MICHIGAN STATE UNIVERSITY

COMPUTER LABORATORY

6000 SCOPE MEMO NO. 135.1

FEBRUARY 28, 1979

INTERACTIVE SUBSYSTEM

1.0 INTRODUCTION

The interactive subsystem is responsible for all interactive jobs. It must coordinate activities for the MISTIC2 users, the FREND 7/32 system and the 6500 operating system. MANAGER is the major component, which oversees the needs of all interactive jobs and satisfies those needs or directs others to. MANAGER also communicates with the FREND system via 1FP and a set of control ports.

In addition to managing interactive jobs, MANAGER drives several line printers connected to the front-end. MANAGER's function for the printers is to get print files from ECS (with help from MAN), read the files in such a way that the dayfile will be printed first, and perform various file positioning commands for FREND. FREND is the supervisor for batch print; MANAGER is a slave processor.

This voids 6SM number 47, MSX, and sections 4.2, 4.4.3-4.4.5 of 6SM number 60, HUSTLER 2, (interactive core).

2.0 EXTERNAL REFERENCE SPECIFICATIONS

The primary external user documentation for the interactive subsystem is Interactive System User's Guide, chapters 2 and 5.

2.1 MANAGER COMMANDS

MANAGER recognizes the following commands as EDITOR commands:

```
OLD         SCRATCH     SYSTEM
SAVE        LIST        RESEQ
MERGE       TAB         TABCH
```

USE
MARGIN
MOVE
EDSTAT
COMP
SET
BATCH
FTN
DUP
LENGTH
READ
EWFLOCK
COMPX
STRING
LISTF
COBOL
DELETE
INSERT
PUNCH
FOLD
COMPER
FORMAT
GO
BASIC

These commands will be sent to EDITOR unless they are prefixed with a "$", then MANAGER will interpret them as SCOPE commands.

The following are MANAGER commands for which MANAGER must do some processing. Then EDITOR or another SCOPE routine may be called in to complete the processing of the command.

EOT
OK
MISTIC
TAPE
READPT
N
LOGOUT
READY
END
TAPEC
TPREAD

2.2 MAN FUNCTIONS

The format of the MAN calls is:

```
18/3LMAN.1/IB.5/0.12/FUNC,24/PARAMETER ADDRESS
```

where IB is the internal bit (set for function 6).
FUNC is the function number.
PARAMETER ADDRESS is the parameter address (MANAGER's RA for function 6).

Parameter word (functions 1-5 only):

```
42/STUFF.11/USER.1/CB
```

where STUFF is dependent on the function.
USER is the user number and
CB is the complete bit.

Function 6 (MISTIC2 idledown) does not use a parameter word.

Function 7 (get an output file) uses a 4-word block:

```
3/0. 3/flags, 18/PRU1. 6/0. 18/PRU2. 6/char, 5/code. 1/CB
```

```
60/0
60/0
60/0
```

Flags = printer type bits for CE.LISP call
PRU1 = first PRU limit
PRU2 = second PRU limit
char = routing character for print file
code = 2 for print output
CB = complete bit

On exit, the 4-word block looks like this:

```
36/0, 12/FNT address. 12/1
60/FNT1
60/FNT2
60/FNT3
```

If no file was found, only the complete bit is set. If a file is found, its FNT address is returned in the first word, followed by the 3-word FNT image.

Function 8 (re-queue an output file) uses a 5-word block:

```
42/file name, 6/new queue, 11/code, 1/complete bit
60/unused
60/unused
60/unused
12/FNT address, 48/unused
```

"New queue" is the source character for the output queue that the file is to be put on. "Code" is 1 to return (unload) the file, or 2 to re-queue it.

2.3 FRONT END COMMANDS

Front End commands can be issued either by the users at their interactive terminals or from a running program on the 6500. The PP routine FER was written for the Front End project to accept a Front End command from a CPU program and via a stack request, send it to the 7/32. A control card callable routine, a FORTRAN callable function, and user macros were written to call FER to issue the Front End command.

2.3.1 FER

The format of the FER call is:

```
18/3LFER,6/20B,12/FUNCTION,6/0,18/PARAMETER ADDRESS
```

FER must be called with recall.

Function 1 is used to transmit Front End commands from the 6500 to FREND. The command and its associated character code are specified as follows in the parameter word:

```
24/0,12/LENGTH.12/CODE,11/ERROR.1/0
```

with the command following the parameter word.
the LENGTH of the command is in CM words and
the CODE is the character code in display code.

ERROR is the Front End error code and is returned to the caller.

2.3.2 FECMD CONTROL CARD

```
FECMD,COMMAND.
```

where COMMAND is the Front End command without the Front End command character.

2.3.3 FECMD FORTRAN FUNCTION

The FECMD function call format is as follows:

```
EC = FECMD(COMMAND[.CODE])
```

FECMD is a real function. The CODE is an optional 2 character connect type. If it is not specified. OM is assumed. The COMMAND is either a hollerith string or an array containing either display code or ASCII characters, terminated by either an end-of-line byte of binary zero or the ASCII equivalent of 4000B.

This function returns the Front End error code to the caller.

2.3.4 FECMD MACROS

FECMD user macros were installed in both SYSTEXT and CPUTEXT. The format of the call follows:

```
FECMD     PARAM
```

where PARAM is the address of the FER parameter word (see section 2.3.1)

2.4 SETCODE

SETCODE will change the character code of a file without modifying the file's connected/non-connected status. SETCODE is available as a control card callable routine. a FORTRAN callable function and a macro call (in both CPUTEXT and SYSTEXT).

2.4.1 CON CALLING SEQUENCE

CON was modified to accept a SETCODE parameter in its input register. Format of the input register is as follows:

```
18/3LCON.6/RCL.12/TYPE.5/S.1/FLAG.18/PARADDR
```

where RCL is an optional recall parameter,
TYPE is 0 if connect. non-zero if discont,
S is 1 if SETCODE function, else ignored,
FLAG is zero if code is to be ignored.
PARADD is the parameter word address.

Parameter word format:

```
42/FILENAME.6/0.11/CODE.1/CB
```

FILENAME is the local file name,
CODE is the 2 letter connect type.
CB is the complete bit.

2.4.2 SETCODE CONTROL CARD

The SETCODE control card syntax is as follows:

```
SETCODE.LFN=CC[.LFN1=CC...].
```

where LFN is the local file name and
CC is the character code.

2.4.3 SETCODE FORTRAN FUNCTION

```
SETCODE(LFN.2LCC)
```

where LFN is the local file name and
CC is the new character code.

If CC is omitted then Old Mistic is used.

2.4.4 SETCODE MACROS

SETCODE macros were added to both SYSTEXT and CPUTEXT. The format of the macro calls:

```
   SETCODE    LFN,CC,R
```

where LFN is the address of a parameter word containing the local file name (left justified).
CC is a 2 letter character code.
R is an optional recall parameter.

A character code may be preset in the parameter word. If CC is omitted, then the preset value is not destroyed. If there is no preset value and CC is omitted, the default connect type is used by CON.

2.5 SPOOLED INPUT/OUTPUT

2.5.1 READPT

```
READPT,LFN[.CC][,NR].
```

where LFN is the local file name.
CC is an optional character code.
NR is an optional no rewind parameter.

MANAGER will spool incoming lines to TTYTTY using CC as the character code. The CPU routine SPOOL will copy TTYTTY to the specified file, rewinding the file unless NR is specified.

2.5.2 TPREAD

```
TPREAD.LFN[.CC][,NR].
```

TPREAD is identical to READPT (section 2.5.1) except MANAGER issues Front End commands READER,ON at the beginning of the spooling process and READER,OFF at the end.

2.5.3 WRITEPT

```
WRITEPT,LFN[.CC][,NR].
```

where LFN is the local file name,
CC is an optional character code, and
NR is an optional no rewind parameter.

WRITEPT will copy the local file name to TTYTTY. If CC is specified. the character code of TTYTTY is changed to CC. MANAGER will then list TTYTTY to the user's terminal in that character code. If the NR parameter is specified, WRITEPT will not rewind the local file before copying it to TTYTTY.

2.6 MSO CALLING SEQUENCES

MSO is used to issue messages to a user's line. There are three calling sequences. Two are for MSG. The other is for CPU routines. The system routines SEND and MESSAGE are the primary users of the CPU calling sequence.

2.6.1 USER CALL

MSO calling sequence for a user call:

input register -

```
18/MSO.1/0,1/R,1/0.3/CP.12/N,2/0,4/B,18/ADD
```

where R is the recall parameter,
CP is the control point assignment,
N is the receiving line's user number,
B is bounce information for the refusable request.
ADD is the address of the parameter word.

parameter word layout -

```
10/0,1/REF,1/AS.1/REP.6/0.18/MADD.11/C,1/CB
```

where REF is the refusable request parameter,
AS is set if the message is in ASCII,
REP is set if the request was refusable and MSO could not send the message.
ADD is the address of the message,
C is the character count of the message,
and CB is the complete bit.

Sending a message to another user's line is restricted to system library routines. A user number of zero is interpreted as the caller's user number.

The refusable call allows system routines (SEND and MESSAGE) to be told (via the REP field) that a line was busy and the message could not be sent. Previously, the caller would swap out waiting output (WT.OUT) on the receiver's line. With this interactive system, MANAGER is told when a specific user's line is ready for more output and MANAGER frees that user if necessary. It does not scan the entire pool looking for lines which are now ready for more output and those waiting to be freed, as 1BR did.

Messages can be in either display code or ASCII. Display coded messages are packed 10 characters per CM word, while ASCII are 5 characters per CM word. The AS parameter specifies which character code is being used.

2.6.2 MSG CALL WITH RECALL

MSO calling sequence for a MSG call with recall:

input register -

```
18/MSO,1/1,2/0,3/CP,13/0,1/AR,4/0,18/PADD
```

where CP is the control point assignment,
AR is the MSG auto-recall flag (set).
PADD is the address of the parameter word.

parameter word layout -

```
12/0,18/MADD,29/0,1/CB
```

where MADD is the address of the message,
and CB is the complete bit.

2.6.3 MSG CALL WITHOUT RECALL

MSO calling sequence for a MSG call without recall:

input register -

```
18/MSO,1/1,2/0,3/CP,13/0,1/AR,4/0,18/MADD
```

where CP is the control point assignment,
AR is the auto-recall flag from MSG (zero)
MADD is the address of the message.

2.7 MSX CALLING SEQUENCES

MSX has two calling sequences. One is for normal errors, which is just specified by an error number (F.ERRXX) and/or a secondary error number. The other is for mode errors.

The secondary error number is used to select a particular I/O error message when the primary error flag is F.ERIO or E.6ES.

2.7.1 NORMAL ERRORS

Input register layout for normal errors -

```
18/3LMSX.1/1.2/0.3/CP.1/MD.5/0,12/SEC.6/ERR,12/USER
```

where CP is the control point assignment,
MD is the flag which distinguishes mode errors (clear),
SEC is the secondary error flag.
ERR is the primary error flag,
USER is the user number.

2.7.2 MODE ERRORS

Input register layout for mode errors -

```
18/3LMSX.1/1.2/0.3/CP.1/MD.5/MDERR,18/ADDR,12/USER
```

where CP is the control point assignment,
MD is the mode error flag (set),
MDERR is the mode of the error,
ADDR is the address of the mode error,
USER is the user number of the job.

3.0 SYSTEM PROGRAMMING CONSIDERATIONS

3.1 INSTALLATION

The installation of the new interactive subsystem was co-ordinated with the installation of the "7/32" FREND system and the associated "6500" support routines (1FP, CPCIO. CP4ES, CP2TT). All of the "6500" routines were installed in LSD 45.18 with bug fixes in LSD's 45.19 and 45.20. The LSD 45.18 system was designed to be as compatible as possible to the previous interactive system. In LSD 46.00. along with the official installation of the FREND system, minor modifications were made to install the final version of the system.

The batch printer subsystem was installed in LSD 48.05 and FREND version 02.00. This involved some modifications to MANAGER and MAN, a lot of new code in FREND, and smaller changes to CMR. IRCP. CPCIO. CP2TT. CP4ES. QDR. 6DP. FECMD, ARGUS, 1EJ. FNT, DISPOSE. and 1FP.

3.2 SYSTEM TABLES

3.2.1 LOW CORE

Pointers and counts referencing interactive input, output, and backup buffers were deleted from P.SHA, P.SHA1, and P.SHA2.

