Gould MPX-32™

Release 3.3

Reference Manual

Volume IV

Unsupported Software

December 1986

Publication Order Number: 323-001554-300

GOULD
Electronics

HISTORY

The MPX-32 Software Release Notes, Publication Order Number **324-004400-100,** (Release 3.2C) was printed December, 1985.

The attachments were removed from the MPX-32 Software Release Notes and now form Volume IV of the MPX-32 Release 3.3 Reference Manual.

The MPX-32 Release 3.3 Reference Manual, Volume IV, Publication Order Number **323-001554-300,** was printed December, 1986.

This manual contains the following pages:

Title page
Copyright page
iii/iv through ix/x
1-1 through 1-15/1-16
2-1 through 2-32

CONTENTS

Documentation Conventions

Notation conventions used in directive syntax and message examples throughout this manual are described below.

lowercase letters

In directive syntax, lowercase letters identify a generic element that must be replaced with a value. For example,

```
!ACTIVATE taskname
```

means replace taskname with the name of a task, e.g.,

```
!ACTIVATE DOCCONV
```

In messages, lowercase letters identify a variable element. For example,

```
**BREAK** ON:taskname
```

means a break occurred on the specified task.

UPPERCASE LETTERS

In directive syntax, uppercase letters specify a keyword must be entered as shown for input, and will be printed as shown in output. For example,

```
SAVE filename
```

means enter SAVE followed by a filename, e.g.,

```
SAVE DOCCONV
```

In messages, uppercase letters specify status or information. For example,

```
taskname,taskno ABORTED

*YOUR TASK IS IN HOLD. ENTER CONTINUE TO RESUME IT
```

Braces { }

Elements placed one under the other inside braces specify a required choice. You must enter one of the arguments from the specified group. For example,

$$\left\{\begin{array}{l}\text{counter}\\ \text{startbyte}\end{array}\right\}$$

means enter the value for either counter or startbyte.

Brackets [ ]

An element inside brackets is optional. For example,

```
[CURR]
```

means the term CURR is optional.

Items placed one under the other within brackets specify you may optionally enter one of the group of options or none at all. For example,

$$\begin{bmatrix}\text{base name}\\ \text{progname}\end{bmatrix}$$

means enter the base name or the program name or neither.

Items in brackets within encompassing brackets specify one item is required only when the other item is used. For example,

```
TRACE [lower address [upper address] ]
```

means both the lower address and the upper address are optional, and the lower address may be used alone. However, if the upper address is used, the lower address must also be used.

Commas between multiple brackets within an encompassing set of brackets are semi-optional; that is, they are not required unless subsequent elements are selected. For example,

```
M.DFCB fcb,lfc [, [a] , [ b ] , [ c ] , [ d ] , [ e ] ]
```

could be coded as

```
M.DFCB FCB12,IN

    or

M.DFCB FCB12,IN,,ERRAD

    or

M.DFCB FCB13,OUT,,ERAD,,PCK
```

Horizontal Ellipsis . . .

The horizontal ellipsis indicates the previous element may be repeated. For example,

```
name [,...,name]
```

means you may enter one or more name values separated by commas.

Vertical Ellipsis .
.
.

The vertical ellipsis specifies directives, parameters, or instructions have been omitted. For example,

```
COLLECT 1
  .
  .
  .
LIST
```

means one or more directives have been omitted between the COLLECT and LIST directives.

Numbers and Special Characters

In a syntax statement, any number, symbol, or special character must be entered as shown. For example,

```
(value)
```

means enter the proper value enclosed in parentheses; e.g., (234).

Underscore

In syntax statements, underscoring specifies the letters, numbers or characters that may be typed by the user as an abbreviation. For example,

```
ACTIVATE  taskname
```

means spell out the directive verb ACTIVATE or abbreviate it to ACTI.

```
RESET
```

means type either RESET or RST.

In examples, all terminal input is underscored; terminal output is not. For example,

```
TSM > EDIT
```

means TSM > was written to the terminal; EDIT is typed by the user.

Subscript Delta $_\Delta$

A subscript delta specifies a required space. For example,

EDT > STO$_\Delta$TSSPGM

means a space is required between O and T.

SECTION I

MPX-32 DEMONSTRATION PACKAGE

1.1 Introduction

The MPX-32 demonstration package illustrates the major features of MPX-32 and establishes the baseline viability of the computer system MPX-32 is running on. The production and operational facilities of MPX-32 are emphasized, as opposed to its program development facilities. The major features exercised in the demonstration are:

Priority CPU scheduling

Swapping

I/O facilities

- Blocked/unblocked I/O
- Wait/no-wait I/O
- I/O to regular and extended memory
- Short and long transfers
- Multiple simultaneous I/O from a single task
- Various I/O functions - read, write, backspace, advance, rewind, write EOF
- Various devices - disc, magnetic tape, terminals, line printer, floppy disc

Shared memory (between tasks)

Message sending/receiving

Software interrupt system (abort receivers, break receivers, etc.)

File locks

Timed scheduling and suspensions

Task activation

Many of the features used in this demonstration are not externally observable, such as message sending between tasks. At the completion of the demonstration, a listing is available which includes a customized banner page, the line printer test messages, and a log of the demonstration activity.

1.2 Demonstration Structure

The MPX-32 demonstration utilizes the test executive called MPXDEMO to run several test tasks according to a predefined script. A complete description of MPXDEMO and its capabilities are described in the MPXDEMO Test Executive section.

The test tasks include IBNNR, IOMUL, IOTST, ITERM, and MIXER. IBNNR produces a banner page for the line printer output of the test session. IOMUL performs multiple no-wait I/O, permanent file creation and allocation, and file locking. IOTST performs virtually all combinations of I/O to a device or file including wait/no-wait, blocked/unblocked, regular memory/extended memory, and variable transfer length I/O. ITERM displays a variety of patterns and banners on a CRT. MIXER provides general system load. Each of these test tasks are described in more detail in Section 1.13.

1.3 Hardware Considerations

The MPX-32 demonstration can be run on any MPX-32 configuration that contains the following minimum hardware:

. CPU

. 128K Memory

. Disc

. Floppy disc or magnetic tape

. Operator's console

. Line printer

The following considerations apply to running the demonstration on systems with additional hardware:

. Additional hardware cannot be used in the demonstration unless it is SYSGENed in.

. An IPU makes the demonstration run slightly faster; however, the accounting information produced in the log is inaccurate and misleading. The demonstration can be altered to illustrate the presence of the IPU more dramatically by adjusting the parameters to MIXER so that MIXER is heavily compute bound.

. A scientific accelerator does not affect the performance of the demonstration beyond the performance increase due to the presence of the high speed floating point unit. However, the compute loop in MIXER is comprised primarily of calls to subroutines accelerated by the scientific accelerator; therefore, if the source or object of MIXER is available, recataloging MIXER with the scientific accelerator library would make it possible to demonstrate its effectiveness.

- Additional memory makes the demonstration run faster, primarily by reducing swapping.

- It is strongly recommended that ITERM should not be used with a hard copy terminal; the demonstration would probably take hours to complete.

- A copy of IOTST should be run with each terminal, disc drive, magnetic tape drive, floppy disc drive, and line printer configured in the system.

- A copy of ITERM should be run for each CRT terminal configured in the system.

- A copy of IBNNR should be run for each line printer configured in the system.

- The timings produced by the demonstration rely on both the real-time clock and the interval timer. These timers must be properly jumpered and SYSGENed.

- The IOTST test has a map block granularity dependency. See the IOTST documentation.

- It is possible for the demonstration to run out of disc space if the available disc space is limited and the temporary disc file assignments are large. If the demonstration seems to cease activity (swap device inactive) and the log is not produced, enter OPCOM and examine the execution queue for tasks in the SWDC state. If the five MPX-32 demonstration test tasks are all in the SWDC state, reboot the system and create additional disc space or reduce the demonstration's requirements.

1.4 Software Considerations

The MPX-32 demonstration can be run on a minimum MPX-32 configuration, such as a starter system. Specific requirements are as follows:

- All devices used in the demonstration must be SYSGENed in.

- There must be approximately 80K of memory available for use by the demonstration. If the demonstration is run with less memory, excessive swapping occurs.

- The demonstration requires an 8K global common partition named GLOBAL80. This partition may be either static or dynamic. Explicit directions on setting up a dynamic partition are given in Section 1.7. If a static partition is desired, see the MPX-32 Reference Manual Volume II.

- Two assumptions are made concerning the real-time clock SYSGEN parameters. First, the MTIM parameter is assumed to be 60. If MTIM is not 60, the fractions of seconds in the times produced in the log are incorrect. Second, the NTIM parameter is assumed to be the same as the MTIM. If NTIM and MTIM differ, the time unit is not one second. The values entered in the /TIMEOUT directive and in the STAGGER and CYCLE fields of the task scheduling directive are actually time units, not seconds.

- It is possible to exhaust the available dispatch queue entries. If during the demonstration MPXDEMO reports it is unable to schedule a test task because of error five, increase the number of dispatch queue entries using the SYSGEN DISP directive or decrease the demonstration size.

. Devices other than disc should not be SYSGENed with SHR (shared) specified in the DEVICE directive. If other devices are shared, improper displays appear on line printers and terminals and can cause tests of floppy disc and magnetic tape to fail.

. The NOCMS SYSGEN directive cannot be used in conjunction with the MPXDEMO software. MPXDEMO aborts with an SV02.

