MICHIGAN STATE UNIVERSITY

COMPUTER LABORATORY

6000 SCOPE MEMO NO. 134.1

July 13, 1979

# 1FP,CP2TT and Associated Routines

## 1.0 Introduction

This 6SM describes 1FP, CP2TT, and associated routines on the 6500 which interface with the FREND system. This includes portions of CPSP and CPCIO. Together these form the system for input/output to and from FREND.

The entire front-end development project consisted also of extensive changes to MANAGER and other interactive support routines (6SM 135) and the development of FREND (6SM 124).

This 6SM replaces 6SM 134.

## 2.0 External Reference Specifications

### 2.1 Major front-end components.

The major components of the 6500 side of the front end can be broken down into 2 major subsystems:

1. The input/output subsystem:
   1FP - front end stack request processor
   CP2TT - issues stack requests for 1FP from CIO requests.
   CPSP - primarily designed to process stack requests for the disks, CPSP also handles stack requests for 1FP.
   CPCIO - processes all CIO calls, and invokes CP2TT for I/O on connected files.

2. The control subsystem:
   MANAGER - the monitor for all interactive jobs
   MAN - a PPU helper for MANAGER

### 2.2 7/32 - 6500 Interface

All communication between the 7/32 and the 6500 is done by 1FP, via the 6500-7/32 DMA interface. 1FP is a dedicated PPU which is initiated by MTR when there is an outstanding stack request on device FD, the front end interface. 1FP remains active until MANAGER is dropped and all requests have been processed. It performs the following functions.

1. CIO calls cause CP2TT to issue the stack request. The request is issued directly by a PPU (O.FWRP). This is done by MSX,MSO, and FER.

2. Transfer data over the control ports for MANAGER, ARGUS, and the stimulator. Control ports are pseudo files on which there is always an outstanding read or write in progress. They are used to transfer special control information between the 6500 and the 7/32. Control ports have no FNT entries.

2.3 CIO interface.

All CIO requests on connected files are trapped by CPCIO and passed directly to CP2TT. CP2TT then formats a special front-end stack request and passes it to CPSP. CPSP then passes this request to 1FP, which processes it.

2.3.1 A read on a connected file is supported by two CIO operations:.

READ (10B) - order code O.FREAD
RDSKIP (20B) - order code O.FRDSK

These operations have the following rules and conventions:

- A READ is issued by a FORTRAN formatted READ statement, and by a COBOL READ. A RDSKIP is issued by a FORTRAN BUFFER-IN statement.
- The maximum line length is 240 characters.
- If the user buffer will not hold the entire line, as much as will fit is transferred. For a normal read, error code 10B (device capacity exceeded) is returned. For a read-skip, no error code is returned.
- A prompt character (*) will be issued if prompt is on, unless typed ahead data is present to satisfy the read when the read is issued.
- The binary/coded flag in the FET has no effect on READ and RDSKIP.
- A single READ or RDSKIP CIO request will transfer only one line of data, regardless of the amount of extra room in the user buffer, and the presence of typed-ahead data on the 7/32 side.
- When the user enters a line on the 7/32 which is longer than the current setting of INLEN (default=240), that line is broken at INLEN characters, as if the user had typed an end-of-line. A program on the 6500 which reads that line cannot tell if it was terminated by an end-of-line, or automatically broken by the 7/32 because of its length.
- If a read is issued and the requested data is not available on the 7/32, the user job is swapped out until that data becomes available.
- Character translation is described in section 2.4.

2.3.2 A write to a connected file is supported by three CIO operations:

WRITE (14B) - order code O.FWRT
WRITER (24B) - order code O.FWRTR
WRITEF (34B) - order code O.FWRTR

These operations have the following rules and conventions:

- A WRITE is issued by a FORTRAN formatted WRITE statement, and by a COBOL WRITE.
- A WRITER is issued by a FORTRAN BUFFER-OUT statement.
- There is no limit on the line length for output. Any write always dumps the entire contents of the user buffer to the terminal, regardless of any line boundaries (or partial lines).
- WRITER and WRITEF always imply that there is an end-of-line following the last data in the buffer. Hence the next data following a WRITER or WRITEF is considered a new line for carriage control purposes.
- WRITE does not imply end-of-line unless the last data in the buffer actually terminates with an end-of-line as defined for that particular character set. Hence, WRITE may write many segments of partial lines.
- Because WRITE does not guarantee that only complete lines are written, writes of partial lines from multiple connected files must not be intermixed.
- Data from a user buffer is transferred directly from the user program to a buffer in the 7/32. When the write operation has completed, the user buffer has been emptied; however, the data has not necessarily all been printed at the user terminal.
- Character translation is described in section 2.4.
- Carriage controls are associated with OM and AF files as described in section 2.4.
- Delay characters (ASCII null) are automatically inserted by the 7/32 after it transmits CR, LF, FF, HT, and VT originating from files of type OM and AF. The number of delay characters is controlled by the 7/32 command CDELAY. It is also set by the 7/32 TERMINAL command.

2.3.3 Block writes to connected files are supported by three CIO operations:

FWTBL (414B) - order code O.FWTBL
FWBER (420B) - order code O.FWRBL
FWBEF (424B) - order code O.FWRBL

These operations have the following rules and conventions:

- A FWTBL will transfer only PRU (100B) blocks from the user's buffer to the front-end.

- Both FWBER and FWBEF transfer data until the user's buffer is empty.
- A FWBER transmits an EOR record after flushing the user's buffer. The level number from the FET is used.
- A FWBEF transmits an EOR level zero (0) after flushing the user's buffer if the operation previous to the FWBEF was a FWTBL or there was data in the user's buffer (EOF with data). In any case, an EOF record is transmitted. This guarantees an EOR before the EOF.
- FWBER and FWBEF imply end-of-line following the last data in the buffer. Since FWTBL is PRU oriented, it of course allows writes of partial lines.
- No translation is done on the data. All data is written verbatim one character per 7/32 byte into the 7/32 buffer. The character code determines how data is to be unpacked from CM.

## 2.4 Character Sets.

There are 4 character sets supported on connected files. The character set is selected by connecting the file with the appropriate code. The default is OM (Old Mistic).

Input: the line is not translated until it is read by a program on the 6500. Data is retained in ASCII by the 7/32; it is translated by 1FP when the data is transferred from the 7/32 to the 6500.

Output: the line is translated by 1FP when it is written to the 7/32. Translation is <u>not</u> done on block writes.

### 2.4.1 OM: Old Mistic

Character code: 6 bit display code, packed 10/CM word.

End-of-line: 12 - 66 zero bits in rightmost position of a CM word.

On output, the 6500 will remove all trailing blanks back to but not including the first character on the input line. On input, trailing blanks are not removed; the input line is translated as described below. All characters in the last CM word past the last valid character are filled with binary zero. If the input was an even multiple of 10 characters, an extra word of zero follows the data.

All OM characters map directly to their ASCII graphic

equivalents, with the exception of the 00B code, which maps to a blank if imbedded in a CM word (trailing 00B characters indicate end-of-line).

On input, any character with no display code equivalent is ignored. Otherwise, the character is mapped to its display code graphic equivalent.

Carriage controls: the 1st character of every OM output line is always interpreted as a carriage control. It is never printed (this may be overidden by the front end command CCTL). Valid carriage controls are:

1 - CR,LF,LF,LF
0 - CR,LF,LF
+ - CR
- - CR,LF,LF,LF
$ and , - no action
any other - CR,LF

The 7/32 will automatically supply the proper number of delay characters after CR or LF.

Right margin processing is done by the 7/32 for OM output lines.

2.4.2 AF: ASCII Fancy.

This is a full ASCII character set (7 bits, 128 characters) with the OM conventions for end-of-line and carriage controls.

Character code: 7 bits, in 12-bit bytes, packed 5 to a CM word. ASCII null (00) is represented as 4000B, all other codes are the verbatim 7-bit ASCII code.

End-of-line: 12 - 60 zero bits in the rightmost bits of a CM word. On output, end-of-line implies that the 1st character of the next line will be a carriage control.

On output, no trailing blanks are stripped.

On input, trailing blanks are not removed--the input line is transmitted verbatim. All characters in the last CM word past the last valid character are filled with binary zero. If the input was an even multiple of 5 characters, an extra word of zero follows the data.

