements and the various options that are available to manage memory and disk resources.

There are several tools available to assist you in determining the resource requirements of your job. The simplest tools are Tables 4-1 and 4-2. These tables present gross estimates of the memory and total disk space requirements of static analyses using default parameters with normal output requests. Other solution sequences will generally have greater requirements.

Table 4-1. Estimated Memory Requirements of Static Analyses.

Degrees of Freedom	Memory Requirements	
	Cray and NEC	Others
DOF < 10 000	3 MW	3 MW
10 000 ≤ DOF < 50 000	4 MW	5 MW
50 000 ≤ DOF < 100 000	6 MW	10 MW
100 000 ≤ DOF < 200 000	11 MW	22 MW

Table 4-2. Estimated Total Disk Requirements of Static Analyses.

Degrees of Freedom	Total Disk Space Requirements
DOF < 10 000	90 MB
10 000 ≤ DOF < 50 000	500 MB
50 000 ≤ DOF < 100 000	1 000 MB
100 000 ≤ DOF < 200 000	2 000 MB

More detailed resource estimates can be obtained from the ESTIMATE program, described in Section 6.1. ESTIMATE will read the input data file and estimate the job's memory and disk requirements. The ESTIMATE program is most accurate in predicting the requirements of static analyses that don't have excessive output requests. The memory requirements for normal modes analyses using the Lanczos Method are reasonably accurate; however, the disk requirements are dependent upon the number of modes. This is a value that ESTIMATE does not know. Memory and disk requirements for other solutions are less accurate.

The best estimates of the memory requirements for a job are available in User Information Message 4157, described in Section 5.5.4, but this requires an MSC/NASTRAN run.

4.4 Using the Test Problem Libraries

Three libraries of test problems are delivered with MSC/NASTRAN.

- The demonstration problem library (DEMO, *install_dir*/msc705/nast/demo) contains a selection of MSC/NASTRAN input files that are documented in the *MSC/NASTRAN Demonstration Problem Manual*.
- The test problem library (TPL, *install_dir*/msc705/nast/tpl) contains a general selection of MSC/NASTRAN input files showing examples of most of the MSC/NASTRAN capabilities, in general, these files are not documented.
- The ARCHIVE library (*install_dir*/msc705/nast/misc/archive) contains MSC/NASTRAN input files that are no longer a part of either the DEMO or TPL libraries (these problems may be incompatible with MSC/NASTRAN V70.5 or use capabilities that are no longer supported).

The DEMO and TPL libraries contain demoidx.dat and tplidx.dat respectively. These files contain one-line descriptions of the library members. Also included are files named tplexec and demoexec, which are scripts used to run the problems. For both libraries, the recommended execution procedure is to copy the file to your own

directory and then execute the problem using the instructions in Section 4.1. Note that several of the library files have “INCLUDE” files that should also be copied.

Some example problems contain references to files that are qualified with the following logical symbols:

TPLDIR
DEMODIR
DBSDIR
OUTDIR

Unless they already exist in your environment as environment variables, the logical symbols DEMODIR and TPLDIR point to the DEMO and TPL libraries respectively. DBSDIR and OUTDIR are always based on the “dbs” and “out” keywords respectively.

4.5 Making File Assignments

Using the ASSIGN statement, you can assign physical files used by MSC/NASTRAN to FORTRAN units or DBset files. The ASSIGN statement is documented in the File Management Section (FMS) of the *MSC/NASTRAN Quick Reference Guide*.

4.5.1 ASSIGN Statement for FORTRAN Files

For FORTRAN files, the format of the ASSIGN statement is

```
ASSIGN logical-name=filename, [ STATUS={NEW|OLD|UNKNOWN}
UNIT=u,
        FORM={FORMATTED|UNFORMATTED}  TEMP  DELETE
SYS=sys-spec ]
```

There are no values of the SYS field defined for FORTRAN files on any UNIX computer.

Table 4-3. FORTRAN Files and Their Default Attributes.

Logical Name6	Physical Name	Unit No.	Form	Status	Assignable	Open	Access	Description
SEMTRN	*sdir*/*data*.f01	1	FORMATTED	NEW	NO	YES	SEQ.	Input Data Copy Unit
LNKSWH	*sdir*/*data*.f02	2	UNFOR-MATTED	NEW	NO	YES	SEQ.	Link Switch Unit
MESHFL	*sdir*/*data*.f03	3	FORMATTED	NEW	NO	YES	SEQ.	Input Data Copy Unit
LOGFL	*out*.f04	4	FORMATTED	NEW	NO	YES	SEQ.	Execution Summary Unit
INPUT	*data*.dat	5	FORMATTED	OLD	NO	YES	SEQ.	Input File Unit
PRINT	*out*.f06	6	FORMATTED	NEW	NO	YES	SEQ.	Main Print Output Unit
PUNCH	*out*.pch	7	FORMATTED	NEW	YES	YES	SEQ.	Default Punch Output Unit
	authorize.dat	8	FORMATTED	OLD	NO	YES	SEQ.	Authorization File
INCLD1		9	FORMATTED	OLD	YES	NO	SEQ.	INCLUDE Statement Unit
CNTFL								Unavailable for Use
INPUTT2	REQ	REQ	UNFOR-MATTED*	OLD	YES	NO	SEQ.	INPUTT2 Unit
OUTPUT2	*out*.op2	12	UNFOR-MATTED*	NEW	YES	YES	SEQ.	OUTPUT2 Unit
INPUTT4	REQ	REQ	UNFOR-MATTED	OLD	YES	NO	SEQ.	INPUTT4 Unit
OUTPUT4	REQ	REQ	UNFOR-MATTED†	NEW	YES	NO	SEQ.	OUTPUT4 Unit
PLOT	*out*.plt	14	UNFOR-MATTED	NEW	YES	YES	SEQ.	Plotter Output Unit
DBMIG								Unavailable for Use
DBC	*out*.xdb	40	UNFOR-MATTED	NEW	YES	YES	DIRECT	Database Converter Unit
DBUN-LOAD	REQ	50	UNFOR-MATTED*	NEW	YES	NO	SEQ.	Database Unload
DBLOAD	REQ	51	UNFOR-MATTED*	OLD	YES	NO	SEQ.	Database Load
USER-FILE	REQ	REQ	REQ	REQ	YES	NO	SEQ.	Any User-Defined File

Table 4-3. FORTRAN Files and Their Default Attributes. (Cont.)

Logical Name	The logical name used by MSC/NASTRAN.
Physical Name	The default name used to open the file. "REQ" means that this parameter is required in the ASSIGN statement from the user.
Unit Number	The default FORTRAN unit number used by MSC/NASTRAN. "REQ" means that this parameter is required in the ASSIGN statement from the user. The valid FORTRAN units range from 1 to 99, excluding those units already used.
Form	The default form used when the file is opened.
Status	The default status used when the file is opened.
Assignable	If "YES", the user may assign a physical file to this logical name. If "NO", the unit and logical names are reserved by MSC/NASTRAN.
Open	If "YES", the file is opened by default. If "NO", the file must be explicitly opened.
Access	If "SEQ.", the file is opened for sequential access. If "DIRECT", the file is opened for direct access.
*	FORMATTED is required for neutral-format files.
†	This must be FORMATTED if the BCD option is selected in DMAP.

4.5.2 ASSIGN Statement for DBsets

```
ASSIGN  logical-name=filename [ TEMP  DELETE  SYS=sys-spec
]
```

See Section 5.4.1 for details on the SYS field for DBsets.

Table 4-4. Default DBsets and Their Default Attributes.

Cray and NEC

DBset	Memory			Buffsize		Physical File Attribute		
	Type	Size	Units	Assignable	Size	Logical_Name	Physi-cal_Name	SIZE
MASTER	RAM	120 000	Words	YES	4097	MASTER	*dbs.*MASTER	5 000
DBALL	N/A	-		YES	4097	DBALL	*dbs.*DBALL	1 000 000
OBJSCR	N/A	-		NO	4097	OBJSCR	*sdir.*OBJSCR	5 000
SCRATCH	SMEM	0	GINO Blocks	YES	4097	SCRATCH	*sdir.*SCRATCH	1 000 000
SCRATCH	N/A	-		YES	4097	SCR300	*sdir.*SCR300	1 000 000
User DBset	N/A	-		YES	4097	DBset	*dbs.DBset*	25 000

All Others

DBset	Memory			Buffsize		Physical File Attribute		
	Type	Size	Units	Assignable	Size	Logical_Name	Physi-cal_Name	SIZE
MASTER	RAM	120 000	Words	YES	2049	MASTER	*dbs.*MASTER	5 000
DBALL	N/A	-		YES	2049	DBALL	*dbs.*DBALL	250 000
OBJSCR	N/A	-		NO	2049	OBJSCR	*sdir.*OBJSCR	5 000
SCRATCH	SMEM	100	GINO Blocks	YES	2049	SCRATCH	*sdir.*SCRATCH	250 000
SCRATCH	N/A	-		YES	2049	SCR300	*sdir.*SCR300	250 000
User DBset	N/A	-		YES	2049	DBset	*dbs.DBset*	25 000

DBset	The DBset.
Memory	The size of open core memory (in words) of the RAM of the MASTER DBset. The size may be modified using the FMS statement, INIT MASTER (RAM = *value*).
Buffsize	The buffer size (words) used for I/O transfer for each DBset. This size may be changed if "YES" is in the Assignable column.
Logical Name	The logical name of the DBset. This name may be set with the ASSIGN or INIT statement.
Physical Name	The name of the file as known to your operating system. This name may be changed by using the ASSIGN statement.
SIZE	The default maximum file size (in GINO blocks) allowed for each DBset. This size may be changed by using the INIT statement.

4.6 Using Databases

MSC/NASTRAN provides a database for the storage and subsequent retrieval of matrices and tables. This facility consists of several database sets (DBsets) that conform to the following specifications:

- The MSC/NASTRAN limit on the maximum number of DBsets for an analysis is 200. Your computer may have a lower limit on the maximum number of open files that a process can open. This limit is displayed as the "Number of open files" by the "limits" special function (see Section 4.1.3).
- Each DBset may consist of 1 to 20 physical files. Again, this is subject to the maximum number of open files that your systems permits.
- The maximum size of each DBset is machine dependent. There are several factors affecting the maximum size a given file can reach. Among these are: the job's file resource limit; the available space of the filesystem containing the file; the maximum file size supported by the operating system. The "df" command lists the maximum space and available space in a filesystem. Your resource limit is displayed by as the "Maximum file size" by the "limits" special function.

On a 32-bit processor, the UNIX operating system's maximum file size has traditionally been 2 gigabytes (actually 2**32-1). In recent years, many systems have switched over to 64-bit processors or now support "large files," i.e., a file that can exceed 2 gigabytes. Table 4-5 lists those versions of MSC/NASTRAN that support large files.

Table 4-5. Database I/O Capabilities

Computer	Large File	File Mapping	Buffered I/O
Convex C-Series	Yes	Yes	Yes
Cray	Yes	No	No
Digital	Yes	Yes	Yes
Fujitsu	Yes	Yes	Yes
HP 9000	Yes[1]	No	Yes
HP Exemplar	Yes	No	Yes
Hitachi	No	No	Yes
IBM	Yes[2]	Yes	Yes
NEC	Yes	No	Yes
Silicon Graphics R4K, R5K	No	Yes	Yes
Silicon Graphics R8K, R10K	Yes[3]	Yes	Yes
Sun	Yes[4]	Yes	Yes

Notes:
1. Large files can only be created on filesystems supporting large files (the flags value from "df -g" must show the 0x10 bit set).
2. Large files are available on AIX 4.2 or later if the filesystem containing the file supports large files. See your system administrator to determine which filesystems, if any, support large files.
3. Large files can only be created on "XFS" filesystems.
4. Large files are available on Solaris 2.6 or later if the filesystem containing the file supports large files. See your system administrator to determine which filesystems, if any, support large files.

The default database provides for five DBsets that are subdivided into two categories (scratch and permanent DBsets) as follows:

- Three DBsets are scratch DBsets that are typically deleted at the end of a run. The logical names for these DBsets are SCRATCH, SCR300, and OBJSCR.
- The remaining two DBsets have the default names of *dbs*.MASTER and *dbs*.DBALL, where *dbs* is set by the "dbs" keyword.

The database may be defined in two different ways:

1. Using the "dbs" keyword on the command line (see Section 4.6.1).
2. Using ASSIGN statements in the FMS section of the input data file (see Sections 4.5.2 and 4.6.2).

4.6.1 Using the "dbs" Keyword

To illustrate the use of the "dbs" keyword, see the TPL file "am762d.dat".

```
ID MSC, AM762D $ JFC 30SEP88
$ DBS=AM762D SPECIFIED WHEN JOB SUBMITTED
TIME 2
SOL 101 $ SUPERELEMENT STATICS
CEND
TITLE = EXAMPLE: SPECIFY DBS=AM762D WHEN JOB SUBMITTED      AM762D
SUBTITLE = COLD START
LOAD = 11
DISPLACEMENT = ALL
ELFORCE = ALL
BEGIN BULK
CBEAM,1,1,10,20,0.,1.,0.
FORCE,11,20,,100.,1.,.8,1.
GRID,10,,0.,0.,0.,,123456
GRID,20,,10.,0.,0.
MAT1,100,1.+7,,.3
PBEAM,1,100,1.,.08,.064,,.1
ENDDATA $ AM762D
```

To run this job, enter

```
msc705 nastran TPLDIR:am762d
```

The default value for "dbs" in this example is "./am762d". The DBALL and MASTER DBsets are created in your directory as shown in the following directory listing:

```
am762d.DBALL       am762d.dat         am762.f06
am762d.MASTER      am762.f04          am762.log
```

To restart from the previously created DBsets, use the following command:

```
msc705 nastran TPLDIR:am762r dbs=am762d
```

The input data for the restart is TPL file am762r.dat. The “dbs” keyword is set to “am762d”. The following is sample input for the am762r.dat file:

```
RESTART VERSION = 1 $ RESTART FROM AM762D
$ DBS=AM762D SPECIFIED WHEN JOB SUBMITTED
ID MSC, AM762R $ JFC 30SEP88
TIME 2
SOL 101
CEND
TITLE = EXAMPLE: RESTART, ATTACH DATABASE VIA DBS=AM762D     AM762R
SUBTITLE = RESTART WITH LARGER LOAD
SELG = ALL $ GENERATE NEW LOAD
SELR = ALL $ REDUCE NEW LOAD
LOAD = 11
DISPLACEMENT = ALL
ELFORCE = ALL
BEGIN BULK
FORCE,11,20,,100.,1.,.8,1.
ENDDATA $ AM762R
```

The following files remain at the end of the run:

```
am762d.DBALL   am762d.dat     am762d.f06     am762r.dat     am762r.f06
am762d.MASTER  am762d.f04     am762d.log     am762r.f04     am762r.log
```

4.6.2 Using the ASSIGN Statement

This section contains two examples using the ASSIGN statement. The first example, TPL file am763d.dat shows how to use the ASSIGN statement to create the database files. The second example shows how to use the ASSIGN statement to assign database files in a restart job.

```
ASSIGN 'MASTER=DBSDIR:am763d.MYMASTER'
ASSIGN 'DBALL=DBSDIR:am763d.MYDBALL'
$
$ DBSETS CREATED WITH DIRECTORIES AND NAMES AS ASSIGNED ABOVE.
$ THIS IS ALTERNATE METHOD TO BE USED INSTEAD OF SPECIFYING DBS = AM763D
$ WHEN JOB IS SUBMITTED.
$ SPECIFY SCR=NO WHEN JOB IS SUBMITTED.
$
ID MSC, AM763D $ FILENAME CHANGED 16SEP88 -- JFC
TIME 2
SOL 101 $ STRUCTURED SUPERELEMENT STATICS WITH AUTO RESTART
CEND
TITLE = EXAMPLE: DATABASE CREATED VIA ASSIGN CARDS             AM763D
SUBTITLE = COLD START.
LOAD = 11
DISPLACEMENT = ALL
ELFORCE = ALL
BEGIN BULK
CBEAM,1,1,10,20,0.,1.,0.
FORCE,11,20,,100.,1.,.8,1.
GRID,10,,0.,0.,0.,,123456
GRID,20,,10.,0.,0.
MAT1,100,1.+7,,.3
PBEAM,1,100,1.,.08,.064,,.1
ENDDATA
```

To submit this job, use the commands:

```
mkdir dbs
DBSDIR=dbs; export DBSDIR
# Korn shell: export DBSDIR=dbs
# C-shell:    setenv DBSDIR dbs
msc705 nastran TPLDIR:am763d
```

The DBsets “mydball” and “mymaster” are placed in the “dbs” directory as shown in the directory listing:

```
am763d.dat   am763d.f06   dbs/am763d.MYDBALL
am763d.f04   am763d.log   dbs/am763d.MYMASTER
```

The second example (TPL file am763r.dat) illustrates a restart that uses the ASSIGN statement:

```
RESTART $ RESTART FROM AM763D, SAVE VERSION 1 ON DATABASE
$ ATTACH AM763D DATABASE WITH ASSIGN COMMANDS BELOW
ASSIGN MASTER='DBSDIR:am763d.MYMASTER'
ID MSC,AM763R $ FILENAME CHANGED 16SEP88 -- JFC
TIME 2
SOL 101
CEND
TITLE = EXAMPLE: RESTART, DATABASE ATTACHED VIA ASSIGN CARDS   AM763R
SUBTITLE = RESTART -- ADD STRESS RECOVERY COEFFICIENTS TO PBEAM
LOAD = 11
DISPLACEMENT = ALL
ELFORCE = ALL
STRESS = ALL
BEGIN BULK
$ WITH STRUCTURED SOLUTION SEQUENCES (SOL 101+), ALL BULK DATA IS STORED
$ ON DATABASE.
$ ON RESTART, ONLY INCLUDE ADDITIONAL CARDS OR CHANGED CARDS.
/,6 $ DELETE OLD PBEAM CARD ON DATABASE, ADD STRESS RECOVERY COEFFICIENTS
$ AND REPLACE AS FOLLOWS.
PBEAM,1,100,1.,.08,.064,,.1,,+PBEAM1
+PBEAM1,0.0,0.5,0.0,-0.5,0.3,0.0,-0.3,0.0,+PBEAM2
+PBEAM2,YES,0.5,1.0,.08,.064,,.1,,+PBEAM3
+PBEAM3,0.0,0.5,0.0,-0.5,0.3,0.0,-0.3,0.0
ENDDATA $ AM763R
```

To submit the above file, issue the command:

```
msc705 nastran TPLDIR:am763r
```

This job uses the DBsets created by am763d with the following files remaining at the end of the run:

```
am763d.dat     am763d.f06     am763r.dat     am763r.f06     dbs/am763d.MYDBALL
am763d.f04     am763d.log     am763r.f04     am763r.log     dbs/am763d.MYMASTER
```

4.6.3 Using the INIT Statement

DBsets are created using the INIT statement, which is documented in the File Management Section (FMS) of the *MSC/NASTRAN Quick Reference Guide*. For example,

```
INIT  DBALL LOGICAL=(DBALL1(2000),DBALL2(300KB))
```

creates and allocates two members DBALL1 and DBALL2 to the DBALL DBset with a size of 2000 GINO blocks for DBALL1 and a size of 300 kilobytes for DBALL2. The size can be specified either as the number of GINO blocks or as a number followed by one of the following modifiers:

M or Mw	Multiply the size by 1024**2, round up to a BUFFSIZE multiple.
Mb	Multiply the size by (1024**2)/*bytes_per_word*, round up to a BUFFSIZE multiple.
K or Kw	Multiply the size by 1024, round up to a BUFFSIZE multiple.
Kb	Multiply the size by 1024/*bytes_per_word*, round up to a BUFFSIZE multiple.
w	Round the size up to a BUFFSIZE multiple.
b	Divide the size by *bytes_per_word*, round up to a BUFFSIZE multiple.

where *bytes_per_word* is 8 on Cray and NEC; 4 on all others. The modifier may be specified using any case combination.

Note: MSC/NASTRAN now uses standard computer units for K and M. Prior releases used engineering units.

4.7 Using INCLUDE Statement

The INCLUDE statement is used to insert a specified file into the input file. This statement is especially useful when you want to partition your input into separate files. The format is

```
INCLUDE filename
```

or

```
INCLUDE 'logical-symbol:filename'
```

The default directory for *filename* is the current directory (i.e., the directory where the nastran command was run). If the filename contains lowercase letters, spaces, or dollar signs, it must be enclosed in single quotes.

The directory *install_dir*/msc705/nast/misc/sssalter contains additional alters that represent client-requested or prototype features that are not yet implemented. These alters can be inserted using the INCLUDE statement and the SSSALTERDIR symbol. For example,

```
INCLUDE 'SSSALTERDIR:zfreqa.dat'
```

In the TPL library, there is a file named tplidx.dat that contains a one-line description of the input files. Also included is a file named tplexec, which is a script used to run the data files. For both libraries, the recommended execution procedure is to copy

the file to your own directory and then execute the problem using the instructions in Section 4.1. Note that several of the library files have “include” files that should be copied.

4.8 Resolving Abnormal Terminations

MSC/NASTRAN generates a substantial amount of information concerning the problem being executed. The F04 file provides information on the sequence of modules being executed and the time required by each of the modules; the LOG file contains system messages. A list of known outstanding errors for Version 70.5 is delivered in the file “*install_dir*/msc705/nast/doc/error.lis”. Please consult this file for limitations and restrictions.

MSC/NASTRAN may terminate as a result of errors detected by the operating system or by the program. If DIAG 44 is set (see the *MSC/NASTRAN Quick Reference Guide*), then MSC/NASTRAN produces a dump when most of the errors occur. Before the dump occurs, there may be a fatal message. The general format of this message is

```
***SYSTEM FATAL ERROR 4276, subroutine-name ERROR CODE n
```

This message is issued whenever an interrupt occurs that MSC/NASTRAN is unable to satisfactorily process. The specific reasons for the interrupt are usually printed in the F06 and/or LOG file; “*n*” is an error code that is explained in Chapter 16 of the *MSC/NASTRAN Reference Manual*.

Whenever the System Fatal Error 4275 or 4276 is associated with a database error, further specific information is written to the F06 file as follows:

```
bio-function ERROR – STATUS = errno, FILX = i, LOGNAME = logical, NSBUF3
j
FILE = filename
BLKNBR = k
ERROR MESSAGE IS ––
error-message-text
```

The FILE and/or BLKNBR lines may not be present, depending upon the bio-function issuing the message.

4.8.1 Interpreting System Error Codes

If an operating system error occurs, an attempt is made to catch the error signal (errno) and place the error number in the LOG file. A description of these error numbers may be obtained with the following command:

IBM:	cat /usr/include/sys/errno.h
Sun:	man –s2 intro
All others:	man 2 intro

4.8.2 Terminating a Job

There may be instances when a running job must be prematurely terminated; this is accomplished using one of the following procedures:

Job Running in the Foreground (batch=no)

Use the interrupt key (on NEC and Silicon Graphics systems this key is usually “Ctrl-\”; on other systems “Ctrl-C”).

Job Running in the Background (batch=yes or after=*time*)

Use the “ps” command to find the process ID (PID) of the MSC/NASTRAN job (i.e., the “*install_dir*/msc705/*arch*/analysis” executable) and use “kill *pid*” where *pid* is the process ID.

Job Running Under NQS or NQE (queue=*queue_name*)

1. Use “qstat -a” to find the request-id of your job.
2. Use “qdel *request-id*” to delete a job that has not yet started; or use “qdel -k *request-id*” to kill a job that has already started where *request-id* is the request ID.

4.8.3 Flushing F04 and F06 Output to Disk (Convex C-Series, Cray, SGI only)

As MSC/NASTRAN writes to the F04 and F06 files, the FORTRAN runtime libraries will buffer this I/O in memory to reduce the amount of time consumed by disk I/O. When the buffers are filled (i.e., MSC/NASTRAN has written a sufficient amount of information to the F04 or F06 file), the buffers will be flushed to the files by the FORTRAN runtime libraries. In a large job, some modules may do substantially more computation than I/O. As a result, the I/O may remain in the FORTRAN buffers (possibly for several hours) before they are written to disk.

Convex C-Series, Cray, and SGI computers support asynchronous flushing of the F04 and F06 files. To do this, enter the command

```
kill –USR1 pid
```

where *pid* is the process ID of the running MSC/NASTRAN job (i.e., the "*install_dir*/msc705/*arch*/analysis" executable). There may be a time delay between the time you issue the kill command and time the files are actually updated.

4.8.4 Common System Errors

The most common system errors encountered during an MSC/NASTRAN job are described below.

Disk I/O Errors

- ERRNO 1 (EPERM) - no permission to file (all systems).

  Please check the ownership and mode of the file or directory with the "ls -l" command. Change either the ownership or permissions of the file or the directories along the path. The chgrp(1) command is used to change the group of a file, chmod(1) is used to change permissions of the file, and chown(1) is used to change ownership of the file.

- ERRNO 27 (EFBIG) - file is too large (all systems)

  This error occurs if a file's size exceeds a resource limit. The resource limits in effect during the job's execution are printed in the LOG file under the heading "Current Resource Limits." Increase the "-lf" and "-lF" parameters on your qsub command if you are running NQS or NQE; ask your system administrator to increase your "Filesystem Space" limit (Cray and NEC) or "File Size" limit (all others).

- ERRNO 28 (ENOSPC) - disk space is completely filled (all systems).