1.5 MPX-32 Demonstration Program (MPXDEMO)

The MPX-32 demonstration program, MPXDEMO, is included with other optional files on the System Distribution Tape (SDT). The load modules required for executing MPXDEMO are in the SYSTEM directory and the source files are in the DEMO directory prefixed by the characters US (Unsupported Source). The load modules required for executing MPXDEMO are as follows:

MPXDEMO	The MPX-32 demonstration executive
IOTST IOMUL ITERM IBNNR MIXER	Five test tasks
BACKROUN	The background task
TESTSCR3	Minimum CONCEPT/32 system test script
TESTSCR4	Typical large CONCEPT/32 system test script
MPXDTSM	TSM macro for running the MPX-32 demonstration
MPXDBAT	Batch stream for running the MPX-32 demonstration

1.6 MPXDEMO Test Executive

MPXDEMO is an MPX-32 test executive which performs the following functions:

. Provides a mechanism for scheduling test tasks for execution at various time intervals

. Changes the operating parameters for a test task, such as logical file code assignments

. Sends a message to the test task

. Monitors the progress of a test task

. Aborts a test task that continues past a specified time-out value

. Collects the results of a test task execution, including a pass/fail indication, execution times, and a message (if any)

. Formats the results of a test task execution into a test session log

Only tasks specially coded for execution with MPXDEMO may be scheduled.

MPXDEMO has two primary functions: scheduling test tasks and monitoring test execution and results. Task scheduling capabilities include:

- Multiple activations
- Message passing to test task
- Alteration of user name, owner name, and priorities
- I/O file and device allocation

Monitoring capabilities include:

- Timing facilities
- Event recording
- Breaking and aborting test tasks that do not respond before the time-out value expires
- Test result collection
- Test session log

1.7 GLOBAL80

MPXDEMO requires a static or dynamic global memory partition. The partition must be 8K words and be mapped into the end of the regular memory address space. To create an appropriate dynamic global memory partition, enter the Volume Manager and use the directive:

```
CREATE C GLOBAL80 16 240 BRIE=N
```

To create an appropriate static global memory partition, see the MPX-32 Reference Manual Volume II.

1.8 MPXDEMO Directives

The MPXDEMO directives are task and group. Directives consist of up to four input lines, each line containing up to 80 characters. The directive continuation character is a hyphen (-) appearing at the end of the line to be continued. A directive cannot be continued from within a parenthetical value, such as a message. For example:

```
TASKNAME=TEST,MESSAGE=(ABC-
DEF)
```

is not allowed.

Any characters following a line continuation hyphen are ignored. Directives may not contain embedded blanks except in messages. For example:

```
TASKNAME=TEST, MESSAGE=(AB)
```

is not allowed, but

```
TASKNAME=TEST,MESSAGE=(A   B)
```

is allowed.

1.8.1 Task Directives

A task directive causes a test task to be scheduled for activation one or more times during the test session.

A task directive has the form:

```
keyfield [,keyfield] . . .
```

which indicates one or more keyfields where keyfield has the form:

```
keyword=value
```

The keywords, their meaning, valid values, and special considerations are listed below.

```
TASKNAME=taskname
```

taskname is a one- to eight-character test task name. taskname is the only required keyfield in a task directive. The test task name can be specified without TASKNAME= if it appears as the first keyfield in the test directive.

```
STAGGER=seconds
```

seconds is a one- to four-digit decimal integer specifying the number of seconds before the test task is scheduled for the first activation. If not specified, the stagger is zero.

```
REPEAT=repetitions
```

repetitions is a one- to two-digit decimal integer specifying the number of times to schedule the test task for activation after the first time. If not specified, the repeat count is zero.

```
CYCLE=seconds
```

seconds is a one- to four-digit decimal integer specifying the number of seconds between the repeat activations. If not specified and the repeat count is greater than zero, an error occurs.

```
MESSAGE=(message)
```

message is a 1- to 72-character message to be sent to the test task when the test task is activated. Interpretation of the message is the responsibility of the test task.

```
PRIORITY=priority
```

priority is a one- to two-digit decimal integer from 1 to 64 that specifies the execution priority of the test task.

```
OWNER=ownername
```

ownername is a one- to eight-character owner name to be associated with the test task.

```
USERNAME=username
```

username is a one- to eight-character user name to be associated with the test task during execution. This feature is sometimes important when using the assignment keyfield.

```
OPTION=(option [,option] . . .)
```

option is a one- to two-character decimal integer from 1 to 32 specifying the options to be set during test task execution. Interpretation of the options is the responsibility of the test task.

```
ASSIGNn=(lfc=info)
```

n is a number from one to four, lfc is a one- to three-character logical file code, and info is identical to the corresponding MPX-32 assignments. For example,

```
$ASSIGN3  OUT=MT10,SAVE,,U
```

is a typical MPX-32 assignment. The corresponding task directive assignment is:

```
. . . ,ASSIGN3=(OUT=MT10,SAVE,,U),. . .
```

A task directive has up to five logical file code (LFC) assignments. Use of the LFCs assigned is the responsibility of the test task.

1.8.2 Group Directives

A group directive affects the operation of a group of directives corresponding to a test session. All group directives start with a forward slash (/) in the first character position. The group directives are listed below.

/END	terminates a directive group. An end-of-file terminates the final directive group.
/TIMEOUT=seconds	seconds is a one- to five-digit decimal integer that specifies the amount of time to wait after activating the last task in the test session for all test tasks to respond. If this value expires, MPXDEMO aborts all outstanding test tasks.
/BACKGROUND	indicates that the background task is to be run during the test session. The background task can be used to measure system performance.

Sample Directive Stream

Session	Command Input
1	Task directives /TIMEOUT=seconds /BACKGROUND /END
2	Task directives /TIMEOUT=seconds /BACKGROUND /END
. . . .	. . . .

Command Summary

Task Directives	Group Directives
TASKNAME = taskname	/TIMEOUT = seconds
STAGGER = seconds	/BACKGROUND
REPEAT = repetitions	/END
CYCLE = seconds	
MESSAGE = (message)	
PRIORITY = priority	
OWNER = ownername	
USER = username	
OPTION = (option [,option] . . .)	
ASSIGNn = (lfc=info)	

1.9 MPXDEMO Special Considerations

MPXDEMO documentation assumes the time unit (as specified during SYSGEN) is one second. If it is not, then wherever the documentation reads seconds, it should be read as time units.

The Text Editor and TSM put characters into columns 73 through 80 of a line. MPXDEMO may be unable to interpret the directives if this occurs.

1.10 MPXDEMO Operation

MPXDEMO schedules test tasks and collects and prints the results. The following file codes are used for input and output:

Unit	Use	Default Assignment	I/O Unit Default Assignment
5	Directive input to MPXDEMO	SYC	ASSIGN2 5=SYC
6	Directive echo and diagnostics	SLO,100	ASSIGN2 6=SLO,100
7	Log of test session	SLO,1000	ASSIGN2 7=SLO,1000

The demonstration begins after MPXDEMO is activated and given a set of directives. MPXDEMO is a standard MPX-32 task, and can be activated in a variety of ways including interactively, batch, with a command file, or with the OPCOM ACTIVATE directive. The demonstration package provides two methods of activation using the files MPXDBAT and MPXDTSM. The directive input is from TESTSCR3 or TESTSCR4.

MPXDEMO can read its directives from a variety of sources including disc files, cards, or terminals.

MPXDEMO accepts directives specifying which test tasks to run, when and how often to run them, what messages to send to them, and which resources, priority, owner name, and user name to associate with them. Additional directive lines set the time-out value, determine whether the background task BACKROUN is to be run, and terminate the directive input. The time-out value determines how long MPXDEMO lets the test session run before terminating it by aborting the outstanding test tasks. The background task uses the available CPU time and reports how much CPU time it used in the test session log.

MPXDEMO operates in response to a directive stream. The directive stream is divided into groups of directives, with each directive group describing a test session. The /END directive, or end of file on the directive input unit, terminates a directive group.

Each directive group is used to make entries in a test task timing table, store a message in a message table, and build a parameter task activation (M.PTSK) block. The timing table has a fixed time increment of 350 seconds and a fixed time span of 4200 seconds. Therefore, activation requests that do not fall on an exact timing table increment are rounded to the nearest one, and activation requests beyond the timing table span are ignored. When all directives in a group are processed, the test session begins. At each time increment, starting at zero, MPXDEMO activates the test tasks scheduled for that time, and then suspends itself until it reaches the next timing table increment containing at least one test task activation.

As test tasks complete, test results are written in GLOBAL80, a static or dynamic global partition. Once MPXDEMO has activated all test tasks in a test session, it waits for their results. If any test tasks have not responded when the time-out value expires, MPXDEMO removes them from the system by sending a break interrupt, and then kills them.

If all test tasks are terminated, MPXDEMO formats and prints the information contained in GLOBAL80 as a test session log, and then processes the next available directive group. One additional capability of MPXDEMO is to activate a background task for the duration of a test session that utilizes all unused CPU time and report the amount of CPU time used in the log. The background task interferes with the test session if any of the test tasks are run at time-distribution priority levels.

Each input file provides a script of directives for MPXDEMO. These scripts can be user customized by editing the MESSAGE keyfield contained in the scripts. Directive usage is described in Section 1.8.

In order to run the MPX-32 demonstration program, perform the following steps:

1. Install the MPX-32 demonstration package.

2. Create the dynamic Global Common definition, GLOBAL80, by using the following Volume Manager directive:

   ```
   VOL>CREATE C GLOBAL80 16 240 BRIE=N
   ```

3. Mount scratch tapes with write rings inserted on all configured tape drives and mount scratch floppy discs in all configured floppy disc drives. Depending on the test script, some of the tape drives and floppy disc drives may not be used.

4. To run the demonstration in the interactive mode, enter TSM and respond:

   ```
   TSM>MPXDTSM testscript
   ```

   testscript is TESTSCR3 or TESTSCR4. See Section 1.5. When running the MPX-32 demonstration in this manner, the originating terminal is unavailable to the test tasks.

5. To run the demonstration in the batch mode, perform the following steps:

   a) Enter the Text Editor and transfer the contents of MPXDBAT into a workfile.

   b) Change the line that reads $ASSIGN1 5=TESTSCR1 to $ASSIGN1 5=testscript where testscript is TESTSCR3 or TESTSCR4. See Section 1.5.

   c) Store the file as MPXDBAT with the editor command:

      ```
      EDT>STORE MPXDBAT SYS UNN
      ```

   d) Enter MPXDBAT into the batch stream with:

      ```
      TSM>!BATCH @SYSTEM(SYSTEM)MPXDBAT
                         or
      ??BATCH @SYSTEM(SYSTEM)MPXDBAT
      ```

   e) Log off the terminal so it can be used by the test tasks, if necessary.