Translation: the 6500 representation of ASCII null (00) is 4000B. All other ASCII codes are the verbatim 7-bit value. Note that only 7 bits are used. On output, an 8th bit will be ignored. (If parity is OFF, the parity bit will always be a zero.)

Carriage controls: the 1st character of every AF line is always interpreted as a carriage control. It is never printed (this may be overidden with the front end command CCTL). Valid carriage controls are:

1 - CR,LF,LF,LF
0 - CR,LF,LF
+ - CR
- - CR,LF,LF,LF
$ and , - no action
any other - CR,LF

The user may generate his own CR and LF with the proper ASCII characters. The 7/32 will automatically supply the proper number of delay characters after CR, LF, HT, VT or FF. Note than an imbedded CR,LF in the output line is not an end-of-line: the next character is not a carriage control. AF carriage controls are identical to central batch and SCOPE 3.4. (SCOPE 3.4 interactive does not support ",".) Right margin processing is done by the 7/32 for AF output lines.

2.4.3 AS: ASCII 7-bit transparent.

This is a 7-bit transparent character set.

Character code: 7-bit ASCII packed into 12-bit bytes, packed 5 per CM word.

End-of-line: 12-60 bits of 4000B in the rightmost bits of a CM word. On input, the last CM word is padded out with as many 4000B bytes as necessary. If the number of characters is a multiple of 5, the data will be followed by an entire CM word of 5 bytes of 4000B. On output, the last word should be padded out with 4000B--these are not sent out to the terminal. End-of-line on output has no real meaning, since there are no carriage controls.

Translation: there is no translation. The 6500 representation is the verbatim 7-bit code. Note that only 7 bits are transmitted or received, regardless of the contents of the 8th bit.

Carriage controls: there are no carriage controls. The output data must contain all the proper ASCII control characters for carriage motion (CR, LF). The 1st character of every line is always printed. The 7/32 will never insert any delay characters. Right margin processing is never done.

2.4.4 BI: Binary 8-bit transparent.

This is an 8-bit transparent character set. The only difference between this character set and AS is that AS is always 7 bits, while BI is always 8 bits.

Character code: 7 bit ASCII packed into 12-bit types, packed 5 per CM word.

End-of-line: 12-60 bits of 4000B in the rightmost bits of a CM word. On input, the last CM word is padded out with as many 4000B bytes as necessary. If the number of characters is a multiple of 5, the data will not be followed by an extra padding word. This allows literal data transmission without spurious imbedded end-of-line bytes. On output, the last word should be padded out with 4000B--these are not sent out to the terminal. End-of-line on output has no real meaning, since there are no carriage controls.

Translation: there is no translation. The 6500 representation is the verbatim 8-bit code. Note that to receive and transmit 8 valid bits of data, the 7/32 must be in PARITY,NONE mode. If the 7/32 is in even parity (the default) or odd parity, the 8th bit will always be the parity bit, regardless of the actual value of the bit sent to or received from the terminal.

Carriage controls: there are no carriage controls. The output data must contain any characters necessary for desired carriage motion. The 7/32 inserts no delay characters after the special ASCII characters CR,LF,HT,VT,FF.

Note: binary mode on the 7/32 (the BINARY command) does not automatically imply a binary BI type file. The type of file is determined strictly by the type of connected file reading the data on the 6500. To transmit pure 8-bit binary data from a terminal to the 6500, it is necessary to place the terminal in PARITY,NONE and BINARY mode, and then read the data into a BI type file on the 6500.

## 2.5 Special CIO interface for control port jobs.

CP2TT is the CIO routine which builds the stack request for all I/O on connected files. This request is built slightly differently for control-port jobs, than for normal user jobs.

For user jobs (HUSTLER jobs), the character set of a connected file is always taken from the file FNT entry. The destination port number is always taken from field C.CPPNBR of the users control point area.

For MANAGER and ARGUS, which use CIO to issue stack requests for all front-end I/O, this scheme is not practical, since they would continually have to manipulate the FNT and control point area. Therefore, for control ports, the following scheme is always followed:

port number - taken from bits 0-11 of word W.FETPT of
the FET.(Q.FETPT,Y.FETPT)
character set - taken from bits 12-17 of word W.FETCCC

of the FET (Q.FETCCC,Y.FETCC)

Note that this implies that for control ports, field C.CPPNBR (port number) of the control point area is ignored. Also, field FDCCC (connected file code) of the FNT is ignored.