  MSC/NASTRAN deletes its scratch files at termination even if the disk space fills up. Therefore, the df(1) command may show a large amount of free space even though the job failed due to lack of disk space. Both the current working directory and the scratch directory need to be checked. Move your files to a disk with more space (see the “out”, “dbs”, and “sdirectory” keywords), or remove unnecessary files from the disk with the rm(1) command.

Inability to Allocate the Requested Amount of Memory (OPEN CORE Allocation Failed)

- Temporary lack of swap space (all systems).

  This error may be caused by too many processes running at the same time. Decrease the number of processes or increase the available swap space.

- The data segment of the process has exceeded the UNIX resource limit (all systems).

  The resource limits in effect during the job’s execution are printed in the LOG file under the heading “Current Resource Limits.” Ask your system administrator to increase your “Data Segment Size” (all), “Real Address Space” (Cray), “Maximum break size” (HP), or “Virtual Address Space” (all others).

- HPALLOC failed in SUBROUTINE FIELDLN (Cray).

  The resource limits in effect during the job’s execution are printed in the LOG file under the heading “Current Resource Limits.” Ask your system administrator to increase your “Real Address Space.”

- memory allocation error: unable to allocate *n* words (HP 9000).

  The resource limits in effect during the job’s execution are printed in the LOG file under the heading “Current Resource Limits.” Check you “Maximum break size”. If this is smaller than the requested memory, ask your system administrator to increase your limit.

  If your limit is large enough, the system wide “shmmax” and “maxdsize” kernel parameters may be too small. These parameters must be large enough to accomodate all simultaneously executing MSC/NASTRAN jobs plus all others users of shared memory. These values are modified using sam(1M), see “Kernel Parameters” under “Configurable Parameters”.

It may also be possible to correct these errors with the following:

- Reduce the amount of memory requested by the “memory” keyword.
- Increase the “-Im” and “-IM” parameters if you submitted your job to NQS or NQE using a “qsub” command.
- Increase the “prmdelta” or “ppmdelta” keyword values if you submitted your job to NQS or NQE using the “queue” command line parameter.

EAG FFIO Errors (Cray Only)

The following error message may appear on Cray systems when FFIO is being used:

```
eie open failure  :  Not enough space for cache pages
```

This message is a consequence of not having enough memory for the eie cache pages. System memory requirements are as follows:

Description	Size	Where Documented
executable	6.5 MW	Appendix C
opencore	memory keyword	Appendix B and Section 5.3
EIE buffers/ Cache		Appendix B

If the job was submitted with the “qsub” command, then the error can be avoided by increasing the IM and Im NQS parameters The value should be at least 6.5 MW **plus** the value specified by the mem keyword **plus** the amount needed for eie. To determine the amount needed for FFIO, consider the following ff_io_opts request:

```
(eie:128:16:1:1:1:0,set:0:0)
```

This request requires an additional:

$$128 \text{ (blocks/page)} \times 16 \text{ (pages)} \times 512 \text{ (words/block)} = 1\,048\,576 \text{ W} = 1 \text{ MW}.$$

If the job was submitted with the nastran “queue” keyword, then the nastran command adjusts the memory request. This message should only appear if the user modified the “ff_io_defaults” or “ff_io_opts” keywords without modifying the “ff_io_cachesize” keyword. This error can be avoided by increasing the value set by the “ppmdelta” keyword (Appendix B) to 6.5 MW **plus** the amount of memory for FFIO.

See the URL file://*install_dir*/msc705/*arch*/ffio.html for a complete description of EAG FFIO.

CHAPTER 5

HOW TO USE THE ADVANCED FUNCTIONS OF MSC/NASTRAN

This chapter discusses the NASTRAN statement, managing MSC/NASTRAN's internal memory allocations, managing the databases, interpreting performance related information in the F04 file, understanding some of the lower-level database messages, and creating alternate delivery databases.

5.1 Using the Advanced Keywords

The following is a partial list of the advanced keywords that may be used on the command line or placed into RC files as appropriate. More basic keywords are listed in Section 4.2; a complete list of all keywords and their syntax is listed in Section B.1 of Appendix B.

5.1.1 All Systems

Keyword	Purpose
buffsize	Specifies the size of database I/O transfers.
bpool	Specifies the number of GINO blocks set aside for buffer pooling.
delivery	Specifies an alternate delivery database name.
exe	Specifies an alternate solver executable.
gmconn	Specifies an external evaluator connection file.
nastran	Specifies NASTRAN statements.
post	Specifies UNIX commands to be executed after job has completed.
pre	Specifies UNIX commands to be executed before job has begun.
proc	Specifies an alternate solver executable file type.
rank	Specifies the rank size for the sparse solvers.
smem	Specifies the number of GINO blocks to set aside for MEMFILE portion of the SCRATCH DBSet.
sysfield	Specifies global SYS parameters (see Section 5.4.1).
sys*n*	Specifies SYSTEM cell values.

5.1.2 Not Available on Fujitsu, Hitachi, and IBM

Keyword	Purpose
parallel	Specifies the number of processors to use in numeric modules.

5.1.3 Cray Only

Keyword	Purpose
ff_io	Invokes Cray's FFIO high performance I/O system.
ff_io_cachesize	Specifies the size of the FFIO cache.

5.1.4 HP Exemplar Only

Keyword	Purpose
subcomplex	Specifies the subcomplex where the job will run.

5.1.5 NEC Only

Keyword	Purpose
hpio_param	Invokes NEC's HPIO high performance I/O system and specifies the parameters.

5.1.6 SGI R8K, R10K Only

Keyword	Purpose
spintime	Specifies the time to wait in a spin loop.
threads	Specifies the preferred number of threads under Dynamic Thread Management.

5.1.7 Queuing Keywords

Note: These capabilities are dependent upon the queue submission commands defined by the "submit" keyword and your queuing system. The keywords may not work on your system.

Keyword	Purpose
ppcdelta	Specifies the per-process CPU time limit delta.
ppmdelta	Specifies the per-process memory limit delta.
prmdelta	Specifies the per-request memory limit delta.
qoption	Specifies other queue command options.
submit	Defines queues and their associated submittal commands.

5.2 Using the NASTRAN Statement

The NASTRAN statement allows you to change parameter values at runtime.

The format of NASTRAN statements is

```
NASTRAN     KEYWORD1=A, KEYWORD2=B, ... KEYWORDi=I
```

An input file may contain more than one NASTRAN statement. A full description of these keywords is in Section 1, Table 1, of the *MSC/NASTRAN Quick Reference Guide.* A brief description of a few of the keywords follows:

AUTOASGN

AUTOASGN is used to determine which DBsets are automatically assigned (see the following table). The default is AUTOASGN=7, which specifies that all DBsets are to be automatically assigned.

Value	Default DBsets	Delivery DBsets	DBLOCATEd DBsets
0			
1	X		
2		X	
3	X	X	
4			X
5	X		X
6		X	X
7 (Default)	X	X	X

Notes: 1. Default DBsets are the use-default DBsets and any DBsets specified by INIT statements (see Table 4-4).

2. Delivery DBsets contain the Structured Solution Sequences and Unstructured Solution Sequences.

3. DBLOCATEd DBsets are the DBsets specified by DBLOCATE statements. See the DBLOCATE FMS statement in Section 2 of the *MSC/NASTRAN Quick Reference Guide*.

BUFFPOOL, SYSTEM(114)

See the “bpool” command line keyword in Section B.1 of Appendix B.

BUFFSIZE, SYSTEM(1)

See the “buffsize” command line keyword in Section B.1 of Appendix B.

PARALLEL, SYSTEM(107)

See the “parallel” command line keyword in Section B.1 of Appendix B.

SYSTEM(128)

SYSTEM(128) specifies the maximum interval of CPU time (in minutes) between database directory updates to the MASTER DBSET when the INIT MASTER(RAM) option is being used. The default is 1 minute on Cray and NEC systems and 5 minutes on all others. See the DBUPDATE FMS statement in Section 2 of the *MSC/NASTRAN Quick Reference Guide* for more information.

SYSTEM(149) (Cray Only)

SYSTEM(149) specifies the default RAW I/O flag for DBsets (see the *UNICOS System Calls Reference Manual* SR-2012). If 0 is specified, all DBsets are opened without the O_RAW option. Otherwise, DBsets are opened up with the O_RAW option. The O_RAW option reduces system CPU time at the possible expense of elapsed time. The default is 1, i.e., enable RAW I/O for all DBsets.

The setting for all DBsets can also be specified by the “sysfield” keyword “RAW”; specific DBset can be set using the INIT FMS statement’s SYS parameter “RAW” (see Section 4.5.2).

Note: RAW I/O should always be used if FFIO is enabled (see the “ff_io” keyword).

SYSTEM(198), SYSTEM(205)

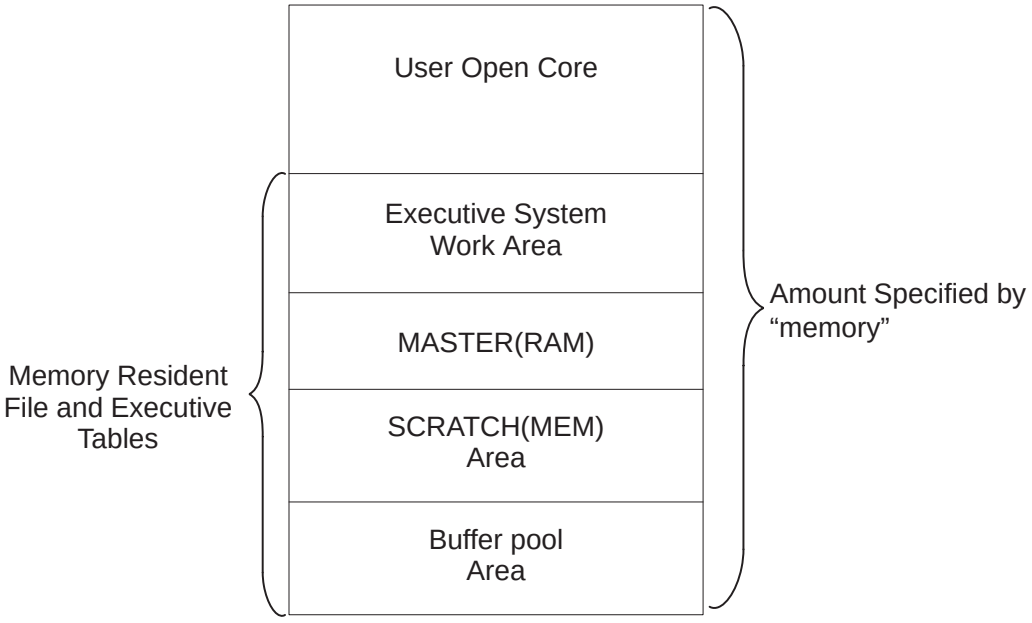
See the “rank” command line keyword in Section B.1 of Appendix B.

SYSTEM(207)

See the “LOCK” keyword in Section 5.4.1.

5.3 Managing Memory

Memory is dynamically allocated at runtime with the “memory” keyword of the nastran command. The memory can be partitioned in a variety of ways (see the memory map at the top of the F04 file for the actual memory allocation used in a job). To make the most effective choice of the sizing parameters, see the following map of MSC/NASTRAN’s memory:



As can be seen in this diagram, the memory available for use by MSC/NASTRAN modules (user open core) is the amount specified by the “memory” keyword (open

core size) less the space required by memory resident files and executive tables. The actual user open core is calculated as follows:

$$\text{User Open Core} = \text{MEM} - (\text{EXEC} + \text{RAM} + \text{SMEM} \times \text{BUFFSIZE} + \text{BUFFPOOL} \times (\text{BUFFSIZE} + 10))$$

MEM — The total size of open core. There is no default. Set by the "memory" keyword.

EXEC — The executive system work area. The size is 70 409 + 4 × BUFFSIZE words.

RAM — NDDL tables. The default is 30 000. Set by the FMS statement INIT MASTER (RAM=*value*).

SMEM — The memory-resident file space for temporary database files. The default is 0 for Cray and NEC; 100 for all others. Set by the FMS statement INIT SCRATCH (MEM=*value*) or the "smemory" keyword.

BUFFSIZE — The maximum BUFFSIZE used for all the DBsets referenced by the job. The default is 4097 on Cray and NEC; 2049 on all others. Set by the "buffsize" keyword.

BUFFPOOL — The buffer pool area for permanent database files. The default size is 27 on Cray and NEC; 37 on all others. Set by the "bpool" keyword.

The INIT statement may be used to size MASTER and SCRATCH memory. Several examples of the INIT statement follow along with an explanation of their uses:

1. If the available memory is a critical resource, then using the following selection reduces memory requirements at the expense of increased CPU and wall-clock time.

```
INIT SCRATCH(NOMEM)      $     temporary database files
```

2. Performance gains may be made by increasing the memory-resident area for the scratch and permanent DBset(s) as follows. Note that the default RAM is sufficiently large and need not be increased.

```
NASTRAN BUFFPOOL = 70    $     increase permanent database
files
INIT SCRATCH (MEM=200)   $     increase scratch memory
```

3. If disk space is critical, then all DBsets may be deleted at the end of the job by specifying "S" on the INIT MASTER statement as follows:

```
INIT MASTER(S) $ delete DBsets at end of job
```

   This statement is identical to specifying "scratch=yes" on the command line.

4. If disk space is critical, but data recovery restarts are required, then a mini database may be created that will support data recovery restarts by setting “scratch=mini” on the command line.

```
msc705 nastran myjob scratch=mini
```

5.4 Managing DBSets

5.4.1 Using the SYS Field

The SYS field is used to specify computer-dependent parameters on ASSIGN statements. If your computer does not recognize a particular parameter, it is ignored without acknowledgement. This keyword is specified as a comma separated list of keyword=value pairs. For example, file locking may be disabled on for a particular DBset with the following statement:

```
ASSIGN 'DBALL=mydball.DBALL' SYS=LOCK=NO
```

A global SYS field for all DBsets can be specified by the “sysfield” keyword described in Section 5.1.

The following tables describe the SYS field parameters recognized on UNIX systems. A complete description of parameters and their syntax is available in Section B.2 of Appendix B.

All Systems

Keyword	Purpose
lock	Lock database files.

Systems Supporting File Mapping (see Table 4-5)

Keyword	Purpose
mapio	Use the virtual memory system to map database files to memory.
wnum	Specifies the default number of maps used on database files.
wsize	Specifies the default size of maps used on database files.

Systems Supporting Buffered I/O (see Table 4-5)

Keyword	Purpose
buffio	Use intermediate buffers to hold database file records (no)
wnum	Specifies the default number of buffers used for database files
wsize	Specifies the default size of buffers used for database files

Cray Only

Keyword	Purpose
raw	Use “raw I/O” to read and write database files.

5.4.2 Using File Mapping

Notes:
1. See Table 4-5 to determine if file mapping is available on your computer.
2. Cray users should use the “ff_io” parameters in Section B.1 of Appendix B as an alternative to file mapping.
3. NEC users should use the “hpio_param” parameter in Section B.1 of Appendix B as an alternative to file mapping.

File mapping is a way to tell the operating system to use the virtual paging system to process a file. From the perspective of the process, file mapping effectively changes

the file I/O operations from synchronous to asynchronous because the paging functions of the operating system perform the I/O as part its normal virtual memory management. File mapping can be used for both permanent and temporary DBsets.

The “wsize” and “wnum” parameters described in Section B.2 of Appendix B specify the size of the window mapping the file to memory and the number of windows or maps that will be used for each file. The larger the window, the less often it must be moved when the file is sequentially read or written. Multiple maps allow several I/O streams to be active in the same file.

File mapping is controlled using the ASSIGN statement SYS field for individual DBsets and, globally, using the “sysfield” command line keyword. These are described in Section 5.4.1.

As an example, if file mapping is to be enabled for all files, the “sysfield” keyword in the command initialization or RC file or on the command line is:

```
sysfield=mapio=yes
```

If file mapping is to be disabled for all files, the “sysfield” keyword is:

```
sysfield=mapio=no
```

If file mapping is to be enabled for all but a specified set of DBsets, both “sysfield” keyword and ASSIGN specifications are required. In the command initialization or RC file or on the command line:

```
sysfield=mapio=yes
```

and, in the MSC/NASTRAN data file:

```
ASSIGN logical-name=filename,SYS=MAPIO=NO
```

for those files to be processed using normal disk I/O processing.

If file mapping is to be disabled for all but a specified set of DBsets, both “sysfield” keyword and ASSIGN specifications are also required. In the command initialization file, RC file, or on the command line:

```
sysfield=mapio=no
```

and, in the MSC/NASTRAN data file:

```
ASSIGN logical-name=filename,SYS=MAPIO=YES
```

for those files to be processed using file mapping.

5.4.3 Using Buffered I/O

Notes:
1. See Table 4-5 to determine if buffered I/O is available on your computer.
2. Cray users should use the “ff_io” parameters in Section B.1 of the Appendix as an alternative to buffered I/O.
3. NEC users should use the “hpio_param” parameter in Section B.1 of the Appendix as an alternative to buffered I/O.

Buffered I/O instructs MSC/NASTRAN to “buffer” or use intermediate memory areas to hold records of a file before either writing them out to disk or copying them to the MSC/NASTRAN internal areas. The primary purpose for using buffered I/O is to increase data reuse and, in some cases, to increase the actual read/write data lengths beyond that normally used by MSC/NASTRAN. Buffered I/O can be used for both permanent and temporary DBSETS.

The “wsize” and “wnum” parameters described in Section B.2 of Appendix B specify the size of the buffer to be used to hold file records and the number of such buffers to be used. The larger the buffer, the less often actual physical read/write operations are needed when the file is sequentially read or written. Multiple buffers allow several I/O streams to be active in the same file.

Buffered I/O is controlled using the ASSIGN statement SYS field for individual DBsets and, globally, using the “sysfield” command line keyword. These are described in Section 5.4.1.

As an example, if buffered I/O is to be enabled for all files, the “sysfield” keyword in the command initialization or RC file or on the command line is:

```
sysfield=buffio=yes
```

If buffered I/O is to be disabled for all files, the “sysfield” keyword is:

```
sysfield=buffio=no
```

If buffered I/O is to be enabled for all but a specified set of DBsets, both “sysfield” keyword and ASSIGN specifications are required. In the command initialization or RC file or on the command line:

```
sysfield=buffio=yes
```

and in the MSC/NASTRAN data file:

```
ASSIGN logical–name=filename,SYS=BUFFIO=NO
```

for those files to be processed using normal disk I/O processing.

If buffered I/O is to be disabled for all but a specified set of DBsets, both “sysfield” keyword and ASSIGN specifications are required. In the command initialization or RC file or on the command line:

```
sysfield=buffio=no
```

and in the MSC/NASTRAN data file:

```
ASSIGN logical–name=filename,SYS=BUFFIO=YES
```

for those files to be processed using buffered I/O.

5.4.4 Interpreting Database File-Locking Messages

All database files are locked using the operating system function “fcntl(2)”. This prevents two or more MSC/NASTRAN jobs from interfering with one another; however, this does not prevent any other program or operating system command from modifying the files.

A read-write (exclusive) lock is requested for every database file that is to be modified. A read-only (shared lock) is requested on every database file that is not modified, e.g., DBLOCATEd databases. If the lock request is denied because another MSC/NASTRAN job is using the file in a potentially conflicting manner, the following fatal error message is written to the F06 file:

```
bio-function ERROR – STATUS = errno, FILX = i, LOGNAME = logical, NSBUF3 = j
FILE = filename
ERROR MESSAGE IS ––
Unable to acquire a lock_type lock.
lock-type-explanatory-text
Process ID pid is holding a conflicting lock.
```

where *lock-type-explanatory-text* is:

- *lock_type* is “read-only”:

  ```
  This operation failed because another process already holds
  a read–write lock on this file.
  ```

- *lock_type* is “read-write”:

  ```
  This operation failed because another process already holds
  a read–write or read–only lock on this file.
  ```

Some systems will deny a file lock because of an internal resource limit. In these cases, the job is allowed to continue, and the following message will be written to the F06 file:

```
bio-function WARNING – STATUS = errno, FILX = i, LOGNAME = logical, NSBUF3 = j
FILE = filename
ERROR MESSAGE IS ––
Unable to acquire a lock_type lock.
computer-specific-text
advisory-text
```

where *computer-specific-text* is:

Cray

```
The file appears to be an NFS file, and remote file locking
is not supported by this system or the remote system.
```

or

```
The system wide maximum number of file locks has been
exceeded.  See the NFLOCKS parameter in the UNICOS kernel
file config.h.
```

Digital Alpha

```
The system wide maximum number of file locks has been
exceeded or the file may be in a partition that does not
support file lock (e.g., an NFS partition).  See ENOLCK in
man 2 fcntl for further information.
```

HP

```
The file appears to be an NFS file, and remote file locking
was denied.  See ENOLCK in man 2 fcntl for further
information.
```

IBM

```
The file appears to be in a Parallel Filesystem partition,
and file locking is not supported in PFS partitions.
```

or

```
The system wide maximum number of file locks has been
exceeded.  See ENOLCK in SC23–2198 Call and Subroutine
Reference.
```

SGI

```
The system wide maximum number of file locks has been
exceeded.  See {FLOCK_MAX} in man 2 intro.
```

Sun

```
The system wide maximum number of file locks has been
exceeded.  See ENOLCK in man –s 2 fcntl.
```

All others

```
The system wide maximum number of file locks has been
exceeded.  See ENOLCK in man 2 fcntl.
```

and *advisory-text* is:

- *lock_type* is “read-only”

  ```
  If another job modifies this file during this run, there is
  the potential for incorrect results to occur in this job.
  ```

- *lock_type* is “read-write”

  ```
  If another job accesses this file during this run, there is
  the potential for the file to be damaged and/or incorrect
  results to occur in both jobs.
  ```

Disabling File Locking

File locking can be disabled by:

- Setting “sysfield=lock=no” in an RC file or on the command line (see Section 5.1). This affects all DBsets in the job.
- Setting SYSTEM(207) to a nonzero value using the NASTRAN statement (see Section 5.2). This affects all DBsets in the job.

  The following informational message is written to the F06 file:

```
*** SYSTEM INFORMATION MESSAGE – BIO
    SYSTEM(207).NE.0 – File locking suppressed.
```

- Setting SYS=LOCK=NO on an FMS INIT statement (see Section 5.4.1). This only affects the specific DBset (s).

5.5 Interpreting the F04 File

MSC/NASTRAN writes information to the F04 file that aids in monitoring and tuning the performance of your job. An overview of the complete F04 file can be found in Section 9.2 of the *MSC/NASTRAN Reference Manual*. This section contains more detailed explanations of selected portions of the F04 file.

5.5.1 Summary of Physical File Information

This summary table describes the physical files used for the DBSets. A sample of this table, located near the top of the F04 file, is shown below.

```
              S U M M A R Y   O F   P H Y S I C A L   F I L E   I N F O R M A T I O N

ASSIGNED PHYSICAL FILE NAME                                  RECL (BYTES)  MODE  FLAGS
------------------------------------------------------------ ------------  ----  -----
/tmp/65872_57.SCRATCH                                              8192   R/W    L
/tmp/65872_57.OBJSCR                                               8192   R/W    L
/tmp/65872_57.MASTER                                               8192   R/W    L
/tmp/65872_57.DBALL                                                8192   R/W    L
/tmp/65872_57.DBALL2                                               8192   R/W    L
/tmp/65872_57.SCR300                                               8192   R/W    L
/msc/msc691/aix/SSS.MASTERA                                        8192   R/O    L
/msc/msc691/aix/SSS.MSCOBJ                                         8192   R/O    L

    FLAG VALUES ARE --
      F   FFIO INTERFACE USED TO PROCESS FILE
      H   HPIO INTERFACE USED TO PROCESS FILE
      L   FILE HAS BEEN LOCKED
      M   FILE MAPPING USED TO PROCESS FILE
      R   FILE BEING ACCESSED IN 'RAW' MODE

** PHYSICAL FILES LARGER THAN 2GB FILES ARE NOT SUPPORTED ON THIS PLATFORM
```

In this summary, “ASSIGNED PHYSICAL FILENAME” is the physical FILENAME with any symbols translated; “RECL” is the record length in bytes; “MODE” is the file access mode, R/W is read-write mode, R/O is read-only mode. The “FLAGS”

column will contain various letters depending on the capabilities of the platform and user requests, the text below the table indicates flag values that are possible on the specific platform.

In this example, an INIT statement was used to create the DBALL DBSet with two files using the logical names DBALL and DBALL2.

Below the summary is a message indicating if large files (see Section 4.6) are available on this platform. On HP-UX 10.20 and IRIX64 6.1, the actual file system containing the file must support large files; this fact is not indicated in the message.

5.5.2 Memory Map

Immediately following the “Summary of Physical File Information” is a map showing the allocation of memory. This map is also described in Section 5.3.

```
** MASTER DIRECTORIES ARE LOADED IN MEMORY.
   USER OPENCORE (HICORE)         =     3804612  WORDS
   EXECUTIVE SYSTEM WORK AREA     =       78605  WORDS
   MASTER(RAM)                    =       30000  WORDS
   SCRATCH(MEM) AREA              =      204900  WORDS (     100 BUFFERS)
   BUFFER POOL AREA (GINO/EXEC)   =       76183  WORDS (      37 BUFFERS)

   TOTAL MSC/NASTRAN MEMORY LIMIT =     4194300  WORDS
```

In this table “USER OPENCORE” is the amount of memory available to the module for computation purposes; “EXECUTIVE SYSTEM WORK AREA” is the space reserved for the executive system; “MASTER(RAM)” is the space reserved to cache datablocks from the MASTER DBSet; “SCRATCH(MEM) AREA” is the space reserved to cache datablocks from the SCRATCH and SCR300 DBSets; “BUFFER POOL AREA” is the space reserved for the buffer pool; “TOTAL MSC/NASTRAN MEMORY LIMIT” is the total space allocated to MSC/NASTRAN’s open core using the “memory” keyword.