The demonstration proceeds. Lines print on the line printers. Various patterns and banners are displayed on the terminals. Tape drives, floppy disc drives and disc drives advance, backspace, and rewind. The demonstration terminates after producing a log of the test task activity on the line printer

The duration of the demonstration varies widely depending on configuration and demonstration complexity. A minimum test script (TESTSCR3) run on a minimum configuration takes approximately 30 minutes.

1.11 Monitoring the MPX-32 Demonstration

During the demonstration there is very little evidence of activity beyond the physical movements of the devices. Using a parallel panel, it is possible to display indications of the CPU's activity by continuously displaying locations in the communications region (such as the memory request queue head cell). The communications region variables are documented in the MPX-32 Technical Manual. The contents of the various state chain head cells are documented in the MPX-32 Reference Manual Volume I.

1.12 Error Conditions

MPXDEMO provides error diagnostic facilities for three error groups: directive, activation, and overflow.

Directive errors occur when the syntax is incorrect. If MPXDEMO is unable to decode a directive, the directive is displayed with continuation characters and the remaining characters removed. The following message is displayed:

```
PROBLEM IN DIRECTIVE:
```

Any directive containing an error is not processed.

Activation errors occur when MPXDEMO activates the tasks as requested using the parameter task activation service, M.PTSK. If an error is detected, the following message is displayed:

```
**** ERROR **** TASK taskname
COULD NOT BE RUN BECAUSE n
```

taskname is the name of the task and n is the error code returned from M.PTSK. These error codes are documented in the MPX-32 Reference Manual Volume I in the section describing M.PTSK.

Overflow errors occur because the communication region, GLOBAL80, has a limited capacity. The total count of the test task activations must not exceed 60. Each activation attempt after the limit is reached causes the following message to be displayed:

```
**** ERROR **** TASK taskname
NOT RUN DUE TO EXCESSIVE TASKS SCHEDULED
```

taskname is the name of the task.

If MPXDEMO encounters any other error conditions, descriptive messages are produced. The test tasks can also encounter error conditions. The log produced by MPXDEMO includes a field which indicates whether the test task passed or failed. If the test task failed, a variety of other messages can appear. Some possibilities are as follows:

. If an error condition such as an I/O or a system service error is detected, some tasks display a message that might include an event code. Beyond retrying the demonstration, there is little that can be done by the user to correct the problem. The message and the event code are intended solely as debugging aids for Development.

. If the test task receives a break interrupt, the following message is displayed:

  ```
  BREAK RECEIVED EVENT CODE n
  ```

  n is the event code. In most cases, this condition occurs when MPXDEMO times out and purges the test tasks from the system. Check the log to see if a time out occurred. If MPXDEMO is timing out, either the /TIMEOUT value needs to be increased or test tasks are hung up and corrective action requires further investigation. If MPXDEMO has not timed out and the break received message is displayed, it indicates a serious problem that requires further investigation.

- If the test task aborts, the following message is displayed:

  ```
  ABORT CODE abortcode extended abort code EVENT CODE n
  ```

  abortcode is a four-character abort code. The extended abort code is an abort message. n is the event code. The following may be helpful in analyzing abort codes:

  - RTxx abort messages may result when an incorrect message is passed to the test task.
  - ALxx abort messages may result when logical file code assignments are incorrect.
  - TIMEOUT and BREAK INTRPT abort messages may result when MPXDEMO times out and sends a break to the test task which does not respond correctly.

See the MPX-32 Reference Manual Appendix C for further descriptions on abort codes.

1.13 MPX-32 Demonstration Test Tasks

1.13.1 IBNNR

IBNNR produces a banner page for the line printer output of the test session. The form of the banner page is:

```
*****************************************************************
*                                                               *
*                                                               *
*                            MPX-32                             *
*                                                               *
*                         DEMONSTRATION                         *
*                                                               *
*                            RUN FOR                            *
*                                                               *
*                         customer name                         *
*                                                               *
*                              OF                               *
*                                                               *
*                            company                            *
*                                                               *
*****************************************************************
```

The message keyfield specified in an MPXDEMO task directive to be sent to IBNNR has the form:

```
MESSAGE = (customer name/company)
```

customer name	is an alphanumeric string that can include blanks and most special characters. However, it may not include a forward slash (/).
company	is an alphanumeric string that may include blanks and most special characters

Both customer name and company are restricted to approximately twenty characters.

I/O units are as follows:

Unit	Use	Default Assignment	Suggested Assignment
DEV	Banner page is written to this unit	None	ASSIGN3 DEV=LP

1.13.2 IOMUL

IOMUL tests multiple no-wait I/O operations from a single task, synchronization file locks, and the creation and access of permanent files.

The message keyfield specified in an MPXDEMO task directive to be sent to IOMUL has the form:

MESSAGE = (filename)

filename	is an arbitrary eight-character name for a disc file to be created (if necessary) and used. If synchronization file locks are to be fully tested, two copies of IOMUL should be activated simultaneously with the same file name specified.

I/O units are as follows:

Unit	Use	Default Assignment	Suggested Assignment
DV1	30 records written to unit	None	ASSIGN3 DV1=DC,100
DV2	30 records written to unit	None	ASSIGN3 DV2=DC,100
DV3	30 records written to unit	None	ASSIGN3 DV3=DC,100
DV4	30 records written to unit	None	ASSIGN3 DV4=LP
DV5	30 records written to unit	None	ASSIGN3 DV5=TY

1.13.3 IOTST

IOTST performs virtually all combinations of I/O to a device or file, including wait/no-wait, blocked/unblocked, regular memory/extended memory, and variable transfer length I/O.

The message keyfield specified in an MPXDEMO task directive to be sent to IOTST has the form:

MESSAGE = (io,files,records,incr,maxrecordsize,mapblocksize)

io	is a one-digit decimal integer indicating whether input, output, or both is to be performed. Values are: 0 - input and output 1 - input only 2 - output only
files	is a four-digit decimal integer indicating the number of files to read and write. A value of zero means the default is three.
records	is a four-digit decimal integer indicating the number of records per file. A value of zero means the default is two.
incr	is a four-digit decimal integer indicating the record size increment. A value of zero means the default is 100.
maxrecordsize	is a five-digit decimal integer indicating the maximum record size. A value of zero means the default is 310.
mapblocksize	is a four-digit decimal integer indicating the map block size in words. The value 2048 must be used for a CONCEPT/32 computer.

I/O units are as follows:

Unit	Use	Default Assignment	Suggested Assignment
DEV	The I/O device	None	All configured devices

1.13.4 ITERM

ITERM displays a banner and a variety of patterns on a CRT.

The message keyfield specified in an MPXDEMO task directive to be sent to ITERM has the form:

MESSAGE = (type,banner)

type	indicates the terminal type. A Hazeltine is indicated by an H and an ADM-3 is indicated by an A. Other terminals are not supported.
banner	specifies the words formatted as a banner. Each word can contain alphanumeric characters and most special characters. Blanks are used to separate words. Each word consists of up to twelve characters and the banner consists of up to three words.

I/O units are as follows:

Unit	Use	Default Assignment	Suggested Assignment
DEV	The CRT device used to display banner and patterns	None	ASSIGN3 DEV=TY or ASSIGN3 DEV=CA

1.13.5 MIXER

MIXER provides general system load by emulating the execution behavior of a typical application. MIXER reads, computes, and writes in a loop.

The message keyfield specified in an MPXDEMO task directive to be sent to MIXER has the form:

MESSAGE = (cycles, cpuseconds, recordsize, recordcount)

cycles	is a five-digit decimal integer specifying the number of times to repeat the loop
cpuseconds	is a five-digit decimal integer specifying the number of seconds to compute during each loop
recordsize	is a five-digit decimal integer specifying the size in words of each record read or written
recordcount	is a five-digit decimal integer specifying the number of records to read or write in each loop

I/O units are as follows:

Unit	Use	Default Assignment	Suggested Assignment
SIN	Read at the beginning of each loop	None	ASSIGN3 SIN=DC,200,
SOT	Written at the end of each loop	None	ASSIGN3 SOT=DC,200,U

Note: The files for SIN or SOT must be large enough to contain the records specified by recordsize and recordcount.

SECTION II

UNSUPPORTED SOFTWARE

The following unsupported software is supplied in this release of MPX-32:

.	ANALYZE	Crash Dump Analyzer
.	AUTOJOB	Automatic Batch Job Submission on Boot-up
.	DD	Disc Dump by Sector
.	DDUMP	Disc Dump by File
.	DOWNDATE	Source Compare Program
.	DOWNDAT2	Source Compare Program/Source Update Output
.	GM.FLOW	Assembly Source Code Flowchart Tool
.	GM.SEAR	Source Search Tool
.	LMINFO	Load Module Information
.	LOGMDT	Log Contents of Rapid File Allocation MDT
.	LOGOFF	Remote Terminal Logoff
.	MIPSMON	Instruction Sequence Timing Tool
.	NEWCOPY	Floppy Disc Duplication Tool
.	POOLSCAN	Memory Pool Monitor
.	PORTPROT	Dial-up Port Protection
.	SPRINT	Serial Printer Formatter/Spooler
.	TERMLOAD	Terminal Initializer/Loader

To access an individual tool, enter the keyword specified in capital letters at the TSM prompt. For example,

```
TSM>DD
```

enters the program Disc Dump by Sector.

2.1 ANALYZE - Crash Dump Analyzer

The MPX-32 crash dump analyzer, ANALYZE, is an interactive task which allows the contents of system tables and queues to be examined after a system crash occurs. It is designed to run after MPX-32 is rebooted.