3.2.2 MANAGER TABLES

The input. output. and backup buffers were deleted. The special input and output characters which were passed to MMM (SOC. and SIC. symbols) were deleted.

The input and output buffer addresses were deleted from the USER table. FRONT END port and socket numbers were added.

For the front-end batch printer subsystem, the USER table was extended to include an entry for each printer. This added length is not reflected in the table length field in P.SHA1, so that only MANAGER knows these entries exist.

4.0 INTERNAL REFERENCE SPECIFICATIONS

The elements of the interactive subsystem are:

1. MANAGER is the interactive line manager. It is responsible for initiating and overseeing the activities and needs of the MISTIC jobs.

   It directs its PPU lackey MAN to set flags and manipulate system tables as needed.

   To determine the status of the terminal end of a MISTIC job. MANAGER communicates with the 7/32 system over a control port.

   Based on information from the 7/32 and from system tables, MANAGER is able to direct the interactive system to serve the needs of MISTIC jobs.

2. MAN is designed solely for use by MANAGER. the interactive system coordinator for MISTIC2 service. MAN performs those functions which require a PPU program because of manipulation of various system tables.

3. MSO is an interactive output routine left over from INTERCOM 1. It is called by CPU programs to move either ASCII or display code messages to an interactive line. It is also called by MSG to display messages.

4. MSX is an interactive error message PP routine left over from INTERCOM 1. It is called by others to issue error flag

(F.ERXXX) message, I/O messages (also issued by 6WM), and CP4ES error messages (also issued by 6DP).

5. Many utility and special function programs. LOGIN processes the user log in procedure. LOGOUT processes the user log out procedure. FECMD was written to allow Front End commands to be issues by a 6500 program. READPT, WRITEPT, and TPREAD provide a means for input and output spooling. SETCODE permits a file's character code to be changed.

The following outline the events which occur when a user logs in:

1. The 7/32 sends MANAGER an open protocol record, indicating a new user wants to log in.

2. MANAGER searches the user table for a free entry and initializes it.

3. MANAGER calls MAN to find and initialize a pool pocket for the new user.

4. MANAGER fakes a user command of LOGIN.

5. MANAGER puts the control card +LOGIN. in the user's swapped out control point area.

6. MAN is called in to start the user job.

7. LOGIN runs and its PP SPN updates the user table entry with the user authorization limits and identifications.

8. The user job is swapped out waiting command (WT.CMD). MANAGER's processing state for this user is now SERV, waiting for a user type in.

In the batch printer system, FREND is the supervisor, and MANAGER is the slave. FREND drives the printers, and MANAGER performs disk file processing tasks such as getting a file from ECS, returning a file to ECS, reading it, positioning it, finding the dayfile, etc.

The batch printers were integrated into MANAGER by creating a USER table entry for each one, as well as some auxiliary tables to contain information unique to the printers. The USER table was used because it already contained many fields pertinent to print jobs, and also because MANAGER is a state-driven program, using the USER table index throughout. In this way, printer states could be added, with only minor changes to MANAGER's state processing code.

The printer USER table entries are contained in an extension to the USER table known only to MANAGER, not to the rest of the system (the length of the table in P.SHA1 did not change.)

The major events in a print job are as follows:

1. When a printer is turned "ON". FREND sends a FP.OPEN protocol record to MANAGER. which causes table entries to be initialized for the printer.

2. When a printer is ready for a print file. FREND sends a FP.GETO protocol record to MANAGER. MANAGER then calls MAN to get an appropriate print file from ECS. When a file is found. MANAGER sends a FP.NEWPR record to FREND, and begins to read the file.

3. If the print file has a dayfile, MANAGER will automatically position the file to the beginning of the dayfile. This causes the dayfile to appear at the beginning of the print file. When EOI is reached. MANAGER automatically rewinds the file before reading again, and when the dayfile is read for the second time. MANAGER sends the FP.EOI record to FREND.

4. When FREND receives the FP.EOI from MANAGER, it finishes printing the file, then sends the FP.ACCT record to MANAGER, containing page and line counts for the job. This is a request for accounting. MANAGER computes the estimated charges. and sends this back in another FP.ACCT record. MANAGER also dayfiles the charge and gets rid of the print file.

4.1 MANAGER

MANAGER is a state driven routine. There are tables which specify the processing state of each user at any time. Each processing state has a serially-reentrant routine associated with it.

MANAGER runs at control point 1. Currently, it cannot be moved. The user table is kept within MANAGER's field length and its absolute address is kept in low core (P.SHA1).

4.1.1 TABLES

MANAGER has several internal tables to keep track of its users. There are pointers in MANAGER's low core to all of its tables. This helps in both debugging and crash analyzing. Also. the zeroth entry of STATUS. AUXSTAT. AUXSTAT2. and PRGCOM contain the name of the table in display code. Again, helping the programmer.

STATUS, AUXSTAT, and AUXSTAT2 are completely internal. Only MANAGER modifies these words.

PRGCOM is a program communications word. For example, through a SYS request a user may set a timed input request. SYS will interlock the PRGCOM word and set the timed input request in it. MANAGER will see the request. process it and free the PRGCOM interlock.

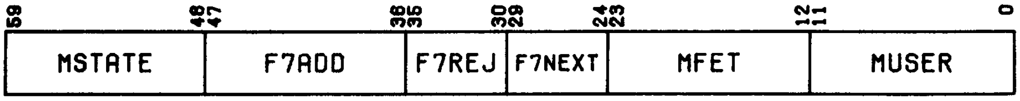
The mechanics of the PRGCOM interlock are: first. the PP routine gets the INTERCOM channel (CH.SHA) and checks the MGRACS flag in the PRGCOM word. If its set. MANAGER is working on a previous request. When the flag is clear, the PP sets its request and MGRACS in PRGCOM. It is now MANAGER's turn. After MANAGER is finished with the request. it will clear the request and the MGRACS bit.

The only exception to this interlock procedure. is SSS and the PRGACT flag. SSS will just clear this flag when the job is swapped out waiting command (WT.CMD). At this point. SSS is the only activity and an interlock is not necessary.

The user table is also within MANAGER's field length. P.SHA1 in low core contains the absolute address of this table. The table is initialized by LOGIN/SPN when the user first logs into the system. MANAGER copies the user's connect time to a separate word in the user table, where it is counted down. This is done to prevent both MANAGER and SPN from writing in the same word.

Several tables are used only for the batch printer system. These tables are used by MANAGER only, and contain flags. counts. values. etc., for the printers. Their names are PRSTAT. PRSTAT2, PRFILE. and DFWRD. These tables each contain one word for each printer. plus an extra word containing the table name in "H" format, for ease in reading dumps. With the USER number in B2, all these tables are indexed via the expression "BASE+B2+tablename."

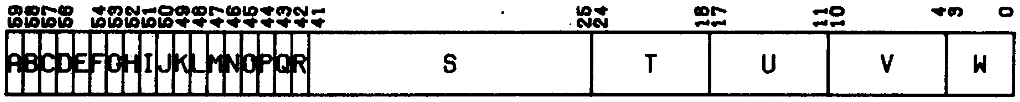
4.1.1.1 STATUS



59 48	47 36	35 30	29 24	23 12	11 0
MSTATE	F7ADD	F7REJ	F7NEXT	MFET	MUSER

MSTATE	users MANAGER state
F7ADD	relative address of 7/32 FET
F7REJ	reject state for a LISTMSG
F7NEXT	next state after a LISTMSG
MFET	relative address of TTYTTY FET
MUSER	relative address of user table entry

4.1.1.2 AUXSTAT



A - TPREAD set if in TPREAD mode

B - NUMBER set if in auto-numbering

C - TAPE set if in any tape mode

D - TAPEC set if in TAPEC

E - READPT set if in READPT mode

F - TTYDTF set if data on TTYTTY

G - LO set if in LOGOUT

H - LI set if in LOGIN

I - 2MW set if 2 minute warning given

J - 5MW set if 5 minute warning given

K - unused

L - TAPER set if commands read during TAPE

M - READY set if OK desired for response

N - SUSPN set when = used in auto-numbering

O - DISC set if user has been disconnected

P - unused

Q - MDISC set if MANAGER disconnected user

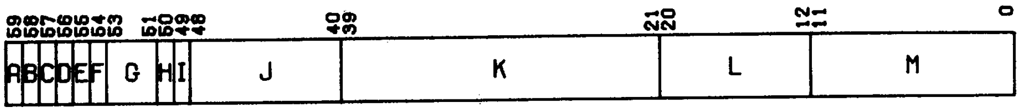
R - unused

S - NI integer portion of number line

T - NF fraction part of number line

U - NII	integer portion of increment
V - NIF	fraction part of increment
W - ABTC	abort count down

4.1.1.3 AUXSTAT2



A - TMOF set when timed input is done

B - CONN set when terminal is connected

C - HAVED set when 7/32 has user input

D - NEEDD set when 7/32 has space for output

E - unused

F - STAT set when 7/32 requests a JOBSTAT

G - TTYCCC TTYTTY character code

H - USABT set for user abort

I - FECRP set when FEC is valid

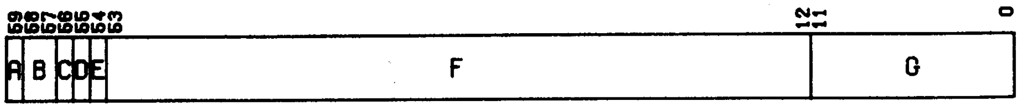
J - FEC F.e. Command error code

K - unused

L - PORT 7/32 port number

M - TMOC timed input count

4.1.1.4 PRGCOM



A -	MGRACS	MANAGER busy with PRGCOM
B -		unused
C -	PRGACT	set when user job is active
D -		unused
E -	SLO	set if LOGOUT is in progess
F -		unused
G -	PTMOC	timed input count

4.1.1.5 USER TABLE

0	USER ID										
1	PROBLEM NUMBER										W.USPAS
2	C.USFL. MAXIMUM FL	C.USTL. A	UNUSED	C.USFIL. B	D	E	PASWORD ORDINAL				W.USPAR. W.USLO
3	C.USRTL. RTL VALUE	UNUSED	F	H	I	J	K	S.USHPO. POOL ORDINAL	ACCUMULATED CP TIME	C.USPTPP. L	W.USTL. W.USCPT
4	RFL VALUE	C.USTT. M	UNUSED	CONNECT TIME FROM AF (LOGIN)							W.USFL. W.USTT
5	UNUSED	C.USSOCK. N	C.USPORT. PORT NUMBER								W.USSOCK. W.USP
6	LOGIN TIME										W.USLGTM
7	UNUSED	REMAINING CONNECT TIME									W.USCNTM
10	SEQUENCE NUMBER										W.USSEQ

A MAXIMUM CP TIME LIMIT
B FILE LIMIT
D S.USLO. LOGOUT BIT
E S.USLOCK. LOCK FLAG
F S.USLNHE. IF SET MEANS HERMES USER
H S.USLNTN. IF SET MEANS TELENET USER.

I S.USLNET. IF SET MEANS NETWORK USER
J S.USLNHS. IF SET THE LINE IS 1200 BAUD
K S.USLNFE. IF SET THE LINE IS AN FE LINE
L POOL POCKET ADDRESS
M STIMULATOR THINK TIME (SECONDS)
N ORIGINATING SOCKET (CC PORT IF MERIT)

4.1.1.6 PRSTAT

PRSTAT TABLE

59 42	41 24	23	22	21	20	19		11 0
DFPRU (18 BITS)	NPRU (18 BITS)	NEWCPY	NEWREC	DAYF	SKB	PREOI	UNUSED	FNTAD (12 BITS)

DFPRU	PRU position of dayfile record
NPRU	PRU count for backspace command
NEWCPY	1 if FREND needs another copy
NEWREC	1 if EOR just read on print file
DAYF	1 while printing the dayfile
SKB	1 if FREND requested a backspace
PREOI	1 if at end of print file
FNTAD	print file FNT address

4.1.1.7 PRSTAT2

PRSTAT2 TABLE

	37	20 19	5 4 3 2 1 0			
UNUSED	LINES (18 BITS)	PAGES (15 BITS)	FPACCT	ENDJ	OWNF	RG

LINES lines printed for this job

PAGES pages printed for this job

FPACCT FREND sent accounting command

ENDJ 1 if job ended by operator

OWNF 1 if user supplied own forms

RG rate group field for print job

4.1.1.8 PRFILE

This table contains the exact copy of the 3rd word of the print file FNT entry. It is kept to record the file size, dayfile-present flag, copies count, and print limit fields.