2.6 Special CIO codes available to control ports.

To facilitate the job done by MANAGER, ARGUS, and the stimulator, which are the 3 control-port jobs which run on the 6500, 2 special CIO codes have been defined. These are extensions of the normal read and write CIO requests on connected files.

These requests are essentailly control-word read and write operations. The record read or written is prefixed by a 1 CM word control word, which conveys various information about that record. Bytes 0-3 of this control word are identical to the 4 byte control word which prefixes all records on the 7/32. (bits 0-7 of each byte are valid). Byte 4 of the control word is unused. The format of this control word is: (see also FESYM)

12/DHBCT, 12/DHTYPE, 12/DHCHC. 12/DHCTL, 12/unused

DHBCT = character count + 4 (L.DTAHDR)
DHTYPE = record type ((FP.xxx symbols)
DHCHC = CC.FDXXX character code
DHCTL = various control flags

2.6.1 IO.FRDNF - read native format.

The 4 byte header is a verbatim copy of the header in the 7/32, execpt that for OM files, the byte count is the number of valid display code characters written to the 6500 (+4).

DHCHC has no meaning.

DHTYPE will be FP.DATA (data), FP.EOF (EOF), or FP.EOR (EOR)

V.DHCEOL and V.DHCNEW in DHCTL will reflect whether this is a new line, or a continuation line.

Like all CIO reads by a control port, the record is translated according to the character set in the FET(FETCCC).

Unlike a normal read or readskip on a connected file, O.FRDNF never sends a prompt (unlock) to the 7/32. Also, O.FRDNF will never receive a "device capacity exceeded" error if the buffer is smaller than the line.

2.6.2 IO.FWTNF - write native format.

This request writes a single line, with control word, to a data

port on the 7/32. The record is translated by 1FP when it is written.

Bytes DHTYPE, DHCHC, and DHCTL are sent verbatim to the 7/32.

Byte DHBCT is reset by 1FP to be the actual translated character count, following the normal rules for translation of the record being written (i.e., the 6500 must supply the proper end-of-line terminator and padding to ensure that the exact number of desired character is written.

Translation is as defined in the FET (FETCCC). The 6500 program should ensure that DHCHC matches FETCCC, to avoid potential confusion by the 7/32.

DHTYPE should only be FP.DATA (data), FP.UNLK (unlock for a read), or FP.FEC (front-end command to the 7/32).

Any desired flags in DHCTL may be set.

## 2.7 Special CIO codes available to console jobs.

The following special CIO codes allow console jobs (including control ports) to perform special functions involving the 7/32.

### 2.7.1 IO.FRDM - read front-end memory.

Memory is read from the 7/32 to the users circular buffer in the 6500. The format is 1 8-bit byte from the 7/32 into 1 12-bit byte on the 6500, right justified. This request is also available to system programmers from normal HUSTLER jobs.

### 2.7.2 IO.FWTM - write front-end memory.

Data is written from the users circular buffer on the 6500 directly to memory in the 7/32. The format is bits 0-7 of each 6500 12-bit byte go to each successive 8-bit 7/32 byte.

For both these requests, the following holds:

1. The first word address of 7/32 memory for the transfer is in word FET+6 (W.FETRRI) of the FET (bits 0-23). The address is not divided by 2. The address must be even; 1FP will remove bit 0 from the address.

2. The length of the transfer is determined by the normal rules of circular I/O.

3. The unused-bit-count may be used to transfer other than a full CM word of data at the end of the transfer.

4. A read of non-existant memory on the 7/32 will return a

status of EOI. A write to non-existant memory will cause error code 10B to be returned to the FET if the EP bit is set, otherwise the job is aborted. 1FP sets the unused bit count at end-of-memory on both read and write.