Initially, ANALYZE must be supplied with a tape or floppy disc produced by H.DMPMT. The H.DMPMT data is read and stored on a named disc file. The disc file is used in subsequent activations for analyzing the crash. The display from ANALYZE is directed to either the SLO device or the terminal; the default is to the terminal.

ANALYZE first prompts for the pathname of the disc file. Then it prompts for tape, floppy, or disc. If the reply is disc, it uses the named disc file which was previously produced by ANALYZE.

If the reply is tape or floppy, it uses the appropriate device. If a disc file exists and a tape or floppy is being used, ANALYZE prompts to delete the disc file (reply Y or N). If the reply is Y or if a disc file does not exist, ANALYZE writes to the disc file. ANALYZE creates the file dynamically. If the reply is N, ANALYZE exits.

Numerous changes were made for MPX-32 Release 3.3. Therefore, system tables and queues sometimes change for each release. ANALYZE checks the revision number of the system where the dump was produced. If the revision number is not 3.3, it displays a continue option prompt (reply Y or N).

Note: All numbers displayed and accepted by ANALYZE are hexadecimal except error codes.

Following is a list of ANALYZE directives and descriptions:

Directive	Description
CDT	Display controller definition tables
COMM	Display communications region variables and/or dispatch queue entries
DQE	Display dispatch queue entries
DUMP	Display contents of memory or a task
EXIT	Exit ANALYZE
HELP	Display useful information
INDEX	Display addresses of data structures
POOL	Display memory pool
PRINT	Divert most output to SLO device
SEARCH	Search for occurrences of a value
SET SAME	Set or reset ANALYZE logic
SMT	Display shared memory table entries
TSA	Display task service areas
UDT	Display unit definition tables
VIEW	Divert all output to user terminal

2.1.1 CDT Directive

Syntax:

```
CDT [IOQ] [devmnc] [,devmnc]...
          [index ] [,index ]
```

IOQ	causes an I/O queue summary to be displayed for each CDT listed
devmnc	is a device type mnemonic, such as TY
index	is the CDT index

The CDT directive displays a text description of the CDTs. If the devmnc and index parameters are not specified, all CDTs are displayed.

The following information is contained in the display:

- Physical address of this CDT
- CDT index (CDT.INDX)
- UDT index for each device on the controller
- Class (CDT.CLAS)
- Device type (CDT.DTC)
- Mnemonic, channel, and subaddress (CDT.CHAN, CDT.SUBA)
- Interrupt priority level (CDT.IPL)
- Number of units on controller (CDT.NUOC)
- Number of outstanding requests (CDT.IORO)
- Flags (CDT.FLGS, CDT.FLG2)
- I/O status (CDT.IOST)
- Interrupt handler address (CDT.SIHA)

The IOQ summary consists of the number of entries in the IOQ list (CDT.IOCT) and the following information. When the following information is displayed, the previous I/O queue (BIOQ) is listed first, followed by the current I/O queue (FIOQ) and any other I/O queues linked in the UDT entry's I/O queue list:

- Indices of the associated CDT and UDT
- Status and flags (IOQ.STAT, IOQ.FLGS)
- FCB or TCPB address (IOQ.FCBA)
- Program number (IOQ.PRGN)

- Handler function words (IOQ.FCT1, IOQ.FCT2, IOQ.FCT3, IOQ.FCT4)
- Number of bytes transferred (IOQ.UTRN)
- If the device is nonextended I/O, number of words in the OS buffer (IOQ.WOSB)
- Buffer addresses (IOQ.FBUF, IOQ.TBUF)
- I/O return status (IOQ.IOST, IOQ.IST1, IOQ.IST2)
- FCB control information (IOQ.CONT)

2.1.2 COMM Directive

Syntax:

```
     COMM [qmnc] [,qmnc] ...
```

qmnc   is a queue mnemonic, such as CURR, SWGQ, etc.

The COMM directive displays a text description of the communications region and all active DQEs. See DQE directive. If parameters are specified, the COMM directive displays a text description of all DQEs in those states.

The following is the default communication region information displayed:

- System name (C.SYSTEM)
- Revision number (C.REV)
- Patch level (C.UPDT)
- Status and system configuration flags (C.BIT)
- Count of outstanding interrupts and traps (C.GINT)
- Count of memory release events (C.RRUN)
- Configuration flags (C.CONF)
- Counts of memory modules available (C.TMAC, C.EMAC, C.HMAC, C.SMAC)
- Counts of memory modules configured (C.TMCC, C.EMCC, C.HMCC, C.SMCC)
- Address of memory pool (C.SBUF)
- Number of words in memory pool for each queue (FREE and USED):
    - Head cell address
    - Number of entries in the queue
    - Maximum cell size

. Minimum cell size

. Total number of words in the queue

2.1.3 DQE Directive

Syntax:

$$\text{DQE} \begin{bmatrix} \text{lmn} \\ \text{index} \end{bmatrix} \begin{bmatrix} \text{,lmn} \\ \text{,index} \end{bmatrix} \ldots$$

lmn	load module name
index	DQE entry number

The DQE directive dumps the specified active DQEs in text format. If no parameters are specified, all active DQEs are dumped.

If any parameters are invalid, an error message is displayed on the user terminal. All valid parameters are executed.

Note: In the case of J.SWAPR, if the DQE state is SUSP, it is not necessarily linked into the SUSP chain.

The following information is provided:

. Physical address of this DQE

. DQE entry number

. Load module name

. Base mode task declaration

. Current user priority (DQE.CUP)

. Base priority of user task (DQE.BUP)

. I/O priority (DQE.IOP)

. Task state (DQE.US)

If the task is in the MRQ, the memory request type (DQE.MRT) is displayed.

If the task is in the SWGQ, the following information is displayed:

. General queue identity (DQE.PRS, DQE.PRM)

. General queue function code (DQE.GQFN). If this is 'queued for volume resource' or 'queued for dual-port lock' or 'queued for synchronous resource lock', then a description of the associated Allocated Resource Table is displayed:

- UDT index (AR.UDTI)
- Resource descriptor block address (AR.BLOCK)
- Current access mode (AR.CACM)
- Resource allocation flags (AR.FLAGS)
- Resource information:

Resource	Display
Volume	MVT entry pointer (AR.MTVA)
Segment definition	Number of blocks in definition (AR.NBLKS)
Memory partition	SMT entry pointer (AR.SMTA)
Device	UDT entry pointer (AR.UDTA)

Else volume name:

- DQE index of the exclusive lock owner (AR.XRL)
- DQE index of the synchronous lock owner (AR.SRL)
- Number of active assignments (AR.ASSNS)
- Number of resource users (AR.USERS)
- Number of dual-processor requests queued (AR.QUE)
- Number of readers currently on this resource (AR.RDRS)
- Current EOF position in this file (AR.EOF)
- Current EOM position in this file (AR.EOM)

- Task activation number (DQE.TAN)
- Owner name (DQE.ON)
- Pseudonym (DQE.PSN)
- User status word (DQE.USW)
- Scheduling flags (DQE.USHF)
- When an abort is in progress, abort code (DQE.ABC)
- Swapping inhibit flags (DQE.SWIF)

- Number of no-wait I/O requests (DQE.NWIO)
- Number of unbuffered I/O requests currently outstanding (DQE.UBIO)
- Number of no-wait mode run requests outstanding (DQE.NWRR)
- Number of no-wait mode message requests outstanding (DQE.NWMR)
- Number of swappable E-class map blocks currently allocated (DQE.CME)
- Number of swappable H-class map blocks currently allocated (DQE.CMH)
- Number of swappable S-class map blocks currently allocated (DQE.CMS)
- Inclusive span of maps in use (DQE.MAPN)
- Map span required for MIDLs and MEMLs (DQE.MSPN)
- Shadow memory flags (DQE.SHF)

2.1.4 DUMP Directive

Dump Data

Syntax:

```
{DUMP [st_addr] [,end_addr]}
{st_addr [,end_addr]       }
```

st_addr	is the physical address where the dump begins. The default is zero.
end_addr	is the physical address where the dump ends. The default is the last data dumped by H.DMPMT.

The DUMP directive writes the specified data in hexadecimal/ASCII format - four words per line on the user terminal, or eight words per line preceded by an index of data structures on the SLO device. See the INDEX directive. When no parameters are specified, all data is dumped.

Dump Load Module

Syntax:

```
DUMP # ,index [,R] [,st_addr] [,end_addr]
```

#	indicates that a load module is to be dumped
index	is the DQE entry number
R	denotes addresses relative to the start of DSECT. If R is not specified, addresses are logical and start at the beginning of TSA.

st_addr	is the address where the dump begins
end_addr	is the address where the dump ends

ANALYZE displays the DQE index followed by a side-by-side ASCII/hexadecimal dump of the load module - four words per line on the user terminal, or eight words per line on the SLO device.

Note: The tasks with hand-built TSAs cannot be dumped in this manner.

2.1.5 EXIT Directive

Syntax:

```
EXIT
```

The EXIT directive exits ANALYZE. A prompt is displayed asking whether to delete the disc file; reply Y or N.

2.1.6 HELP Directive

Syntax:

```
HELP [topic]
```

topic	is the directive to be described

The HELP directive displays information pertaining to the topic. If the topic is not supplied, a list of topics is displayed.

2.1.7 INDEX Directive

Syntax:

```
INDEX
```

The INDEX directive displays the physical addresses of various data structures on the user terminal regardless of print or view execution. The addresses displayed are:

- Allocated resource table (AR.)
- Channel definition table (CHT.)
- Communication region (C.)
- Controller definition table (CDT.)
- CPU scratchpad image
- Device type table (DTT.)
- Dispatch queue (DQE.)

- End of data read in
- End of operating system
- Interrupt timer table (ITT.)
- Map image list for operating system
- Master process list
- Memory allocation table (MEM.)
- Module address table
- Mounted volume table (MV.)
- Resourcemark table
- Shared memory table (SMT.)
- Unit definition table (UDT.)