4.1.1.9 DFWRD

This table contains a copy of the second word of the dayfile record, for print files that have dayfiles. It is used as an aid in recognizing the dayfile record when it is read for the second time. The dayfile is actually recognized by its PRU position in the file, but keeping this word for comparision means that no file position check (FNTSTAT request) need be done for records which do not match this word.

4.1.2 MAIN LOOP

The main loop consists of processing each user, followed by control port processing, disk I/O checking, EXPORT processing, clock processing, operator requests (CFOs), and then a small recall loop.

Each user has a processing state associated with it (MSTATE). The individual processor is jumped to via a jump table at JUMPTAB. The state processors return to MAINLP (the top of the main loop).

4.1.2.1 CONTROL PORT PROCESSING

Subroutine CTLPORT is called from the main loop to process incoming control port messages. If the control port has been set complete, the 7/32 has died. A dayfile message is issued and MANAGER aborts.

Subroutine RDCTLP is called to read a control port message. The record type (FP.XXX), the port number and first parameter are isolated. The individual record type processor is called. They all return to CTLP where another message is read and processed. This continues until the control port is empty.

An FP.CLO record causes the disconnect flags to be set (MDISC and DISC).

An FP.OTBS record causes the NEEDD flag to be set if there are L.DTOT free buffers and NEEDD is cleared if there are not.

An FP.INBS record causes the HAVED flag to be set if there is user data in the 7/32 and cleared if there is not.

An FP.ABT record causes the USABT flag to be set.

An FP.OPEN record causes MANAGER to find an empty user table entry and initialize it. An open accept response is sent back to the 7/32 if MANAGER found an empty user table entry. An open reject is returned if the user table is full.

An FP.STAT record causes the STAT flag to be set.

An FP.FRCP record causes the FECRP (Front End command reply) flag and the FEC (Front End command error code) to be set.

An FP.CPOPN record is ignored. This is just a response from MANAGER's outgoing control port open. Since 1FP ensures the 7/32 is running and tells MANAGER when FREND has died, MANAGER does not need to keep track of these response records.

Several protocol types are used only for the front-end batch printers. These are processed exactly as are any others. Most result in various flags and fields being set in the printer tables, which in turn cause the printer states to do various things. The new record types are as follows.

FP.GETO is FREND's "get me a print file" command. The record contains the source code to use, and 2 PRU limits. These are passed to MAN in the function 7 call, and MAN does the real work of getting an appropriate print file. The printer type is also passed to MAN, so that a file of the correct disposition can be gotten, but this information came to MANAGER as the "OT.XXX" value in the FP.OPEN record for the printer.

FP.NEWPR is MANAGER's answer to FP.GETO. It may contain all zeros, telling FREND there was no print file, or it may contain the file name, size, copies count, print limit, and dayfile-present flag.

FP.ENDJ - this is sent from FREND to MANAGER, to tell MANAGER that a print file has been ended or killed by the operator, or that a file has hit page limit. MANAGER will stop reading the file, and return it to ECS, in the "R" queue.

FP.SKB is a command to backspace a print file, sent from FREND to MANAGER. It contains a count of prus to skip.

FP.ACCT is FREND's request for print file charges from MANAGER. It is sent to MANAGER with page and line counts; MANAGER sends it back with approximate charges and rate group. This record also causes MANAGER to dayfile the print charges.

FP.COPY is sent from FREND to MANAGER after EOI, if another copy of the file must be printed. It causes MANAGER to rewind the file, reposition to the dayfile if any, and begin reading.

4.1.2.2 DISK I/O PROCESSING

Subroutine CHKFETS is called from both the main loop and from CIOCALL to manage the TTYTTY FETs and requests.

First all of the active FETs are processed by subroutine CFET one at a time. If the FET is complete, the routine responsible for cleanup after the function is complete is called. The macro IOFUNC defines the initiating and terminating routines associated with each I/O function issued on a TTYTTY file.

Next, unassigned FETs are assigned to users needing a FET. FQUEUE contains the chain of users.

4.1.2.3 EXPORT PROCESSING

Subroutine CHKEXP2 is called from MANAGER's main loop to check on EXPORT activity. If EXPORT is up and running subroutine ALC is called to manage the EXPORT lines.

4.1.2.4 CLOCK AND COUNTERS

Subroutine CLOCK is called from the main loop. It is responsible for update various MANAGER counters.

If a second has not elapsed since that last call, CLOCK just exits.

Otherwise, CLOCK loops through each user table entry and decrements until zero the connect time, the abort count, the input timeout count.

Then CLOCK issues a control port open record (FP.CPOPN) every 30 seconds to inform FREND that MANAGER is alive and running.

4.1.2.5 CFO PROCESSING

Subroutine CFO is called from the main loop to check for operator requests. Currently, the only request is LOGOUT, i.e. start the idle down procedure and inform users that MISTIC service is terminated.

Under previous systems, any time MANAGER dropped out, MMM would inform users that service had ended. With the FREND system, a request from the 6500 must be sent to FREND to notify users that service had ended. In most cases, MANAGER was aborted and a valid dump of both MANAGER and the 7/32 is needed. Sending a service terminated message could invalidate the dump.

4.1.3 USER PROCESSING STATES

Each user has a processing state associated with it, so that MANAGER remembers what has been done and what needs to be done with the user's job.

A set of states waits for, parses, and processes a command line. Another set of states initiates the user job from the parsed command line. A single state is responsible for overseeing the needs of the running job. A set of states then lists TTYTTY to the terminal. Then MANAGER goes back to the states which wait for and process command lines.

4.1.3.1 SERV, WAITIN, DOSERV, AND STASH

This set of states accepts a command line from the user's terminal, parses it and processes it up to the point of, but not including, initiating the user job.

State SERV waits for a command line. It processes disconnects, user aborts, and status requests. If the user is in either TAPE mode or auto-numbering, the user abort indicates the end of that mode. If user input is present in the 7/32, SERV starts a read and goes to state WAITIN to wait for the read to complete.

State WAITIN waits for the read to complete. If the user is in either TAPE mode or in auto-numbering, any end-of-file records are changed into the display code equivalent strings *EOR, *EOF, *EORN, or *EORNN (where N and NN are the level numbers).

State DOSERV parses the command line into spooled input, EDITOR text lines, EDITOR commands, NUCLEUS library commands, system commands, and unrecognized garbage.

DOSERV puts the commands in the user's swapped out ECS control point area.

State STASH puts the spooled input and text lines on the user's TTYTTY file, buffering it through ECS.

4.1.3.2 MGRCC, NEXTC, COMR2, COMR3, GOJOB, AND CMDCHK

This set of states initiates an interactive job and processes MANAGER directives.

State MGRCC is entered whenever a MISTIC control card is found in the control card buffer or from state NEXTC after a MANAGER directive was processed. MGRCC checks the next control card for a MANAGER command. If it needs to be processed by MANAGER, the correct processor is called. All processors exit to state NEXTC. If it is not a MANAGER directive, MGRCC exits to state COMR2.

State NEXTC is the common exit for all MANAGER command processors. It checks the user's control card buffer for more control cards. If more are present, it exits to MGRCC, else to state RMSG to issue the OK or READY message.

State COMR2 is the first step in initiating the user job. It checks for data on TTYTTY. If there is none, it exits to state GOJOB, else to state COMR3.

State COMR3 just rewinds the TTYTTY file and exits to state GOJOB.

State GOJOB waits for TTYTTY to become complete. It then calls MAN to start the user job. It exits to state CMDCHK.

CMDCHK waits for a reply from MAN. MAN will detect file limit and time limit and not start the user job. CMDCHK exits to state ACTIVE if there are no errors. Otherwise, it issues the time limit or file limit message and exits to state RMSG to put out the OK or READY message.

4.1.3.3 ACTIVE

In state ACTIVE, the user's job is running. MANAGER processes PRGCOM requests, user aborts, disconnects, and status requests (JOBSTAT). If the job is in a wait state (WT.XX), MANAGER is responsible for freeing the job when the wait condition is no longer true.

If the job is waiting input (WT.IN), MANAGER waits for the HAVED flag to be set and then calls MAN in to free the job.

If the job is waiting output (WT.OUT), MANAGER waits for the NEEDD flag to be set and then calls MAN in to free the job.

If the job is waiting Front End command (WT.FEC), MANAGER waits for the FECRP flag to be set and for the job to completely swap out. It moves the error code to the swapped out control point area and calls MAN in to free the job. MSO issues a Front End command and does not swap the job out waiting command. To reduce overhead, if MANAGER has a command reply from the 7/32 and the job is swapped out not waiting Front End command, MANAGER will clear the reply from its tables.

4.1.3.4 LIST, PRELST, LISTTY, AND ENDLST

These states are responsible for listing the user's TTYTTY file to the terminal.

State ACTIVE exits to state LIST after the user has swapped out waiting command (WT.CMD). LIST merely rewinds TTYTTY.

State PRELST waits for the rewind to complete. PRELST exits to state MGRCC if there is no data on TTYTTY. Otherwise, PRELST sets the correct character code in TTYTTY and exits to state LISTTY.

In state LISTTY, the actual spooling occurs. The TTYTTY file is spooled through ECS to the user's terminal. LISTTY exits to ENDLST when TTYTTY has been listed, a disconnect has occurred, or a user abort was entered.

State ENDLST cleans up after the listing process. The character code of TTYTTY is reset to old Mistic, the ECS buffer pointers are cleared, and TTYTTY is evicted. ENDLST then exits to state MGRCC.

4.1.4 PRINTER PROCESSING STATES

MANAGER processes the state for each printer, on each pass through the main loop, after all interactive users are processed.

The following sections describe the printer states, as nearly as possible in the order of flow in a print job. The complete set of states is a cycle, beginning with an idle printer, going through print file retrieval, dayfile processing, normal printing, and end-of-job processing.

4.1.4.1 PRINTER TURNED OFF

When a printer is turned off, it is in state "NULL," or zero. There is no processor for this state, and the main loop simply skips any printer (or interactive user, for that matter) whose state is zero.

When FREND sends a FP.OPEN on a printer socket, the printer is turned on, and MANAGER puts it in state IDLEPR.

4.1.4.2 IDLE PRINTER STATES

IDLEPR This is the state for a printer between jobs, waiting for a job to print. IDLEPR waits for the DISC flag to be set, indicating that a FP.CLO was sent by FREND, turning the printer off. When this happens, this state clears out the printer tables, and goes to state NULL.

GETO A FP.GETO protocol record from FREND in state IDLEPR sends the printer to this state. GETO makes a MAN call to request a print file, then goes to state WAITPR.

WAITPR WAITPR waits for the MAN call to be complete, then checks the return. It will be either that no file was found, or a file was found, and its FNT image is in the MAN call parameter block. WAITPR sends the FP.NEWPR to FREND, with or without print file information. (FREND needs a response to the FP.GETO, even if there is no file to print). If no file is found, WAITPR goes back to state IDLEPR. If a file was found, it goes to state PREDF1, the first dayfile positioning state. Note that PREDF1 will check for a dayfile present before positioning.

4.1.4.3 DAYFILE POSITIONING STATES

These four states are responsible for positioning the print file at the beginning of the dayfile record (always the last record in the file.) the first of them executes at the beginning of every print job, and at the beginning of each subsequent copy, and it decides to do the positioning only if the file has a dayfile (the dayfile-present bit is set in the output FNT.)

These states ensure that the dayfile appears at the beginning of the print on front-end printers. The positioning sequence is: skip to end-of-information, backspace one record, and make an FNTSTAT call to record the file position.

PREDF1 This state sets the need-data flag (NEEDD), which causes reading to begin after the dayfile is positioned. It sets the new-record flag (NEWREC), which will trigger the disk read routines to remember the second word of the dayfile record for later comparison. Then it checks to see if the file has a dayfile, and goes straight to state PRINT if there is none. If there is a dayfile record, this state begins a SKIPEI operation, and goes to state PREDF2.

PREDF2 This state waits for the SKIPEI that was begun by PREDF1 to be complete, then starts a SKIPB of one record, and goes to state PREDF3.

PREDF3 This state waits for PREDF2's SKIPB to complete, then issues an FNTSTAT call, to get the position of the dayfile record in the file.