5.5.3 Day Log

The Day Log portion of the F04 is a DMAP execution summary. This log, in table format, contains the vast majority of the information in the F04. The beginning of the Day Log is shown below:

```
DAY TIME  ELAPSED  I/O MB   DEL_MB   CPU SEC  DEL_CPU   SUB_DMAP/DMAP_MODULE MESSAGES

10:32:16     0:16    13.6       .3       .8       .0    SESTATIC    20  IFPL    BEGN
10:32:16     0:16    13.7       .1       .8       .0    IFPL        29  IFP1    BEGN
10:32:16     0:16    13.7       .0       .8       .0    IFPL        39  XSORT   BEGN
```

In the Day Log, “DAY TIME” is the time of day of the entry; “ELAPSED” is the elapsed time since the start of the job; “I/O MB” is the megabytes of I/O to the databases since the start of the job; “DEL_MB” is the delta I/O since the previous entry; “CPU SEC” is the total CPU seconds since the start of the job; “DEL_CPU” is the delta CPU since the previous entry; “SUB_DMAP/DMAP_MODULE” indicates

the DMAP statement being executed; and “MESSAGES” are any messages issued by the module, “BEGN” is the start of the module and “END” is the end.

Notes: 1. The “I/O MB” value is computed by multiplying SYSTEM(85), which is incremented by one for each GINO I/O, by BUFFSIZE. This value will lose accuracy if the the DBSets do not have the same BUFFSIZE.

2. If SYSTEM(84) is set to 0, the “I/O MB” column will be the number of GINO I/Os.

3. The “I/O MB” column will be scaled by gigabytes and a “G” will be appended after each number if the value is greater than or equal to 100 000.

4. Prior to Version 69, the “I/O SEC” value was computed by multiplying SYSTEM(85) by SYSTEM(84) (a pseudo-I/O rate).

5.5.4 User Information Messages 4157 and 6439

The UIM 4157 text provides decomposition estimates upon completion on the preface of the decomposition module. This message has a counterpart, UIM 6439, which provides actual information from the completed decomposition process. These two messages are interspersed within the Day Log at each decomposition. The following example is from a sparse decomposition.

```
*** USER INFORMATION MESSAGE 4157 (DFMSA) ---PARAMETERS FOR SPARSE DECOMPOSITION OF DATA BLOCK  KLL     (
TYPE=RDP ) FOLLOW
                      MATRIX SIZE =      64 ROWS             NUMBER OF NONZEROES =     260 TERMS
           NUMBER OF ZERO COLUMNS =       0         NUMBER OF ZERO DIAGONAL TERMS =       0
                 CPU TIME ESTIMATE =       0 SEC                I/O TIME ESTIMATE =       0 SEC
       MINIMUM MEMORY REQUIREMENT =      20 K WORDS              MEMORY AVAILABLE =    3804 K WORDS
    MEMORY REQR'D TO AVOID SPILL =      30 K WORDS
    EST. INTEGER WORDS IN FACTOR =       1 K WORDS           EST. NONZERO TERMS =       1 K TERMS
    ESTIMATED MAXIMUM FRONT SIZE =      11 TERMS                RANK OF UPDATE =      16
*** USER INFORMATION MESSAGE 6439 (DFMSA) ---ACTUAL MEMORY AND DISK SPACE REQUIREMENTS FOR SPARSE SYM. DE-
COMPOSITION
        SPARSE DECOMP MEMORY USED =      30 K WORDS           MAXIMUM FRONT SIZE =      11 TERMS
          INTEGER WORDS IN FACTOR =       1 K WORDS      NONZERO TERMS IN FACTOR =       1 K TERMS
   SPARSE DECOMP SUGGESTED MEMORY =      30 K WORDS
```

The most important elements of the UIM 4157 message are the “MINIMUM MEMORY REQUIREMENT”, which is an estimate of the user open core memory that will allow the decomp to run, but with heavy spilling to disk. The “MEMORY REQR’D TO AVOID SPILL” will allow the decomposition to run in “in core”, i.e., without spilling to disk. These two values represent the extremes of memory requirements, the memory for optimal CPU performance is between the two. The “ESTIMATED MAXIMUM FRONT SIZE”, a function of the model, affects the memory estimates; the minimum memory is a function of the front size, and the memory to avoid spill is a function of the square of the front size. The “NUMBER OF NONZEROES” is the size of the input matrix, multiply this value by 8 to estimate the size of the input file in bytes. The sum of “EST. INTEGER WORDS IN FACTOR” and “EST. NONZERO TERMS” is the size of the output matrix, multiply the integer value by 8 on Cray and NEC and 4 on other machines, and the nonzero value by 8 to estimate the size of the output file in bytes. The “RANK OF UPDATE” is the

number of rows that will be simultaneously updated during the decomposition. This value is set by either the “rank” keyword or SYSTEM(205).

> Note: Setting SYSTEM(69)=64 will cause MSC/NASTRAN to terminate after printing UIM 4157. This can be useful for determining a job’s memory and disk space requirements.

In UIM 6439, “SPARSE DECOMP MEMORY USED” states the actual memory used in the decomposition process. Based on the execution of the module, the “SPARSE DECOMP SUGGESTED MEMORY” will result in optimal throughput performance.

5.5.5 Memory and Disk Usage Statistics

These tables are written after the job has completed, and indicate the maximum memory used by any sparse numerical module and the maximum disk used by any module during the job. A sample follows.

```
 +---------- SPARSE SOLUTION MODULES -----------+         +------------- MAXIMUM DISK USAGE -------------+
    HIWATER              SUB_DMAP        DMAP               HIWATER              SUB_DMAP        DMAP
    (WORDS)   DAY_TIME     NAME          MODULE               (MB)   DAY_TIME     NAME          MODULE
     517786   04:35:44    SEKRRS     18  DCMP              15.625   04:35:48    SESTATIC  186  EXIT
```

In the left hand table, “HIWATER WORDS” is the maximum amount of open core used by certain sparse numerical modules; “DAY_TIME” is the time of day the module ran. “SUB_DMAP NAME” is the name of the SUBDmap; “DMAP MODULE” indicates the line number and module name that made the maximum request. Similarly, in the right hand table, “HIWATER (MB)” is the maximum amount of disk space used by any module; “DAY_TIME” is the time of day the module ran. “SUB_DMAP NAME” is the name of the SUBDmap; “DMAP MODULE” indicates the line number and module name that made the maximum request.

5.5.6 Database Usage Statistics

These statistics, provided in table format, summarize the I/O activity for the DBSets.

```
*** DATABASE USAGE STATISTICS ***

+------------------- LOGICAL DBSETS -------------------+   +-------------------------- DBSET FILES --------------------------+
 DBSET      ALLOCATED   BLOCKSIZE       USED    USED      FILE       ALLOCATED     HIWATER       HIWATER  I/O TRANSFERRED
             (BLOCKS)     (WORDS)    (BLOCKS)     %                   (BLOCKS)    (BLOCKS)         (MB)             (GB)

 MASTER         5000        2048         143    2.86     MASTER         5000         143         1.117             .010
 DBALL        250000        2048           9     .00     DBALL        250000           9          .070             .000
                                                         DBALL2          300           1          .008             .000
 OBJSCR         5000        2048         121    2.42     OBJSCR         5000         121          .945             .003
 SCRATCH      500100        2048          19     .00    (MEMFILE         100          81          .633            .000)
                                                         SCRATCH      250000           1          .008             .000
                                                         SCR300       250000           1          .008             .000
                                                                                                            ============
                                                                                                 TOTAL:          .013
```

This statistical table contains two parallel tables. The “LOGICAL DBSETS” table lists each DBset while the “DBSET FILES” tables lists the component files of the DBSet. In these tables, “DBSET” is the name of the DBSet; “ALLOCATED” is the MSC/NASTRAN DBSet size limit in blocks; “BLOCKSIZE” is BUFFSIZE of the DBSet minus one. “USED (BLOCKS)” and “USED %” are the number of blocks and percent of the DBSet actually used; “FILE” is the file’s logical name associated with the DBSet to the left. Additionally, “ALLOCATED” is the number of blocks allocated by MSC/NASTRAN to the file; while “HIWATER (BLOCKS)” and “HIWATER (MB)” are the number of blocks and megabytes actually used in the file. “I/O TRANSFERRED” is the amount of I/O to the file. The last line of the DBSet Files table lists the “TOTAL I/O TRANSFERRED”.

In this example, the MASTER and OBJSCR DBSets are each composed of one file. The DBALL DBSet is composed of two files, DBALL and DBALL2; and the SCRATCH DBSet has three components, MEMFILE, SCRATCH, and SCR300.

This table can be used to determine if the DBSets and files are appropriately sized and the amount of I/O activity associated with each file. Best elapsed time performance can be obtained if the files with the greatest activity are on different physical devices (and better yet, separate I/O controllers or busses).

5.5.7 Summary of Physical File I/O Activity

This summary describes the physical file I/O for each database file.

```
*** SUMMARY OF PHYSICAL FILE I/O ACTIVITY ***

ASSIGNED PHYSICAL FILENAME                                  RECL (BYTES)  READ/WRITE COUNT  MAP WSIZE (NUM)   MAP COUNT
------------------------------------------------------------ ------------  -----------  ---------------  -----------
/tmp/65872_57.SCRATCH                                           8192             1          128KB ( 4)            1
/tmp/65872_57.OBJSCR                                            8192           378          128KB ( 4)           24
/tmp/65872_57.MASTER                                            8192          1247          128KB ( 4)           11
/tmp/65872_57.DBALL                                             8192            26          128KB ( 4)            1
/tmp/65872_57.SCR300                                            8192             1          128KB ( 4)            1
/msc/msc691/aix/SSS.MASTERA                                     8192           162        N/A              N/A
/msc/msc691/aix/SSS.MSCOBJ                                       8192           202         N/A               N/A
```

In this summary, “ASSIGNED PHYSICAL FILENAME”, “RECL”, and “MAP WSIZE and NUM” are repeated from the “Summary of Physical FIle Information” table. “READ/WRITE COUNT” is the number of GINO reads and writes that were performed on the file and “MAP COUNT” is the number of times the map window had to be remapped (these columns are only present on systems supporting mapped I/O).

This summary can be used to tune I/O performance. For mapped I/O systems, if the map count approaches the number of reads and writes, the map size and/or the number of maps should be increased. Increasing the number of maps is suggested if a module simultaneously accesses more data blocks or matrices in a file than there are windows. Increasing the size of the windows is suggested if a file contains

very large data blocks or matrices. Best elapsed time performance, with or without mapping, can be obtained if the files with the greatest activity are on different physical devices (and better yet, separate I/O controllers or busses).

5.6 Improving Network File System (NFS) Performance

The Network File System (NFS) is software allowing file systems on remote computers to appear as if they were mounted on the local computer. There are two daemons that handle NFS traffic: “nfsd” handles file system access requests by the local computer to remotely mounted file systems; “biod” handles requests by remote computers to access local file systems.

These daemons have been designed so that multiple executing copies of each daemon increase NFS traffic capacity. Two of the possible causes of poor NFS performance are a lack of sufficient daemons to handle NFS requests made by the local computer to remotely mounted file systems (nfsd), or a lack of sufficient daemons to handle NFS requests of local file systems by remote computers (biod) . The default number of daemons for nfsd and biod is typically four of each. This default is usually fine for a stand alone workstation used by one person. If you or others are accessing many remote file systems or run many MSC/NASTRAN jobs accessing file systems on file servers or remote workstations, you may need to increase the number of nfsd and biod daemons on both systems to increase NFS performance.

If you are running three or more MSC/NASTRAN jobs accessing disks on remote computers via NFS, MSC recommends increasing both nfsd and biod daemons above the standard defaults. A good starting point is twelve (12) nfsd daemons and eight (8) biod daemons per CPU on client and server computers, respectively.

Your system administrator can change both system’s configurations to start additional NFS daemons. The administrator can also monitor network statistics with “nfsstat” to ensure network traffic is being handled efficiently. Additional daemon tuning may be necessary for your specific network needs.

5.7 Creating and Attaching Alternate Delivery Databases

MSC/NASTRAN uses one delivery database, the Structured Solution Sequences (SSS), located in *install_dir*/msc705/*arch*. You may modify and store a tailored solution sequence by creating a new delivery database. This procedure is also useful to eliminate unwanted solutions from the delivery database or add additional solution sequences.

The following files are delivered in the *install_dir*/msc705/nast/del/ directory:

Filename	Description
buildsss	Script used to build delivery database. Script Used to Build Delivery Database.
*.dat	SubDMAP source.
*.dck	SubDMAP source that must be preprocessed by MSCFPP.
*.ddl	NDDL source.

Rebuild Using MSC-Supplied Source

To rebuild the delivery database using the MSC-supplied source, the following procedure is used:

1. Change the working directory to a directory other than the *install_dir*/msc705/nast/del or the *install_dir*/msc705/*arch* directories. For example,

   ```
   cd $HOME/new-del
   ```

2. Rebuild the delivery database.

   ```
   msc705 buildsss
   ```

Upon completion of this procedure, the delivery files SSS.MASTERA, SSS.MSCOBJ, and SSS.MSCSOU are created. These files are attached with the “delivery” keyword (see Appendix B).

These file may be installed in the master architecture directory (if you have write access) with the command:

```
cp SSS.* install_dir/msc705/arch
```

Rebuild Using Modified Source

To build a modified delivery database, the following procedure is used:

1. Change the working directory to a directory other than the *install_dir*/msc705/nast/del or the *install_dir*/msc705/*arch* directories. For example,

   ```
   cd $HOME/new-del
   ```

2. Copy the subDMAP and NDDL source files that are to be modified to the current directory.

   ```
   cp install_dir/msc705/nast/del/subDMAP.dat .
   cp install_dir/msc705/nast/del/subDMAP.dck .
   cp install_dir/msc705/nast/del/nddl.ddl .
   ```

   where *subDMAP* and *nddl* are the specific files to be modified.

3. Modify the desired subDMAP and/or NDDL source files.

   ```
   vi *.dat *.dck *.ddl
   ```

4. Rebuild the delivery database.

   ```
   msc705 buildsss src=.
   ```

Upon completion of this procedure, the delivery files SSS.MASTERA, SSS.MSCOBJ, and SSS.MSCSOU are created. These files are attached with the “delivery” keyword (see Appendix B).

These files may be installed in the master architecture directory (if you have write access) with the command:

```
cp SSS.* install_dir/msc705/arch
```

CHAPTER 6

HOW TO USE THE UTILITY PROGRAMS

This chapter describes how to use the various MSC/NASTRAN utility programs. These utilities are grouped by function as follows:

1. Moving results database (XDB) files between dissimilar computers.
   - RECEIVE
   - TRANS
2. Converting MSC/NASTRAN plot files to PostScript.
   - NEUTRL
   - PLOTPS
3. Converting a neutral-format OUTPUT2 file to binary format.
   - RCOUT2
4. Reformatting MSC/NASTRAN Version 67 heat-transfer and optimization data files into current formats.
   - HEATCONV
   - OPTCONV
5. Graphical user interface for submitting and monitoring MSC/NASTRAN jobs.
   - XMONAST
   - XNASTRAN
6. Estimating the requirements of an MSC/NASTRAN job and making suggestions on improving its performance.
   - ESTIMATE

7. Accumulating and summarizing MSC/NASTRAN accounting data.
   - MSCACT
8. Compiling the message catalog.
   - MSGCMP

Descriptions on using the utilities follow in alphabetical order. At the end of the section, instructions on how to build the source code utilities are included.

6.1 Using ESTIMATE

ESTIMATE may be used to estimate the memory and disk requirements for MSC/NASTRAN jobs and make suggestions on improving the performance of these jobs. ESTIMATE will read the input data file and estimate the job's memory and disk requirements. The ESTIMATE program is most accurate in predicting the requirements of static analyses that do not have excessive output requests. The memory requirements for normal modes analyses using the Lanczos Method are reasonably accurate; however, the disk requirements are dependent upon the number of modes, this is a value that ESTIMATE cannot determine. Memory and disk requirements for other solutions are less accurate.

The basic format of the "estimate" command is

```
msc705 estimate input_file [keywords]
```

where *input_file* is the name of the data file. If the file suffix of the input data file is ".dat", it may be omitted from the command line.

ESTIMATE processes keywords using the following precendence to resolve conflicts when keywords are duplicated (with 1 representing the highest precendence):

1. The bulk data file.
2. The command line.
3. The nastran INI and RC files (if "nastrc=yes" is specified).
4. *data-file-directory*/.estimaterc (where *data-file-directory* is the directory containing the input data file).
5. $HOME/.estimaterc
6. estimate.ini (in the directory containing the ESTIMATE executable).

6.1.1 Keywords

application application=NASTRAN Default: NASTRAN

Defines the application. The default is based on the SOL or LINK name specified in the executive section.

Note: This keyword should only be set to “NASTRAN”.

bpool bpool=*value* Default: 27 (Cray and NEC); 37 (all others)

Same as MSC/NASTRAN keyword (see Section B.1 of Appendix B). This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.

buffsize buffsize=*value* Default: 4097 (Cray and NEC); 2049 (all others)

Same as MSC/NASTRAN keyword (see Section B.1 of Appendix B). This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.

dskco dskco=*value* Default: 1.0

Allows you to specify a constant factor that is either more or less conservative than the default.

Example: `msc705 estimate example dskco=2`

This will double the disk space estimate.

enable The “enable” keyword can be used to explicitly enable rules. This may be useful to enable a rule that was automatically suppressed when a value was assigned. For example, the following command will now calculate the estimated memory requirements for a job even though a value for memory was specified on the command line:

Example: `msc705 estimate myjob memory=5mb enable=10`

estimatedof estimatedof={yes|no} Default: No

Indicates if the number of degrees of freedom are to be estimated. By default, ESTIMATE will count the DOF. This process takes time, but it is generally more accurate. Specifying “estimatedof=no” will result in a less accurate, but faster, estimate of the DOF. The presence of any MESH entries in the Bulk Data will force “estimatedof=yes”.

memco memco=*value* Default: 1.0

Allows you to specify a constant factor that is either more or less conservative than the default.

Example: `msc705 estimate example memco=2`

This setting will double the memory estimate.

memory memory=*memory_size* Default: 4MW

Same as MSC/NASTRAN keyword (see Section B.1 of Appendix B). This keyword cannot appear in an ESTIMATE RC file if "nastrc=yes" is specified.

method method=*sid* Default: *None*

Selects a METHOD for dynamics jobs if a METHOD Case Control command is not present or multiple METHOD Case Control commands are present in the data file. By default, ESTIMATE will choose the first METHOD found.

mode mode={estimate|suggest|modify} Default: suggest

Selects the program operating mode. Specifying "mode=estimate" will result in memory and disk estimates only. Specifying "mode=suggest", the default, will estimate memory and disk requirements for the current job configuration, suggest modifications to improve the performance, and provide estimates for the memory and disk requirements of the suggested configuration. Specifying "mode=modify" does all that "mode=suggest" does plus actually make the suggested changes to your data file. See "out" to specify the new data file's name and information on organizing your input file.

Note: If "mode=modify" is specified, and ESTIMATE detects errors in the input file or encounters valid Bulk Data that is not understood by ESTIMATE, the program will revert to "mode=suggest".

Example: `msc705 estimate example mode=estimate`

The memory and disk requirements for the current job are displayed.

Example: `msc705 estimate example`

The memory and disk requirements for the current job, suggestions for improving performance, and memory and disk requirements for the suggested configuration are displayed.

Example: `msc705 estimate example mode=modify`

The memory and disk requirements for the current job, suggestions for improving performance, and estimates of memory and disk

requirements for the suggested configuration are displayed. If, and only if, modifications to “example.dat” are suggested, the original input file is versioned (given indices) and the revised data file is written to “example.dat”.

mpc mpc=*sid* Default: *None*

Selects an MPC if an MPC Case Control command is not present or multiple MPC Case Control commands are present in the data file. By default, ESTIMATE will choose the first MPC found.

nastrc The “nastrc” keyword allows you to select the type of RC file processing invoked by the ESTIMATE utility. Setting “nastrc=yes”, the default, will process the standard MSC/NASTRAN RC files before the standard ESTIMATE RC files, i.e., $HOME/.estimaterc and “*data-file-directory/*.estimaterc”, are processed. Setting “nastrc=no” will only process the standard ESTIMATE RC files.

out out=*pathname* Default: *input filename*

Specifies the name of the output file if “mode=modify” is specified and modifications of the data file are actually required. By default, the original file is versioned (given indices) and the revised data file is written to the original input file’s name. (See Section 4.1.1.)

Example: `msc705 estimate example mode=modify`

If modifications to “example.dat” are suggested, the original input file is versioned (given indices) and the revised data file is written to “example.dat”.

Example: `msc705 estimate example mode=modify out=modified`

The revised data file is written to “modified”.

Note: In order to minimize the amount of data duplicated between the original input file and the modified file, MSC recommends that the Bulk Data that is not subject to modification by ESTIMATE (i.e., all Bulk Data except PARAM and EIGRL entries) be placed in an INCLUDE file.

An example of the recommended input file organization is:

NASTRAN statements
FMS statements
Executive
CEND
Case Control
BEGIN BULK
PARAM,...
$
EIGRL,...
$
INCLUDE *file.bulk*
$
ENDDATA

pause	pause=keyword	Default: no

Pause ESTIMATE before exiting to wait for the “Enter” or “Return” key to be pressed. This can be useful when ESTIMATE is embedded within another program. The values are “fatal”, “information”, “warning”, “yes”, and “no”. Setting “pause=yes” will unconditionally wait; “pause=fatal” will only wait if a fatal message has been issued by ESTIMATE; “pause=information” and “pause=warning” will similary wait only if an information or warning message has been issued. The default is “pause=no”, i.e., do not wait when ESTIMATE ends.

real	Same as MSC/NASTRAN keyword (see Section B.1 of Appendix B). This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.	
report	report={normal|keyword}	Default: Normal

Specifies the program’s report format. The “report=normal” format is intended to be read by you. The “report=keyword” format is intended to be read by a program.

smemory	smemory=*value*	Default: 0 (Cray and NEC); 100 (all others)

Same as MSC/NASTRAN keyword (see Section B.1 of Appendix B). This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.

spc — spc=*sid* — Default: *None*

Selects an SPC if an SPC Case Control command is not present or multiple SPC Case Control commands are present in the data file. By default, ESTIMATE will choose the first SPC found.

suppress — suppress=*list* — Default: *None*

Specifies rules that are to be suppressed when “mode=suggest” or “mode=modify” is specified. See Section 6.1.2 for the list of rules. If no value is specified, i.e., “suppress=”, then any rules previously suppressed are enabled. Multiple rules can be suppressed by using the keyword multiple times or by specifying a comma-separated list.

Example:
```
msc705 estimate example suppress=1
```

Suppress rule 1, the rule controlling BUFFSIZE.

Example:
```
msc705 estimate example suppress=1,6
```
or
```
msc705 estimate example suppress=1 suppress=6
```
or
```
msc705 estimate example suppress=2 suppress=
suppress=1,6
```

Suppress rules 1 and 6.

verbose — verbose={yes|no} — Default: No

Specifies the amount of information to be displayed. Specifying “verbose=yes” will generate a much larger amount of output. The additional information includes a more detailed summary of the input file, the parameters used in estimating the memory and disk requirements, and the estimates for the original file, even when “mode=suggest” or “mode=modify” is specified.

version — version=*string* — Default: 70.5 (MSC/NASTRAN)

Specifies the version of MSC/NASTRAN for which the estimates are to be targeted. The version will affect the estimated memory requirements and the actions of various rules (see Section 6.1.2). This keyword cannot appear in an ESTIMATE RC file if “nastrc=yes” is specified.

Note: Supported versions are 68.2, 69, 69.1, 70 and 70.5.

wordsize — wordsize={32|64}[,{32|64}] — Default: 64 (Cray and NEC); 32 (all others)

Specifies the word size of the estimate’s target computer. By default, ESTIMATE’s calculations will be appropriate the the current computer. This keyword may be used to specify estimates for a computer with a different word size. A comma-separated list of values may be specified when estimates and suggestions for

multiple machines are desired. If “mode=modify” was specified, the modification are based on the last word size specified.

6.1.2 Rules

ESTIMATE has a fixed rule base that it uses to make suggestions for improvement. Any of the rules may be suppressed with the “suppress” keyword. The current rules are:

1. Set recommended BUFFSIZE.

BUFFSIZE=2049	*DOF* < 2 000 and *wordsize* = 32
BUFFSIZE=4097	*DOF* < 2 000 and *wordsize* = 64
BUFFSIZE=4097	2 000 ≤ *DOF* and *DOF* < 40 000
BUFFSIZE=8193	40 000 ≤ *DOF* and *DOF* < 100 000
BUFFSIZE=16385	100 000 ≤ *DOF* and *DOF* < 400 000
BUFFSIZE=32769	*DOF* ≥ 400 000

2. Use default BPOOL.

BPOOL=37	*wordsize* = 32	
BPOOL=20	*wordsize* = 64	and *version* < 70.5
BPOOL=27	*wordsize* = 64	and *version* ≥ 70.5

3. Suppress symmetric decomposition if not enough memory for sparse.

   SYSTEM(166)=0

4. Make all open core available to modules.

   Delete HICORE.