2.1.8 POOL Directive

Syntax:

$$\text{POOL} \begin{Bmatrix} \text{FREE} \\ \text{USED} \end{Bmatrix}$$

FREE	are free cells in memory pool
USED	are used cells in memory pool

The POOL directive displays the start and end physical addresses and the number of free or used cells in the memory pool. It then displays the forward and backward links, the address of the cell header, and the number of words for each cell followed by a side-by-side hexadecimal/ASCII dump of the cell's data area.

2.1.9 PRINT Directive

Syntax:

```
PRINT
```

The PRINT directive prints all requested data on the SLO device until a VIEW directive is executed.

2.1.10 SEARCH Directive

Syntax:

```
SEARCH [st_addr], [end_addr], value [,mask]
```

st_addr	is the address where the search begins. The default is zero.
end_addr	is the address where the search ends. The default is the last address dumped by H.DMPMT.
value	is the value to be matched
mask	is a hexadecimal word mask. The default is 'XFFFFFFFF'.

The SEARCH directive searches within the specified addresses for a match to the supplied value.

2.1.11 SET SAME Directive

Syntax:

$$\text{SET SAME=} \begin{Bmatrix} \text{Y} \\ \text{N} \end{Bmatrix}$$

Y	enables SAME logic. Default.
N	disables SAME logic

The SET SAME directive sets or resets the same logic for a hexadecimal/ASCII dump. The SAME logic compares the last word of the previous line to the entire current line. If they are the same, the current line is not printed and SAME is displayed. For all consecutive lines that match, SAME is displayed once. SAME logic remains set or reset until it is changed by the SET SAME directive.

2.1.12 SMT Directive

Syntax:

$$\text{SMT} \begin{bmatrix} \text{pname} \\ \text{owner} \end{bmatrix} \begin{bmatrix} \text{,pname} \\ \text{,owner} \end{bmatrix} \ldots$$

pname	is a partition name
owner	is an owner name or task number

The SMT directive displays a text description of the Shared Memory Table (SMT) for the specified partition, owner name, or task number. If no parameters are specified, all SMT entries are displayed.

The following information is provided:

. Physical address of the SMT
. Partition name (SMT.NAME)
. Owner name or task number associated with the partition, if any (SMT.TNUM)
. Owner name of partition creator (SMT.OWNR)
. Project group name associated with partition (SMT.PROJ)
. Entry index (SMT.IND)
. Flags (SMT.FLAG)
. Number of tasks sharing this memory that are not outswapped (SMT.UCNT)
. Number of map blocks (SMT.MAPN)
. Starting map register number (SMT.MAPS)
. If dynamic memory partition, memory type (SMT.MTY)
. Total number of pages (SMT.PTOT)
. Start address of map image descriptor list (SMT.MIDL)

2.1.13 TSA Directive

Syntax:

$$\text{TSA} \quad \begin{bmatrix}\text{lmn}\\ \text{index}\end{bmatrix} \begin{bmatrix}\text{,lmn}\\ \text{,index}\end{bmatrix} \ldots$$

lmn	is a load module name
index	is the DQE entry number

The TSA directive dumps TSAs in text format. If no parameters are specified, all the active TSAs are dumped. If any parameters are invalid, a message is displayed on the user terminal. All the valid parameters are executed.

Note: The TSA for J:SWAPR cannot be dumped.

The following information is provided:

. Task's load module name
. Task's DQE index
. Contents of the pushdown stack (registers and PSD for each level)
. Logical start address of TSA

- Physical start address of TSA
- Address of task's file assignment table (T.FATA)
- Address of task's file pointer table (T.FPTA)
- Address of task's segment definition table (T.SEGA)
- Number of FAT/FPT pairs associated with task (T.FILES)
- Start address of task's DSECT area (T.BIAS)
- End address of task's DSECT area (T.END)
- Transfer address of task's main segment (T.TRAD)
- End address of the TSA (T.TEND)
- Status bits (T.BIT1, T.BIT2, T.BIT3, T.BIT4, T.BIT5)
- Map image descriptor list and memory attribute list (T.MIDL, T.MEML)
- DSECT origin within T.MEML/T.MIDL (T.DSOR)
- DSECT size in map blocks (T.DSSZ)
- CSECT origin within T.MEML/T.MIDL (T.CSOR)
- CSECT size in map blocks (T.CSSZ)
- Extended address origin in T.MEML/T.MIDL (T.EAOR)
- Extended address size in map blocks (T.EASZ)

The following is also provided if the task was a base mode task:

- Task's prelocation delta (T.PREL)
- Start address of the shared image descriptors (T.SHIMDA)
- Number of shared image descriptors (T.NSI)

2.1.14 UDT Directive

Syntax:

```
     UDT    [IOQ]    ⎡devmnc⎤ ⎡,devmnc⎤ . . .
                     ⎣index ⎦ ⎣,index ⎦
```

IOQ	causes an I/O queue summary to be displayed for each UDT listed
devmnc	is a device type mnemonic, such as TY
index	is the UDT index

The UDT directive displays a text description of the UDTs. If the devmnc and index parameters are not specified, all UDTs are displayed.

The following information is provided:

. Physical address of the UDT

. UDT index (UDT.UDTI)

. CDT index (UDT.CDTI)

. Unit status (UDT.STAT, UDT.STA2)

. If the device is allocated (see flags):

  . DQE index

  . Load module name

  . Associated ART data (see DQE directive)

. Device type code (UDT.DTC)

. Mnemonic, channel, and subaddress (UDT.CHAN, UDT.SUBA)

. Flags (UDT.FLGS, UDT.BIT2)

. Service interrupt handler address (UDT.SIHA)

The IOQ summary consists of the number of entries on the list (UDT.IOCT) plus any I/O queue data supplied by the CDT directive.

2.1.15 VIEW Directive

Syntax:

```
VIEW
```

The VIEW directive displays all requested data on the user terminal until the PRINT directive is executed.

Examples:

Using the tape for the first time:

```
TSM>ANALYZE
PLEASE ENTER THE DISC FILE PATHNAME
ANA>@TB00(SYSTEM)DUMP
USE TAPE, FLOPPY OR DISC FILE?
ANA>TAPE                          Reads tape to above disc file.
DISC FILE ALREADY EXISTS          Only displayed if the disc file already
SHALL I DELETE THE DISC FILE?     exists.
ANA>Y
```

```
THE SUPPLIED DATA IS FROM A          Only displayed if revision is not 3.3.
REVISION xxxx SYSTEM.
CONTINUE ? (Y/N)
ANA>Y
ANA>TSA                              Displays all active TSAs to user terminal.
ANA>DUMP 180000,                     Displays memory from 180000 (hexadec-
                                     imal) to the end of the data from
                                     H.DMPMT to the user terminal.
ANA>TSA J.TSM,A,J.SOUT               Displays the TSAs of J.TSM, the task with
                                     DQE entry number A and J.SOUT to the
                                     user terminal.
ANA>DUMP #,5                         Displays the load module having DQE
                                     number five to the used terminal.
ANA>PRINT                            Diverts display to the SLO device.
ANA>CDT IOQ 5,TY                     Prints the CDTs for CDT number five and
                                     all teletypewriters with I/O queue
                                     summaries to the SLO device.
ANA>SMT GLOBAL01                     Prints the SMT of GLOBAL01 to the SLO
                                     device.
ANA>EXIT
SHALL I DELETE THE DISC FILE?
ANA>N
TSM>
```

Using the previously produced disc file:

```
TSM>ANALYZE
PLEASE ENTER THE DISC FILE PATHNAME
ANA>@TB00(SYSTEM)DUMP
USE TAPE, FLOPPY OR DISC FILE?
ANA>DISC
THE SUPPLIED DATA IS FROM A          Only displayed if revision is not 3.3.
REVISION xxxx SYSTEM.
CONTINUE ? (Y/N)
ANA>Y
directives for ANALYZE
ANA>X
SHALL I DELETE THE DISC FILE?
ANA>Y                                If you have finished with the disc file
                                     enter Y; otherwise, enter N.
TSM>
```

2.2 AUTOJOB - Automatic Batch Job Submission on Boot-up

AUTOJOB is activated after J.TSM by the SYSGEN directive SEQUENCE=(AUTOJOB). The program searches the system volume and directory for the file M.BATCH. If the file is not found, AUTOJOB exits. If it is found, AUTOJOB scans each line and processes the directives which are syntactically correct. The OPTION directive can be used to prevent error messages from being displayed at the console and to abort AUTOJOB if an error occurs.

Note: This program is written in FORTRAN 77+ Revision 4.x and uses only native services.

Syntax:

```
AUTOJOB
```

Command file syntax:

```
cmd=arg [com]
     or
[-] path [com]
```

cmd	is the command keyword terminated by the equal sign
arg	is the command argument
com	is the comment field. Comments are ignored by AUTOJOB.
-	inhibits the message which indicates that the job was sent to the batch stream. Must be used before a pathname.
path	is the full pathname of the file to be sent to the batch stream at boot-up. The files are preprocessed by J.SSIN and can be edit stored or saved, Toolkit card image, or default format with trailing blanks removed.

Command Summary:

Syntax:

```
* [com]
```

com	is a comment. Any line starting with the asterisk (*) character is ignored by AUTOJOB.

Syntax:

```
NOTE=[com]
```

com	is a text message up to column 72 which is sent to the console in the format AUTOJOB:com

Note: This command is used primarily to produce milestone messages for M.BATCH file processing.

Syntax:

$$\text{OPTION} = \begin{bmatrix}\text{MESSAGE}\\ \text{NOMESSAGE}\end{bmatrix}\ [\text{com}]$$

or

$$\text{OPTION} = \begin{bmatrix}\text{ABORT}\\ \text{NOABORT}\end{bmatrix}\ [\text{com}]$$

MESSAGE	sends messages to the console. Default.

NOMESSAGE	prohibits message output to the console
com	is an optional comment
ABORT	aborts command processing if an error is detected
NOABORT	continues command processing if an error is detected. Default.

Note: The OPTION flags can be enabled or disabled at any time to control processing.