5. Valid address ranges may be read from the DST as long as 1FP has been called in at least once (since 1FP set these fields):

C.DSTFCL = low core LWA+1/2**15
C.DSTFIF = lmbi FWA/2**15
C.DSTFLI = lmbi size/2**15
C.DSTFIF+C.DSTFLI/2**15 = LMBI LWA+1

2.7.3 IO.FHL - halt-load the front-end.

Up to 16 bytes of data are read from the user circular buffer and loaded into the LSU of the 7/32. The data is in the standard 8-in-12 format as described above. Then the 7/32 is halt-loaded via the 6500 interface. Data is read from the buffer as described for IO.FWTM, however, there is no 7/32 address in word FET+6 - only the UBC field is valid.

If the buffer is empty, no data is sent to the interface, but the halt-load is still done.

The following operation occurs on an IO.FHL:

1. The 7/32 system clear signal is brought up for 1/2 to 2 seconds. This is a master-clear to all the 7/32 I/O devices.

2. The 7/32 reads up data from its device 5, which is the interface RAM (LSU) which has been loaded with data from the circular buffer on the 6500. The 16 bytes are used as follows: (all bytes not supplied by the user are zeroed.)

bytes 0 - 1 are loaded into bits 16-31 of the Program Status Word.

bytes 2 - 3 are loaded into bits 48-63 of the PSW (the location counter). Bits 0-15 and 32-47 are set to zero.

bytes 4 - 5 become the FWA of the load.

bytes 6 - 7 become the LWA of the load.

bytes 8 - 15 are loaded into 7/32 core, starting at FWA. The ram is read circularly until LWA is reached, duplicating bytes 0 - 15.

The halt-load operation is identical to pressing the INI button on the 7/32 console, except that O.FHL loads the RAM but the INI button does not.

2.8 The stack request.

A special format of the standard 1SP stack request is used for requests for communicating with 1FP. This request replaces the disk position information with front-end information. The first word of the request is:

12/FEPT,12/FSTA,1/,1/FEC,1/,1/RES,1/NTA,1/PRMT,6/CCC,1/,1/EP,4/LEV,6/OC,
6/CP,1/FER,5/

FEPT = front-end port number
FSTA = FST address
FEC = 1 for front-end command. On O.FWRT, O.FWRTR, and O.FWRP, this causes the record written to the 7/32 to be type FP.FEC, rather than FP.DATA.
RES = 1 for restricted access (not implemented)
NTA = 1 for non-throw-away. On user abort on O.FWRT and O.FWR, tthis causes flag V.DCHNTA to be set in the record written to the 7/32.
PRMT =1 to send prompt character (*).
On O.FREAD and O. FRDSK, this causes 1FP to send an * as a prompt character when it writes on unlock to 7/32 to request input data.
CCC = connected character code (CC.XXXXX).
This determines the translation done by 1FP on all requests exept O.FRDM, O,FWTM, and O.FHL. Also, for O.FWRT, O.FWRTR, and O.FWRP, CCC is set into byte B.DHCHC of the record sent to the 7/32.
EP = 1 for error processing
LEV = EOR level number
OC = order code
FER = 1 (front-end request flag)

The second word of the request is the same as the standard 1SP disk request format.

Valid stack request fields based on the order code:

| | |
|---|---|
| O.FREAD | FEPT, FSTA, PPMT, CCC |
| O.FRDSK | FEPT, FSTA, PPMT, CCC |
| O.FWRT | FEPT, FSTA, FEC, NTA, CCC |
| O.FWRTR | FEPT, FSTA, FEC, NTA, CCC |
| O.FRDM | NONE |
| O.FWTM | NONE |
| O.FHL | NONE |
| O.FWRP | FEPT, FSTA, FEC, NTA, CCC |
| O.FRDNF | FEPT, FSTA, CCC |
| O.FWTNF | FEPT, FSTA, CCC |
| O.FWTBL | FEPT,FSTA,CCC |
| O.FWRBL | FEPT,FSTA,CCC,LEV |

OC, FER, and EP are valid on all requests.

2.9 PPU writes to the 7/32.

Any PPU may write data to a port on the 7/32 using the O.FWRP stack request. There are certain rules associated with this request:

1. Only 1 line of data may be written. 1FP always sets the new line (V.DHCNEW) and end-of-line (V.DHCEOL) flags in the record. The transfer stops at the LWA indicated by the request. For OM, any imbedded EOL's are converted to blanks.