5. Select the sparse solver.

Delete SPARSE	*density* ≤ 12.0
Delete USPARE	*density* ≤ 12.0
SPARSE=1	*density* > 12.0
USPARSE=0	*density* > 12.0

6. Force default rank size.

   Delete SYSTEM(198)
   Delete SYSTEM(205)

7. Do not sequence.

PARAM,NEWSEQ,-1	*release* < 69

8. Use default Lanczos parameters.

   EIGRL,...,V1=“”
   EIGRL,...,MAXSET=15

9. Use default SMEMORY.

   INIT SCRATCH (MEM=100) *wordsize* = 32
   INIT SCRATCH (MEM=0) *wordsize* = 64

10. Use estimated memory size.

    memory=*estimated memory*

11. Use default RAM.

    INIT MASTER (RAM=30000)

12. Real.

    Delete REAL.

13. Do not use Supermodule.

    Delete PARAM,SM,YES.

14. Do not use Parallel Lanczos.

    Delete NUMSEG.

6.1.3 Examples

The ESTIMATE program can be used in several ways. The default mode will make suggestions on improving the performance of MSC/NASTRAN and estimate the resource requirements of the job assuming the suggested parameters.

```
msc705 estimate myjob
```

To get an estimate of the job using the current parameters, use the command:

```
msc705 estimate myjob mode=estimate other_estimate_keywords
```

To have a new input file generated with the suggested changes, use the command:

```
msc705 estimate myjob mode=modify other_estimate_keywords
```

To run MSC/NASTRAN with the memory estimated by ESTIMATE, use:

```
msc705 nastran myjob memory=estimate other_nastran_keywords
```

6.2 Using HEATCONV

HEATCONV may be used to reformat an existing heat-transfer Bulk Data file used in MSC/NASTRAN prior to Version 68 into a format compatible with Version 68 or later. The operations performed by this program are described in the

MSC/NASTRAN Release Notes for Version 68. The basic format of the “heatconv” command is

```
msc705 heatconv input_file [keywords]
```

where *input_file* is the name of the heat-transfer data file. If the file suffix of the old data file is “.dat”, it may be omitted from the command line.

6.2.1 Keywords

output=*output_file* Default: *input_file*

This option specifies the name of the reformatted data file. By default, the old output file is renamed by appending the file suffix “.old”; the new file is the original name of the input file. If an output file is specified using this option, the original input filename is unchanged.

6.2.2 Examples

To execute the program, enter the following command:

```
msc705 heatconv example
```

The Version 68-compatible output is written to

```
example.dat
```

The original data file is renamed to example.dat.old.

6.3 Using MSCACT

MSCACT may be used to generate usage reports from the accounting files generated by MSC/NASTRAN when the “acct=yes” keyword is used. The basic format of the “mscact” command is

```
msc705 mscact [keywords] acc-file [acc-file ...]
```

where *acc-file* are the names of the accounting file(s) to be summarized.

Note: The keywords only affect files listed after the keyword.

6.3.1 Keywords

perfile perfile={yes|no} Default: No

Specifies the summary is to be printed on a per file basis. If “perfile=yes” is specified, a summary of each file will be individually printed. By default, the summary will include all files.

sortby sortby=keyword Default: name

Sort the report as specified by the keyword. The keywords are:

Keyword	Sort Order
count	Sort by third report column.
cpu	Sort by second report column.
name	Sorth by first report column.
none	Do not sort report; report is ordered as found in data file.

Setting “sortby=none” will produce a report very similar to the previous versions of this utility.

summary summary=*keyword* Default: None

Selects the type of summary. If “summary=none” is specified, the total CPU for all entries will be displayed. Otherwise, one of the following summary types may be selected:

Keyword	Type of Summary
acdata	By acdata
acid	By account ID (acid)
date	By execution date
jid	By job name
product	By product name
sol	By SOL
user	By user name
version	By product name and version

Note: Prior to MSC/NASTRAN V70.5, the UNIX syntax “-s *keyword*” was used, V70.5 dropped support for that syntax.

6.3.2 Examples

To summarize accounting data across all files:

```
cd install_dir/acct
msc705 mscact file1 file2
 file1 file2:
  Total: cpu-sec count
```

where *filei* are the filenames, *cpu-sec* is the total CPU seconds across all files, and *count* is the number of entries accumulated across all files.

To summarize accounting data from individual files:

```
cd install_dir/acct
msc705 mscact perfile=yes file1 file2
 file1:
  Total:     cpu-sec count
 file2:
  Total:     cpu-sec count
```

where *file*, is the name of each file, *cpu-sec* is the total number of CPU seconds, and *count* is the number of entries in each file.

To summarize accounting data in individual files by user:

```
cd install_dir/acct
msc705 mscact summary=user perfile=yes file1 file2
 file1:
  user1:     cpu–sec1 count1
  user2:     cpu–sec2 count2
   ...
  Total:     cpu-sec count
 file2:
  user1:     cpu–sec1 count1
  user2:     cpu–sec2 count2
   ...
  Total:     cpu-sec count
```

where *filei* are the filenames of each file, *useri* are the names, *cpu-seci* are the total CPU seconds for each user, *counti* are the number of entries accumulated for each user, *cpu-sec* is the number of total CPU seconds, and *count* is the number of entries in each file.

6.3.3 Accounting File Format

A separate file is created for each month of each year and is named *install_dir*/acct/msc*yymm*.acc where *yy* are the last two digits of the year and *mm* is the month (01 to 12). Each month's file is independent of every other file.

The accounting file begins with three header records followed by detail records, one detail record for each MSC/NASTRAN job run during the given month and year.

Comments, indicated by a hash mark "#" as the first character of the line, may be placed anywhere in the file.

Detail records (any line after the third line that does not begin with a hash mark) include the following data:

1. The day the job was started (i.e., Sun., Mon., Tue., Wed., Thu., Fri., or Sat.).
2. The month the job was started (i.e., Jan., Feb., Mar., Apr., May, Jun., Jul., Aug., Sep., Oct., Nov., or Dec.).
3. The date of the month the job was started (i.e., 01 through 31).
4. The time the job was started (i.e., hh:mm:ss, where hh is 00 through 23, mm is 00 through 59, and ss is 00 through 59).
5. The time zone (i.e., the "TZ" environment variable).
6. The year the job was started (four digits).
7. The name of the user running the job.
8. The job's output filename.
9. The analysis application, e.g., MSC/NASTRAN.
10. The version of the application (e.g., 70.5).
11. The SOL used by the job (e.g., 101, SESTATICS).
12. The total CPU time, in seconds, of the job (from the F04 file).
13. The cumulative CPU time, in seconds, of all detail records up to and including this record.
14. The cumulative CPU time, in minutes, of all detail records up to and including this record.
15. The account ID as specified by the nastran command's "acid" keyword.
16. The account data as specified by the nastran command's "acdata" keyword.

Note: The cumulative times (fields 13 and 14) are for historical purposes only. These values are ignored.

6.4 Using MSGCMP

MSGCMP compiles a text message file and generates a binary message catalog. The basic format of the command is

```
msc705 msgcmp text_file [message_catalog]
```

where *text_file* is the name of an existing text message file or is "-" to read from stdin, and *message_catalog* is the optional name of the message catalog that will be written. The suffix of the text file must be ".txt". If a message catalog is not named, the message catalog will be written in the local directory as "*text_file*.msg". The message catalog can be tested using the "msgcat" keyword described in Appendix B.

The utility can also regenerate a text file from an existing message catalog using the command

```
msc705 msgcmp message_catalog.msg [text_file]
```

where *message_catalog*.msg is the name of an existing message catalog and *text_file* is the optional name of a text file that will be written. The suffix of the message catalog must be ".msg" and must be entered on the command line. If a text file is not named, the text file is written to stdout.

The text source file for the standard message catalog is "*install_dir*/msc705/util/analysis.txt". The standard message catalog is "*install_dir*/msc705/*arch*/analysis.msg".

6.4.1 Examples

The following command will compile the message catalog from a text file named "myfile.txt"

```
msc705 msgcmp myfile
```

The message catalog will be named "myfile.msg". This catalog may be used with the nastran command

```
msc705 nastran myjob msgcat=myfile.msg
other_nastran_keywords
```

Note: Message catalogs are machine dependent. Table 6-1, "Binary File Compatibility" identifies the systems that are binary compatible; binary compatible systems can use multiple copies of the same message file.

6.5 Using NEUTRL

NEUTRL converts a binary-format plot file into a neutral-format plot file. The basic format of the “neutrl” command is

```
msc705 neutrl binary_plot_file [keywords]
```

where *binary_plot_file* is the name of a binary plot file. If the file suffix of the plot file is “.plt”, it may be omitted from the command line.

6.5.1 Keywords

dump={no|yes} Default: no

This option enables a raw print of each plot command to be made before it is processed. This print is used for debugging purposes only.

output=*output_file* Default: *binary_plot_file*.neu

This option specifies the name of the neutral-format file. If “out=–” is specified, the neutral plot file is written to stdout. By default, the output file is the name of the input file with the new suffix “.neu”.

verbose={no|yes} Default: yes (output is a disk file)
no (output is stdout)

This option specifies whether processing messages are to be written.

6.5.2 Examples

To execute the program, enter the following command:

```
msc705 neutrl example1
```

The name of the output file is

```
example1.neu
```

6.6 Using OPTCONV

OPTCONV may be used to reformat an existing optimization Bulk Data file used in MSC/NASTRAN prior to Version 68 into a format compatible with Version 68 or later. The operations performed by this program are described in the *MSC/NASTRAN Release Notes for Version 68*. The basic format of the "optconv" command is

```
msc705 optconv input_file [keywords]
```

where *input_file* is the name of the dynamic-optimization data file. If the file suffix of the old data file is ".dat", it may be omitted from the command line.

6.6.1 Keywords

output=*output_file* Default: *input_file*

This option specifies the name of the reformatted data file. By default, the old output file is renamed by appending the file suffix ".old"; the new file is the original name of the input file. If an output file is specified using this option, the original input filename is unchanged.

6.6.2 Examples

To execute the program, enter the following command:

```
msc705 optconv example
```

The Version 68-compatible output is written to

```
example.dat
```

The original data is renamed to example.dat.old.

6.7 Using PLOTPS

PLOTPS reads plotting commands from a single MSC/NASTRAN binary- or neutral-format plot file and produces a file that can be printed on a PostScript device. The basic format of the “plotps” command is

```
msc705 plotps input_plot_file [keywords]
```

where *input_plot_file* is the name of the plot file generated by MSC/NASTRAN or NEUTRL. A neutral-format plot file can be read from stdin by specifying “–” as the filename. The plot file suffix “.plt” does not have to be specified on the command line.

6.7.1 Keywords

begin=*first_frame_to_plot* Default: 1

end=*last_frame_to_plot* Default: 999999

These two options can be used to plot a selected range of plot frames.

color={no|yes} Default: no

Enables or disables color pens. Setting “color=no”, the default, will assign a solid line to pen 1 and various dashed lines to pens 2, 3, and 4. Setting “color=yes” will assign black to pen 1, red to pen 2, green to pen 3, and blue to pen 4. All text and axes will always be written with a solid black pen.

cscale=*character_scale_factor* Default: 1.0

This option specifies a scale factor for all characters and special symbols on the plot. By default, characters and special symbols are 9 points (about 0.125 inch). The scale value, if specified, is also applied to characters and special symbols.

dump={no|yes} Default: no

This option enables a raw print of each plot command before it is processed. This print is used for debugging purposes only.

format={binary|neutral} Default: binary

The option is used to specify the input file format. If the file type of the input file is “.neu” or the plot file is read from stdin, then “format=neutral is assumed.

height=*printable_page_height* Default: 10.0 inches

The option is used to specify the printable page height. The actual page is assumed to be 1 inch larger.

output=*output_file* Default: *plot-file*.ps

This option specifies the name of the PostScript output file. If a neutral-format plot file is read from stdin, the default output filename is “plotps.ps”. If “out=–” is specified, the PostScript output is written to stdout. By default, the output file is named the name of the input file with the new suffix “.ps”.

rotate={automatic|no|yes} Default: automatic

This option controls the orientation of the generated image. If “rotate=automatic” is specified, the program orients the image so that the long direction of the image is aligned with the long direction of the page. If “rotate=no” is specified, the image is generated with the horizontal axis aligned with the bottom edge of the page. If “rotate=yes” is specified, the image is generated with the horizontal axis aligned with the right edge of the page.

scale=*plot_scale_factor* Default: 1.0

This option specifies a scale factor for all elements of the plot.

Note: The program will not attempt to print a multipage image if this option is used to enlarge the image beyond the size of the available page.

verbose={no|yes} Default: yes (output is a disk file)
no (output is stdout)

This option specifies whether processing messages are to be written.

width=*printable_page_width* Default: 7.5 inches

The option is used to specify the printable page width. The actual page is assumed to be 1 inch larger.

6.7.2 Examples

- To translate a binary-format plot file named example1.plt into PostScript, use

```
msc705 plotps example1
```

The name of the output file is

```
example1.ps
```

- To translate a neutral-format plot file named example2.neu into PostScript, use

```
msc705 plotps example2.neu
```

The name of the output file is

```
example2.ps
```

6.8 Using RCOUT2

RCOUT2 is used to convert a neutral-format OUTPUT2 file generated by MSC/NASTRAN into a binary-format OUTPUT2 file. Since MSC/NASTRAN can read and write binary-format and neutral-format OUTPUT2 files, this utility is generally used to construct a binary OUTPUT2 file for a third-party program that can only read a binary OUTPUT2 file. The basic format of the "rcout2" command is

```
msc705 rcout2 neutral_output2_file [keywords]
```

where *neutral_output2_file* is the name of the neutral-format OUTPUT2 file. If the file suffix of the OUTPUT2 file is ".on2", it may be omitted from the command line.

> Note: Prior to MSC/NASTRAN V70.5, the suffixes ".neut" and ".out2" were used, V70.5 changed them to the more transportable ".on2" and ".op2," respectively.

6.8.1 Keywords

output=*binary_file* Default: *neutral_file*.op2

This option specifies the name of the binary OUTPUT2 file. By default, the output file is the name of the input file with the new suffix ".op2".

6.8.2 Examples

To execute the program, enter the following command:

```
msc705 rcout2 example
```

The name of the output file is

```
example.op2
```

6.9 Using RECEIVE

RECEIVE is used to convert a neutral results database file (NDB) into a binary results database file (XDB). The basic format of the "receive" command is

```
msc705 receive neutral_xdb_file [keywords]
```

where *neutral_xdb_file* is the name of the NDB file. If "–" is specified as the neutral format database file, the file is read from stdin. If the file suffix of the NDB file is ".ndb", it may be omitted from the command line.

Note: Prior to MSC/NASTRAN V70.5, the suffix ".ntrl" was, V70.5 changed this to the more transportable ".ndb".

6.9.1 Keywords

output=*binary_xdb_file* Default: *neutrl_xdb_file*.xdb

This option specifies the name of the binary results database file. By default, the output file is the name of the input file with the new suffix ".xdb". If the neutral format database file was read from stdin, the default output filename is "receive.xdb". A binary XDB file cannot be written to stdout.

verbose={no|yes} Default: yes (output is a disk file)
no (output is stdin)

This option specifies whether processing messages are to be written.

6.9.2 Examples

To execute the program, enter the following command:

```
msc705 receive example
```

The name of the output file is

```
example.xdb
```

An XDB file can be transferred directly from a remote system with the following command:

Cray, HP

```
$ remsh node msc705 trans binary_xdb_file out=– \
      | msc705 receive – out=binary_xdb_file
```

See the remsh(1) man page for further information.

NEC

```
$ /usr/ucb/rsh node msc705 trans binary_xdb_file out=– \
      | msc705 receive – out=binary_xdb_file
```

See the rsh(1) man page for further information.

All others

```
$ rsh node msc705 trans binary_xdb_file out=– \
      | msc705 receive – out=binary_xdb_file
```

See the rsh(1) man page for further information.

6.10 Using TRANS

A results database file (XDB) may be exchanged between computer systems that have binary file compatibility as defined by the following table. Otherwise, the TRANS utility is required. TRANS converts an XDB file that is generated by MSC/NASTRAN to an equivalent character file that can be sent across a network to another computer. RECEIVE converts the character file back into an XDB file for postprocessing.

Binary File Compatibility

The following table lists the compatibility of binary files between various computer systems supported by current or previous versions of MSC products. Note that not all of these combinations have been tested by MSC. Please report any compatibility problems encountered to your MSC representative.

Table 6-1. Binary File Compatibility.

MSC/NASTRAN	Architecture			Postprocessor Platform						
	IEEE	Byte Order	Word Size	Digital Alpha	Digital VAX	HP	IBM RS/6000	SGI	SUN SPARC	Windows NT
Cray UNICOS	No	Big	64	TR	TR	TR	TR	TR	TR	TR
Cray UNICOS IEEE T90	Yes	Big	64	TR	TR	TR	TR	TR	TR	TR
Digital Alpha UNIX	Yes	Little	32	Copy	TR	TR	TR	TR	TR	Copy
Digital Alpha OpenVMS	Yes	Little	32	TR	TR	TR	TR	TR	TR	TR
Digital VAX OpenVMS	No	Little	32	TR	Copy	TR	TR	TR	TR	TR
Fujitsu UXP	Yes	Big	32	TR	TR	TR	TR	TR	TR	TR
HP 9000	Yes	Big	32	TR	TR	Copy	Copy	Copy	Copy	TR
HP CONVEX Exemplar	Yes	Big	32	TR	TR	Copy	Copy	Copy	Copy	TR
HP CONVEX C-Series	Yes	Big	32	TR	TR	Copy	Copy	Copy	Copy	TR
Hitachi S-Series HI-OSF/1-MJ	No	Big	32	TR	TR	TR	TR	TR	TR	TR
IBM MVS/XA, VM	No	Big	32	TR	TR	TR	TR	TR	TR	TR
IBM RS/6000 AIX	Yes	Big	32	TR	TR	Copy	Copy	Copy	Copy	TR
NEC Super-UX	Yes	Big	64	TR	TR	TR	TR	TR	TR	TR
SGI IRIX	Yes	Big	32	TR	TR	Copy	Copy	Copy	Copy	TR
SUN SPARC Solaris	Yes	Big	32	TR	TR	Copy	Copy	Copy	Copy	TR
Windows NT	Yes	Little	32	Copy	TR	TR	TR	TR	TR	Copy

Notes: 1. Copy indicates that XDB files can be transferred between the systems without using TRANS and RECEIVE.

2. TR indicates that XDB files must be transferred between the systems using TRANS and RECEIVE.

3. Windows NT includes both Intel and Digital Alpha processors.

The first column on the left of the table lists various MSC/NASTRAN platforms. The second and third columns list basic architectural features of the computer, specifically whether the computer conforms to ANSI/IEEE Standard 754-1985 (the *IEEE Standard for Binary Floating-Point Arithmetic)* and byte ordering methods (Big Endian or Little Endian) used by the computer. The remaining columns list postprocessor platforms.

Running TRANS

TRANS is used to convert a binary results database file (XDB) into a neutral results database file (NDB) that may be copied to any other computer. The basic format of the "trans" command is

```
msc705 trans binary_xdb_file [keywords]
```

where *binary_xdb_file* is the name of the XDB file. An XDB file cannot be read from stdin. If the file suffix of the XDB file is ".xdb", it may be omitted from the command line.

6.10.1Keywords

alphabet={48|64} Default: 64

Choose the 48- or 64-character conversion table.

output=*neutral-xdb-file* Default: *binary-xdb-file*.ndb

This option specifies the name of the neutral format database file. If "out=–" is specified, the neutral-format database file will be written to stdout. By default, the output file name is the input file name with the new suffix ".ndb".

Note: Prior to MSC/NASTRAN V70 the suffix ".ntrl" was used, V70 changed this to the more transportable ".ndb".

verbose={no|yes} Default: yes (output is a disk file)
no (output is stdout)

This option specifies whether processing messages are to be written.

6.10.2Examples

To execute the program, enter the following command:

```
msc705 trans example
```

The name of the output file is

```
example.ndb
```

An XDB file can be transferred directly to a remote system with the following commands:

Cray, HP

```
msc705 trans binary_xdb_file out=– \
      | remsh node [–l user] msc705 receive –
out=binary_xdb_file
```

See the remsh(1) man page for further information.

NEC

```
msc705 trans binary_xdb_file out=– \
      | /usr/ucb/rsh node [–l user] msc705 receive –
out=binary_xdb_file
```

See the rsh(1) man page for further information.

All others

```
msc705 trans binary_xdb_file out=– \
      | rsh node [–l user] msc705 receive –
out=binary_xdb_file
```

See the rsh(1) man page for further information.

6.11 Using XMONAST

XMONAST is a simple OSF/Motif GUI to monitor MSC/NASTRAN jobs. The Motif runtime libraries along with an X-capable terminal/monitor are required to run XMONAST. The basic format of the “xmonast” command is:

```
msc705 xmonast list_of_files &
```

XMONAST is a point-and-click text file viewer that can view the output of your MSC/NASTRAN job as it progresses. The viewer can be started in three ways:

- From the command line:

```
msc705 xmonast list_of_files &
```

where *list_of_files* are text files that will be displayed. XMONAST will read stdin if "-" is specified.

- From the nastran command (with manual termination of XMONAST):

```
msc705 nastran data_file ... xmon=yes
```

where only the LOG file will be displayed. See the "xmon" keyword in Section B.1 of Appendix B for more details on this method.

- From the nastran command (with automatic termination of XMONAST when the MSC/NASTRAN job ends):

```
msc705 nastran data_file ... xmon=kill
```

where only the LOG file will be displayed. See the "xmon" keyword in Section B.1 of Appendix B for more details on this method.

The selected files will be displayed in scrollable windows. Once the entire file as it currently exists has been displayed, XMONAST will enter an infinite loop waiting for additional text. This process will continue until the "Exit" push button is selected, or until the MSC/NASTRAN job has completed if XMONAST is started from the nastran command with "xmon=kill".

You may temporarily suspend updates to the scrollable windows (e.g., to browse the output) by selecting the "Pause Output" push button. To resume output, select the same button, now labeled "Continue Output".

If a LOG file is being displayed, the "Kill Job" push button may be used to cancel a running MSC/NASTRAN job. This will send an interrupt kill signal (SIGKILL) to your MSC/NASTRAN job. Unless started by "xmon=kill", you may still scroll through the output data files after terminating a job.

To exit XMONAST, select the "Exit" push button or select "File→Exit" from the menu bar.

6.11.1 Menu Bar Commands

File

Re-Open Files	This command rereads the input files from the beginning (does not function for stdin).
Exit	This command writes various resources to "$HOME/Xmonast" and exits XMONAST.

Kill

Sure Kill — This command sends signal SIGKILL (9) to the MSC/NASTRAN job. This command is only enabled if an MSC/NASTRAN Version 68.1 (or later) LOG file is being displayed in one of the panes. If more than one LOG file is being displayed, only one of the jobs will be killed.

Help

On-line Documentation... — This command starts the online documentation application. The command used by XMONAST to start the application is specified by the “Xmonast*docname” resource. The default value is “mne”, i.e., the MSC/NASTRAN Encyclopedia (a separately installed product).

Program Version... — This command displays the program version in a pop-up window. Press the “OK” button to dismiss the pop-up window.

6.11.2 Buttons

Pause Output — This button suspends output to the panes so that they can be examined. The button will change to “Continue Output” while the output is paused. Pressing “Continue Output” will resume output to the panes.

Kill — This button sends signal SIGKILL (9) to the MSC/NASTRAN job. The button is only enabled if an MSC/NASTRAN V68.1 (or later) LOG file is being displayed in one of the panes. If more than one LOG file is being displayed, only one of the jobs will be killed.

Exit — This button writes the current resource settings to “$HOME/Xmonast” and exits.

6.11.3 Examples

To monitor the F06, F04, and LOG files of an already running job named example, use:

```
msc705 xmonast example.f06 example.f04, example.log &
```

To run an MSC/NASTRAN job named example in the background and monitor the LOG file as the job progresses, use:

```
msc705 nastran example batch=yes xmonast=yes
```

XMONAST will continuously display the LOG file until the “Exit” push button is selected.

6.11.4 Resources

The default resource file, “/usr/lib/X11/app-defaults/Xmonast”, and, if it exists, your resource file, “$HOME/Xmonast”, are read at application startup. Your resource file is completely rewritten if XMONAST is terminated using the “File→Exit” menu item or the “Exit” button at the bottom of the window. Your resource file is not written if you terminate XMONAST using the “window→Close” menu item. Documentation of the XMONAST resources can be found in the standard MSC resource file, “*install_dir*/msc705/*arch*/Xmonast”.

6.12 Using XNASTRAN

XNASTRAN is a simple OSF/Motif Graphical User Interface to submit MSC/NASTRAN jobs. The Motif runtime libraries along with an X-capable terminal/monitor are required to run XNASTRAN. The basic format of the “xnastran” command is:

```
msc705 xnastran &
```

The XNASTRAN command allows you to select the input file, set job options (i.e, command line keywords), and submit the job to MSC/NASTRAN.

6.12.1 Menu Bar Commands

File

Exit — This command and exits XNASTRAN.