Syntax:

```
[-] path [com]
```

-	inhibits the message which indicates that the job was sent to the batch stream
path	is the pathname of the file submitted to the batch stream. Must begin in column one and be followed by a blank.

Resource Summary:

Unit CT	is the LFC used to write messages to the console. It is dynamically allocated and should not be reassigned.
Unit BAT	is the LFC used to read the M.BATCH file. It is dynamically allocated and should not be reassigned.

M.BATCH file example:

```
*
** Prepare the system for application
*
OPTION=ABORT                    If the following job fails, abort
NOTE=                           Start up application program
@USER1(SETUP)BUILD              Build application environment
OPTION=NOABORT                  Do not abort if errors occur
@USER1(SETUP)COPY               Copy files to application disc
@USER1(SETUP)START              Start up application
-@USER1(SETUP)CLEANUP           Clean up everything except messages
OPTION=NOMESSAGE                Do not send messages to the console
*
** Process System Administrator tasks
*
@SYSTEM(SYSTEM)CLEANUP          Clean up workfiles
@SYSTEM(SYSTEM)BUILDMD          Initialize and mount memory disc
@SYSTEM(SYSTEM)MONITOR          Activate background monitor
OPTION=MESSAGE                  Send messages to the console
NOTE=                           Log all the disc files
@SYSTEM(SYSTEM)LOGDISC          Log all the files on the system
NOTE=                           The job is complete
```

Notes:

1. All directives must start in column one. They are processed up to column 72.
2. The M.BATCH file must be edit stored or Toolkit card image.
3. The jobs submitted must mount required volumes by the JCL in the particular batch job or must refer to volumes mounted by the M.MOUNT file.
4. The OPTION = NOMESSAGE directive does not disable error messages. The hyphen (-) option is not required if this directive is used.
5. The following defaults are in effect at start-up:

```
OPTION = NOABORT
OPTION = MESSAGE
```

2.3 DD - Disc Dump by Sector

DD creates a side by side hexadecimal/ASCII dump of any system-configured disc.

The disc modification option is activated by hitting any key except return after the TSM ENTER CR FOR MORE message.

Syntax:

```
DD addr [,qty [,dev [,str] ] ]
```

addr	is the absolute sector to start the dump (hexadecimal)
qty	is the number of sectors to dump (decimal or hexadecimal if preceded by an X). The default is all.
dev	is the device where the dump occurs (DM0800). The default is system disc.
str	is a one- to eight-character string to search for

Option 13 allows interactive modification of data on disc for the System Administrator only; no modification if not set.

Optional Assignments:

```
$ASSIGN OT TO LFC=UT    ! DEFAULT ASSIGNMENT (4 WORD WIDE DUMP)
or
$ASSIGN OT TO SLO       ! SLO ASSIGNMENT (8 WORD WIDE DUMP)
```

Aborts:

RM02	ACCESS MODE NOT ALLOWED

Someone other than the System Administrator attempted to modify the disc.

2.4 DDUMP - Disc Dump by File

DDUMP creates a side by side hexadecimal/ASCII dump of a disc file.

A file can be patched by hitting any key except return at the TSM ENTER CR FOR MORE message.

Syntax:

```
DDUMP file [,stad [,enda] ]
```

file	is a one- to sixteen-character file name of a file in the current working directory or the system directory that is to be dumped
stad	is a hexadecimal starting address relative to the first block of the file (zero) where the dump starts
enda	is a hexadecimal ending address relative to the first block of the file where the dump stops

Optional Assignments:

```
$ASSIGN OT TO LFC=UT    ! DEFAULT ASSIGNMENT (4 WORD WIDE DUMP)
or
$ASSIGN OT TO SLO       ! SLO ASSIGNMENT (8 WORD WIDE DUMP)
```

Aborts: Various system aborts. No internal aborts.

2.5 DOWNDATE - Source Compare Program

DOWNDATE is used to identify changes between program source files.

Note: This program is written for FORTRAN 77+ Revision 4.x and requires native services.

Syntax:

```
DOWNDATE
or
$EXECUTE DOWNDATE ! BATCH MODE
```

Default Assignments:

```
$ASSIGN 1 TO ORIGINAL      ! OLD FILE (PATHNAME) (static assignment required)
$ASSIGN 2 TO UPDATED       ! NEW FILE (PATHNAME) (static assignment required)
$ASSIGN 5 TO DEV=NU        ! SYNCHRONIZATION (normally not used)
$ASSIGN 6 TO SLO           ! OUTPUT FROM PROGRAM (optional)
```

Aborts: Various FORTRAN run-time aborts.

Note: If there are too many changes, the program loses sync and is not able to resync.

2.6 DOWNDAT2 - Source Compare Program/Source Update Output

DOWNDAT2 is an enhanced version of DOWNDATE that identifies changes between source files. The output format is slightly different from DOWNDATE. Generated output can be processed by Source Update.

Note: This program is written for FORTRAN 77+ Revision 4.x and requires native services.

Syntax:

```
     DOWNDAT2
     or
     $EXECUTE DOWNDAT2 ! BATCH MODE
```

Default Assignments:

```
$ASSIGN 1 TO ORIGINAL      ! OLD FILE (PATHNAME) (static assignment required)
$ASSIGN 2 TO UPDATED       ! NEW FILE (PATHNAME) (static assignment required)
$ASSIGN 5 TO DEV=NU        ! SYNCHRONIZATION (normally not used)
$ASSIGN 6 TO SLO           ! OUTPUT FROM PROGRAM (optional)
$ASSIGN 7 TO DEV=NU        ! OUTPUT FOR SOURCE UPDATE PROCESSING
                             (optional)
```

Aborts: Various FORTRAN run-time aborts.

Note: If there are too many changes, the program loses sync and is not able to resync.

2.7 GM.FLOW - Assembly Source Code Flowchart Tool

GM.FLOW reads an uncompressed, nonbase assembly source code file and produces a display of all forward and backward references. The output consists of up to 29 visible levels of forward references, 72 bytes of source code, and up to 29 visible levels of backward references. This report is followed by a summary displaying unresolved references and unreferenced symbols; for example, BU *ADDRESS,X1 cannot be resolved. This report also contains simple statistics collected during run time; for example, forward references found and records processed.

Optionally, a sequence number requiring eight spaces is placed in the source output starting in any column from 1 to 65. When used, the sequence number overlays the source output and is relative to the first record actually processed. The record where processing starts and stops can also be optionally specified.

Note: This program is written for FORTRAN 77+ Revision 4.x and requires native services.

Syntax:

```
     GM.FLOW [-S [column] ] [start] [stop]
```

-S	indicates that sequence numbers are to be placed in the text section of the output
column	specifies a column from 1 to 65 as the starting location of the sequence number. If not specified, the default is one.

start	indicates the record number where processing starts. Records skipped are ignored and not listed. If not specified, the default is the first record in the file. A zero indicates that the default is used.
stop	indicates the record number where processing stops. Records after the stop are not read. If not specified, default is the last record in the file. A zero indicates the default is used.

Resource summary:

```
$AS SI TO SOURCE_FILE         ! REQUIRED ASSIGNMENT
$AS SO TO SLO                 ! DEFAULT ASSIGNMENT.  RE-ASSIGNABLE
```

Example of output:

```
          *
          ** RUN LOOP GM.FLOW DEMO
          *
         START                    LW               R1,DATA
+<<<<<<<<<<<<<<<<<<<<<<<BZ        MERGE
!         REDO        LI          R1,1<<<<<<<<<<<<<<<<<<<<<<<<< +
! +<<<<<<<<<<<<<<<<<<<<<BU        HERE                          !
! !      DATA           REZ                          1W         !
+======   MERGE         BU        LOOP                          !
! +>>>>   HERE          BU        REDO========================= +
+>>>>>>   LOOP          TRN       R1,R4                         !
          DO            STW       R4,DATA<<<<<<<<<<<<<<<<<<<+   !
                        BIB       R4,DO>>>>>>>>>>>>>>>>>>>>>+   !
                        BU        HERE>>>>>>>>>>>>>>>>>>>>>>>>> +
```

2.8 GM.SEAR - Source Search Tool

GM.SEAR searches for two different strings within a blocked, uncompressed source file. The search can be by columns if the string starts within a predefined column range.

Output can be displayed as follows:

. The matching line with seven lines preceding and following it are displayed

. Only the matching line is displayed. This display resembles edit when used for searching.

. All lines beginning at a specified line are listed

Additional functions include:

. Echo of screen data to SLO. Can be enabled and disabled.

. On-line help and program status inquiry

. File movement through back record, skip record, or rewind functions

. Support of full pathnames

. Comments can be sent to SLO to document echoed messages

. Support of standard terminal and ANSI mode terminal screen controls

Note: This program is written for FORTRAN 77+ Revision 4.x and requires native services.

Syntax:

```
    GM.SEAR [-A]
```

-A indicates that the terminal requires ANSI screen control codes

Note: Additional documentation is resident within the program. It can be examined by requesting help at the first prompt or by selecting help from the command menu.

2.9 LMINFO - Load Module Information

LMINFO displays load module information from the preamble after receiving responses from the following prompts:

```
INPUT THE VOLUME THAT THE LOAD MODULE RESIDES ON:
INPUT THE DIRECTORY THAT THE LOAD MODULE RESIDES ON:
INPUT THE LOAD MODULE NAME:
```

Syntax:

```
LMINFO
```

2.10 LOGMDT - Log Contents of Rapid File Allocation MDT

LOGMDT displays the address, size, and contents of the Rapid File Allocation MDT area. It can optionally include a hexadecimal dump of all active and inactive Resource Descriptor (RD) entries in the MDT.

LOGMDT gets and displays the MDT address and entry parameters from the MPX-32 communications area. If the MDT base address is non-zero, it then displays the pathnames and entry numbers of files with entries (active or inactive) in the MDT. Entry numbers are typically non-linear because RD entries are hashed into the MDT. LOGMDT checks all of the possible entry positions for any non-zero word, and will report any entry position that is not totally zeroed.