PREDF4 This state waits for PREDF3's FNTSTAT call to complete, then records the PRU position of the beginning of the dayfile record in the DFPRU field of the PRSTAT table. The position is remembered, so that the dayfile can be recognized the second time it is read. Then, MANAGER will simulate EOI on the print file, to avoid reprinting the dayfile at the end of the job.

4.1.4.4 NORMAL PRINTING STATES

These states execute in a cycle, starting with the setting of the need-data (NEEDD) flag by the receipt of the FP.OTBS record from FREND. When NEEDD is set, MANAGER begins a read on the print file, and transfers a block of data to FREND.

PRINT This is the state most usually occupied by a print job during printing. It is the idle state between blocks of print file data sent to FREND. PRINT monitors the NEEDD flag, which means that FREND has room in the printer port for a buffer-full of data from the print file. When NEEDD is set, PRINT reserves a pair of FETs and a buffer for the transfer, fills in the file names and the disk file FNT address, and begins a READ on the disk file. The next state is BL2FE.

PRINT performs other functions besides checking the NEEDD flag and initiating disk reads. For descriptions of these other functions, see the following sections on end of print job processing, file skipping, and abnormal terminations.

BL2FE This state just calls subroutine WTBLFE, which does some complicated processing. What it needs to do is wait for the disk READ to complete, and begin a WRITE. WRITER or WRITEF to flush the buffer to the connected file (to the printer port in FREND). But it also has to check for EOI, and it must recognize when the dayfile record is being read for the second time, and treat this as the end of the print file.

At the disk file EOI on a print job with no dayfile, BL2FE releases the FETs and buffer, and sets printer state PREOI. This will cause the FP.EOI record to be sent to FREND, signalling end-of-job, or at least end of this copy of it.

At EOI on a job with a dayfile, this is really not the end of the job, but only the end of the dayfile, which has been printed at the beginning of the job. WTBLFE rewinds the print file, and goes back to state PRINT (via ENDDSK, which waits for the rewind to complete), with the NEEDD flag still set. Thus a new disk read will be started immediately, without having to wait for another FP.OTBS from FREND.

At the beginning of each record (NEWREC = 1), WTBLFE calls subroutine CK4DAYF, which does the processing necessary to recognize the dayfile when it is recognized the second time. This subroutine saves the image of the 2nd word of the dayfile record on the first pass through, then compares the 2nd word of each subsequent record with it. When a match is found, CK4DAYF begins a FNTSTAT call, to get the current position of the print file, then changes state to CKDAYF.

CKDAYF This state waits for the FNTSTAT call started by CK4DAYF to complete, then compares the current file position with the known position of the dayfile record, as remembered by state PREDF4. If the position is the same, then we are reading the dayfile record for the second time this copy, and we must end the print job here. CKDAYF then releases the FETs and buffer, and goes to state PREOI, which will send the FP.EOI record to FREND, to signal the end of the job (or copy). If the position does not match, CKDAYF simply returns to state BL2FE, which will send the record in the buffer to FREND.

ENDBL This state is entered from WTBLFE (BL2FE) after the write to the connected file is begun. It simply waits for the write to be complete, releases the FETs and buffer, and returns to state PRINT. One problem that can arise here is that 1FP will refuse to do the write because of a shortage of buffers in FREND. In this case the FET is set complete, but the buffer is still full of data. ENDBL will then wait, without changing states or releasing the FETs, until the NEEDD flag is set again, which will occur when FREND has cleared out enough of its memory to make more buffers for output. Then ENDBL reissues the write, waits for it to complete, and so on.

4.1.4.5 END OF PRINT JOB STATES

At the end of a print job, certain things must happen: FREND must be told there is no more data, we must decide whether to print another copy, and if not, we must do accounting and close the print file. These states accomplish all this.

PREOI This state is entered upon reaching EOI on the print file, either real (if there is no dayfile) or simulated (when the dayfile is read for the second time). It sends the FP.EOI protocol record to FREND, which tells FREND to expect no more data, clears the NEEDD flag, sets the PREOI flag, and goes to state PRINT.

PRINT This state behaves differently when the PREOI flag is set. The EOI condition allows PRINT to check for the receipt of the FP.ACCT protocol record from FREND, which signals that the last copy has been printed, and final print job accounting should be done. PREOI also makes PRINT sensitive to the NEWCPY flag, which is set by the FP.COPY protocol from FREND. It tells MANAGER to rewind the print file and transmit it again for another copy.

When the FP.ACCT protocol record is received, state PRINT sends a FP.ACCT back to FREND with the estimated print charge, so that FREND can print it on the listing (subroutine SENDACCT does this). Then subroutine PRTDAYF is called, to dayfile the accounting message for the job.

After the accounting, PRINT returns the print file (subroutine RETURNPR) and resets all tables for this job (RESETPR). This leaves the printer "ON," in state IDLEPR, waiting for a request for a new file to print.

If FP.COPY is received (instead of FP.ACCT), PRINT begins a REWIND on the print file, and goes to state COPY.

COPY This state waits for the completion of the REWIND that PRINT started. Then it changes state to PREDF1, to begin transmission of the print file for a new copy.

4.1.4.6 PRINT FILE SKIPPING

Forward skipping (the %SKIP front-end command) is done entirely by FREND, with no special processing in MANAGER. It is done by throwing away lines until the target PRU is reached. Note that it is FREND, not MANAGER, that counts PRUs in a print file and keeps track of the current position.

Backward skipping (the %BKSP command) is done by MANAGER, because there is a CIO function to skip a file backward a number of prus. This is done when the FP.SKB protocol record is received in state PRINT.

PRINT State PRINT checks the SKB flag (set by receipt of the FP.SKB protocol) at all times, unless the printer has been rewound or accounting has been requested by FREND. When the SKB flag is set, PRINT starts a backspace operation on the print file, for the requested number of prus. PRINT then goes to state ENDDSK to wait for completion.

ENDDSK This is a general purpose state, which just waits for any operation on the print file to complete, releases the FETs and buffer, and goes back to state PRINT.

4.1.4.7 ABNORMAL PRINT JOB TERMINATIONS

There are 3 ways by which a print job can terminate abnormally:

1. The printer may be rewound.

2. The print job may be ended or killed.

3. The print job may reach page limit.

PRINT This state checks the DISC flag. DISC is set when a FP.CLO is received from FREND. This means either that the printer was turned off (and no file was printing) or that the job currently printing was rewound. When PRINT sees DISC = 1, it makes a MAN call to put the print file back in ECS (without changing queues), then clears all tables for that printer. The printer is then in the NULL state.

PRINT also checks the ENDJ flag, which is set by receipt of the FP.ENDJ protocol record from FREND. This record indicates that the print job has been abnormally ended, either by hitting print limit, or by operator %END or %KILL. When ENDJ is set, PRINT recognizes the accounting request (FP.ACCT) as if the file were at EOI. After accounting is done, PRINT again checks the ENDJ flag, and if set, calls MAN to return the print file to the "R" queue, instead of unloading the file.

4.1.5 PRINTER FILE HANDLING

4.1.5.1 DISK FILE FNTS

While FREND is printing a job, there are always 2 FNT entries for the job in CM. One is the FNT as it came from ECS, and it is kept in CM for recovery purposes. It is never used. The other FNT is a scratchpad created by MAN function 7, which is used for all I/O on the print file.

The reason this is done is that disk I/O destroys several fields in the output FNT, which must be preserved if the system or MANAGER crashes. These fields are the PN and PN ordinal, print limit, file size, etc. IRCP and MAN function 6 (MISTIC2 idledown) have both been modified to recognize dual-FNT print files at MANAGER's control point. They will zero out the local scratchpad FNT, and put the output FNT back in ECS.

4.1.5.2 PRINTER FET AND BUFFER MANAGEMENT

MANAGER has a number of "block-transfer FETs" which are meant to be shared between processes which transfer data in a blocked fashion between disk and the front-end. The number of them is determined by the symbol N.BLFET, which is currently set to 1. Only the batch printers use them, but they may someday be used for E/I200 data, TTYTTY output, and a file transfer process.

Each "FET" is really a pair of FETs, one for a disk file and one for a connected file, with a shared 6-PRU buffer. The 6-PRU size was chosen for the batch printers, because it fits evenly into an RB, and nearly fills a front-end port, and so optimizes disk transfer efficiency for these files.

A block-transfer FET may be reserved to a "user" by calling subroutine GETBLFET with the user number in B2. If a FET is free, GETBLFET places the user number in the FET, and places the FET address in the F7ADD field of the STATUS table for that user. A FET may be released by calling RELBLFET.

At initialization time, MANAGER connects a file for each printer. The file name is derived from the USER number of the printer. Each time a FET is reserved, the two file names, the printer port number and the disk file FNT address must be placed into the FET. This is done for printers by subroutine INIBLFET.

4.1.6 ECS USAGE

MANAGER has its own ECS partition, which it uses to hold buffers for each user. Each buffer is 2 prus in size, and is used to hold text lines for spooling to or from TTYTTY.

When in command mode, all numbered text lines are written to the user's ECS buffer, which is periodically written to the TTYNNN file (for user NNN). This then becomes the user's TTYTTY file when the job is swapped in. When in TPREAD, READPT, or tape mode, all input lines are written directly to the user's ECS buffer, which is then periodically flushed to TTYNNN.

When listing TTYTTY, the proper TTYNNN file (for user NNN) is read into MANAGER and written directly to the user's ECS buffer. Then it is read from ECS in small chunks of about 20 words, and written to the 7/32 via 1FP and CIO.

This ECS access is coordinated by the TTYTTY support routines in MANAGER.

MANAGER also accesses the job's control point area, which is kept in ECS when the job is swapped out. When a line of commands has been entered, MANAGER parses them and writes them directly to the user's control card buffer in the swapped out control point area.

4.1.7 COMMUNICATION WITH FREND

All communication between MANAGER and the 7/32 is handled by 1FP. MANAGER communicates with the 7/32 through 2 distinct interfaces: the control ports and standard CIO requests.

4.1.7.1 CONTROL PORTS

Control ports are special files which carry continuous control information between MANAGER and the 7/32. As long as these control port FETs are set busy, 1FP continuously transfers information over them between MANAGER and the 7/32. The MANAGER control port FETs are pointed to by word W.CPFE in the control point area. There are 2 FETs, one for information from MANAGER to the 7/32, and one for information from the 7/32 to MANAGER. MANAGER is control port 1 (PTN.MAN), which corresponds to port 1 in the 7/32. Any information entered by MANAGER into its FET for transfer to the 7/32 is automatically written to port 1 in the 7/32, where task CTLPT is started to process it. Any information which the 7/32 writes to its port 1 is automatically transferred to MANAGER. This automatic transfer occurs as long as both FETs are busy, and 1FP believes that the 7/32 is alive.

Data transferred over control ports is called PROTOCOL. A protocol record generally is very short, and is of the form:

8/BC, 8/TYPE, 8/0, 8/0, 8/PT, 8/P1 ...

BC = byte count

TYPE = protocol record type

PT = port number to which the record applies

P1 ... = parameters

All incoming protocol records are processed by the MANAGER subroutine CTLPT. The major records processed are:

FP.OPEN - sent by the 7/32 when a user dials in.

FP.CLO - sent by the 7/32 when a user hangs up.

FP.INBS - indicates the number of lines of user input on the 7/32.

FP.OTBS - indicates the number of lines of user output to be sent on the 7/32.

Outgoing protocol records are written by subroutine WTCLPT. The major records sent to the 7/32 are:

FP.ORSP - open response, indicating successful open.

FP.CLO - requests the 7/32 to hang up the user.

FP.TIME - sets the time/date in the 7/32.

FP.CPOPN - tells the 7/32 that MANAGER is alive.

FP.INBS - requests input buffer status from the 7/32.

FP.OTBS - requests output buffer status from the 7/32.

A full description of all protocol records is found in FESYM. ARGUS and the stimulator are also control ports, and communicate with the 7/32 in a similar manner.

4.1.7.2 DATA TRANSFER

All data transfer between the 7/32 and MANAGER is done using standard CIO requests. These requests cause 1FP to transfer data between FETs and buffers in MANAGER, and ports on the 7/32. MANAGER has 4 FETs and associated buffers which it uses to read from or write to the 7/32. These are shared between all users on a first-come/first-served basis. Once an I/O operation is completed, the FET is released for use by another user.