Setup

MSC/NASTRAN Version... — This command allows you to enter the “MSC/NASTRAN Version Label” defining the product name, the default is “MSC/NASTRAN V70.5” and the “Run Command” that submits a job, the default is “/usr/bin/msc705 nastran”. The “Select File” button will bring up a standard file selection tool allowing you to find the run command file. The “Accept” button will accept the changes and cancel

the dialog, the “Cancel” button will cancel the dialog with making any changes, and the “Help” button will bring up a help window; select the “Close” button to dismiss the help dialog.

System Default — This command resets various defaults, including the window size, the “MSC/NASTRAN Version Label”, and the “Run Command”.

Save — This command writes all the current settings, including the various entries, to the “$HOME/Xnastran” resource file. These values will be reloaded the next time you enter XNASTRAN.

Help

Online Documentation... — This command starts the online documentation application. The command used by XNASTRAN to start the application is specified by the “Xnastran*docname” resource. The default value is “mne”, i.e., the MSC/NASTRAN Encyclopedia (a separately installed product).

Program Version... — This command displays the program version in a pop-up window. Select the “OK” button to dismiss the pop-up window.

6.12.2 Main Window Items

Each item in the main window includes a “Help” button. Selecting the help item will bring up a short help dialog; the dialog is dismissed with the “Close” button.

The items in the main window are listed below as they appear from the top of the window to the bottom.

Input Data FIle — This subpane allows you to enter the name of the input file using the keyboard or with a file selection tool if the “Select File” button is selected.

Scratch Directory — This subpane allows you to enter the name of the scratch directory using the keyboard or with a directory selection tool if the “Select Directory” button is selected. This sets the “sdirectory” keyword.

Database Prefix — This subpane allows you to enter the prefix of the database files using the keyboard. This sets the “dbs” keyword; if the text field is empty, the “dbs” keyword is not set.

Monitor Output	This subpane allows you to start the XMONAST utility to monitor the F06, F04, and LOG files. This sets “xmon=yes” or “xmon=no”.
Background Process	This subpane allows you to run the job in the background. This sets “batch=yes” or “batch=no”.
Combine Files	This subpane allows you to append the F06, F04, and LOG files into a single OUT file. This sets “append=yes” or “append=no”.
Delete Databases	This subpane allows you to delete the user databases at the completion of the MSC/NASTRAN job or select a “mini” database. This sets “scratch=yes”, “scratch=no”, or “scratch=mini”. See Chapter 12 of the *MSC/NASTRAN Reference Manual* for further details on the “mini” database
Display News	This subpane allows you to display the MSC/NASTRAN system news in the F06. This sets “news=yes” or “news=no”.
Print Output Files	This subpane allows you to print the F06, F04, and LOG files at the completion of the MSC/NASTRAN job. This sets “prt=yes” or “prt=no”.
Send Notification	This subpane allows you to receive notification when the MSC/NASTRAN job completes. This sets “notify=yes” or “notify=no”.
Save Previous	This subpane allows you to version old output files before the MSC/NASTRAN job begins. This sets “old=yes” or “old=no”.
Output Prefix	This subpane allows you to enter the prefix of the output files using the keyboard. This sets the “out” keyword; if the text field is empty, the “out” keyword is not set.
Start Time	This subpane allows you to select a job starting time using the keyboard. This sets the “after” keyword; if the text field is empty, the “after” keyword is not set.
Queue Name	This subpane allows you to select the starting time of the job using the keyboard. This sets the “after” keyword.
Advanced Keywords	This subpane allows you to enter any additional keywords using the keyboard. You must enter the complete text of any keywords to be set. If the text field is empty, no additional keywords are set.
Memory Size	This subpane allows you to enter the memory allocation using the keyboard. The pop-up menu allows you to select the units modifier, i.e, none, “Kb”, “Kw”, “Mb”, “Mw”, “Gb”, “Gw”. This sets the “memory” keyword.

Submit MSC/NASTRAN This button submits the job using the parameters displayed in the window.

6.12.3 Resources

The default resource file, "/usr/lib/X11/app-defaults/Xnastran", and, if it exists, your resource file, "$HOME/Xnastran", are read at application startup. Your resource file is completely rewritten if you select the "Setup→Save" menu item. Documentation of the XNASTRAN resources can be found in the standard MSC resource file, "*install_dir*/msc705/*arch*/Xnastran".

6.13 Building the Utilities Delivered in Source Form

Several of the utilities (i.e., PLOTPS, NEUTRL, RCOUT2, and MSCACT) are delivered in source and executable form. The source code allows these utilities to be customized or built for other platforms. A script and makefile are provided to build and install these utilities. The script determines the architecture of current platform and invokes the make utility to perform the actual compilation, link, and installation.

The utility program source files are located in the directory "*install_dir*/msc705/util". This directory is an optional component of the MSC/NASTRAN installation. This directory includes the following files:

Table 6-2. Utility Program Source Files.

File	Description
install_dir/msc705/util/util	Script to Build Source Utility Programs.
install_dir/msc705/util/ld.f	Source for RCOUT2 Utility Routines.
install_dir/msc705/libfmsc.F	Source for FORTRAN Utility Library Routines.
install_dir/msc705/util/makefile	Makefile to Build Source Utility Programs.
install_dir/msc705/util/mattst.f	Source for Sample OUTPUT2 File Reader MATTST (see Section 7.6).
install_dir/msc705/util/mscact.c	Source for MSC Accounting Programs.
install_dir/msc705/util/neutrl.F	Source for NEUTRL Utility.
install_dir/msc705/util/ngtarg.F	Source for Command Line Utilities.

File	Description
install_dir/msc705/util/plotps.F	Source for PLOTPS Utility.
install_dir/msc705/util/rcout2.F	Source for RCOUT2 Utility.
install_dir/msc705/util/tabtst.f	Source for Sample OUTPUT4 File Reader TABTST (see Section 7.7).
install_dir/msc705/util/neutrl.F	Source for NEUTRL Utility.

Three steps are required to build and install the source utilities. Make sure that you are in the *install_dir*/msc705/util directory.

1. The first step compiles and links all of the source utility programs. Enter the command

   ```
   msc705 util build
   ```

   If only one utility is to be built, use the name of the utility (i.e., “mscact,” “neutrl,” “plotps,” or “rcout2”) instead of “build.” For example,

   ```
   msc705 util plotps
   ```

   will only build the PLOTPS utility.

2. After the programs are generated in the current directory, you can install the executable programs into the architecture directory for you computer (i.e., *install_dir*/msc705/*arch*). Enter the command

   ```
   msc705 util install
   ```

3. The third step deletes all object files and temporary files created by the “make” process. Enter the command

   ```
   msc705 util clean
   ```

The building and installation process can be repeated if you want to build the utilities for other computer architectures at your site.

If you want to build the utilities on another computer that does not have MSC/NASTRAN installed, you can copy the complete utilities directory to the other computer. Since the msc705 command will not be available, you must directly run the util script. Before you do, however, set the environment variable *MSC_ARCH* to the name of a supported architecture as shown in Table 3-1. The “install” option cannot be used.

CHAPTER 7

HOW TO BUILD AND USE THE SAMPLE PROGRAMS

This section describes how to build and use the various MSC/NASTRAN sample programs. The sample programs are grouped by function as follows:

1. Reading and displaying OUTPUT2 and OUTPUT4 files.
   - MATTST.
   - TABTST.
2. Reading and displaying XDB results database files. These sample programs are part of MSC/ACCESS and show how to use the database library routines.
   - DDLPRT.
   - DDLQRY.
   - DEMO1.
   - DEMO2.
   - SMPLR.
3. Implementing user-defined bar and beam elements for MSC/NASTRAN.
   - BEAMSERV.

Descriptions on building and using the sample programs follow in alphabetical order. At the end of the section, instructions on building the sample programs are included.

7.1 Building and Using BEAMSERV

Note: This program is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

BEAMSERV implements a user-defined beam element for MSC/NASTRAN. Unlike the other sample programs, a beam server is not a stand alone program run by the user from the command line. Instead, the beam server is started and controlled by MSC/NASTRAN. In the current UNIX implementation, communications between MSC/NASTRAN and the beam server are accomplished through pipes, with MSC/NASTRAN reading and writing BEAMSERV's stdout and stdin units, respectively.

Notes:
1. The MSC/NASTRAN job invoking the beam server and the beam server itself must run on the same computer.
2. Your program may not read from stdin (FORTRAN logical unit 5) nor write to stdout (FORTRAN logical unit 6).
3. The beam server cannot write to the F06, F04, or LOG files of the MSC/NASTRAN job that started the beam server.
4. Debugging must be accomplished by writing to a disk file, or connecting to the running beam server executable with a debugger (this is not available on all systems).

7.1.1 Building BEAMSERV

The BEAMSERV program source files located are in the directory "*install_dir*/msc705/bmsrv" (see Section 7.9). To build the program, change the working directory to the bmsrv directory and type the command:

```
msc705 bmsrv build
```

If you do not have write access to *install_dir*/msc705/bmsrv, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
msc705 ./bmsrv build
```

7.1.2 Using BEAMSERV

MSC/NASTRAN is made aware of the beam server by the “gmconn” keyword and an external evaluator connection file. Entries in the connection file for piped communications are formatted as follows:

```
groupname,pipe,pathname
```

where *groupname* is the group name defined on the CONNECT FMS statement and *pathname* is the pathname of the beam server executable.

> Note: The group name on the CONNECT statements and in the external evaluator connection file must match exactly, including character case. To use a mixed or lower case group name, the name on the CONNECT statement must be in quote marks; the name in the external evaluator connection file is never quoted.

To use the sample beam server and data file, create the file “samp_eval” with the following line:

```
LOCBMLS,pipe,pathname
```

where *pathname* is the pathname of the beam server built above, i.e., *install_dir*/msc705/*arch*/beamserv or ./beamserv”.

MSC/NASTRAN is then run using the following command:

```
msc705 nastran sample gmconn=samp_eval
```

7.2 Building and Using DDLPRT

DDLPRT illustrates the mass retrieval of data from the MSC/ACCESS Data Definition Language (DDL) database (see the *MSC/ACCESS User’s Manual* for further details).

7.2.1 Building DDLPRT

The DDLPRT program source code is in the file “*install_dir*/msc705/access/ddlprt.F” (see Section 7.10). To build the program, change the working directory to the access directory and type the command:

```
msc705 access ddlprt
```

If you do not have write access to *install_dir*/msc705/access, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
msc705 ./access ddlprt
```

7.2.2 Using DDLPRT

DDLPRT is run with the “ddlprt” command. The format of the “ddlprt” command is

```
msc705 ddlprt [ddl_xdb_file] [keywords]
```

If the DDL XDB file is not specified, the program uses the default MSC/ACCESS DDL file, *install_dir*/msc705/*arch*/dbc.xdb. The optional keywords are:

print=*print_file* Default: *ddl_xdb_file*.prt

This keyword specifies the name of the print file documenting the format of every MSC/ACCESS relation. By default, the print file uses the basename of the input DDL XDB file with the new file type “.prt”. Note, the size of this file is approximately one megabyte.

toc=*table_of_contents_file* Default: *ddl_xdb_file*.toc

This keyword specifies the name of the print file’s table of contents. By default, the toc file uses the basename of the input XDB file with the new file type “.toc”.

To execute the program, enter the command

```
msc705 ddlprt
```

The program displays the filename, version, and compilation date of the DDL file as well as the names of the print and table of contents files. Once these files are generated, the program exits. The print and table of contents files may then be printed once DDLPRT has completed.

7.3 Building and Using DDLQRY

DDLQRY illustrates the interactive retrieval of data from the MSC/ACCESS Data Definition Language (DDL) database (see the *MSC/ACCESS User's Manual* for further details).

7.3.1 Building DDLQRY

The DDLQRY program source code is in the file "*install_dir*/msc705/access/ddlqry.F" (see Section 7.10). To build the program, change the working directory to the access directory and type the command:

```
msc705 access ddlqry
```

If you do not have write access to *install_dir*/msc70/access, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
msc705 ./access ddlqry
```

7.3.2 Using DDLQRY

DDLQRY is run with the "ddlqry" command. The format of the "ddlqry" command is

```
msc705 ddlqry [ddl_xdb_file]
```

If a file is not specified, the program uses the default MSC/ACCESS DDL file, *install_dir*/msc705/*arch*/dbc.xdb.

The program displays the filename, version, and compilation date of the DDL file and prompts you for the name of a DDL object:

```
Enter Object Name (null to quit)
```

After you enter the name of each object, the format of the object is displayed. The program repeats the prompt until a blank line is entered.

7.4 Building and Using DEMO1

Note: This program is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

DEMO1 prints information about a results database (XDB) file produced by MSC/NASTRAN. DEMO1 is part of MSC/ACCESS, described in the *MSC/ACCESS User's Manual*.

7.4.1 Building DEMO1

The DEMO1 program source code is in the file "*install_dir*/msc705/access/demo1.f" (see Section 7.10). To build the program, change the working directory to the access directory and type the command:

```
msc705 access demo1
```

If you do not have write access to *install_dir*/msc705/access, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
msc705 ./access demo1
```

7.4.2 Using DEMO1

DEMO1 is run using the "demo1" command. The installed version of the program is run with the command:

```
msc705 demo1
```

You are prompted for the input graphics database filename.

```
Enter the database path name:
```

Running MSC/NASTRAN with a101x.dat (in *install_dir*/msc705/access) produces a101x.xdb that may be used as input to this program.

7.5 Building and Using DEMO2

Note: This program is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

DEMO2 prints information about a results database (XDB) file produced by MSC/NASTRAN. DEMO2 is part of MSC/ACCESS, described in the *MSC/ACCESS User's Manual*.

7.5.1 Building DEMO2

The DEMO2 program source code is in the file "*install_dir*/msc705/access/demo2.f" (see Section 7.10). To build the program, change the working directory to the access directory and type the command:

```
msc705 access demo2
```

If you do not have write access to *install_dir*/msc705/access, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
msc705 ./access demo2
```

7.5.2 Using DEMO2

DEMO2 is run using the "demo2" command. The installed version of the program is run with the command:

```
msc705 demo2
```

You are prompted for the input graphics database filename.

```
Enter the database path name:
```

Running MSC/NASTRAN with a61x.dat (in *install_dir*/msc705/access) produces a101x.xdb that may be used as input to this program.

7.6 Building and Using MATTST

> Note: This program is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

MATTST reads a binary format OUTPUT4 matrix.

7.6.1 Building MATTST

The MATTST program source code is in the file "*install_dir*/msc705/util/mattst.f" (see Section 6.1.2). To build the program, change the working directory to the util directory and type the command:

```
msc705 util mattst
```

If you do not have write access to *install_dir*/msc705/util, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
msc705 ./util mattst
```

7.6.2 Using MATTST

MATTST is run with the "mattst" command. The installed version of the program is run with the command:

```
msc705 mattst
```

You are prompted for the number of matrices.

```
Please enter the number of matrices:
```

You are prompted for the input filename.

```
Please enter the INPT4 FILENAME:
```

You are prompted for the output binary filename.

```
Please enter the output binary filename:
```

You are prompted for the output text filename.

```
Please enter the output text filename:
```

Running MSC/NASTRAN with um54.dat (in the DEMO library, *install_dir*/msc705/nast/demo) produces um54.f11 that may be used as input to this program.

7.7 Building and Using TABTST

Note: This program is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

TABTST reads a binary format OUTPUT2 file (do not confuse this program with RCOUT2, described in Section 6.8).

7.7.1 Building TABTST

The TABTST program source code is in the file "*install_dir*/msc705/util/tabtst.f" (see Section 6.1.2). To build the program, change the working directory to the util directory and type the command:

```
msc705 util tabtst
```

If you do not have write access to *install_dir*/msc705/util, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
msc705 ./util tabtst
```

7.7.2 Using TABTST

TABTST is run with the "tabtst" command. The installed version of the program is run with the command:

```
msc705 tabtst
```

You are prompted for the input filename.

```
Please type the INPUT2 filename:
```

You are prompted for the output filename.

```
Please type the output filename:
```

Running MSC/NASTRAN with tabtsta.dat (in the TPL library, *install_dir*/msc705/nast/tpl) produces tabtsta.f11 that may be used as input to this program.

7.8 Building and Using SMPLR

Note: This program is only provided as a simple example illustrating basic concepts. It is not intended to be a complete or usable program.

SMPLR reads a results database (XDB) file produced by MSC/NASTRAN. SMPLR is part of MSC/ACCESS, described in the *MSC/ACCESS User's Manual*.

7.8.1 Building SMPLR

The SMPLR program source code is in the file "*install_dir*/msc705/access/smplr.f" (see Section 7.10). To build the program, change the working directory to the access directory and type the command:

```
msc705 access smplr
```

If you do not have write access to *install_dir*/msc705/access, copy the entire directory to another location, change the working directory to the new location, and issue the command:

```
msc705 ./access smplr
```

7.8.2 Using SMPLR

SMPLR is run using the "smplr" command. The installed version of the program is run with the command:

```
msc705 smplr
```

You are prompted for the input filename.