If the MDT is not configured (the base address equals zero), LOGMDT terminates after displaying the address and entry parameters.

Note: This package consists of a main program written in FORTRAN 77+ Revision 4.2 and three subroutines written in non-base assembler, and uses only native services. The program runs in privileged mode.

Syntax:

```
    LOGMDT [ V ]
```

V invokes the verbose option. Each entry logged includes the file's owner name, project name, file type, and a hexadecimal dump of the file's 192-word RD.

Resource Summary:

Unit LO	Log information and error output. The default is to LFC=UT (static, reassignable). For hard-copy output, use $ASSIGN LO TO SLO.

Error Reporting and Termination:

Internal Aborts:	None
Error Processing:	The program reports error messages and ends execution for the following conditions:

- A consistency check of two MDT "C." parameters fails
- The internal buffers cross map block boundaries
- Improper status from external subroutines

Examples of output:

The following output is produced when the MDT has two active entries:

```
************************************************
 LOGMDT  date  LOG RAPID FILE ALLOCATION - MDT-

MDT INSTALLED, CURRENT PARAMETERS ARE:

MDT BEGIN ADDRESS (REAL)    =  001E8000 HEX
MDT TOTAL LENGTH (BYTES)    =  00006000 HEX   24576 DEC
MDT END ADDRESS (REAL)      =  001EE000 HEX

# MDT ENTRIES CURRENTLY USED        =    2
# ENTRIES NOW AVAIL. (FREE)         =   30
TOTAL # ENTRIES (MDT SIZE)          =   32

ENTRY #   27, ACTIVE !
VOL/DIR/FIL  =  MPX3.3          RAPFILE          MT02

ENTRY #   28, ACTIVE !
VOL/DIR/FIL  =  MPX3.3          RAPFILE          MT01

ENTRIES NOT LOGGED ARE ALL ZEROS !!
A TOTAL OF   32 ENTRIES SEARCHED.

********************END OF LOGMDT*****************
```

The following output is produced when the MDT has one active entry and one inactive entry:

```
*************************************************
 LOGMDT  date  LOG RAPID FILE ALLOCATION - MDT -

MDT INSTALLED, CURRENT PARAMETERS ARE:

MDT BEGIN ADDRESS (REAL)    =  001E8000 HEX
MDT TOTAL LENGTH (BYTES)    =  00006000 HEX      24576 DEC
MDT END ADDRESS (REAL)      =  001EE000 HEX

# MDT ENTRIES CURRENTLY USED  =          1
# ENTRIES NOW AVAIL. (FREE)   =         31
TOTAL # ENTRIES (MDT SIZE)    =         32

ENTRY #    27, *** NOT ACTIVE ***
VOL/DIR/FIL  =  MPX3.3            RAPFILE            MT02

ENTRY #    28, ACTIVE !
VOL/DIR/FIL  =  MPX3.3            RAPFILE            MT01

ENTRIES NOT LOGGED ARE ALL ZEROS !!
A TOTAL OF    32 ENTRIES SEARCHED.

*********************END OF LOGMDT****************
```

2.11 LOGOFF - Remote Terminal Logoff

LOGOFF terminates outstanding I/O at a specified TY address and logs off the current user. UDT linked devices must also be TSM devices for this program to run. The source code (US.LOGO) can be modified to specify System Administrator use only. This program runs privileged only when required.

Note: This program is written in nonbase assembler and requires native services.

Syntax:

```
LOGOFF TYccss
```

cc	is the channel
ss	is the subchannel

Aborts:

US01	Fatal error detected. See user's terminal for text message.
US02	Unrecoverable I/O error on user's terminal

2.12 MIPSMON - Instruction Sequence Timing Tool

MIPSMON enables a user to time individual instruction sequences with great accuracy. Instruction sequences are defined in an assembly language program using MIPSMON macros. This program is linked with the MIPSMON Executive to form a load module which is then run in a stand alone environment. The instructions are timed using an RTOM. An output file is produced. Another program, MIPREP, reduces the output file and presents the results of the measurement in terms of time (nanoseconds) and machine cycles.

MIPSMON contains the following software:

File	Description
MIPSMONX32.M	MIPSMON base object module
MIPS_LIB	MIPSMON nonbase object library
MIPS_DIR	MIPSMON nonbase object directory
MIPMACX32	Base mode macro library
MIPMAC	Nonbase mode macro library
MIPMSYS	MIPS system JCL
MIPREP	MIPS report generator

The user interface to MIPSMON consists of a JCL file, MIPSYS, and a load module, MIPREP. MIPSYS reads an instruction sequence file containing the instruction sequences to be timed. It outputs a load module. When the load module is run, an output file is produced which contains the results of the timing. MIPREP produces a report from this output file.

MIPSYS

Syntax:

```
MIPSYS file,lm,[base], [defout]
```

file	is the instruction sequence file that is input
lm	is the load module that is output
base	indicates base register instructions. The default is nonbase register.
defout	is the output file. If omitted, the timing results are output to MIP.OUT.

MIPREP

Syntax:

```
MIPREP defout [>PRINT]
```

defout	is the output file
>PRINT	indicates that output is to the printer. Default is to the terminal.

Instruction sequence files are assembly language programs containing the instructions to be timed interspersed with MIPSMON macros which direct the MIPSMON Executive to time the instruction sequences as specified.

An instruction sequence file has the following format:

```
PROGRAM prgname
MIPINIT
   .
   .
   .
One or more sequences to be measured
   .
   .
   .
M.DONE
END
```

The MIPINIT macro must appear once in the file under the PROGRAM statement. It initializes certain data structures and assembly variables. There are no parameters for this macro.

The M.DONE macro must appear once in the file following the last instruction sequence to be measured.

Each instruction sequence has the following format:

```
NEW name
M.INIT
(Initialization instructions)
TRSW 0
M.INITL
(Initialization instructions)
TRSW 0
M.INST
(Instruction sequence to be measured)
M.INSTE
```

The NEW macro introduces each new measurement sequence and assigns it an identifier which is printed in the report. The identifier is a character string containing a maximum of eight letters and is the parameter for the NEW macro.

The optional macro, M.INIT, introduces a subroutine which is called before entering the timing loops (dummy and real). It should contain any loop independent initialization required by the sequence being timed. The subroutine is called by a branch and link and must contain a TRSW 0 to return.

The optional macro, M.INITL introduces a subroutine which is called within the timing loops (dummy and real). It should contain any loop dependent initialization required by the sequence being timed. The subroutine is called by a branch and link and must contain a TRSW 0 to return.

The M.INST macro introduces the sequence to be timed. There are no parameters for this macro.

The M.INSTE macro indicates the end of the sequence to be timed. There are no parameters for this macro.

MIPSMON inserts the user's instructions into a sequence of instructions that block interrupts and clear the instruction pipeline so the user's instructions can be timed without interference. The sequence is executed repeatedly and then averaged to obtain an accurate value.

The timing of instruction sequences is dependent on input values. An input value of zero should not be used because it does not represent typical instruction processing.

M.INIT and M.INITL should be used to avoid underflow and overflow conditions which lead to incorrect timing statistics.

Example:

The following instruction sequence file times the LW instruction:

```
                               PROGRAM LWTIME
                               MIPINIT
*
*    INITIALIZATION DATA
*
T                              DATAW  E'2.1'
TP                             DATAW  E'2.1'
E                              DATAW  E'1.877493'
J                              DATAW  E'.499975'
*
*    LW/MPFW/STW TIMING
*
                               NEW TIMING
*
                               M.INITL
                               LW     3,TP
                               STW    3,T
                               TRSW   0
*
                               M.INST
                               LW     3,T
                               MPFW   3,E
                               MPFW   3,J
                               STW    3,T
                               M.INSTE
*
                               M.DONE
                               END
```

The above example times the sequence consisting of a load word instruction, two multiply floating point word instructions, and a store word instruction. The M.INITL routine initializes the variable T to avoid the overflow which occurs from continued multiplications. By resetting T to its original value, the sequence of instructions is timed with the same input values each time.

MIPSMON allows general instruction sequences with the following exceptions:

- The values of R7, B4, B5, and B6 cannot be changed.
- The STRUCT label is reserved for MIPSMON.
- The value of register zero must be preserved.
- There cannot be more than 100 words between M.INST and M.INSTE.
- Branches require special handling.

2.13 NEWCOPY - Floppy Disc Duplication Tool

NEWCOPY performs the following functions:

- Archives floppy discs to a hard disc
- Duplicates floppy to floppy
- Copies from archive to floppy disc
- Performs general maintenance on archives
- Formats floppy discs

Note: This program is written for FORTRAN 77+ Revision 4.x and requires compatible services.

Syntax:

```
    NEWCOPY
```

Optional Assignments:

```
$ASSIGN LO TO SLO               ! AUDIT TRAIL (DEFAULT)
$ASSIGN OT TO LFC=UT            ! PROMPT/MESSAGE OUTPUT (DEFAULT)
$ASSIGN IN TO LFC=UT            ! COMMAND INPUT (DEFAULT)
```

All other assignments for resources are dynamic.

Aborts: Examine source file for details.

2.14 POOLSCAN - Memory Pool Monitor

POOLSCAN monitors the current usage and state of memory pool. It can be used to maintain a historical profile of memory pool usage. The profile can be used to project memory pool requirements.

POOLSCAN uses an updating screen output method that is compatible with the following terminals: ADM-3A, Hazeltine 1200, TeleVideo 910, and TeleVideo 921. It can work with other terminals as well.

Syntax:

```
POOLSCAN [del] [,H] [,-A]
               [,D]
```

del	specifies a time delay of 0 to 60 seconds between screen updates. If not specified, the default is ten seconds.
H	specifies that the display is hexadecimal. Default.
D	specifies that the display is decimal
-A	specifies that the terminal requires ANSI control codes, such as TeleVideo 922 or TeleVideo 970

Notes:

1. If the zero second delay time is used, POOLSCAN degrades system performance because it changes its priority level and does system context switch inhibits while scanning memory pool.