4.1.7.2.1 INPUT

MANAGER reads a line of input from the 7/32 when an FP.INBS protocol record tells him that the user has data in the 7/32. An FET is reserved, and a CIO request is made for a native format read (IO.RDNF). This is similar to a standard CIO read but it also returns the record header from the 7/32. (see the description in CP2TT and FESYM). 1FP transfers the data from the indicated 7/32 port to the buffer in MANAGER. The data is then processed as a command or a text record.

4.1.7.2.2 OUTPUT

For single line messages, MANAGER writes data to the 7/32 using a native format write. An FET is reserved, and an IO.WRTNF CIO request is issued. This causes 1FP to transfer the data from MANAGER to the associated data port on the 7/32. If the data will not fit in the 7/32, 1FP completes the FET, but does not transfer any data. MANAGER immediately releases the FET, and waits until the 7/32 sends an FP.OTBS indicating that there is room in the 7/32. Then MANAGER regenerates the message and repeats the above steps.

When listing TTYTTY, MANAGER writes data to the 7/32 in 20 word blocks, using a standard CIO write. Hence, partial lines are often written.

This saves MANAGER the chore of locating the end of a line (note that 4 different file types are supported) and then transferring the data a line at a time.

4.1.7.2.3 PRINTER OUTPUT

All output to the batch printers is done using CIO codes for connected block write. The codes are:

IO.FWTBL	front-end block WRITE
IO.FWBER	front-end block WRITER
IO.FWBEF	front-end block WRITEF

To the 6000 CPU routine, the rules for these transfers are practically identical to those for IO.WRITE, IO.WRITR and IO.WRITF on a disk file. For a write, data from the buffer is written to the front-end in PRUs, until there is less than one PRU left in the buffer.

1FP transmits the data to FREND verbatim, without any translation or EOL-byte checking. The data from the CM buffer is packed into 240-character 7/32 buffers (24 CM words to a buffer, in 6-bit mode), filling each 7/32 buffer until the data remaining to transfer won't fill 240 characters. This last buffer then is short. IO.FWBER and IO.FWBEF requests generate FP.EOR and FP.EOF protocols at the end of the data in the port.

1FP can write 12 CM words to a 7/32 buffer in 8-bit mode, if the character set in the FET is not OM. However this mode is not used for the printers, since the character set of a print file is not known until FREND determines it on a line-by-line basis. Therefore, so as to avoid losing bits from OM data, all transfers are done in 6-bit mode. ASCII data, which is packed 8-in-12, must then be repacked into 7/32 bytes by FREND.

4.1.8 EXPORT PROCESSING

All EXPORT related subroutines and tables are assembled in MANAGER under the qualifier EXPORT.

Subroutine CHKEXP2 is called from MANAGER's main loop to check on EXPORT activity. If EXPORT is up and running subroutine ALC is called to manage the EXPORT lines.

If EXPORT is not up. CHKEXP2 will read up the EXPORT on/off flag from the installation area (W.INSAF). CHKEXP2 will only read this flag once a second to keep overhead down. If EXPORT is to be brought up. subroutine ALCINIT is called to initialize and then subroutine ALC is called to manage the lines.

4.1.9 INITIALIZATION

There are 3 separate initialization sequences which must be performed: MANAGER. Front End. and EXPORT.

The initialization code is overlaid. The 7/32 initialization code is assembled in a use block called LASTLAST which will be returned to the system after initialization has completed or become part of the EXPORT buffers.

The initialization control code is assembled in the TTYTTY file buffers.

4.1.9.1 MANAGER INITIALIZATION

Currently, MANAGER must only read up the ECS partition word (W.ECMGR) for the MANAGER partition. This is used to ensure all ECS references are within the partition before issuing the ECS read/write request.

4.1.9.2 FRONT END INITIALIZATION

When called without any parameters. MANAGER will always automatically load and start the 7/32. The 7/32 is loaded with the current production version of the FREND system.

For each LSD. there is a permanent file FREND-LSD XXYY. where XXYY is the current LSD number. This file contains the name of another permanent file, the one containing the Front End system which should be loaded into the 7/32. The name of this file will always be FREND-VER NNNN, where NNNN is the version number. MANAGER will attach this file. and verify that it is infact the proper version. (the 4 digit version number is in the 1ST 4 digits of word 8 of the prefix table)

The correct permanent file is loaded into the 7/32 by simply reading it into MANAGER and writing it to the 7/32 using 1FP and the IO.FWTM CIO request.

If the NOLOAD parameter was specified, MANAGER will not attempt to load the 7/32.

If the PFN= parameter was specified, MANAGER will load the 7/32 with the FREND system on the specified permanent file.

MAN is called in to initialize low core and MANAGER control point area pointers.

MANAGER's control ports and 7/32 scratch files are initialized. The correct date and time are sent to the 7/32.

After FREND is running, MANAGER calls subroutine FEBM (in comdeck FEBM) to initialize the banner message. This subroutine attaches a permanent file containing the message, checks its expiration date. and writes the message directly into 7/32 memory, in the banner message table in the LMBI.

Banner message processing, including comdeck FEBM, is fully described in 6SM 94.1.

4.1.9.3 EXPORT INITIALIZATION

The initial IOD parameter word is set up. The XJ parameter word to read the EXPORT flags (W.INSAF) is set up for subroutine CHKEXP2. A field length request is made to reduce the field length to the run time field length.

4.1.10 DEBUG OPTIONS

There are three MANAGER debug options. TRACE. TRAP, and BID.

4.1.10.1 TRACE

TRACE dayfiles trace messages for selected users. A trace message is in the form:

XX MESSAGE

where XX is decimal user number.

Trace messages are issued by the TRACE macro.

Currently, trace messages can be issued at the exit to every new processing state, as each incoming protocol record is processed, and requesting and releasing a 7/32 FET.

Trace messages for user N are reouested by setting bit N of word RA+3 of MANAGER's field length. Trace can be operated during production, but issuing dayfile messages does slow MANAGER down.

TRACE is controlled by the TRACE$ assembly option which is currently assembled on.

4.1.10.2 TRAP

TRAP is a user requested breakpoint which is meant to be used in conjunction with DIS when debugging MANAGER at the console. The TRAP macro plants a trap. The BKP. switch controls whether the trap's default is on or off.

TRAP is under control of the IFTRAP assembly option, which is currently assembled off.

4.1.10.3 BID

The BID$ assembly option invokes code which allows MANAGER to be debugged interactively using the BID package. This option is currently assembled off.

4.2 MAN

4.2.1 OVERALL OPERATION

When MANAGER starts, it calls MAN function 5 (initialization). This function will set up low core and the control point area of MANAGER as needed. It also starts MAN function 6 in the delay stack on a 5 second recall.

While MANAGER is running, MAN function 6 bounces every 5 seconds. Its sole function is to control MISTIC2 idledown. When MANAGER is dropped or aborted, MAN function 6 initiates MISTIC2 idledown. and then ensures that MANAGER remains at a control point until all MISTIC2 jobs are cleared out of the system and all front end output files are returned to ECS.

MANAGER calls MAN function 7 when it wants an output job to send to the 7/32 for printing.

If MANAGER needs to evict an output file or to return an output file to ECS, it calls MAN function 8.

When a new user logs in, MAN function 2 is called by MANAGER to create a pool table pocket entry for the new job.

In the normal course of running an interactive job, MANAGER calls upon functions 1, 2, and 4. Function 1 starts a job executing a set of control cards which the user has entered. These have been moved to the job's ECS resident control point area by MANAGER.

Function 2 will abort an executing user job by setting the appropriate error flag in the pool table or job control point area.

Function 4 is called by MANAGER to wake up a job which is swapped out waiting for input, output, or a Front End command reply. MANAGER is informed by the 7/32 that the terminal has input, needs output, or has just processed a Front End command. If the job is waiting for one of these conditions, function 4 is called, which clears the wait state in the pool table pocket, thus restarting the job.

4.2.2 FUNCTION 1 - START UP A JOB

Function 1 initiates a user's control card record.

The first control card is read from the user's swapped out control point area in ECS. Function 1 determines if the control card needs special processing, either a 1AJ internal function or EDITOR which gets a higher job weight. Function 1 sets the initial swap-in field length, based on the control card from ECS and what libraries the user has defined in the control point area.

Checks are made to ensure the job has not exceeded its file or time limits.

If no errors were found, then the job's TTYTTY file is moved from MANAGER's control point to the user's. The job's pool pocket is set up for execution.

Note that the ECS and user table buffers for function 1 overlay the rest of MAN's functions.

4.2.3 FUNCTION 2 - ABORT A JOB

Function 2 is given a user number and an error flag. From the user number, MAN finds the job's pool pocket.

If the job is swapped out waiting command (WT.CMD), MAN sets the error flag in the pool pocket and sets the swap-in field length to 200B. The small field length will provide a quicker abort.

If the job is swappped in, MAN sets the control point error flag.

If the job is swapping, MAN delays and checks the swap state again until the job is either swapped in or out.

4.2.4 FUNCTION 3 - CREATE A POOL POCKET

Function 3 finds an empty pool pocket for a new user.

It gets a pool id from MTR (M.SEQ). Function 3 then scans the pool ensuring this user now has a unique pool id and user number. If the pool id is not unique, MAN just requests another id and repeats the scan.

If the user number is not unique, MAN returns an error to MANAGER.

Subroutine EIPOOL is called to setup the pool pocket. Subroutine UPUTBL updates the user table with the pool pocket ordinal. Subroutine F3INST is called to process any instrumentation.

4.2.5 FUNCTION 4 - RESTART A JOB

Function 4 will free a job which has been waiting output (WT.OUT), input (WT.IN), or Front End command (WT.FEC).

Subroutine FREEJOB is called to free the job. An error is returned to MANAGER if the job was not waiting. Subroutine INST is called to process instrumentation.

4.2.6 FUNCTION 5 - INITIALIZE MANAGER

Function 5 first checks if MANAGER's RA is control point zero's FL, i.e. there is no unassigned storage between control point zero and MANAGER. If there is, function 5 will abort MANAGER. Otherwise, MANAGER's control point could be moved. The pointer in low core to the user table is an absolute address.

Function 5 sets the absolute address of the user table in low core (P.INT1), the number of interactive users (also in P.INT1). The MGROK and SSSRPY flags are set in P.INT.

MANAGER's job name is changed to MANAGER.

MAN function 6 is placed on the delay stack with a 5 second delay and MANAGER's original RA in its input register.

4.2.7 FUNCTION 6 - MISTIC2 IDLE DOWN

Function 6 remains in the delay stack at MANAGER's control point as long as MANAGER is running and is at its original RA.

If either an error flag is set or MANAGER has ended, function 6 begins the idle down procedure.

First, the MGROK flag is cleared in P.INT. This signals 1RA to abort all running interactive jobs. When all have disappeared, 1RA clears the SSSRPY flag (in P.INT).

Now that all MISTIC2 jobs are logged out, MAN function 6 will not be recalled again. Now the function turns its attention to clearing out front end output files. It begins by searching the CM FNT for output files at MANAGER's control point. When one is found, function 6 examines it to see if it could be a front end output file.

Function 6 starts the examination by calling subroutine FNMATCH to search the CM FNT for another entry with the same name, control point, and RBT pointer as the output FNT entry in question. If a matching FNT entry is found, function 6 verifies that it has a file type of local (since a front end output file must have a local or scratch-pad FNT entry and an output FNT entry). If no matching FNT entry is found, or if the match is of the wrong file type, function 6 assumes that the output FNT doesn't belong to a front end output file, and it continues to search the CM FNT for other output FNT entries at MANAGER's control point.

When function 6 is satisfied that it has found the two FNT entries for a front end output file, it tries to set the file busy by setting the complete bit in the local FNT entry. If the file is already busy, function 6 checks the PP activity count at MANAGER's control point; if it is the only PP active, it ignores the busy and seizes the file. If other PPs are still running at the control point, function 6 forgets about the FNT entries it has found and begins its search of the CM FNT over again. (Under normal circumstances, MAN function 8 will be the only PP which could busy a front end output file. When function 8 finishes with the file, it will delete its FNT entries from CM; function 6 will then be able to continue.)

If function 6 is able to busy the file, it calls subroutine REQUEUE to return the output FNT entry to ECS. Then it calls subroutine DELFNT to delete both FNT entries from CM. Function 6 then continues to search the CM FNT for more output FNT entries.

When function 6 has searched the entire CM FNT, it clears P.INT1 in low core and W.CPFE in MANAGER's control point area. Then it drops out.