```
Enter the database name to process:
```

Running MSC/NASTRAN with *install_dir*/msc705/access/a101x.dat produces a101x.xdb that may be used as input to this program.

7.9 Beam Server Source Files

The BEAMSERV program source files are located in the directory "*install_dir*/msc705/bmsrv". This directory is an optional component of the MSC/NASTRAN installation. This directory includes the following files:

Table 7-1. Beam Server Sample Program Source Files.

File	Description
install_dir/msc705/*arch*/libbmsrv.a	Object Library Used When Building Beam Servers.
install_dir/msc705/bmsrv/bmsrv	Script to Build the Sample Beam Server Program.
install_dir/msc705/bmsrv/brtucd.f	Source for Sample Beam Server Subroutine BRTUCD.
install_dir/msc705/bmsrv/brtugd.f	Source for Sample Beam Server Subroutine BRTUGD.
install_dir/msc705/bmsrv/brtuid.f	Source for Sample Beam Server Subroutine BRTUID.
install_dir/msc705/bmsrv/brtupd.f	Source for Sample Beam Server Subroutine BRTUPD.
install_dir/msc705/bmsrv/bsbrcd.f	Source for Sample Beam Server Subroutine BSBRCD.
install_dir/msc705/bmsrv/bsbrgd.f	Source for Sample Beam Server Subroutine BSBRGD.
install_dir/msc705/bmsrv/bsbrid.f	Source for Sample Beam Server Subroutine BSBRID.
install_dir/msc705/bmsrv/bsbrpd.f	Source for Sample Beam Server Subroutine BSBRPD.
install_dir/msc705/bmsrv/bsbrt.f	Source for Sample Beam Server Subroutine BSBRT.
install_dir/msc705/bmsrv/bscon.f	Source for Sample Beam Server Subroutine BSCON.
install_dir/msc705/bmsrv/bsgrq.f	Source for Sample Beam Server Subroutine BSGRQ.
install_dir/msc705/bmsrv/bsmsg.f	Source for Sample Beam Server Subroutine BSMSG.
install_dir/msc705/bmsrv/makefile	Makefile to Build the Sample Beam Server Program.
install_dir/msc705/bmsrv/main.c	Source for Sample Beam Server Main Program.
install_dir/msc705/bmsrv/mevbrd.f	Source for Sample Beam Server Subroutine MEVBRD.
install_dir/msc705/bmsrv/msbrcd.f	Source for Sample Beam Server Subroutine MSBRCD.
install_dir/msc705/bmsrv/msbrgd.f	Source for Sample Beam Server Subroutine MSBRGD.
install_dir/msc705/bmsrv/msbrid.f	Source for Sample Beam Server Subroutine MSBRID.
install_dir/msc705/bmsrv/sample.dat	MSC/NASTRAN Sample Data File.

7.10 MSC/ACCESS Source Files

The MSC/ACCESS sample source files located are in the directory "*install_dir*/msc705/access". This directory is an optional component of the MSC/NASTRAN installation. This directory includes the following files:

Table 7-2. MSC/ACCESS Sample Program Source Files.

File	Description
install_dir/msc705/access/a101x.dat	MSC/NASTRAN Data File.
install_dir/msc705/access/access	Script to Build MSC/ACCESS Sample Programs.
install_dir/msc705/access/ddlprt.F	Demonstration Database Dictionary Print Program.
install_dir/msc705/access/ddlqry.F	Demonstration Database Dictionary Query Program.
install_dir/msc705/access/demo1.F	Source for Sample MSC/NASTRAN Database Reader.
install_dir/msc705/access/demo2.F	Source for Sample MSC/NASTRAN Database Reader.
install_dir/msc705/access/makefile	Makefile to Build MSC/ACCESS Sample Programs.
install_dir/msc705/access/smplr.F	Source for Sample MSC/NASTRAN Database Reader.
install_dir/msc705/*arch*/grspbd.o	Object to Be Used When Building Database Readers.
install_dir/msc705/*arch*/libdbio.a	Object Library of Input/Output Routines to Be Used When Building Database Readers.

APPENDIX A

GLOSSARY OF TERMS

3060	A User Fatal Message indicating that authorization to run MSC/NASTRAN has been denied (see Section 3.2).
6080	A User Warning Message indicating that timing blocks must be generated for your computer (see Section 3.10).
acct	MSC accounting file directory, "*install_dir*/acct". Also, the program (*install_dir*/msc705/*arch*/acct) that updates the current month's accounting data file "*install_dir*/acct/msc*yymm*.acc". See MSCACT for the program source.
architecture RC file	The RC file "*install_dir*/conf/*arch*/nast705rc". See Table 3-1 for a listing of architecture names.
archive	A test problem library (*install_dir*/msc705/misc/archive) that contains test decks that are no longer part of either the DEMO or TPL libraries. These files may be incompatible with MSC/NASTRAN V70.5 or may use features that are no longer supported.
ASSIGN	A File Management Section (FMS) statement that is used to assign physical files to DBsets or FORTRAN files.
authorize	Command line and RC file keyword that is used to set the authorization code required to run MSC/NASTRAN.
basename	The part of a pathname exclusive of the directory and suffix (e.g., the basename of /temp/myfile.dat. is "myfile").
BUFFPOOL	The NASTRAN statement keyword that sets the size of the buffer pool (see Section 5.2).
buffer pool	A disk cache of GINO blocks.
BUFFSIZE	One plus the number of words in a GINO physical record. Also, the NASTRAN statement keyword that sets the default buffer size (see Section 5.2).

conf	The MSC conference file directory (*install_dir*/conf) contains the system, architecture, and node RC files and other site-specific files.
counted license	A counted license is a FLEXlm license that limits the number of concurrent executions of MSC/NASTRAN. Counted licenses always require a FLEXlm license server.
.dat	The ".dat" file suffix describes a finite element model.
daemon	A UNIX program that runs in the background and provides services to the operating system and to users. Daemons are generally started when the system is bootstrapped and terminate when the system shuts down.
DBset	Database file set.
DDLPRT	Utility program that prints the contents of the results database (XDB) data definition language database (*install_dir*/msc705/*arch*/dbc.xdb) and illustrates the batch recovery of the data definition language.
DDLQRY	Utility program that prints the contents of the results database (XDB) data definition language database (*install_dir*/msc705/*arch*/dbc.xdb) and illustrates the interactive recovery of the data definition language.
del	Delivery database library,
DEMO	The demonstration problem library (*install_dir*/msc705/nast/demo) contains a selection of MSC/NASTRAN input files that are documented in the *MSC/NASTRAN Demonstration Problem Manual*.
DEMO1	Sample program that prints information from a graphics database file.
DEMO2	Sample program that prints information from a graphics database file.
DMAP	Direct Matrix Abstraction Program, which is the programming language of the MSC/NASTRAN solution sequences.
doc	Documentation file directory.
EAG FFIO	Engineering Applications Group Flexible File I/O. (See ffio in Appendix B)
ESTIMATE	Utility that estimates memory and disk requirement of a data file and make suggestions on improving the performance of MSC/NASTRAN.

file locking	A mechanism to prevent multiple MSC/NASTRAN jobs from interfering with one another. For example, two jobs attempting to write to the same DBset interfere with one another, whereas two jobs reading the delivery database do not interfere with one another.
file mapping	A mechanism to use the system's virtual paging system to access a file. MSC/NASTRAN can use file mapping to access GINO files. See Table 4-5 for a listing of systems that support file mapping.
F04	The F04 file is created by MSC/NASTRAN and contains a module execution summary as well as a database information summary. The F04 file has the suffix ".f04".
F06	The F06 file is created by MSC/NASTRAN and contains the numerical results of the analysis. The F06 file has the suffix ".f06".
FMS	File Management Section of the input file, which is used to attach and initialize DBsets and FORTRAN files.
gentim2	MSC/NASTRAN job that determines the timing constants for your computer.
GINO	The MSC/NASTRAN database subsystem.
GINO block	A block of data transferred by GINO.
HEATCONV	Utility program that converts pre-MSC/NASTRAN V68 heat-transfer data files to the MSC/NASTRAN Version 68 format.
IEEE	Institute of Electrical and Electronics Engineers, Inc. A professional society. The floating point formats and, to a lesser extent, algorithms used on many MSC/NASTRAN computers are defined by IEEE Standard 754.
INCLUDE	A File Management Section (FMS) statement that inserts an external file into the input file.
INIT	The INIT statement is part of the File Management Section (FMS) and is used to create a temporary or permanent DBset.
large file	A file on a 32-bit system that can be 2 gigabytes or larger. All files on a 64-bit system can be large files. See Table 4-5 for a listing of systems that support large files.
local RC file	The RC file ".nast705rc" in the directory containing the input data file.

LOG	The LOG file is created by MSC/NASTRAN and contains system information as well as system error messages. The LOG file has the suffix ".log".
MATTST	Sample program that reads the OUTPUT4 matrix files.
memory	Command line keyword that is used to define the amount of memory allocated for open core.
MPL	The module properties list is a table that defines the properties of DMAP modules.
MSC/ACCESS	FORTRAN-callable subroutine library that reads and writes results database (XDB) files.
MSCACT	Utility program that generates accounting reports. The source for this utility and the accounting file update program are maintained in the same file (*install_dir*/msc705/util/mscact.c).
MSGCMP	Utility program that compiles a text file to create a message catalog.
NAO	The Network Authorization Option of MSC/NASTRAN. The implementation in MSC/NASTRAN Version 70.5 is not compatible with earlier versions of NAO.
.ndb	Default neutral-format XDB file suffix.
.neu	Default neutral-format plot file suffix. Only created by NEUTRL.
NEUTRL	Utility program that converts binary plot (.plt) files to neutral plot (.neu) files.
node RC file	The RC file "*install_dir*/conf/net/*nodename*/nast705rc".
NUSR	The node-locked license enforcement of the maximum number of users concurrently running MSC/NASTRAN. See Section 3.3.2 for additional information.
open core	Amount of working memory in words.
OPTCONV	Utility program that converts pre-MSC/NASTRAN V68 optimization and design-sensitivity data files to the MSC/NASTRAN Version 68 format.
.on2	Default neutral-format OUTPUT2 file suffix.
.op2	Default binary-format OUTPUT2 file suffix.
.pch	Default punch file suffix.
PLOTPS	Utility program that converts binary (.plt) or neutral (.neu) plot files to PostScript (.ps) files.

.plt	Default binary-format plot file suffix.
.ps	Default PostScript plot file suffix.
RC file	Runtime configuration file that is used by MSC/NASTRAN to control execution parameters.
RCOUT2	Utility program that converts a neutral OUTPUT2 (.np2) file to a binary OUTPUT2 (.op2) file.
RECEIVE	Utility program that converts neutral results database (.neu) files to binary results database (XDB) files.
RFA	Rigid-format alter library, "*install_dir*/msc705/nast/rfa". (This directory is now empty.)
sdir	Keyword that is used to set the directory for temporary scratch files produced by MSC/NASTRAN.
smemory	Command line keyword to set SMEM.
SMEM	Scratch memory area for memory-resident database files.
SMPLR	Sample program that reads graphics database files.
sysfield	The global SYS parameter that can be specified on the command line or in an RC file.
SYS	An INIT statement parameter that is used to specify special machine-dependent information. File locking and file mapping of GINO files are controlled through the SYS parameter.
system RC file	The RC file "*install_dir*/conf/nast705rc".
SYSTEM(x)	System cells that are used by MSC/NASTRAN to control analysis parameters.
SSS	Structured Solution Sequences. The delivery database files (SSS.MASTERA, SSS.MSCSOU, and SSS.MSCOBJ) are found in "*install_dir*/msc705/*arch*" and the source files are found in "*install_dir*/msc705/nast/del".
SSSALTER	Additional alter and error corrections library, "*install_dir*/msc705/misc/sssalter".
suffix	The part of the pathname exclusive of the directory and basename (e.g., the suffix of /tmp/myfile.dat is ".dat").
TABTST	Sample program that reads binary-format OUTPUT2 files.
TPL	The test problem library (TPL, *install_dir*/msc705/nast/tpl) contains a general selection of MSC/NASTRAN input files

	showing examples of most of the MSC/NASTRAN capabilities, in general, these files are not documented.
TRANS	Utility program that converts binary results database (XDB) files to neutral results database (.neu) files.
UFM	A User Fatal Message that describes an error severe enough to terminate the program.
UFM 3060	A User Fatal Message indicating that authorization to run MSC/NASTRAN has been denied (see Section 3.2).
UIM	A User Information Message that provides general information.
uncounted license	An uncounted license is a FLEXlm license that allows any number of concurrent executions of MSC/NASTRAN on a given node. An uncounted license does not require a FLEXlm license server.
user RC file	The RC file "${HOME}/.nast705rc".
util	Utility program library, "*install_dir*/msc705/util".
UWM	A User Warning Message that warns of atypical situations. You must determine whether a problem exists in the analysis.
UWM 6080	A User Warning Message indicating that timing blocks must be generated for your computer (see Section 3.10).
version	A file is "versioned" by appending a dot followed by a version number to the file's name. The latest version of a file does not have a version number, all earlier versions do, with the oldest having the smallest version number and the latest having the highest version number.
XDB	The XDB file is created by MSC/NASTRAN and contains results information for use by various post-processing programs. See the "POST" parameter in Section 6 of the *MSC/NASTRAN Quick Reference Guide* for further information on generating XDB files. XDB files are not versioned. The XDB file has the suffix ".xdb".

APPENDIX B

KEYWORDS AND ENVIRONMENTAL VARIABLES

B.1 Keywords

The following is a complete list of the keywords that may be used on the command line or placed into RC files as appropriate. Keywords that use yes/no values accept partial specification and case-independent values. For example, “yes” may be specified as“y”, “ye”, or “yes” using uppercase or lowercase letters.

acct acct={yes|no} Default: No

Indicates solution accounting is to be performed. The new “lock” keyword may be used to ensure that all jobs have solution accounting enabled.

For example, the following RC file lines will force all jobs to use accounting:

Example:
```
acct=yes
lock=acct
```

The first line turns accounting on. The second line ensures accounting is on for every job, see the “lock” keyword for more details.

acdata acdata=*string* Default: None

Specifies site defined accounting data. See your system administrator to determine if and how this keyword is to be used. See Section 3.3.1 for additional information.

acid acid=*string* Default: None

Specifies site defined account ID. See your system administrator to determine if and how this keyword is to be used. See Section 3.3.1 for additional information.

acvalid acvalid=*string* Default: *None*

Note: This keyword can only be set in the command initialization file, see Sections 3.3.2 and 3.5.

Indicates account ID validation is to be performed. If “acvalid” is not defined, or is null, then no checks are made of the account ID. If “acvalid” is defined, then account ID validation is performed. See Section 3.3.2 for information on defining this keyword.

after after=*time* Default: *None*

Holds the job until the time specified by *time*. See the description of the “at” command in your system documentation for the format of *time*.

Example: `msc705 nastran example after=10:00`

The job is held until 10:00 AM.

append append={yes|no} Default: No

Combines the F04, F06, and LOG files into a single file after the run completes. If “no” is specified, the files are not combined. If “yes” is specified, the files are combined into one file with the suffix “.out”.

Example: `msc705 nastran example append=yes`

The F04, F06, and LOG files are combined into a file named “example.out”.

application application=NASTRAN

Notes: 1. This keyword may only be specified on the command line or in the command initialization file (see Section 3.5).

2. This keyword should always be set to “NASTRAN”.

Specifies the application to be run.

authinfo authinfo=*number* Default: 0

Specifies the amount of information written to the LOG during authorization processing. Values greater than zero indicate additional information is to be written.

authorize authorize=*pathname* Default: *install_dir*/conf/authorize.dat

Specifies the name of the node-locked authorization file if a file path name is specified. If a directory is specified, the program assumes that either “authorize.dat” or “license.dat” is in the specified directory.

Example: `msc705 nastran example auth=myauthfile`

The job is run using the node-locked authorization code in "myauthfile".

authqueue authqueue=*number* Default: 20

Note: When a job is waiting for either a seat to become available, the job is consuming computer resources such as memory, swap file space, disk space, etc. Too many jobs waiting for licenses could have a severe impact on the system.

All systems except Cray and NEC: Specifies the time in minutes to wait for a seat to become available. If the seat becomes available before this specified time period expires, the job will be allowed to continue. If not, the job will be terminated.

Example: `msc705 nastran example auth=myfile`

The job is run using the node-locked authorization code in "myauthfile". If a seat is not available within 20 minutes, the job will be terminated.

Example: `msc705 nastran example auth=myfile authqueue=10`

The job is run using the node-locked authorization code in "myauthfile". If a seat is not available within 10 minutes of the start of the job, the job will be terminated.

batch batch={yes|no} Default: Yes

Indicates how the job is to be run. If "yes" is specified, the job is run as a background process. If "no" is specified, the job is run in the foreground. If the "aft" or "queue" keywords are specified, the batch keyword is ignored. Jobs submitted with "batch=yes" will run under nice(1).

Note: If the job is already running in an NQS or NQE batch job, the default is "no".

Example: `msc705 nastran example batch=no`

The job is run in the foreground.

bpool bpool=*value* Default: 27 (Cray and NEC); 37 (all others)

Specifies the number of GINO and/or executive blocks that are placed in buffer pool; see the *MSC/NASTRAN Reference Manual*, Section 3.3.1 for more information.

Example: `msc705 nastran example bpool=100`

Space for 100 GINO buffers is reserved for the buffer pool.

buffsize buffsize={*value*|estimate} Default: 4097 (Cray and NEC); 2049 (all others)

Specifies the physical record size, in words (1 word = 8 bytes on Cray UNICOS and NEC systems; 4 bytes on all others), of all MSC/NASTRAN DBsets except those specified with INIT statements and MSCOBJ (which uses 4097 on Cray and NEC systems and 2049 on all others). The physical record size is BUFFSIZE-1 words. If "buffsize=estimate" is specified, ESTIMATE will be used to determine *value*.

The following table provides the recommended BUFFSIZE for a job based on the number of degress of freedom.

Table B-1. Recommended BUFFSIZE.

Degrees of Freedom	Suggested BUFFSIZE	
	Cray and NEC	Others
DOF < 10 000	4097	2049
10 000 ≤ DOF < 50 000	4097	4097
50 000 ≤ DOF < 100 000	8193	8193
100 000 ≤ DOF < 200 000	16385	16385
DOF ≥ 400 000	32769	32769

BUFFSIZE must reflect the maximum BUFFSIZE of all DBsets attached to the job including the delivery database, which is generated with a BUFFSIZE of 4097 words on Cray and NEC or 2049 words on all others. If you generate your own delivery database, this default may be higher. The maximum value of BUFFSIZE is 65537 words. BUFFSIZE must be one plus a multiple of the disk block size. The disk block size may be determined with the "system" special function described in Section 4.1.3.

Example: `msc705 nastran example buffsize=16385`

The BUFFSIZE is set to 16385 words.

config config=*number* Default: *Computer dependent*

Specifies the configuration (CONFIG) number used by MSC/NASTRAN to select timing constants. You can change this value to select the timing constants of a different computer model. A

configuration number of zero is considered undefined by the nastran command. See Sections 3.9 and 3.10 for additional information.

cputime cputime=*cputime* Default: *None*

Note: The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Specifies the maximum amount of CPU time that the complete job is permitted to use when the “queue” keyword is used. This time includes the execution of the driver program, the MSC/NASTRAN executable, plus any commands specified by the “pre” and “post” keywords. See your system’s queuing documentation for the format of *cputime*.

The value can be specified as either “*hours*:*minutes*:*seconds*”, “*minutes*:*seconds*”, or “*seconds*”, and will always be converted to the number of seconds.

Example: `msc705 nastran example queue=small cputime=60`

This example defines the maximum CPU time for the complete job as 60 seconds.

Example:
```
msc705 nastran example queue=small cpu=1:15:0
msc705 nastran example queue=small cpu=75:0
msc705 nastran example queue=small cpu=4500
```

These examples all define the maximum CPU time for the complete job as one hour and fifteen minutes.

dbs dbs=*pathname* Default: `.`

Creates database files (see Table 4-4) using an alternate file prefix. If “dbs” is not specified, database files are created in the current directory using the basename of the input data file as the prefix. If the “dbs” value is a directory, database files are created in the specified directory using the basename of the input data file as the filename.

Note: If “dbs” is specified and “scratch=yes” is specified, a warning will be issued and “scratch=no” assumed. This is a change from the operation of prior releases, which would ignore the “dbs” keyword if “scratch=yes” was set.

In the following examples, assume the following directory contents:

```
example.dat              mydir/            other/
other/example.dat
```

where “mydir” and “other” are directories.

Example: `msc705 nastran example`
or: `msc705 nastran other/example`

Database files are created in the current directory with name example, e.g., ./example.DBALL.

Example: `msc705 nastran example dbs=myfile`

Database files are created in the current directory with name myfile, e.g., ./myfile.DBALL.

Example: `msc705 nastran example dbs=mydir`

Database files are created in the mydir directory with name example, e.g., mydir/example.DBALL.

Example: `msc705 nastran example dbs=mydir/myfile`

Database files are created in the mydir directory with name myfile, e.g., mydir/myfile.DBALL.

delivery — delivery={*pathname*|MSCDEF}Default: MSCDEF

Specifies an alternate delivery database option. (See Section 5.7 for further information on alternate delivery databases.)

Example: `msc705 nastran example del=mysss`

The job is run using a solution sequence from the delivery database “mysss.MASTERA”.

display — display=*display_name* Default: *Current display.*

Specify a display for XMONAST. This value may also be set with the *DISPLAY* environment variable. The environment variable overrides the RC files; the command line overrides the environment variable.

executable — executable=*pathname* Default: *computer dependent*

Specifies the name of an alternate solver executable. This keyword overrides all architecture and processor selection logic. If a directory is not specified in the pathname and the file does not exist in the current directory, the default architecture directory is assumed.

Example: `msc705 nastran example exe=analysis.um`

The job is run using the executable “analysis.um”. Since a directory was not specified, this file must exist in the either current directory or “*install_dir*/msc705/*arch*”.

fbsopt fbsopt=*number* Default: *See the description below.*

Selects the forward-backward substitution methods. This value may also be set with the “sys705” command line keyword. See the *MSC/NASTRAN Quick Reference Guide* for information on the default value and legal values for this keyword.

ff_io ff_io={yes|no|append} Default: Yes

Cray only. Indicates if EAG FFIO is to be enabled. EAG FFIO can provide a substantial elapsed-time performance increase. If “ff_io=yes” is set and “ff_io_opts” is not set, a default value for the FF_IO_OPTS environment variable will be determined. This value will include both the default permanent and scratch DBsets; will use the cache size specified by the “ff_io_cachesize” keyword; and will consider the device geometries of the disks containing the “dbs” and “sdirectory” directories. If “ff_io=append” is set, the calculated FF_IO_OPTS value will be appended to the user’s FF_IO_OPTS value. If “ff_io_no” is specified, any values for FF_IO_OPTS and FF_IO_DEFAULTS will be suppressed.

Note: Because of the difficulty in setting the FF_IO_OPTS value, especially the striping partitions, you are *strongly* urged to remove any FF_IO_OPTS settings you may have been using.

Additional documentation, supplied by Cray Research, Inc., on EAG FFIO can be found in the HTML files “*install_dir*/msc705/unicos/ffio.html”, “*install_dir*/msc705/unicosc90/ffio.html” or “*install_dir*/msc705/unicosts/ffio.html”. The default parameters are: *share*=1; *stride*=1; *max_lead*, *alloc*, and *set* (i.e., *cblks* and *cbits*) are based on disk device geometry; *page_size* and *num_pages* are based on BUFFSIZE and “ff_io_cachesize”.

ff_io_cachesize ff_io_cachesize=*memory_size*
Default: 1MW

Cray only. Specifies the size of the EAG FFIO cache only if “ff_io=yes” is set and neither the “ff_io_opts” keyword nor the the FF_IO_OPTS environment variable is set. This value will be added to the “prm” and “ppm” values. The *memory_size* can be specified either as the number of words or as a number followed by one of the following modifiers:

G or Gw Multiply *memory_size* by 1024**3, round down to multiple of 512.

Gb Multiply *memory_size* by (1024**3)/8, round down to multiple of 512.

M or Mw Multiply *memory_size* by 1024**2, round down to multiple of 512.

Mb Multiply *memory_size* by (1024**2)/8, round down to multiple of 512.

K or Kw Multiply *memory_size* by 1024, round down to multiple of 512.

Kb Multiply *memory_size* by 1024/8, round down to multiple of 512.

w Round *memory_size* down to multiple of 512.

b Divide *memory_size* by 8, round down to multiple of 512.

The modifiers may be specified using any case combination.

Note: The minimum cache size is 512 000 words.

Example: `msc705 nastran example ff_io=yes ff_io_cachesize=2mw`

The job is run with a 2 megaword EAG FFIO cache.

ff_io_defaults ff_io_defaults=*string* Default: *None*

Cray only. Specifies the EAG FFIO default options to be used. This value must be a valid FFIO specification string; no error checking is performed before MSC/NASTRAN starts.

This value may also be set by the FF_IO_DEFAULTS environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

ff_io_opts ff_io_opts=*string* Default: *See "ff_io"*

Cray only. Specifies the EAG FFIO options to be used. This value must be a valid EAG FFIO specification string; no error checking is performed before MSC/NASTRAN starts.

This value may also be set by the FF_IO_OPTS environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

Note: Because of the difficulty in setting the FF_IO_OPTS value, especially the striping partitions, you are *strongly* urged to remove any FF_IO_OPTS settings you may have been using.

gmconn gmconn=*pathname* Default: *None*

Specifies the name of the external evaluator connection file. External geometric and bar or beam element evaluators may be specified. See the *MSC/NASTRAN Version 69 Release Guide* for additional information on external bar or beam elements and Section 7.1 of this document for information on running an MSC/NASTRAN job using a beam server.

Example: `msc705 nastran example gmconn=mybeam-server`

The job is run with the external evaluators specified in “mybeamserver”.

hpio_param hpio_param=*string* Default: *None*

NEC only: Specifies the HPIO control string. The control string is composed on one or more filename-options pairs of the form:

file_template (*p1*:*p2*:*p3*:*p4*:*p5*:*p6*:*p7*)

where

file_template: blank separated list of filename templates, there is no default. Examples are “*DBALL” to match all files ending in “DBALL” and “*DBALL *SCR*” to match all files with “SCR” anywhere in the name.

p1: Number of cache pages for each file, the default is 5.

p2: The size of each cache page, the default is “8m” or 8 megabytes. This value is the number of bytes, or a number followed by “k” for kilobytes, or “m” for megabytes.

p3: The maximum number of cache pages to read ahead. The default is 1. This value must be less than p1. A rule of thumb is $p3 < 0.5 \times p1$.

p4: “nolog” or “log”, the default is “nolog”. Note, the “log” option is intended for tuning and debugging purposes only, it can generate large amounts of output.

p5: XMU cache working directory, there is no default. The filesystem containing this directory must be an SFS/H filesystem.

p6: Number of XMU cache pages, the default is 5. This value must be greater than 0 if p5 is specified.

p7: Number of buffers for XMU cache access, the default is 2.

The additional main memory consumed by the HPIO facility is

($p1 + 1) \times p2 \times number\_of\_files$ (without XMU)

or

($p1 + 1 + p7 + 2) \times p2 \times number\_of\_files$ (with XMU)

The space on the XMU consumed by the HPIO facility is

$p6 \times p2$

Example: `msc705 nastran example hpio_param='*SCR*(9::1:nolog)'`

The job is run with HPIO enabled for all files with SCR in their name, e.g., *sdir*/example.SCRATCH and *sdir*/example.SCR300. HPIO will allocate ten cache pages (*p1* + 1) of 8 megabytes per page per file, the cache will read ahead 1 page. Assuming no other files use HPIO, an additional 160 MB of memory will be required by this job.

Note: If invalid XMU fields are specified, a message will be printed in the LOG file and will continue without using HPIO.

ja ja={yes|no} Default: No

Cray only. Enable job accounting using the ja(1) utility. See the ja(1) man page for additional information on this utility.

Example: `msc705 nastran example ja=yes`

The job is run with the the job accounting system enabled.

jid jid=*pathname* Default: *None*

Specify the name of the input data file. An input file must be defined on the command line. Any command line argument that does not have a keyword is assumed to be the input file. Only the last filename is used.

Example: `msc705 nastran example`

The input file “example.dat” is used.

Note: If the input file is specified as “example” and the files “example.dat” and “example” both exist, the file “example.dat” will be chosen. In fact, it is *impossible* to use a file named “example” as the input data file if a file named “example.dat” exists.

lock lock=*keyword* Default: None

The “lock” keyword can be used by a site or a user to prevent modification of a keyword’s value.

For example, the following RC file lines will force all jobs to use accounting by setting the “acct” keyword on and then preventing the keyword from being changed later:

Example:
```
acct=yes
lock=acct
```

As soon as the second line is read, any attempt to set the “acct” keyword later in the same RC file, in an RC file read after this file, or on the command line will be silently ignored. RC files and the command line are processed in the following order:

1. *install_dir*/conf/nast705rc
2. *install_dir*/conf/*arch*/nast705rc
3. *install_dir*/conf/net/*node*/nast705rc
4. $HOME/.nast705rc
5. Specified by “rcf” keyword, default is *jid-dir*/.nast705rc
6. *command line*

The “lock” keyword may appear anywhere a keyword is accepted. The lock keyword itself can be locked with “lock=lock”.

massbuf massbuf=*number* Default: *See the description below.*

Sets half the number of buffers to set aside for storing the mass matrix in memory. This value may also be set with the “sys199” command line keyword. See the *MSC/NASTRAN Quick Reference Guide* for information on the default value and legal values for this keyword.

memory memory={*memory_size*|estimate} Default: 4MW. *See the description below.*

Note: See Section 4.3 for information on estimating a job’s memory requirements.

Specifies the amount of open core memory to allocate. If “memory=estimate” is specified, ESTIMATE will be used to determine *memory_size*. Otherwise, the *memory_size* can be specified either as the number of words or as a number followed by one of the following modifiers:

G or Gw	Multiply *memory_size* by 1024**3.
Gb	Multiply *memory_size* by (1024**3)/*bytes_per_word*.
M or Mw	Multiply *memory_size* by 1024**2.
Mb	Multiply *memory_size* by (1024**2)/*bytes_per_word*.
K or Kw	Multiply *memory_size* by 1024.
Kb	Multiply *memory_size* by 1024/*bytes_per_word*.
w	Use *memory_size* as is.
b	Divide *memory_size* by *bytes_per_word*.

where *bytes_per_word* is 8 on Cray and NEC; 4 on all others. The modifier may be specified using any case combination.

If a value has not been assigned to the “memory” keyword, the nastran command will assume “memory=estimate”. If an explicit null value has been set, the nastran command will not assume a default value and the run will fail.

This allows your site to establish a default policy for memory as follows:

- If your site sets the memory keyword to a non-null value, you can choose to override this default by explicitly setting the memory keyword on the command line or in an RC file. You can accept the default by not setting a new value.
- If your site sets the memory keyword to a null value, i.e., “memory=”, in an RC file, you must set the “memory” keyword to a non-null value in one of your RC files or on the command line. If you do not set a non-null value, i.e., you leave the null value, the nastran command will report a fatal error.
- If no value for the “memory” keyword in has been set in any RC file or on the command line, “memory=estimate” will be assumed.

Note: MSC/NASTRAN now uses standard computer units for K, M, and G. Prior releases used engineering units.

Example: `msc705 nastran example memory=25mw`

The job is run using an open core memory size of 25 megawords, or 25 600 kilowords, or 26 214 400 words.

The maximum *memory_size* is limited as shown in Table B-2 (less the size of the executable and I/O buffers).

Table B-2. Maximum *memory_size*.

Cray	Available physical memory
Digital	8 gigabytes
Fujitsu	Lesser of “real” memory or 2 gigabytes
Hitachi	Lesser of “real” memory or 2 gigabytes
NEC	Available physical memory
SGI R8K, R10K	8 gigabytes
All others	2 gigabytes

mpyad mpyad=*number* Default: *See the description below.*

Selects/deselects multiplication method selection. This value may also be set with the “sys66” command line keyword. See the *MSC/NASTRAN Quick Reference Guide* for information on the default value and legal values for this keyword.

msgcat msgcat=*pathname* Default: *install_dir*/msc705/*arch*/analysis.msg

The “msgcat” keyword specifies an alternate message catalog. The message catalog contains the message text used for many MSC/NASTRAN messages. A site or user can modify the message file to include message text that is more appropriate to their operations, compile the new catalog using the MSGCMP utility, and invoke the new catalog using this keyword.

Example: msgcat=mycat.msg

This example will use the file “mycat.msg” as the message catalog. See Sections 3.8 and 6.4 for additional information on customizing the message catalog and using the MSGCMP utility.

Note: Message catalogs are machine dependent. Table 6-1, “Binary File Compatibility” identifies the systems that are binary compatible; binary compatible systems can use multiple copies of the same message file.

nastran nastran *keyword*=*value* Default: *None*

Specifies a value for the NASTRAN statement.

Note: This keyword can only be specified in an RC file. If the last character of the keyword value is a comma, or a quote or parenthetic expression is open, the next line in the RC file is considered a continuation. The continuation will continue until the quote or parenthetic expression is closed and a line that is not ended by a comma is found.

ncmd ncmd=*command_string* Default: "echo *msg* | write *user tty*"

Specifies an alternate job completion notification command (see the "notify" keyword). If the specified command contains embedded spaces, enclose the entire *command_string* in quotes.

If the specified command contains the two-character sequence { }, the sequence is replaced by the text "MSC/NASTRAN job *name* completed".

Note: The following example may not work on your system. The "mail" utility on HP 9000 systems does not accept the "-s" option, and the "whoami" command does not exist on UNICOS.

Example:

```
msc705 nastran example notify=yes \
                ncmd="echo {} | mail -s {}
'whoami'"
```

At the end of the job, mail is sent to the user submitting the job.

The braces in the "ncmd" value are replaced by the job completion text, and the modified command is run:

```
echo "MSC/NASTRAN job example completed" |\
      mail -s "MSC/NASTRAN job example completed"
user
```

news news={yes|no|auto} Default: Yes

Displays the news file (*install_dir*/msc705/nast/news.txt) in the F06 file. If "auto" is specified, the news file is only displayed if it has been modified since the last time it was displayed for you. If "yes" is specified, the news file is displayed in the F06 file regardless of when it was last changed. If "no" is specified, the news file is not displayed in the F06 file.

Example:

```
msc705 nastran example news=yes
```

The news file is displayed in the F06 file after the title page block.

Note: The news file can also be displayed on the terminal by using the command:

```
msc705 nastran news
```

node node=*nodename* Default: *None*

Execute the job on the specified node. Use the “username” keyword to specify an alternate user name on the remote node. This keyword may only be specified on the command line.

Example: `msc705 nastran example node=othernode`

This example runs the job on node “othernode”. The following special processing will occur:

1. The specified node must have a valid MSC/NASTRAN installation and the command name used to start the analysis on the local node, e.g., nastran, must be available on the remote node. Since the remote shell command is used to run the analysis on the remote node, normal login processing (i.e., .login for C-Shell and .profile for Bourne/Korn shell) is not performed. The “/etc/hosts.equiv” and your “.rhosts” files on the remote system must allow access from the local system.
2. All RC files on the local node are ignored.
3. If your current username on the local node is not valid on the remote node, specify the remote node username with the “username” keyword.
4. The value of the “scratch” keyword is forced to “scratch=yes”.
5. If “sdirectory” is not set on the command line, the RC files on the remote node are used to determine the scratch directory.
6. The input file is copied to the remote node directory specified by the “sdirectory” keyword.
7. The analysis is run on the remote node with all output and disk files directed to the scratch directory.
8. If “old=yes” was specified on the command line, the current F06, F04, LOG, OP2, PLT, PCH, and OUT files are saved.
9. The F06, F04, LOG, OP2, PCH, PLT, and XDB files are copied back to the local node and name as specified by the “out” keyword. The XDB file is copied as specified by the “trans” keyword on the command line.
10. The F06, F04, and LOG files are handled as specified by the “append” keyword on the command line.

notify notify={yes|no} Default: Yes

Sends notification when the job is completed. See also the “ncmd” keyword in Appendix B.

Note: If the job is queued using the “queue” keyword, or the job is already running in an NQS batch job, the default is “no”.

Example: `msc705 nastran example notify=yes`

A message is sent when the job completes.

old old={yes|no} Default: Yes

Saves previous copies of the F04, F06, LOG, OP2, OUT, PCH, and PLT output files using sequence numbers. Sequence numbers are appended to the keyword filename and are separated by a period.

If “yes” is specified, the highest sequence number of each of the output files is determined. The highest sequence number found is incremented by one to become the new sequence number. Then, all current output files that do not include sequence numbers are renamed using the new sequence number as a suffix.

Example: `msc705 nastran example old=yes`

For example, the user’s directory contains the following files:

v2401.dat	v2401.f04	v2401.f04.1	v2401.f04.2	v2401.f06
v2401.log	v2401.log.1	v2401.log.2	v2401.log.3	v2401.f06.1

Apparently, the user ran the job four times, but deleted some of the files. When the next job is run, the following files are renamed: v2401.f04 is renamed to v2401.f04.4, v2401.f06 is renamed to v2401.f06.4, and v2401.log is renamed to v2401.log.4. The sequence number 4 is used because it is one greater than the highest sequence number of all of the selected files (the highest being v2401.log.3). Using this method, all files related to a single run will have the same sequence number.

out out=*pathname* Default: **.**

Saves the output files using a different file prefix or in a different directory. If “out” is not specified, the output files are saved in the current directory using the basename of the input data file as a prefix.

If the “out” value is a directory, output files are created in the specified directory using the basename of the input data file as the filename. In the following examples, assume the following directory contents:

```
example.dat             mydir/          other/
other/example.dat
```

where “mydir” and “other” are directories.

Example: `msc705 nastran example`
or: `msc705 nastran other/example`

Output files are created in the current directory with name example, e.g., ./example.f06.

Example: `msc705 nastran example out=myfile`

Output files are created in the current directory with name myfile, e.g., ./myfile.f06.

Example: `msc705 nastran example out=mydir`

Output files are created in the mydir directory with name example, e.g., mydir/example.f06.

Example: `msc705 nastran example out=mydir/myfile`

Output files are created in the mydir directory with name myfile, e.g., mydir/myfile.f06.

parallel parallel=*value* Default: 0

Not available on Fujitsu, Hitachi, or IBM. Specifies the maximum number of CPUs selected for parallel processing in several numeric modules. Parallel processing reduces elapsed time at the expense of CPU time. The default is 0, which specifies no parallel processing. If “parallel=1”, the parallel algorithms are used on one processor.

Note: If you need to vary the number of CPUs during a job, you must set either the “parallel” keyword or SYSTEM(107) on a NASTRAN statement to the maximum number of CPUs that will be requested. Some systems cannot process a DMAP request for CPUs in excess of this initial value.

Example: `msc705 nastran example parallel=2`

The job is run on a maximum of two CPUs.

pause pause=keyword Default: no

Pause the nastran command before exiting to wait for the “Enter” of “Return” key to be pressed. This can be useful when the nastran command is embedded within another program. The values are “fatal”, “information”, “warning”, “yes”, and “no”. Setting “pause=yes” will unconditionally wait; “pause=fatal” will only wait if a fatal message has been issued by the nastran command; “pause=information” and “pause=warning” will similary wait only if an information or warning message has been issued. The default is “pause=no”, i.e., do not wait when the nastran command ends.

pcmd pcmd=*command_string* Default: lpr -f

Specifies an alternate print command. The command must be a valid Bourne-shell command. The F04, F06, and LOG files are piped to the specified command. If the specified command contains embedded spaces, enclose the entire *command_string* in quotes.

Example: `msc705 nastran example prt=yes pcmd="enscript –f"`

The output files are printed using the enscript command.

pdel pdel={yes|no} Default: No

Indicates if the output files F04, F06, and LOG are to be deleted after printing. If "prt" is not set to "yes", this keyword has no effect.

Example: `msc705 nastran example prt=yes pdel=yes`

The output files are printed using the default print command (see the "pcmd" keyword in Appendix B) and are then deleted.

post post=*command_string* Default: *None*

Runs the specified command after the job has completed and after the F06, F04, and LOG files have been concatenated if append=yes is specified. The command must be a valid Bourne shell command. If the specified command contains embedded spaces, enclose the entire *command_string* in quotes. Each occurrence of the "post" keyword will be concatenated together to form a sequence of commands. Specify a null value, i.e., "post=" to erase all of the previously entered commands. Typical uses of this keyword are to run postprocessing programs or to compress the output files to save space.

Example: `msc705 nastran example post='gzip example*'`

At the end of the job, the command gzip example* is run to compress all files beginning with "example".

The value of the "out" keyword is available for use by the "post" keyword. The example "post" keyword could also have been written as post='gzip $MSC_OUT.*'. If app=yes was specified, post='gzip $MSC_OUT.out' would only compress the output file.

See Section B.3 for a list of environmental variables that may be used in the post command.

Note: In order to allow the "post" keyword to operate on the output files, the standard output from the post commands is not written to the output files.

ppcdelta ppcdelta=*time* Default: *None*

Note: The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Specifies the amount of time to subtract from the specified CPU time to determine the per-process cpu time limit. This subtraction will ensure that MSC/NASTRAN does not consume all of the time allocated to the job.

The value can be specified as either “*hours*:*minutes*:*seconds*”, “*minutes*:*seconds*”, or “*seconds*”, and will always be converted to the number of seconds.

Example: `msc705 nastran example queue=small cpu=1000 ppcdelta=5`

The job is submitted to the small queue with a total CPU time limit of 1000 seconds; the MSCNASTRAN job will be limited to 995 seconds.

ppmdelta ppmdelta=*memory_size* Default: *105% of executable size*

Note: The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Specifies the amount of memory to add to the “memory” value to determine “ppm”, the per-process memory value. The per-process limit is the total amount of memory that each process may acquire. This includes the executable, open core memory (via the “memory” keyword), disk file buffers, and etc. (Cray systems also include EAG FFIO cache).

The *memory_size* can be specified either as a number of words or as a number followed by one of the following modifiers:

G or Gw	Multiply *memory_size* by 1024**3.
Gb	Multiply *memory_size* by (1024**3)/*bytes_per_word*.
M or Mw	Multiply *memory_size* by 1024**2.
Mb	Multiply *memory_size* by (1024**2)/*bytes_per_word*.
K or Kw	Multiply *memory_size* by 1024.
Kb	Multiply *memory_size* by 1024/*bytes_per_word*.

w	Use *memory_size* as is.
b	Divide *memory_size* by *bytes_per_word*.

where *bytes_per_word* is 8 on Cray and NEC; 4 on all others. The modifiers may be specified using any case combination.

Note: MSC/NASTRAN now uses standard computer units for K, M, and G. Prior releases used engineering units.

If *memory_size* is less than 1000, then “ppmdelta” equals *memory_size* divided by 100 and multiplied by the size of the executable, i.e., 105 specifies the default 105% of executable size.
If *memory_size* is greater than 1000, but less than the size of the executable, then “ppmdelta” equals *memory_size* plus the executable size.
If *memory_size* exceeds the size of the executable, then “ppmdelta” equals *memory_size*.

Example: `msc705 nastran example queue=small mem=100m ppmdelta=10m`

The job is submitted to the small queue with a open core size of 100 megawords, and a per-process memory limit of 110 megawords.

pre pre=*command_string* Default: *None*

Runs the specified command before the job begins. The command must be a valid Bourne-shell command. If the specified command contains embedded spaces, enclose the entire *command_string* in quotes. Each occurrence of the “pre” keyword will be concatenated together to form a sequence of commands. Specify a null value, i.e., “pre=” to erase all of the previously entered commands.

Note: The following example may not work on your system. The “whoami” command does not exist on UNICOS.

Example:
```
msc705 nastran example \
pre=”echo Job beginning | mail ‘whoami‘”
```

Sends mail to the submitting user immediately before beginning the job.

See Section B.3, for a list of environmental variables that may be used in the “pre” command.

prmdelta prmdelta=*per_request_mem_delta* Default: 5120

Note: The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Specifies the amount of memory to add to the specified “ppm” value to determine “prm”, the per-request or per-job memory value. The per-job limit is the total amount of memory that all processes in the job may acquire. This includes the MSC/NASTRAN process plus any other concurrent or parent processes. The minimum value is 5120.

The *memory_size* can be specified either as a number or words or as a number followed by one of the modifiers described above.

Example: `msc705 nastran example queue=small prmdelta=10k`

The per-job memory limit is 10 kilowords larger than the per-process memory limit.

processor processor=*file_type* Default: *Computer dependent*

Specifies the file type of the solver executable. On some computers, MSC/NASTRAN provides more than one executable. The baseline executable has the filename “analysis” while the other executables are named “analysis.*file_type*”, e.g., “analysis.power2” on IBM, “analysis.ultra” on Sun systems. The nastran command will select the correct executable based on the current computer. In some cases, it may be desirable to use one of the other executables. For example, to run the baseline executable on an advanced system, specify “proc=”. To run the advanced executable on a new computer not correctly identified by the nastran command, specify “proc=*file_type*”.

Note: This keyword overrides the processor selection logic.

prt prt={yes|no} Default: No

Prints the output files F04, F06, and LOG at the end of the run. See also the “pcmd” and “pdel” keywords in Appendix B.

Example: `msc705 nastran example prt=yes`

The files example.f04, example.f06, and example.log are printed using the default print command.

qoption qoption=*string* Default: *None*

Note: The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Defines the options to add to the queue submittal command. See the “submit” keyword in Section 3.11.

Example: `msc705 nastran example queue=small qoption=–mu`

The job is run with the additional job submission parameter “-mu” if the keyword reference %qopt% was included in the queue’s command definition.

queue queue=*string* Default: *None*

Note: The following capability is dependent upon the queue submission commands defined by the “submit” keyword and your queuing system. The capability or examples may not work on your system.

Specifies the name of the queue to use for job submittal. This keyword requires the submit keyword to define the available queues and queue-submittal commands. See Section 3.11 for details on the “submit” keyword.

Example: `msc705 nastran example queue=small`

This example submits the job to the small queue.

rank rank=*number* Default: *See Appendix C*

Sets both SYSTEM(198) and SYSTEM(205) to the specified value. SYSTEM(198) and SYSTEM(205) set the minimum front size and number of rows that are simultaneously updated, respectively, in sparse symmetric decomposition and FBS. The sparse solver will build a front, a $k \times k$ submatrix, until k is at least as large as SYSTEM(198). Once a sufficiently large front has been built, it is updated m rows at a time, where m is the value of SYSTEM(205). For best performance, SYSTEM(205) should always be greater than or equal to SYSTEM(198); the optimal values for these system cells is problem and processor dependent. The default values for these system cells are set by MSC/NASTRAN to processor-dependent values and are always equal. The actual value used for SYSTEM(205) may be found in the F04 file in the text of USER INFORMATION MESSAGE 4157 as the RANK OF UPDATE value. See Table C-20 of Section C.3 in Appendix C for the default values of these system cells.

real real=*memory-size* Default: 0

Specifies the amount of open core memory that certain numerical modules will be restricted to. This keyword may be used to reduce paging, at the potential expense of spilling. The keyword may also be set with the “sys81” keyword. See the *MSC/NASTRAN Quick Reference Guide* for further information.

The memory_size set using this keyword can be specified as a number of words or as a number followed by one of the following modifies:

G or Gw	Multiply *memory_size* by 1024**3.
Gb	Multiply *memory_size* by (1024**3)/*bytes_per_word*.
M or Mw	Multiply *memory_size* by 1024**2.
Mb	Multiply *memory_size* by (1024**2)/*bytes_per_word*.
K or Kw	Multiply *memory_size* by 1024.
Kb	Multiply *memory_size* by 1024/*bytes_per_word*.
w	Use *memory_size* as is.
b	Divide *memory_size* by *bytes_per_word*.

where *bytes_per_word* is 8 on Cray and NEC; 4 on all others. The modifiers may be specified using any case combination.

rcf rcf=*pathname* Default: *None*

Specifies the name of the local RC file. If this keyword is not specified, the .nast705rc file from the data file directory is used.

Example: `msc705 nastran example rcf=nast.rc`

scratch scratch={yes|no|mini} Default: No

Deletes the database files at the end of the run. If the database files are not required, “scratch=yes” can be used to remove them preventing cluttering of the directory with unwanted files. If “mini” is specified, a reduced size database that can only be used for data recovery restarts will be created. See Chapter 12 of the *MSC/NASTRAN Reference Manual* for further details on the “mini” database.

Example: `msc705 nastran example scratch=yes`

All database files created by the run are deleted at the end of the job in the same way as the FMS statement INIT MASTER(S).

sdirectory sdirectory=*directory* Default: *See the description below.*

Note: See Section 4.3 for information on estimating a job's total disk space requirements.

Specifies the directory to use for temporary scratch files created during the run. MSC/NASTRAN can create very large scratch files, the scratch directory should contain sufficient space to store any scratch files created during a run. You must have read, write, and execute privileges to the directory.

The default value is taken from the TMPDIR environment variable if it is set to a nonnull value. Otherwise the computer's default temporary file directory is chosen; this is usually /tmp, but on Silicon Graphics systems, it is /var/tmp.

Example: `msc705 nastran example sdir=/scratch`

Scratch files are created in the directory /scratch.

smemory smemory=*value* Default: 0 (Cray UNICOS and NEC); 100 (all others)

Specifies the default number of GINO blocks to reserve for scratch memory.

Note: This keyword is overridden by the FMS statement ASSIGN SCRATCH(MEM=*value*).

Example: `msc705 nastran example smem=200`

This example reserves 200 GINO blocks for scratch memory.

sparse sparse=*number* Default: *See the description below.*

Sparse matrix method selection. This value may also be set with the "sys126" command line keyword. See the *MSC/NASTRAN Quick Reference Guide* for information on the default value and legal values for this keyword.

spintime spintime=*value* Default: 10 000 000

For Silicon Graphics R8K, R10K systems only. Specifies the number of times to wait in a spin-wait loop before blocking the thread. MSC/NASTRAN slave threads (i.e., the threads that will run MSC/NASTRAN subtasks) spin wait until there is work to do. This makes them immediately available when a parallel region is reached. However, spin waiting wastes processor resources. After a specified spin-wait time has elapsed, the threads block themselves using a system call. Note that blocking is transparent to

MSC/NASTRAN; blocked threads are automatically unblocked when a parallel region is reached. Once a thread is blocked, another system call is required to activate it again. This makes the response time much longer when starting up a parallel region. The default value will take approximately one half second on a 75MHz R8K processor. If the value is set to zero, the slave threads will block themselves immediately.

This value may also be set by the MSC_SPINTIME environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

subcomplex subcomplex=*name* Default: *current complex*

For Hewlett Packard Exemplar systems only. Specifies the name of the subcomplex where the MSC/NASTRAN job is to be run. This may be desirable if specific subcomplexes have been configured to run MSC/NASTRAN.

Example: `msc705 nastran example subcomplex=crunch`

This example runs the MSC/NASTRAN job on the subcomplex named “crunch”.

submit submit=[*queue_list*=]*queue_definition*

Defines the command and options used to run a job when the “queue” keyword is specified. The “submit” keyword, only specified in RC files, consists of an optional queue list, followed by the command definition for the specified queues as shown below:

```
submit=queuelist=command_definition
submit=command_definition
```

When specified, the *queuelist* contains one or more “queue” names separated by commas. If a queue list is not supplied, the *command_definition* applies to all queues.

The command definition section of the “submit” keyword value defines the command used to run a job when a “queue” keyword is supplied that matches a queue name in the queuelist. The command definition section can contain keyword names enclosed in percent “%” signs that are replaced with the value of the keyword before the command is run. A complete description of the command definition is found in Section 3.11.

symbol symbol=*symbolic_name*=*string* Default: *None*

Defines a symbolic (or logical) name used on ASSIGN and INCLUDE statements and in command line arguments. This statement can only be specified in initialization or RC files. It *cannot* be specified on the command line (although logical symbols defined using this keyword may be used on the command line). Symbolic

names must be 16 characters or less, the value assigned to the symbolic name must be 256 characters or less. If the symbolic name used in ASSIGN or INCLUDE statements or in command line arguments is not defined, it is left in the filename specification as is.

For example, many of the TPL and DEMO input data files have ASSIGN statements, such as the following:

```
ASSIGN  'MASTER=DBSDIR:abc.master'
```

The string "DBSDIR:" specifies a symbolic name that is to be replaced by another string. The replaced string is defined by the "symbol" keyword in the initialization or RC file or as an environment variable. For example,

```
SYMBOL=DBSDIR=/dbs
```

When the previous ASSIGN statement is processed, the filename assigned to the logical name MASTER is "/dbs/abc.master". An alternate way of defining symbolic names is through the use of environment variables. For example, issuing the following command at a Bourne or Korn shell prompt