2. POOLSCAN is privileged, but only runs in the privileged mode while scanning memory pool. Otherwise, POOLSCAN runs in the unprivileged mode.

3. BREAK terminates POOLSCAN.

4. POOLSCAN should not be run in the batch mode or on a hard copy device.

5. Setting the page size to zero before running POOLSCAN is not needed. A combination of overprint and other carriage controls are used to avoid line counting.

Aborts:

FREE	indicates that a bad linkage was detected in the memory pool free linked list (only forward pointer checked)
USED	indicates that a bad linkage was detected in the memory pool used linked list (only forward pointer checked)

2.15 PORTPROT - Dial-up Port Protection

PORTPROT adds an extra level of system protection by requiring passwords for selected terminal ports. Documentation for customizing and protecting an individual system is located in the source file.

2.16 SPRINT - Serial Printer Formatter/Spooler

SPRINT formats data for a serial printer that has been connected to an 8512 (1 or 2) eight-line controller. While SPRINT does not support all the functions of the J.SOUT spooler, it provides a clean interface to a serial printer. SPRINT was designed to run with NEC 7715 and NEC 8815 printers and the H.F8XIO handler. It uses WXON and REMOTE as additional J.TINIT parameters.

Note: This program is written for FORTRAN 77+ Revision 4.x and uses only native services.

The form control characters that can be processed by SPRINT in the formatted input mode are:

1	Form feed
0	Double space
+	Overprint
-	Title. When encountered, the first one causes a form feed; the following have no effect until a line without a title character is found. Title lines are not retained and printed at each top of form as with J.SOUT.

All others are treated as a single space.

Syntax:

```
    SPRINT
```

Prompts:

```
    PATHNAME TO PRINT . . . . . . . . . . .  :
```

Enter any uncompressed file where the user has read access. The records can contain up to 132 bytes; any excess is truncated.

```
    FORMATTED ? (Y/N) . . . . . . . . .  :
```

Enter Y to indicate that column one should be interpreted as a carriage control character. Enter N to indicate that the first byte is data to be printed. The default is N.

```
    EMBEDDED FORMS CONTROL . . .  :
```

Enter Y to indicate that the data in the file contains embedded form control characters (files output from ROFF contain embedded form control characters). Internal line counting is inhibited.

Y is normally entered only with the FORMATTED = N mode. Enter N for embedded form control characters to have no effect. The default is N.

```
    DEVICE ADDRESS . . . . . . .  :
```

Enter the device address where the printer is configured or the default that is defined at compile time.

```
    COPIES (1 TO 20) . . . . . . . . . .  :
```

Enter the number of copies to print of the specified file. When making large numbers of copies, the terminal is tied up for an extended period of time.

LINE LENGTH (1 TO 133) . . :

Enter the maximum line length to output to the serial printer. This value must not exceed the line size defined for the device.

Aborts:

RT10	The value entered at the line length prompt exceeds the defined line size.

2.17 TERMLOAD - Terminal Initializer/Loader

TERMLOAD sends predefined control data to a terminal to set up a software controllable operation mode and/or load soft function keys. It can be run in the interactive or independent modes. In the interactive mode, the user specifies a command file and loading options by the command line. The result of command file processing is sent to the terminal where activation was performed. If activated independently by OPCOM or SYSGEN, the system file M.CTLOAD is sent to the system console. If SYSGEN activation is desired, use the SEQUENCE directive for best results. If the file does not exist, nothing is done.

Note: This program is written for FORTRAN 77+ Revision 4.x and uses only native services.

Syntax:

```
TERMLOAD [-] fil
```

-	inhibits a hardcopy audit trail from being spooled to SLO by LFC LO. If errors are encountered, a spool file is still created.
fil	is the pathname of an uncompressed file that contains commands for the TERMLOAD processor

Command file syntax:

```
cmd arg [,arg] [,'str'] [com]
```

cmd	is a three-character command followed by a single blank or a comment character (!,*, or C) in the first byte. See command summary.
arg	is a one- or two-character argument terminated by a comma or a blank (or byte 72) to indicate the end of the argument list. Two-character arguments are processed as ASCII/hexadecimal, and one-character arguments are plain ASCII characters.
str	ASCII (X'20' to X'7E') text not including the single quote (X'27') started by a single quote and terminated by another quote or EOL (byte 72)
com	is a comment following the argument list termination blank

Command summary:

ABS arg|str [,arg|str]...[comment]

Indicates absolute load. The data in the argument list is sent to the unit connected to LFC OT without any leading or trailing data.

DEL arg [comment]

Delay command processing for the number of seconds defined by the hexadecimal argument (hexadecimal only). The value of the argument can be from 1 to 255 seconds.

DEV arg|str [comment]

Indicates device redefinition. The target device is changed to another TY device, such as TY7EA0.

END [comment]

Terminates the command file.

HED arg|str [,arg|str]...[comment]

Indicates header definition. The data in the argument list is retained (unless overwritten) and used as a prefix for data supplied by the MID directive.

MID arg|str [,arg|str]...[comment]

Indicates middle definition. The data in the argument is prefixed by the current HED data and suffixed by the current TER data. The result is retained and sent to the unit connected to LFC OT. If HED and TER directives have not been performed previously, a warning error is produced and the directive is ignored. If the sum of the HED, TER, and MID argument lists exceeds 80 bytes, a warning message is produced and the directive is ignored.

MSG [comment]

Any text following the MSG directive (through byte 72) is echoed to the unit connected to LFC OT.

REP arg [comment]

The last valid ABS or MID directive is repeated the number of times defined by the hexadecimal argument (hexadecimal only). The value of the argument can be from 1 to 255.

REM [comment]

Any text following the REM directive is treated as a comment. !, *, and C are also valid.

TER arg|str [,arg|str]...[comment]

Indicates terminator definition. The data in the argument list is retained (unless overwritten) and used as a suffix for data supplied by the MID directive.

Resource summary:

Unit SI	is a dynamically allocated file that contains uncompressed 80 byte records, numbered or unnumbered. There is no default (specified by directive line) unless it is OPCOM or SYSGEN activated, so SI will be assigned to the system file M.CTLOAD.
Unit LO	is an audit trail and error message unit. All but error messages can be suppressed. The default is to SLO (static, reassignable).
Unit OT	is the destination for directive output. The default is to UT (user's terminal, dynamic, or device CT if SYSGEN or OPCOM activated).

Notes:

1. All of the directives can have on-line comments, as long as they are preceded by the argument list termination blank.

2. Errors always produce output to unit LO.

3. All directives start in the first byte position and end in byte position 72.

Directive file example:

```
*
** Termload example
*
ABS 2B,1A,1B,1C,08,20,0D          Clear the screen
MSG                               The first three function keys of the terminal
                                    are about to be loaded
DEL 02                            Wait a few seconds
HED 1B,1B,|                       Define lead in for the terminal
TER 19                              and terminator
MID 1,1,!,L,I,S,T,0D              OPCOM LIST in key one
MID 2,1,E,X,I,T,0D                EXIT directive in key two
MID 3,1,'SIGNAL',0D               SIGNAL directive in key three
ABS 2B,1A,1B,1C,08,20,0D          Clear the screen
ABS 2B,07,0D                      Load a beep
REP 10                            Send it 16 times
DEV 'TY7EC3'                      Change devices
REP 10                            Send it 16 beeps
MSG LOAD COMPLETE
END                               End
```

Gould Inc., Computer Systems Division
6901 W. Sunrise Blvd.
P. O. Box 409148
Fort Lauderdale, FL 33340-9148
Telephone (305) 587-2900

GOULD
Electronics

Users Group Membership Application

USER ORGANIZATION: ______________________________

REPRESENTATIVE(S): ______________________________

____

____

ADDRESS: ______________________________

____

TELEX NUMBER: ______________ PHONE NUMBER: ______________

NUMBER AND TYPE OF GOULD CSD COMPUTERS: ______________________________

____

OPERATING SYSTEM AND REV. LEVEL: ______________________________

____

APPLICATIONS (Please Indicate)

1. EDP
   - A. Inventory Control
   - B. Engineering & Production Data Control
   - C. Large Machine Off-Load
   - D. Remote Batch Terminal
   - E. Other

2. Communications
   - A. Telephone System Monitoring
   - B. Front End Processors
   - C. Message Switching
   - D. Other

3. Design & Drafting
   - A. Electrical
   - B. Mechanical
   - C. Architectural
   - D. Cartography
   - E. Image Processing
   - F. Other

4. Industrial Automation
   - A. Continuous Process Control Op.
   - B. Production Scheduling & Control
   - C. Process Planning
   - D. Numerical Control
   - E. Other

5. Laboratory and Computational
   - A. Seismic
   - B. Scientific Calculation
   - C. Experiment Monitoring
   - D. Mathematical Modeling
   - E. Signal Processing
   - F. Other

6. Energy Monitoring & Control
   - A. Power Generation
   - B. Power Distribution
   - C. Environmental Control
   - D. Meter Monitoring
   - E. Other

7. Simulation
   - A. Flight Simulators
   - B. Power Plant Simulators
   - C. Electronic Warfare
   - D. Other

8. Other

Please return to:

Users Group Representative

Date: ______________

Gould Inc., Computer Systems Division Users Group. . .

The purpose of the Gould CSD Users Group is to help create better User/User and User/Gould CSD communications.

There is no fee to join the Users Group. Simply complete the Membership Application on the reverse side and mail to the Users Group Representative. You will automatically receive Users Group Newsletters, Referral Guide and other pertinent Users Group activity information.

Fold and Staple for Mailing

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 947 FT. LAUDERDALE, FL

POSTAGE WILL BE PAID BY ADDRESSEE

GOULD INC., COMPUTER SYSTEMS DIVISION
ATTENTION: USERS GROUP REPRESENTATIVE
6901 W. SUNRISE BLVD.
P. O. BOX 409148
FT. LAUDERDALE FL 33340-9970

Fold and Staple for Mailing

(Detach Here)

GOULD
Electronics