MANAGER's control card record is set up to exit to a DUMPFE control card which will dump the 7/32 whenever MANAGER aborts.

MANAGER's original RA is kept in function 6's input register. If MANAGER's control point has moved, then routines which look at or modify the user table are getting or clobbering other parts of core. Function 6 will adjust the absolute address of the user table in low core and then abort MANAGER.

MANAGER's control point will only move if an EDITLIB or a PP interpret is being done. These should only happen during system's time and not during production.

4.2.8 FUNCTION 7 - GET AN OUTPUT JOB

Function 7 is part of the front-end batch print subsystem. MANAGER calls it whenever it wants to print a job. In the call, MANAGER passes flags specifying which printers the desired file can be printed on; a routing character which identifies the source of the desired file; and primary and secondary PRU limits.

First, function 7 calls subroutine CALLISP with the primary PRU limit as a parameter. CALLISP formats a call to CP.LISP function GETO (unless the PRU limit is infinite, in which case function GET is used) to search the ECS FNT entries for a desirable file; that is, one that satisfies MANAGER's specifications. If no file which is smaller than the primary PRU limit can be found, function 7 calls CALLISP again with the secondary PRU limit. If no file smaller than the secondaary PRU limit can be found, function 7 tells MANAGER that no file is available and drops out.

If CALLISP found a suitable file, function 7 massages the FNT entry. First it changes the control point field so that the file will be assigned to MANAGER's control point. Next it sets the "nextjob" bit, so that if the file is returned to ECS before it has finished printing, it will be one of the first jobs printed when printing resumes. Then it copies the FNT entry twice: once into the CM FNT, and once into MANAGER's field length. These copies are referred to as "verbatim" FNT entries.

Next function 7 creates a "scratch-pad", or "local", FNT entry for the same file by changing the file type to "local", setting the scratch-pad bit and clearing the rest of the APF pointer, and clearing the PN and password ordinal fields and all of the third word of the FNT entry. This scratch-pad FNT entry is then placed in the CM FNT and its address is returned to MANAGER.

4.2.9 FUNCTION 8 - RELEASE AN OUTPUT QUEUE JOB

Function 8, which is also a part of the front-end batch printer subsystem, is called by MANAGER whenever MANAGER is no longer interested in a file. Function 8 can either evict the file, if it was completely printed, or it can return the file to ECS if its printing was interrupted.

In the call to function 8, MANAGER passes a file name, the address of a scratch-pad FNT entry for that file, a new routing character for the file (if it is to be returned to a different queue than the one it came from), and a code specifying whether the file is to be evicted or returned to ECS.

Function 8 starts by checking to see if MANAGER's error flag is set. If it is, function 8 quietly drops out, since Function 6 will return any front-end output files to ECS before letting MANAGER drop.

Function 8 proceeds to verify that the file name passed in the call matches the file name in the scratch-pad FNT entry. Then subroutine FNMATCH is called to find the verbatim FNT entry for the file: the FNT entry with the same name and RBT pointer, but with a file type of "output". If a verbatim FNT entry cannot be found for the file, then function 8 aborts MANAGER.

If a verbatim FNT entry is found, function 8 tries to set the file busy by clearing the complete bit of the scratch-pad FNT. (If the file is already busy, function 8 goes back to its beginning, rechecks MANAGER's error flag, and continues from there.)

Next function 8 must decide if it has been called to evict the file or to return it to ECS. It checks a code passed to it by MANAGER. For an evict request, function 8 calls subroutine RELCHN to evict the RBT chain. For a requeue request, function 8 first checks to see if MANAGER told it to send the file to a new queue; if MANAGER did request a new queue, function 8 takes the routing character for that queue and substitutes it for the second character in the file name of the verbatim FNT entry. Then function 8 calls subroutine REQUEUE, which will create and place a CP.LISP call which will return the FNT entry to ECS.

For both evict requests and requeue requests, function 8 completes its task by calling subroutine DELFNT to delete both the scratch-pad and verbatim FNT entries from CM.

4.3 FRONT END COMMANDS

4.3.1 FER

FER is a Front End task PP, similar to SYS. Currently, there is only one function.

Although there is only one function, the design of FER will allow additional functions to be added easily. The "FUNCTION" macro defines the beginning of a function. It enters the address of the function into a function jump table and qualifies the function. The functions must appear in numeric order.

Front End related tasks which are too small to dedicate a entire PP to, are the type of functions which should be added to FER.

4.3.1.1 FUNCTION 1

Function 1 is used to transmit Front End commands to from the 6500 to FREND.

FER reads the command from the user's field length to its internal buffer. After ensuring a valid request, FER builds and enters a stack request to issue the Front End command.

FER swaps the job out (WT.FEC) with FER in the delay stack. After MANAGER receives the Front End command reply from the 7/32, it moves the error code to the user's swapped out control point area and frees the job. FER senses this is the second portion of processing because the internal bit is set. FER will move the reply from the control point area to the user's parameter word and sets the complete bit.

4.3.2 FECMD CONTROL CARD

The command is translated by FECMD to ASCII because of problems transmitting the percent character in display code (it is a rubout when output'ed).

FER is called in to send the command to FREND.

If the error return from FREND is non-zero, FECMD issues an appropriate error message and aborts.

Whenever an Front End command error code (EC.XXX) is added to FESYM and FREND, it must also be added to FECMD.

FECMD error messages are generated by the ERRMSG macro. All Front End command errors are of the form:

```
     FECMD ERROR - XXXXXXXX
```

where XXXXXXXX is an English explanation of the error.

The ERRMSG macro call is as follows:

```
     ERRMSG   ERRNUM,TEXT
```

where ERRNUM is the XXX portion of the EC.XXX error code and TEXT is the appropriate error message.

ERRADD is a table of error message addresses, indexed by the EC.XXX error number. ERRMSG enters the address of the message into this table and then generates the message.

To add another error message it is necessary only to add another call to ERRMSG.

4.3.3 FECMD FORTRAN CALLABLE FUNCTION

FECMD builds the parameter word in a local buffer and moves the command there. It calls FER to issue the Front End command.

FER returns the Front End command error code to the parameter word. FECMD converts the integer error code to type real and returns it to the caller.

4.3.4 FECMD USER MACROS

The FECMD macros are straight forward. Since the caller must set up the FER parameter word, the macros must just issue the FER call for function 1 with recall.

4.4 SETCODE

Control card callable, FORTRAN callable, and macros were written to issue SETCODE functions. The PP routine CON actually changes the character code.

4.4.1 CON MODIFICATIONS

CON, first, checks if the call is for a disconnect, then for a SETCODE and finally for a connect.

The SETCODE and connect paths are identical except SETCODE does not change the device type to connected (DT.CON).

4.4.2 SETCODE CONTROL CARD

The syntax for the SETCODE control card is identical to that of the CONNECT control card. SETCODE was added as a second entry point in CONNECT. The existing code was rearranged slightly into more subroutines, which CONNECT and SETCODE share.

4.4.3 SETCODE FORTRAN CALLABLE SUBROUTINE

The SETCODE subroutine was added to the CONDIS deck on the FTNLIB program library. This places SETCODE with CONNEC and DISCONT subroutines. The syntax of SETCODE and CONNEC is identical, so SETCODE is a small routine which calls an existing CONNEC subroutine (DOPARM) to crack the parameters and setup the CON parameter word.

4.4.4 SETCODE MACROS

The SETCODE macros insert the specified character (if present) into the the parameter word and issue the CON call.

4.5 SPOOLED INPUT/OUTPUT

MANAGER spools input to the user's TTYTTY file and then gives the TTYTTY file to EDITOR if it consists of EDITOR text or to SPOOL if it is input for READPT or TPREAD.

To spool output, a program writes on TTYTTY. MANAGER will then list TTYTTY to the user's terminal after the program has completed. WRITEPT, an entry point in SPOOL, will copy a file to TTYTTY and cause it to be listed at the terminal in whatever character code is specified on the control card.

READPT and TPREAD are MANAGER commands as well as SCOPE commands. MANAGER recognizes the commands as a flag to start spooling input. At the end of the input steam, MANAGER allows the job's control card record to continue and SPOOL executes. See sections 4.1.3.2 and 4.1.3.3 for full details of MANAGER's spooling process.

4.5.1 READPT

READPT is an entry point in SPOOL and accepts spooled input on TTYTTY from MANAGER and copies to the specified file.

Subroutine CHKLFN is called to verify the local file name. If the NR parameter is not specified, the local file is rewound. The comdeck routine CPY= is used to copy from TTYTTY to the local file.

The CC option is used by MANAGER to determine the character code of TTYTTY when it is used to spool incoming lines for READPT.

4.5.2 TPREAD

TPREAD is another entry point in SPOOL. It and READPT (section 4.5.1) are the same routine in SPOOL. The difference between the two is the manner MANAGER spools the input to TTYTTY. MANAGER will issue READER.ON and READER.OFF Front End commands at the beginning and the end of the TPREAD spooling process.

Subroutine CHKLFN is called to verify the local file name. If the NR parameter is not specified, the local file is rewound. The comdeck routine CPY= is used to copy from TTYTTY to the local file.

The CC option is used by MANAGER to determine the character code of TTYTTY when it is used to spool incoming lines for TPREAD.

4.5.3 WRITEPT

WRITEPT is another entry point in SPOOL. It shares subroutines with TPREAD and READPT.

Subroutine CHKLFN is called to validate the local file name. Subroutine FNDCC is called to validate the character code. SETCC sets the character code in TTYTTY. If the no rewind parameter was not specified, the local file is rewound. The comdeck routine CPY= is used to copy the local file to TTYTTY.

MANAGER will then list the TTYTTY file to the user's terminal using the specified character code.

4.6 MSO

Since most of MSO dealt with finding and manipulating user output and backup buffers, it was decided to resequence and move MSO to the SCOPE program library. Almost all of MSO was rewritten for the Front End project.

4.6.1 OVERALL FLOW

All subroutines are designed to exit with the accumulator set to non-zero if a fatal error occurred. The subroutine is responsible for issuing the error message. If no errors were found, the accumulator is set to zero.

MSO determines the type of call (user, MSG with recall or MSG without recall) this is. Subroutines USRCALL (section 4.6.3), MSGW (section 4.6.2) and MSGWO (section 4.6.2) handle the individual calling sequences and parameter processing. Each returns to the main routine with the message in MSO's buffer, the character count, the receiver's user number, and the character code of the message.

The stack request is built and issued. Subroutine RECALL is called to process any stack request errors. The only legitimate error is "no room in port". Any other is a Front End dead, which causes MSO to drop out so that MANAGER may reinitialize.

Subroutine SWOOUT is called to recall the job if there is no room for the message in the user's 7/32 port.

If the request is not a refusable request, MSO swaps the job out waiting output (WT.OUT) on that port, with MSO in the delay stack.

If the request was a refusable request, MSO will swap out waiting time (WT.TIME) up to 2 times. Each time it bounces in, MSO will retry the request. After it has exhausted its bounce limit, it returns an error to the caller indicating the receiver was busy.

4.6.2 MSG CALL PROCESSING

Subroutine MSGWO validates the address of the message for the MSG call without recall and sets the direct cell MESSADD to it. MSGCALL is then, called to do the processing.

Subroutine MSGW validates the parameter word address and then the message address for the MSG call with recall. Again, MESSADD is set to the address of the message and subroutine MSGCALL is called to process the request.

Subroutine MSGCALL performs the usual validation of the message. Since MSG does not supply a character count, MSGCALL must determine it.

The message is at most 8 words long or until the end of the field length. MSGCALL must find the terminating zero byte and then strip off trailing blanks. In addition, MSGCALL prefixes the messsage with 2 blanks (for carriage control).

The port of the receiver is found in the control point area.

Only Old Mistic messages are permitted.

The bulk of this subroutine is taken directly from the original MSO.

4.6.3 CPU CALL PROCESSING

Subroutine USRCALL processes user calls. All parameters are validated. Refusable calls and calls to other users are restricted to system library routines.

There are two special character counts left over from the previous interactive subsystem. If SC.ONBS (3773B) is the character count. then the backspace function should be turned on. If SC.OFBS (3774B) is the character count, the backspace function should be turned off. These were left in MSO due to problems with APL which uses them. Subroutines ONBKSP and OFFBKSP move the appropriate Front End command to the buffer as though it were a user message and set the Front End command bit in the stack request.