```
DBSDIR=/dbs; export DBSDIR
```

or the following at a C shell prompt

```
setenv DBSDIR /dbs
```

is equivalent to the above "symbol" keyword.

Note: If a symbolic name is defined by both an RC file and an environment variable, the symbol statement value will be used.

Section B.3 contains a list of environment variables that are automatically created by the nastran command. Of particular importance to the logical symbol feature are the OUTDIR and DBSDIR variables. These variables refer to the directory that will contain the output files (set using the "out" keyword) and the directory that will contain the permanent database files (set using the "dbs" keyword), respectively.

sysfield sysfield=*string* Default: *None*

Defines a global SYS value that is applied to all DBsets. See Section 5.4.1 or B.2 for further details on the SYS parameter and the values legal on your computer.

Example: `msc705 nastran example sysfield=lock=no`

This example disables file locking for all DBsets.

sys*n* sys*n*=*value* Default: *None*

Sets the system cell *n* to *value*. This keyword may be repeated any number of times. All nonrepeated cells are used, but only the last repeated cell is used. Each keyword-value string must be less than or equal to 256 characters in length. The form used in the prior release, "system(*n*)=*value*", may also be used, but the entire keyword-value string must be quoted when used on the command line.

Example: `msc705 nastran example sys2=19`

This example sets system cell 2 to the 19.

threads threads=*value* Default: *None*

For Silicon Graphics R8K, R10K systems only. A number of features are provided in the R8K, R10K version of MSC/NASTRAN that allow sophisticated users to override multiprocessing defaults and tailor a job's parallelism to their particular requirements.

Threads are used by IRIX to implement MSC/NASTRAN tasks. For maximal performance, there should be one thread per MSC/NASTRAN task and one processor per thread. An excess number of threads will not help performance; if there are more MSC/NASTRAN tasks than threads or more threads than processors, a longer elapsed time will result.

The Dynamic Thread Management feature is available only in the MSC/NASTRAN Rank-N sparse solver (see the "rank" keyword). The Rank-N sparse solver is used widely in linear static analysis and Lanczos eigenvalue analysis jobs. Other MSC/NASTRAN parallel modules will run with a constant number of threads specified by the PARALLEL keyword.

The "threads" keyword specifies the suggested number of threads to be maintained by the Dynamic Thread Management feature. Setting a value for "threads" causes the runtime library to create an additional asynchronous "monitor" process that periodically awakens to monitor system load. When idle processors exist, this monitor process increases the number of threads up to the maximum that is specified by the "parallel" keyword. As the system load increases, the monitor process decreases the number of threads, possibly to as few as one. If "threads" has not been set, this feature is disabled and the constant number of threads specified via the "parallel" keyword will be used.

This value may also be set by the MP_SUGNUMTHD environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

thread_max thread_max=*value* Default: *parallel*

Silicon Graphics R8K, R10K systems only. Specifies an upper bound on the number of threads that a job will use when “threads” is also set. The value must satisfy the relation *thread_min* ≤ *thread_max* ≤ *parallel*, where *parallel* is the value specified by the “parallel” keyword.

This value may also be set by the MP_SUGNUMTHD_MAX environment variable. The environment variable overrides the RC files, and the command line overrides the environment variable.

thread_min thread_min=*value* Default: 1

Silicon Graphics R8K, R10K systems only. Specifies a lower bound on the number of threads a job will use when “threads” is also set. The value must satisfy the relation 1 ≤ *thread_min* ≤ *thread_max*.

This value may also be set by the MP_SUGNUMTHD_MIN environment variable. The environment variable overrides the RC files, the command line overrides the environment variable.

thread_verbose thread_verbose={yes|no}
Default: No

Silicon Graphics R8K, R10K systems only. Controls the output of informational messages. If “thread_verbose=yes” is set, the monitoring process will write messages to the LOG file whenever it changes the number of threads.

This value may also be set by the MP_SUGNUMTHD_VERBOSE environment variable. The environment variable overrides the RC files, the command line overrides the environment variable.

trans trans={yes|no|auto} Default: no *(local)*; auto *(remote)*

If the “node” keyword is not specified, this keyword indicates the XDB file is to be translated to a neutral-format file using the TRANS utility. The output file will have the file type “.ndb”.

If the “node” keyword is specified, this keyword indicates how an XDB file is to be copied back to the local node. If “trans=auto” is specified, the XDB file will be copied using TRANS/RECEIVE if the two computers use different floating point formats or by a binary copy if the floating point formats are the same. If “trans=yes” is specified, the XDB is always copied using TRANS on the remote node and RECEIVE on the local node (this may be needed if the floating point formats are identical but the file formats are not). If “trans=no” is specified, the XDB file will not be copied back

Example: `msc705 nastran example trans=yes`

This example will run MSC/NASTRAN and then convert the XDB file, if written, to neutral format using TRANS.

Example: `msc705 nastran example node=othernode trans=yes`

This example will run MSC/NASTRAN on node othernode and copy the XDB file back using TRANS/RECEIVE.

username — username=*name* — Default: *Current user name*

Specifies an alternate username on the remote host when the “node” keyword is specified. This keyword may only be specified on the command line.

Example: `msc705 nastran example node=othernode user=fred`

This example will run MSC/NASTRAN on node othernode as user “fred”.

usparse — usparse=*number* — Default: *See the description below.*

Unsymmetrix sparse matrix method selection. This value may also be set with the “sys209” command line keyword. See the *MSC/NASTRAN Quick Reference Guide* for information on the default value and legal values for this keyword.

version — version=*version_number* — Default: *Latest installed version*.

Specifies the version number. The keyword may only be specified on the command line or in the command initialization file (see Section 3.5).

Example: `msc705 nastran example version=68.2`

This example will run MSC/NASTRAN V68.2 assuming it has been installed in the same installation base directory as the MSC/NASTRAN V70.5.

xhost — xhost={yes|no} — Default: No

Indicates if the xhost(1) command is to be run. The xhost command may be required if the “node” keyword and either “xmon=yes” or “xmon=kill” are specified. The argument to xhost(1) will be the node specified by the “node” keyword. This keyword is ignored if the “node” keyword is not specified.

xmonast — xmonast={yes|no|kill} — Default: No

Indicates if XMONAST is to be run to monitor the MSC/NASTRAN job. If “xmonast=yes” is specified, XMONAST will be automatically started; you must manually exit XMONAST when the MSC/NASTRAN job has completed. If “xmonast=kill” is specified, XMONAST will start and will automatically exit when the MSC/NASTRAN job has completed.

Example: `msc705 nastran example xmon=kill`

This example runs the XMONITOR utility while the MSC/NASTRAN job is running. Once the job completes, the XMONITOR program is automatically terminated.

B.2 SYS Parameter Keywords

buffio — buffio={yes|no|must} — Default: No

Systems supporting Buffered I/O (see Table 4-5). Specifies if the file is to be buffered. If “buffio=yes” is specified and a memory allocation operation fails, then unbuffered disk I/O will be used. If “buffio=must” is specified and a memory allocation operation fails, then a fatal error will be issued and the job terminated. See Section 5.4.3 for further information on buffered I/O.

lock — lock={yes|no} — Default: No for Delivery DBsets; Yes for all others.

Specifies if the file is to be locked when it is opened. Locking a file prevents two or more MSC/NASTRAN jobs from interfering with one another; however, this does not prevent any other program or operating system command from modifying the file.

SYSTEM(207) can also be used to globally control DBset locking. Setting SYSTEM(207)=1 will disable locking unless overridden for a specific file by SYS=LOCK=YES on an INIT FMS statement. Setting SYSTEM(207)=0 will enable locking of read-write DBsets unless overridden for a specific file by SYS=LOCK=NO on an INIT FMS statement.

mapio — mapio={yes|no|must} — Default: No

Systems supporting File Mapping (see Table 4-5). Specifies if the file is to be mapped. If “mapio=yes” is specified and a mapping operation fails, then normal disk I/O will be used. If “mapio=must” is specified and a mapping operation fails, then a fatal error will be issued and the job terminated. See Section 5.4.2 for further information on file mapping.

raw — raw={yes|no} — Default: Yes.

Cray only. Indicates if RAW I/O is to be used to read and write the file. RAW I/O should be on for files accessed via EAG FFIO.

wnum — wnum=*number* — Default: 4

Systems supporting File Mapping or Buffered I/O (see Table 4-5). Specifies the number of windows or buffers that will be maintained

for each mapped or buffered file. The use of multiple windows or buffers permits multiple I/O streams to target a file (e.g., simultaneously reading one matrix and writing another) without forcing an excessive number of window remap operations or buffered read/writes. The number must be between 1 through 16 inclusive, values outside of this range are ignored without acknowledgement.

wsize wsize=*memory_size* Default: 128 KB

Systems supporting File Mapping or Buffered I/O (see Table 4-5).

File Mapping. Specifies the size of the window mapping the file into memory. The window is that portion of the file that is visible through the map. If the window is the same size as the file, then the entire file is visible. If the window is smaller than the file, then any portion of the file within the window or windows can be directly accessed; the rest of the file cannot be accessed until a window is remapped to include the desired file location.

Buffered I/O. Specifies the size of the buffer read from or written to disk. If the buffer is the same size as the file, then the entire file is memory resident. If the buffer is smaller than the file, then any portion of the file within the buffer or buffers can be directly accessed; the rest of the file cannot be accessed until a buffer is read to include the desired file location.

The window or buffer size is limited to 25% of the available address space. The address space limit is displayed by the “limits” special function (see Section 4.1.3) as the “Virtual Address Space” limit. If “wsize=0” is specified for a read-only file, the entire file will be mapped or buffered into memory (subject to the 25% address space limit). The *memory_size* can be specified either as the number of words or as a number followed by one of the following modifiers:

M or Mw	Multiply *memory_size* by 1024**2.
Mb	Multiply *memory_size* by (1024**2)/4.
K or Kw	Multiply *memory_size* by 1024.
Kb	Multiply *memory_size* by 1024/4.
w	Use *memory_size* as is.
b	Divide *memory_size* by 4.

The modifier may be specified using any case combination.

If *memory_size* is less than the file’s BUFFSIZE, then *memory_size* is multiplied by BUFFSIZE.

B.3 Environment Variables

The following environment variables will affect the execution of the nastran command.

Name	Purpose
DISPLAY	The default display for xmonast.
FF_IO_DEFAULTS	Overrides “ff_io_defaults” keyword in an RC file.
FF_IO_OPTS	Overrides “ff_io_opts” keyword in an RC file.
HOME	The user’s home directory.
LOGNAME	The user ID.
MSC_BASE	If set, the script will use this directory as the *install_dir*.
MSC_NOEXE	If set, the nastran command will build the execution script but will not actually execute it. This may be useful for debugging purposes.
MSC_VERSD	MSC use only.
TMPDIR	If set, this is the default value for the “sdirectory” keyword. If not set, use the system default temporary file directory as the default value.
USER	The user’s home directory (if LOGNAME is not set or is a null string).

The following environmental variables are available for use by the “pre” and “post” keywords.

Name	Purpose
DBSDIR	The directory part of MSC_DBS, i.e., the directory that will contain the permanent database files.
DELDIR	Directory containing the solution sequence source files (*install_dir*/msc705/nast/del).
DEMODIR	Directory containing DEMO library (*install_dir*/msc705/nast/demo).
JIDDIR	Directory containing the input file.
MSC_APP	yes,no
MSC_ASG	MSC use only.
MSC_ARCH	The actual architecture used by the nastran command.
MSC_AUTH	Pathname of authorization file or “*MSC-SL*” (*port@host*).
MSC_BASE	The actual *install_dir* used by the nastran command.
MSC_DBS	Default prefix of permanent databases.
MSC_EXE	Executable path.
MSC_JID	Input data file path.
MSC_MEM	Open core memory size in words.
MSC_OLD	yes,no
MSC_OUT	Prefix of F06, F04, and LOG files.
MSC_SCR	yes,no
MSC_SDIR	Default prefix of scratch databases.
MSC_VERSD	MSC use only.
OUTDIR	Output file directory.
SSSALTERDIR	Directory containing SSS alters (*install_dir*/msc705/nast/misc/sssalter).
TMPDIR	Your default temporary directory, or “sdirectory” if not set.
TPLDIR	Directory containing TPL library (*install_dir*/msc705/nast/tpl).

B.4 Other Keywords

The following keywords are available for use by the nastran command and script templates. You will generally not need to set or use these values.

Keyword	Purpose
0	Pathname of driver program.
0.ini	Command initialization file pathname.
0.lcl	Local job template pathname.
0.rmt	Remote job template pathname.
0.tmplt	Alternate template pathname, overrides local/remote template selection logic.
a.appdir	Application specific base pathname relative to MSC_BASE.
a.archdir	Architecture specific base pathname relative to MSC_BASE.
a.estimate	ESTIMATE executable filename relative to “a.archdir”.
a.fms	Comma-separated list of FMS keywords recognized in RC files.
a.k	Multiplier for K factor.
a.news	News filename relative to “a.appdir”.
a.rc	RC file basename. User RC files are prefixed by “.”.
a.receive	RECEIVE executable filename relative to “a.archdir”.
a.release	Release number, same as MSC/NASTRAN version number.
a.solver	Solver executable filename relative to “a.archdir”.
a.sss	Delivery database filename relative to “a.archdir”.
a.tier	MSC internal variable.
a.touch	News file touch pathname.
a.trans	TRANS executable filename relative to “a.archdir”.
a.xmonitor	XMONAST executable filename relative to “a.archdir”.
dcmd	Debugger.
debug	Run solver under debugger.
job	Job script filename, created in out directory.
j.base	Job basename.
j.command	Job submittal command string.
j.dat	Default input data file suffix. The default is “.dat”.
j.dir	Job directory.

Keyword	Purpose
j.env	Job environment variable list.
j.msg	Job completion message.
j.mtdel	List of suffixes. Delete these files if empty.
j.nascar	List of NASTRAN entries.
j.news	News file pathname.
j.out	Appended output file type.
j.rcfiles	Comma-seperated list of RC files.
j.shell	Shell debugging flag.
j.suffix	Space seperated list of file types to be versioned.
j.title	Title of XMONAST icon.
j.tty	TTY name.
j.unique	Job unique name.
log	Pathname of LOG file.
msg	System message destination.
nprocessors	Number of processors.
ppc	Per-process CPU time limit.
ppm	Per-process memory limit.
prm	Per-request memory limit.
PWD	Current working directory.
r.jid	“jid” on remote node.
r.out	“out” on remote node.
s.arch	System architecture name.
s.block	Words per disk block.
s.bpw	Bytes per word.
s.config	CONFIG number.
s.hostname	Simple hostname.
s.model	System model name.
s.modeldata	Pathname of site specific model data.
s.numeric	Encoded numerical format.
s.nproc	Number of processors.
s.os	OS name.
s.osv	OS version.

Keyword	Purpose
s.pmem	Physical memory in words.
s.proc	Default processor subtype.
s.rawid	Raw configuration number.
s.rsh	Remote shell command.
s.type	System description.
tcmd	Timing command.

APPENDIX C

SYSTEM DESCRIPTION

This appendix presents quantitative information useful when evaluating the processing requirements of MSC/NASTRAN.

C.1 System Descriptions

Table C-1. System Description - Cray C90, T90 (Not IEEE T90).

Items	Descriptions
Supported Machine Model(s)	C90, D90, T90
Models With Timing Constants Installed.	C90, T90
Operating System(s)	UNICOS 9.0
Compiler Used	f90 3.0.2
Compiler Options	-ev -Ooverindex -Oaggress -Oscalar3,vector3
Word Length	64 bits
Memory Management	Real
Size of Executable	6.9 MW (3.7 MW shared)
Maximum Size of Open Core	Lesser of physical memory or available swap space

Table C-2. System Description - Cray IEEE T90.

Items	Descriptions
Supported Machine Model(s)	T90
Models With Timing Constants Installed.	T90
Operating System(s)	UNICOS 9.1
Compiler Used	f90 3.0.2
Compiler Options	-ev -Ooverindex -Oaggress -Oscalar3,vector3
Word Length	64 bits
Memory Management	Real
Size of Executable	6.8 MW (3.9 MW shared)
Maximum Size of Open Core	Lesser of physical memory or available swap space

Table C-3. System Description - Cray J90, Y-MP.

Items	Descriptions
Supported Machine Model(s)	J90, EL, Y-MP
Models With Timing Constants Installed.	J90, EL, Y-MP
Operating System(s)	UNICOS 9.0
Compiler Used	f90 3.0.1
Compiler Options	-ev -Ooverindex -Oaggress -Oscalar3,vector3
Word Length	64 bits
Memory Management	Real
Size of Executable	6.3 MW (3.6 MW shared)
Maximum Size of Open Core	Lesser of physical memory or available swap space

Table C-4. System Description - Digital.

Items	Descriptions
Supported Machine Model(s)	EV4, EV5
Models With Timing Constants Installed.	3000/800, 3000/500, 2100–4/275, 2100-5/300, 500/400, 2100-5/300, 500/500
Operating System(s)	UNIX Version 4.0B-564
Compiler Used	f77: Digital FORTRAN V4.1-92 cc: DEC C V5.2-033
Compiler Options	-assume noaccuracy -O4 -tune ev5 -math_library fast -om -assume dummy
Word Length	32 bits
Memory Management	Virtual
Size of Executable	28.3 MB
Maximum Size of Open Core	Lesser of 8GB or up to available swap space

Table C-5. System Description - Fujitsu

Items	Descriptions
Supported Machine Model(s)	VPP300, VX
Models With Timing Constants Installed	310
Operating System(s)	UXP/V V10L10
Compiler Used	frt V10L10
Compiler Options	-c -sc -Wv,-Of -Oe,-e,-p,-u -Ab
Word Length	32 bits
Memory Management	Virtual
Size of Executable	49.6 MB
Maximum Size of Open Core	Up to available swap space

Table C-6. System Description - HP 9000.

Items	Descriptions
Supported Machine Model(s)	PA-RISC 1.1, PA-RISC 2.0
Models With Timing Constants Installed.	710, 712, 715, 720, 730, 735, 778, 819, 889
Operating System(s)	PA-RISC 1.1: HP-UX 10.20 PA-RISC 2.0: HP-UX 10.20 PA-RISC 2.0: HP-UX 11.0
Compiler Used	PA-RISC 1.1: f77 10.20 HP-UX 10.20: PA-RISC 2.0: f77 10.20 PA-RISC 2.0, HP-UX 11.0: f77 1.2
Compiler Options	PA-RISC 1.1: +ppu -K -O3 +OParallel_env +Onolimit +DS770 +DA1.1 +Olibcalls +Q PA-RISC 2.0: +ppu -K -O3 +OParallel_env +Onolimit +DS2.0a +DA2.0N +Olibcalls +Q PA-RISC 2.0, HP-UX 11.0: +ppu -K +02 +Olibcalls +DS2.0 +DA2.0 +Onolimit +Odataprefetch
Word Length	32 bits
Memory Management	Virtual
Size of Executable	PA-RISC 1.1: 28.5 MB PA-RISC 2.2: 28.0 MB PA-RISC 2.0, HP-UX 11.0: 29.7MB
Maximum Size of Open Core	Up to available swap space

Table C-7. System Description - HP Exemplar.

Items	Descriptions
Supported Machine Model(s)	2000, 2200
Models With Timing Constants Installed.	2000
Operating System(s)	SPP-UX 5.1
Compiler Used	f77 V1.1 970109, cc 10.32.03
Compiler Options	+ppu -K +O2 +Onolimit +Olibcalls +DS2.0 +DA2.0
Word Length	32 bits
Memory Management	Virtual
Size of Executable	28.7 MB
Maximum Size of Open Core	Up to available swap space

Table C-8. System Description - IBM.

Items	Descriptions
Supported Machine Model(s)	Power Classic, Power 2, PowerPC
Models With Timing Constants Installed.	320H, 375, 390, 530, 560, 580, 590, 950, 980E, 990
Operating System(s)	AIX 4.1.4
Compiler Used	xlf 3.2
Compiler Options	Power: -03 -qextname -qstrict -qflttrap:ov:zero:inv:en:imp Power2: -03 -qextname -qstrict -qflttrap:ov:zero:inv:en:imp -qarch:pwrx
Word Length	32 bits
Memory Management	Virtual
Size of Executable	Power: 29.7 MB Power2: 31.3 MB
Maximum Size of Open Core	Up to available swap space

Table C-9. System Description - NEC.

Items	Descriptions
Supported Machine Model(s)	SX-4
Models With Timing Constants Installed.	SX-4
Operating System(s)	SUPER-UX 7.2
Compiler Used	f77 Rev.134, f77sx Rev.175 patch 003-005
Compiler Options	-c -eb -daCDb -Nacet -hnoary -memlayout0 -NE -NF -float0 -f0 -Ng -Np -Npi -w -Cvopt -Wf,-O,nomsg,-i,-pvctl,nomsg,noassume,vwork=stack, -w,double16
Word Length	64 bits
Memory Management	Real
Size of Executable	96.5 MB
Maximum Size of Open Core	Lesser of physical memory or available swap space

Table C-10. System Description - Silicon Graphics R4X00, R5K.

Items	Descriptions
Supported Machine Model(s)	R4K, R5K
Models With Timing Constants Installed.	IP19, IP20, IP22
Operating System(s)	IRIX 5.2, IRIX 6.2
Compiler Used	f77 4.0.1
Compiler Options	-G 0 -static -mips2 -Olimit 2300 -N1210
Word Length	32 bits
Memory Management	Virtual
Size of Executable	29.6 MB
Maximum Size of Open Core	Up to available swap space

Table C-11. System Description - Silicon Graphics R8K, R10K.

Items	Descriptions
Supported Machine Model(s)	R8K, R10K
Models With Timing Constants Installed.	IP21
Operating System(s)	IRIX 6.2, 6.9
Compiler Used	f77 6.2
Compiler Options	-G 0 -O3 -TENV:X=1 -static -mips4 -64
Word Length	32 bits
Memory Management	Virtual
Size of Executable	33.1 MB
Maximum Size of Open Core	Lesser of 8GB or available swap space

Table C-12. System Description - Sun.

Items	Descriptions
Supported Machine Model(s)	SuperSPARC, UltraSPARC
Models With Timing Constants Installed.	SuperSPARC, UltraSPARC
Operating System(s)	SuperSPARC: Solaris 2.4, 2.6 UltraSPARC: Solaris 2.5, 2.6
Compiler Used	SuperSPARC f77 4.2 UltraSPARC f77 4.2
Compiler Options	SuperSPARC: -fast -dalign -libmil -Bstatic -Nc50 -Nq300 -O4 -w -xarch=v8 -xchip=super -xcache=16/32/1:1024/32/1 UltraSPARC: -fast -dalign -libmil -Bstatic -Nc50 -Nq300 -O4 -w -xarch=v8plus -xchip=ultra -xcache=16/32/1:512/64/1
Word Length	32 bits
Memory Management	Virtual
Size of Executable	SuperSPARC: 29.4 MB UltraSPARC: 30.7 MB
Maximum Size of Open Core	Up to available swap space

C.2 Numerical Data

**Table C-13. Numerical Data - 32-bit, big endian, IEEE.
(All but Cray, Digital, Hitachi, NEC, and Windows NT)**

Item	Description
INTEGER Bit Representation	0 1 31 S | Integer
REAL Bit Representation	0 1 8 9 31 S | Exponent | Mantissa
Exponent Range for a REAL Number	±38
Precision of a REAL Variable	6 digits (24 bits)
DOUBLE PRECISION Bit Representation	0 1 11 12 31 S | Exponent | Mantissa 32 63 Mantissa (cont.)
Exponent for a DOUBLE PRECISION Number	±308
Precision of a DOUBLE PRECISION Variable	15 digits (53 bits)

Table C-14. Numerical Data - 32-bit, little endian, IEEE. (Digital, Windows NT)

Item	Description
INTEGER Bit Representation	31 30 0 S | Integer
REAL Bit Representation	31 30 23 22 0 S | Exponent | Mantissa
Exponent Range for a REAL Number	±38
Precision of a REAL Variable	6 digits (24 bits)
DOUBLE PRECISION Bit Representation	63 62 52 51 32 S | Exponent | Mantissa 31 0 Mantissa (cont.)
Exponent for a DOUBLE PRECISION Number	±308
Precision of a DOUBLE PRECISION Variable	15 digits (53 bits)

Table C-15. Numerical Data - 64-bit, big endian, Cray.
(All Cray except IEEE T90)

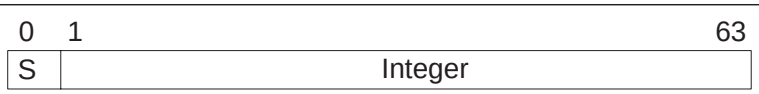
Item	Description
INTEGER Bit Representation	
REAL Bit Representation	
Exponent Range for a REAL Number	±2466
Precision of a REAL Variable	14 digits (48 bits)
DOUBLE PRECISION Bit Representation	
Exponent for a DOUBLE PRECISION Number	±2466
Precision of a DOUBLE PRECISION Variable	28 digits (96 bits)

Table C-16. Numerical Data - 64-bit, big endian, Cray, IEEE.

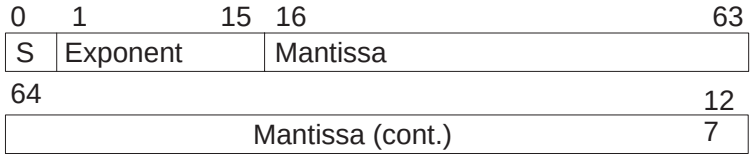
Item	Description
INTEGER Bit Representation	
REAL Bit Representation	
Exponent Range for a REAL Number	±308
Precision of a REAL Variable	15 digits (53 bits)
DOUBLE PRECISION Bit Representation	
Exponent for a DOUBLE PRECISION Number	±4932
Precision of a DOUBLE PRECISION Variable	33 digits (112 bits)

Table C-17. Numerical Data - 64-bit, big endian, NEC, IEEE.

Item	Description
INTEGER Bit Representation	
REAL Bit Representation	
Exponent Range for a REAL Number	±308
Precision of a REAL Variable	15 digits (53 bits)
DOUBLE PRECISION Bit Representation	
Exponent for a DOUBLE PRECISION Number	±308
Precision of a DOUBLE PRECISION Variable	31 digits (96 bits)

C.3 Computer Dependent Defaults

These tables list the computer-dependent default values for MSC/NASTRAN. The default rank values are listed in Table C-21.

Table C-18. Computer-Dependent Defaults, All But Cray and NEC.

Parameter	Input File Settings	Command Line Settings	Default	Comment
BUFFPOOL	`NASTRAN BUFFPOOL=`*n*	`bpool=`*n*	37	GINO Blocks
BUFFSIZE	`NASTRAN BUFFSIZE=`*n*	`buffsize=`*n*	2049	Max: 65537
BUFFSIZE Increment	`NASTRAN SYSTEM(136)=`*n*	`sys136=`*n*	128	Words
DBALL Size	`INIT DBALL LOGICAL= (DBALL(`*n*`))`	-	25 000	GINO Blocks
DBS Update Time	`NASTRAN SYSTEM(128)=`*n*	`sys128=`*n*	5	
Lanczos HPO	`NASTRAN SYS-TEM(193)=`*n*	`sys193=`*n*	0	Save
Lanczos HPO	`NASTRAN SYS-TEM(194)=`*n*	`sys194=`*n*	0	Pack/Unpack
SCRATCH Size	`INIT SCRATCH LOGICAL= (`*logname*`(`*n*`)), SCR300= (`*logname*`(`*n*`))`	-	175 000	GINO Blocks
SMEM	`INIT SCRATCH (MEM=`*n*`)`	`smem=`*n*	100	GINO Blocks

Table C-19. Computer-Dependent Defaults, Cray.

Parameter	Input File Settings	Command Line Settings	Default	Comment
BUFFPOOL	`NASTRAN BUFF-POOL=`*n*	`bpool=`*n*	27	GINO Blocks
BUFFSIZE	`NASTRAN BUFFSIZE=`*n*	`buffsize=`*n*	4097	Max: 65537
BUFFSIZE Increment	`NASTRAN SYSTEM(136)=`*n*	`sys136=`*n*	512	Words
DBALL Size	`INIT DBALL LOGICAL= (DBALL(`*n*`))`	-	1 000 000	GINO Blocks
DBS Update Time	`NASTRAN SYSTEM(128)=`*n*	`sys128=`*n*	1	
Lanczos HPO	`NASTRAN SYS-TEM(193)=`*n*	`sys193=`*n*	1	Recompute
Lanczos HPO	`NASTRAN SYS-TEM(194)=`*n*	`sys194=`*n*	1	Read/Write
SCRATCH Size	`INIT SCRATCH LOGICAL= (`*logname*`(`*n*`)), SCR300= (`*logname*`(`*n*`))`	-	1 000 000	GINO Blocks
SMEM	`INIT SCRATCH (MEM=`*n*`)`	`smem=`*n*	0	GINO Blocks

Table C-20. Computer-Dependent Defaults, NEC.

Parameter	Input File Settings	Command Line Settings	Default	Comment
BUFFPOOL	`NASTRAN BUFFPOOL=`*n*	`bpool=`*n*	27	GINO Blocks
BUFFSIZE	`NASTRAN BUFFSIZE=`*n*	`buffsize=`*n*	4097	Max: 65537
BUFFSIZE Increment	`NASTRAN SYSTEM(136)=`*n*	`sys136=`*n*	512	Words
DBALL Size	`INIT DBALL LOGICAL=(DBALL(`*n*`))`	-	1 000 000	GINO Blocks
DBS Update Time	`NASTRAN SYSTEM(128)=`*n*	`sys128=`*n*	1	
Lanczos HPO	`NASTRAN SYSTEM(193)=`*n*	`sys193=`*n*	1	Recompute
Lanczos HPO	`NASTRAN SYSTEM(194)=`*n*	`sys194=`*n*	1	Read/Write
SCRATCH Size	`INIT SCRATCH LOGICAL=(`*logname*`(`*n*`)), SCR300=(`*logname*`(`*n*`))`	-	1 000 000	GINO Blocks
SMEM	`INIT SCRATCH (MEM=`*n*`)`	`smem=`*n*	0	GINO Blocks

Table C-21. Computer-Dependent Default Rank Values.

Computer Type	Model	SYS198	SYS205
Cray	All	8	2
Digital	All	8	8
Fujitsu	All	32	32
HP 9000	All	36	36
HP Exemplar	All	36	36
IBM	All	16	16
NEC	All	16	16
Silicon Graphics R4K, R5K	All	16	16
Silicon Graphics R8K, R10K	R8K R10K	24 64	24 64
Sun	All	32	32

APPENDIX D

PRODUCT TIMING DATA

If User Warning Message 6080 is printed in the .F06 file, please fill out this form and mail it along with the GENTIM2.F04, .F06, .LOG, and .PCH files (see Section 3.10) on tape or cartridge to MSC at the address below.

Client: _____

Site: _____

Computer: _____

Model: _____

Submodel: _____

Operating System: _____

Operating Level: _____

Thank you.

Client Support
The MacNeal-Schwendler Corp.
815 Colorado Blvd.
Los Angeles, CA 90041

ERROR REPORT OR COMMENTS AND SUGGESTIONS

MSC/NASTRAN Configuration and Operations Guide

Version 70.5 UNIX Edition

Page: _____

Please describe error or suggestion:

Thanks for your feedback. Please FAX or mail to:

MSC Documentation Department
The MacNeal-Schwendler Corporation
815 Colorado Blvd.
Los Angeles, CA 90041
FAX: (213) 259-3838

Client Support
The MacNeal-Schwendler Corporation
4300 W. Brown Deer Rd.
Milwaukee, WI 53223
FAX: (414) 357-0347