Subroutine READMSG determines the number of CM words in the message from the character count and character type. ASCII messages are packed 5 character per CM word and Old Mistic 10 characters per CM word. The message is read to the buffer.

4.6.4 FRONT END COMMANDS

MSO will issue two Front End commands. One to turn the backspace function off and one to turn it back on. This was left in MSO for a version of APL which required it. For more details see section 4.6.4.

4.7 MSX

The bulk of MSX dealt with finding and manipulating user output and backup buffers. Since that had to be changed, it was decided to resequence and move MSX to the SCOPE program libirary. Almost everything was rewritten except the tables of error messages.

4.7.1 OVERALL FLOW

All subroutines are designed to exit with the accumulator set to non-zero if a fatal error occurred. The subroutine is responsible for issuing the error message. If no errors were found, the accumulator is set to zero.

Subroutine VALDATE is called to validate the call. If no errors were found, MOVEMSG is called to move the correct message from the error message table to the buffer. For mode errors, the P register is inserted in the message.

MOVEMSG calls MODERR, IOERR, ESDPERR, or NORMERR (depending on the type of error) to locate the correct error message and insert the P register when necessary for each of the different types of errors. See sections 4.7.3 through 4.7.6 for more details about the individual error message processors. Subroutine MOVEBUF then moves the error message to the buffer.

The stack request is build and issued. Subroutine ERRPROC deals with stack request errors. The only acceptable error is "no room in port". MSX will swap the job out waiting output (WT.OUT). Otherwise, the Front End is assumed to be inoperative. MSX will then, drop out and allow the interactive service to complete its idle down procedures and be reinitialized.

4.7.2 ERROR MESSAGE TABLE STRUCTURES

Basically, there is a table of address of error messages at MSGTAB. The error number is used to index into the table to find the correct error message.

The MESSAGE macro is used to generate entries in MSGTAB and then the messages. MSGTAB is in a USE block called TAB. The messages are generated in the program block.

The format of the MESSAGE call is as follows:

```
     LABEL    MESSAGE  ERRSYM,TEXT
```

where LABEL is the address of a predefined message (if specified it will be entered in MSGTAB),
ERRSYM is the error's index into MSGTAB,
and TEXT is the error message.

The set symbol LTAB is the last entry in MSGTAB so far and is updated by MESSAGE. It is used by the other message generating macros to determine the next available entry in MSGTAB.

The normal error messages are generated with the MESSAGE macro and must be defined before any I/O, CP4ES/6DP, or MSX internal messages because the F.ERXX error number is used directly to index into the message table.

The internal error messages are next. These are the specific messages for mode 1, mode 2, and mode 4 errors. They are generated with the INTMESS macro (call follows).

```
     INTMESS   SYMB,TEXT
```

where F.SYMB will be defined as the offset for the error message and TEXT is the message.

The I/O messages are next. The beginning of this section is defined with a symbol STIOM which is used by subroutine IOERR as the base of the I/O error entries. The macro IOMESS (call follows) is used to generate the entries and messages.

```
     IOMESS   TEXT
```

where TEXT is the message.

IOMESS also calls the MESSAGE macro. The error are added one by one to the end of the MSGTAB table. There are no symbols defined for the secondary error numbers. New I/O errors should be added to the end of the I/O error message section. The same message must be added to 6WM (the batch error message processor).

The CP4ES/6DP errors follow. They are defined with the ESMES macro (call follows).

```
     ESMES    TEXT
```

where text is the error message. The ESMES macro is the same as the IOMES macro. The names are different to emphasize these are two distinct subsets of messages.

Like the I/O error section, the CP4ES/6DP section begins with a base symbol, STESM (which is used by subroutine ESDPERR). There are no symbols defined for the secondary error number. New error messages should be added to the end of the CP4ES/6DP message section. The same message must be added to 6DP (the batch error message processor).

4.7.3 NORMAL ERROR PROCESSING

Subroutine NORMERR is called to process normal errors. These are the standard F.ERRXX errors (except for F.ERAR).

The error number is the offset in MSGTAB which contains the address of the error message.

The P register is inserted into the program stop or exec call error message if needed.

4.7.4 MODE ERROR PROCESSING

Subroutine MODEERR is called to process mode errors. Mode errors 1, 2. and 4 have their own internal messages. Other mode errors have a general mode error message. The P register is inserted in the correct error message.

The offset of the error message in MSGTAB is returned to the caller.

4.7.5 I/O ERROR PROCESSING

Subroutine IOERR processes the I/O errors. These have F.ERIO as their primary error number. The offset in MSGTAB is the secondary error number + 1 + STIOM (the base of the I/O errors).

There is no special processing involved.

4.7.6 CP4ES ERROR PROCESSING

Subroutine ESDPERR processes the CP4ES/6DP errors. These have E.6DES as the primary error number.

The offset in MSGTAB is the secondary error number + 1 + STESM (the base of the CP4ES/6DP MESSAGES).

There is no special processing involved.

4.8 MINOR MODIFICATIONS

This section deals with minor modifications to several existing routines on the system.

4.8.1 CON

4.8.1.1 SETCODE modifications

SETCODE modifications are described in section 4.4.1.

4.8.1.2 Other Modifications

CON was modified to allow console jobs to connect files. This was needed so that MANAGER, ARGUS, DUMPFE, and the stimulator could communicated with FREND over connected files.

The new connected file type AF and BI were added to the table of valid connect types in CON.

4.8.2 LOGOUT

LOGOUT was modified to accept a DIS parameter which indicates the user is being logged out after a disconnect has occurred. An end-of-file status is used to indicate the user has disconnected during the logout process.

4.8.3 SSS

SSS was modified to move the Front End port number to the control point area upon swap out.

4.8.4 SEND/MESSAGE

SEND and MESSAGE were modified. first not to call the PP program MSI in to read input from the user's input buffer. They now use connected files to read user input.

Under previous systems. a user could swap out waiting output to complete on another user's terminal by 'SEND"ing to her. Because of changes to the method of freeing jobs which are swapped out waiting output, a user could hang. SEND and MESSAGE were modified to user a refusable MSO call, which will not swap the caller out. Instead, MSO will return an error and SEND and MESSAGE will inform the sender that the other terminal is busy now.

See appendices 1 and 2 to 6SM 77.0, SEND and appendices 1 and 2 to 6SM 81.0. MESSAGE for more details.

4.8.5 LOGIN

LOGIN was modified to dayfile the user number, the sequence number, the Front End socket and port numbers. and the pool id of the MISTIC job after logging in.

The sequence and line number message, which LOGIN prints at the user's terminal, was changed to include the Front End socket and port numbers.

4.8.6 SITUATE

SITUATE was modified to print the Front End socket and port numbers in addition to the line number, sequence number and user id which SITUATE has previously printed for the operator. The SITUATE user output remained the same (only a list of user ids).

See appendix 1 to 6SM 80.0, SITUATE, for more details.

4.8.8 EDITOR

EDITOR was modified to recognize a standard SCOPE TTYTTY input file. Previously, the TTYTTY input file contained a character count followed by the text line. The length of an interactive line (L.TTYLEN) was changed from 7 to 16. This causes EDITOR to output full length text lines. The Front End command RMARGIN, is used to determine if and where the line should be folded at the terminal.

See appendices 1 and 2 to 6SM 91 for more details.

4.8.9 MISTIC PP'S

4.8.9.1 INTCOM

The MISTIC PPU routines call a common deck, INTCOM, which contains useful macros. These macros were modified to not refer to the input and output buffers and P.SHA2 which were removed from the "6500" and moved to the "7/32".

4.8.9.2 TBL

The code in TBL which allowed a system routine to read a copy of a user's input and output buffers in MANAGER was deleted. Appropriate error messages are dayfiled instead.

4.8.9.3 DELETED ROUTINES

MMM, MMS, 1BR, 2TT. and MSI were removed from the system.

MSO and MSX were resequenced and moved to the SCOPE program library. See sections 4.6 and 4.7 for details.

4.8.10 OTHER ROUTINES MODIFIED FOR BATCH PRINTERS

4.8.10.1 IRCP. CMR

Subroutine PUTIO, which occurs in slightly different forms in both IRCP and CMR, was modified to process new disposition codes for the printers. PUTIO is a function in CE.LISP, which puts a file in the correct I/O queue and column, based on its file type and disposition code.

Printer disposition codes are now as follows:

PR	40B	print on any available printer
PAU	41B	print on 64-character front-end printer
PAF	73B	print on 96-character front-end printer

PA	74B	print on any front-end (ASCII) printer

PUTIO now recognizes all these codes as printer codes.

4.8.10.2 DISPOSE

DISPOSE was modified to create print files in the new dispositions, as described in section 4.8.10.1.

4.8.10.3 FNT, 6DP

These routines were both modified to use the CE.LISP function PUTIO to put output files in ECS, instead of PUT. 6DP was also modified to change the obsolete disposition code 42B (P2) to 40B (PR).

4.8.10.4 QDR

QDR was modified in the way it retrieves output files from ECS for dumping, so as to dump the new PAF and PA files as print files.

4.8.10.5 1EJ

1EJ was modified to write an EOR on the output file before the beginning of the dayfile. This ensures that MANAGER can always find the dayfile as the last record of a print file.

4.8.10.6 ARGUS AND 1AR

ARGUS and 1AR were modified to put ended or page-limited jobs back in the "R" queue, as does MANAGER.

1AR was modified to print all non-PA jobs on the 501 printers, since the old P1 disposition code (41B) is now used for the front-end printers.

4.8.10.6 SYS

The FNTSTAT function was modified to use the FNT address in the "FET" when locating the CM FNT entry for the file. This is because MANAGER does FNTSTAT calls on its print files, which each have 2 FNT entries. MANAGER uses the FNT address field in the FET to ensure that the local scratch-pad FNT entry is always used, instead of the output FNT copy.

5.0 OPERATOR COMMUNICATIONS AND PROCEDURES

5.1 MANAGER PROCEDURES

5.1.1 MANAGER CONTROL CARD OPTIONS

Both control card options deal with the 7/32 load procedure. Normally, MANAGER will automatically load the correct FREND system.

The option "NOLOAD" will prevent MANAGER from attempting to load the 7/32.

The option "PFN=XXXX" (where XXX is the permanent file name of a FREND system) will cause MANAGER to load the specified FREND system.

MANAGER will always dayfile a FREND identification message which which includes the version number, date and time of assembly of the FREND system MANAGER has just loaded into the 7/32.

5.1.2 MISTIC2 TERMINATION

To terminate interactive service, the operator should type

1. CFO LOGOUT

which will cause MANAGER to send termination messages to all users and begin the idle down sequence.

5.1.3 SYSTEM TROUBLE MESSAGES

MANAGER no longer accepts "CFO"S from the operator as a system trouble message. To send a global trouble message, the operator should use the Front End command SENDALL. See the FREND Operator's Guide for complete details.

5.2 SITUATE_CHANGES

The Front End socket and port numbers were added to information SITUATE outputs for the operator. The format is as follows:

```
LINE - SOCKET - PORT - SEQUENCE NUMBER - USER ID
```

6.0 USER_ASPECTS

The new features FECMD. SETCODE, and WRITEPT were added. The capibilites of READPT and TPREAD were expanded to allow input spooling in any of four character codes.

Many outstanding problems with the previous interactive subsystem were solved.

These changes were necessary to install the 7/32 Front End system.

7.0 DAYFILE CHANGES

7.1 LOGIN

USER XXX - SSXXXXX - S XXX - P XXX - POOL-ID XX

Issued by LOGIN when a user successfully logs in.
USER is the MANAGER user number.
S is the Front End socket number, and
P is the Front End port number.

7.2 TBL

TBL - INPUT BUFFER ACCESS SUPPORT DROPPED

Issued by TBL when a request is made to access the input buffers which previously resided in MANAGER.

TBL - OUTPUT BUFFER ACCESS SUPPORT DROPPED

Issued by TBL when a request is made to access the output buffers which previously resided in MANAGER.

7.3 MSO

MSO - USER NOT LOGGED IN

Detected by MSO when requested to transmit a message to a user who is not logged in.

7.4 SPOOL

READPT/TPREAD/WRITEPT FAILURE

Issued by SPOOL when it has failed to copy to end of information.

INVALID FILE NAME - XXXXXX

Issued by SPOOL when an invalid file name is specified. -XXXXXXX- is the file name.

WRITEPT - INVALID CHARACTER CODE - CC

Issued by SPOOL when an invalid character code is specified on a WRITEPT call. CC is the character code.

WRITEPT - SYSTEM ERROR, CANNOT SET CHARACTER CODE

Issued by SPOOL when WRITEPT encounters an error while attempting to set the character code.

7.5 FECMD

FECMD ERROR - UNRECOGNIZED COMMAND

Issued by FECMD when the Front End command is not recognized by the 7/32.

FECMD ERROR - INVALID KEYWORD

Issued by FECMD when the Front End command contains an invalid keyword.

FECMD ERROR - UNSUPPORTED CHARDEF CHARACTER

Issued by FECMD when the Front End command contains an unsupported chardef character.

FECMD ERROR - UNSUPPORTED CHARDEF FUNCTION

Issued by FECMD when the Front End command contains an unsupported chardef function.

FECMD ERROR - PARAMETER MUST BE ON OR OFF

Issued by FECMD when the Front End command contains a parameter which must be on or off and is not.

FECMD ERROR - ILLEGAL TERMINAL TYPE

Issued by FECMD when the Front End command contains an invalid terminal type.

FECMD ERROR - CHARACTER DELAY GT 256

Issued by FECMD when the Front End command contains a character delay of 257 or more.

FECMD ERROR - UNAUTHORIZED COMMAND

Issued by FECMD when the user is not authorized to issue the Front End command.

FECMD ERROR - VALUE MUST BE 1 TO 240

Issued by FECMD when the Front End command contains a parameter value which should be between 1 and 240 and is not.

FECMD ERROR - YOU HAVE ONLY 1 CONNECTION

Issued by FECMD when the Front End command attempts to change connections and only 1 connection exists.

FECMD - NO FRONT END COMMAND SPECIFIED

Non-fatal error message issued by FECMD when no Front End command was specified on the control card.

FECMD SYSTEM ERROR - UNKNOWN ERROR CODE

Issued by FECMD when the Front End when an unknown error code is returned by FER.

7.6 MANAGER

ATTEMPTED AUTO LOAD OF NON-PRODUCTION FREND SYSTEM

Issued by MANAGER when it discovers it is trying to load a non-production FREND system during its automatic load sequence.

```
FREND SYSTEM - VER  NNNN          DDDDDDDDDDTTTTTTTTTT
```

where NNNN is the FREND version number
DDDDDDDDDD is the date of assembly of the FREND system
TTTTTTTTTT is the time of assembly of the FREND

system
Issued by MANAGER to identify the FREND system, MANAGER is loading into the 7/32.

7/32 HAS DIED

Issued by MANAGER after 1FP has set MANAGER's control ports complete, indicating the 7/32 system has died.

FRONT-END TURNED OFF

Issued by MANAGER when it dies, due to the front-end being turned off in the EST.

OPEN REJECTED BY MANAGER

Issued by MANAGER when the 7/32 requests an open and there are no available MISTIC pool pockets.

CONTROL PORT MESSAGE FOR USER NOT LOGGED IN, FP = XXX

Issued by MANAGER when it receives a control port message for a user who is not logged in. XXX is the numeric protocol record type.

MANAGER - CONTROL CARD ERROR

Issued by MANAGER when it does not recognize a control card parameter.

7.7 MSX

MSX - FRONT END DEAD

Occurs when MSX receives an error return other than port full after attempting to send an error message to an interactive user.

MSX - MAXIMUM NUMBER OF USERS IS ZERO

MSX has found the low core cell which holds the maximum number of interactive users to be zero.

7.8 FER

FER - UNDEFINED FUNCTION NUMBER

FER was called for other than function 1.

FER - MUST BE CALLED WITH AUTO-RECALL

FER was called without auto-recall.

FER - CALLED BY NON-MISTIC JOB

FER called by a non-interactive job.

FER - PARAMETER WORD NOT IN FIELD LENGTH

FER was called with the address of its parameter word outside the job field length.

FER - PORT NUMBER IS ZERO

This is a system error.

FER(1) - COMMAND NOT ENTIRELY IN FIELD LENGTH

The command passed to FER did not lie entirely within the job field length.

FER(1) - COMMAND LENGTH IS ZERO

The length of the command passed to FER is zero.

FER(1) - COMMAND LENGTH IS TOO LARGE

The length of the command passed to FER is greater than 240 characters.

FER(1) - UNDEFINED CONNECT TYPE XX

XX is the connect code which the user specified for a front-end command. The code is other than OM, AS. AF. or BI.

7.9 MAN

MAN(5) - FL VIOLATION

Issued by MAN function 5 (initialization) when its parameters are not within MANAGER's field length.

MAN(5) - UNASSIGNED STORAGE BEFORE MANAGER

Issued by MAN function 5 (initialization) when unassigned storage exists between MANAGER and control point zero.

MAN(6) - ILLEGAL CALL

Issued by MAN function 6 when called without the internal bit set.

MAN(6) - MANAGER HAS MOVED

Issued by MAN function 6 when it discovers MANAGER's control point has moved. This will only occur when an EDITLIB or a PP interpret has been initiated.

MAN - ILLEGAL CALL

Issued by MAN when called by any one other than MANAGER.

XXIIFFFFWWWWMMMM

Instrumentation messages issued by MAN when freeing a job in the pool.

XX = the 2 letter message code
XC - job freed after waiting command
IN - job freed after waiting for user input
OU - job freed after waiting for user output
SP - job freed waiting after user output spool
FC - job freed after waiting front-end command
II = the HUSTLER pool table id in display code
FFFF = octal job requested field length
WWWW = octal initial job weight (bottom 6 bits of 12)
MMMM = octal 12 bit millisecond clock/100B

NJXX LINENNNN

Instrumentation message issued by MAN when creating a pool pocket for a new MISTIC job.
XX = the pool id of the new job
NNNN = the user number of the new job

MAN ERROR - USER XXXX DUPLICATE USER

Issued by MAN when it discovers two jobs with the same user number.
XXXX = the duplicate user number

MAN ERROR - USER XXXX NO ROOM FOR POCKET

Issued by MAN when called in to create a pool pocket for a job and there were no free pockets.
XXXX = the user number of the job involved

MAN ERROR - USER XXXX JOB NOT WAITING

Issued by MAN when called in to free a job which was suppose to be waiting input, output, spool, or Front End command and was not.
XXXX = the jobs user number

MAN(6) - OUTPUT FNT HUNG BUSY

Issued when MAN function 6, on discovering that it is the only PP at MANAGER's control point, ignores the fact that the complete bit is not set in the FET and seizes the file.

MAN(7) - ECS ERRORS

Occurs when MAN function 7 detects ECS errors when trying to retrieve a front end output file.

MAN(7) - NO FNT SPACE

Occurs when MAN function 7 cannot find a free slot in the file name table, and MANAGER has died.

MAN(7) - INVALID PRINT CODE

Occurs when MAN function 7 is passed an invalid code by MANAGER. Currently the only legal code requests a print file to be retrieved from ECS.

MAN(7) - PRU LIMIT TOO LARGE

Occurs when MANAGER passes a PRU limit which is too large to be legal to MAN function 7.

MAN(7) - ADDRESS WAS VALID LAST TIME

Occurs when MAN function 7 detects a bounds error when trying to rewrite a status word in MANAGER's field length. after failing to find a requested file.

MAN(7) - ADDRESS WAS GOOD LAST TIME

Occurs when MAN function 7 detects a bounds error when trying to rewrite a status word in MANAGER's field length, after successfully finding a requested file.

MAN(7) - ADDRESS WAS IN RANGE LAST TIME

Occurs when MAN function 7 detects a bounds error when trying to copy a FNT entry to MANAGER's field length.

MAN(8) - FILE NAMES DO NOT MATCH

Occurs when the file name passed to MAN function 8 does not match the name of the FNT whose address is passed.

MAN(8) - MATCHING FNT NOT FOUND

MAN function 8 dayfiles this message when it cannot find a FNT entry with the same name, RBT pointer and control point as the FNT entry passed to it by MANAGER.

MAN(8) - ADDRESS WAS GOOD LAST TIME

Occurs if MAN function 8 gets a bounds error when rewriting a status word in MANAGER's field length.

7.10 CONNECT/SETCODE

INVALID FILE NAME---XXXXXXX

Issued by CONNECT and SETCODE when either detects an invalid file name specified on their control cards. XXXXXXX is the file name.

INVALID TYPE CODE---CC

Issued by CONNECT and SETCODE when either detects an invalid character code on their control cards. CC is the character code.

8.0 REFERENCES

Software Modification Proposals:

28 - Front End
41 - Front End Phase 1 Detail
60 - Front End Command and Control

Other 6000 Scope Memos:

60.2 - HUSTLER 2.0
77.0 - SEND. appendices 1 and 2
80.0 - SITUATE. appendix 1
81.0 - MESSAGE. appendices 1 and 2
91.1 - EDITOR. appendices 1 and 2
94.1 - Banner Messages
124.0 - Frend
134.0 - 1FP and friends
131.0 - Merit Network Support

CDC publications:

Internal Maintenance Specification. INTERCOM 1

Operator Guides

FREND Operator's Guide

WRITTEN BY John K. Renwick
John K. Renwick

APPROVED BY Richard R. Moore
Richard R. Moore

There is no page 70

SECTION MAP

SECTION MAP

SECTION MAP

SECTION MAP

SECTION MAP

SECTION PAGE

SECTION MAP

PL	IDENT	ROUTINE
SCOPE	JKRASCMGR	MANAGER
SCOPE	JKRMGRED	MANAGER

Installed in LSD 48.25, 49.37

Check one:

□ M4 No. ______

□ Appendix number 1 to 6SM 135.1

MSU MINI MOD MEMO MANAGER changes to support ASCII EDITOR

Coded by: J. Renwick JKR CM

Date: 11 / 26 / 79
Mo. Day Yr.

1.0 DESCRIPTION (What was done; why; general effects; references.)

MANAGER was modified to pass EDITOR commands and text to EDITOR in ASCII. Conditionally assembled code supports three MANAGER commands to switch modes between ASCII and Display-code. The default was originally set to Display, to allow ASCII EDITOR development. When ASCII EDITOR was installed in LSD 49.37, JKRMGRED changed the default to ASCII.

2.0 EXTERNAL CHANGES (Operational changes; changes affecting the rest of the system.)

New MANAGER commands are:

ASEDTXT - Pass text to EDITOR in ASCII (Default)
ASEDCMD - Pass commands to EDITOR in ASCII (Default)
OMEDIT - Pass commands and text to EDITOR in Display-code.

MANAGER now writes TTYTTY entirely in ASCII unless OMEDIT is selected. EDITOR commands in the control card buffer (except "EDITOR" and "END" cards) are in ASCII unless OMEDIT is selected.

Dayfile & C.E. Error File Changes

None

New symbols; Table Usage Changes

Two new bits in MANAGER's "AUXSTAT" table:
OMCMD - Display-code EDITOR commands
OMTXT - Display-code EDITOR text.

3.0 INTERNAL SPECIFICATIONS (How the change works. Include cautions and assembly options.)

State SERV reads all input from the terminal in AF (except for TPREAD and READPT input).

State DOSERV processes ASCII input, ignoring leading control characters and blanks as it decides what the input is. Subroutine CMDSTASH translates any non-EDITOR commands to Display-code, but puts EDITOR commands into the control-card buffer in ASCII.

State OUTNUM sends EDITOR line numbers to the terminal in ASCII for the "N" command.

The new macro "ASTODC" translates one character from ASCII to Display-code, using the translate table at "ASDCTAB."

The new subroutine ADDNUM inserts an ASCII line number at the beginning of a line of text. This is used by OUTNUM to generate the prompts, as well as by STASH to add numbers to incoming text.

The new subroutine NXTCCH gets the next character from an ASCII line, and translates it to Display-code.

*** Use an M4 only if the information fits on it, and user changes are insignificant. Final copy should be typed and printed. ***

Assembly option: If the symbol ASSW equals one, code is assembled to:

Process the ASEDTXT, ASEDCMD and OMEDIT commands. These commands just set and clear the OMCMD and OMTXT bits in the AUXSTAT table;

Translate EDITOR commands to Display-code (in CMDSTASH) whenever OMCMD=1;

Translate EDITOR text to Display-code in state STASH. This is done by the new subroutine ASDC, which is conditionally assembled.

The ASSW option allows the pre-ASCII EDITOR to run, and can be turned off whenever we no longer need to use the old EDITOR.

I believe that "M4" stood for MSU Mini-Mod Memo.

mRR December 2003