2. The record will be a front-end command if FEC is set in the request.

3. The NTA flag will be set if NTA is set in the stack request.

4. 1FP always translates the record based on CCC in the stack request.

5. The PPU must set FEPT for the port to which it wants to send the data.

6. An FST is optional, as indicated by the appropriate flags in the 2ND word of the stack request.

7. If there is no room for the data in the 7/32 the request is rejected by setting the code/status in the PPU communications word to complete and error code 10B. The PPU is responsible for reissuing the request, swapping the job, etc. 1FP never initiates a swapout on O.FWRP.

2.10 Stack request error processing.

Connected I/O errors are handled through the regular CPSP/1SY mechanism. That is, 1FP sets the BX bit in the stack request, which tells CPSP the request is complete. 1FP puts an internal error code in the stack request code-status field, but does not set the complete bit. This tells CPSP to do error processing. CPSP interprets the error code, and sets the FET and FNT complete with the correct external error code.

## 3.0 System Programming Considerations

3.1 Installation History.

Modifications to CPSP (RFBSPFE), described in Section 4.5, were installed in LSD 45.09.

Installed in LSD 45.17 were:

| ident | routine |
|---|---|

```
RFBMTRFE MTR
RFB      CPCIO
         CP2TT
         1FP
```

Block transfer support code was added by IDENT DMKPRT1FP in LSD 48.05.

3.2 Assembly Options.

None

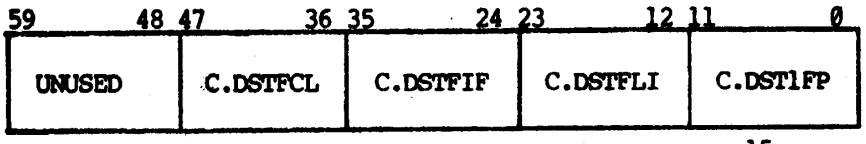
3.3 Cautions

None

3.4 DST usage by 1FP.

Stack requests are linked onto the DST for 1FP in the same manner as for disks. However, certain fields have a different meaning, as diagrammed below.

DST Word 2:




C.DSTFCL = 7/32 Low Core size in bytes, divided by $2^{15}$

C.DSTFIF = 7/32 LMBI FWA, divided by $2^{15}$

C.DSTFLI = 7/32 LMBI size in bytes, divided by $2^{15}$

C.DST1FP = 1 if 1FP is active on this DST entry.
(Set by MTR, cleared by 1FP)

## 4.0 Internal Reference Specifications

4.1 Flow of control for a read from a connected file:

1. User job issues a CIO RA+1 request on a connected file.

2. CPCIO begins processing of the call. This involves buffer pointer validation and preliminary FNT changes. CPCIO then determines that the request is on a connected file (device type=61B) and inititates CP2TT.

3. CP2TT uses information from the FET, the FNT, and the control point area for the job, and constructs a stack request describing the input operation. The request is built in the local TDB (task descriptor block) for CP2TT. CP2TT then passes this request to CPSP by linking it onto the CPSP request chain.

4. CPSP is initiated by MTR. When CPSP processes the stack request, it expands it to an 8 word format, moves it to the request stack area, and links it onto the stack request chain for the front-end DST entry (for device FD).

5. If 1FP is not already resident in a PPU, MTR initiates 1FP.

6. 1FP picks up the request from the DST chain. It moves to the control point of the job issuing the request.

7. 1FP checks the port on the 7/32 associated with the request. If there is no data in the port, 1FP writes an UNLOCK (with an optional prompt character of * if prompt is on) to the output side of the port, and sends a HEREIS command to the 7/32. Then the job is swapped out in WT.IN state. (If the UNLOCK could not be written, the job swaps out in WT.OUT state). 1FP places the reconstructed CIO request in the delay stack of the job. This will cause the original request to be reissued when the job is started up again.

8. When the user enters data, the 7/32 will send an INBS protocol record to the manager control port on the 6500. Manager will see the INBS, and since the job is swapped out in WT.IN state, MANAGER will call MAN to free the job.

9. The job will swap in and the CIO read request is reissued, retracing steps 1-6 exactly.

10. If there is data in the port on the 7/32, 1FP reads that data directly from the 7/32. It then translates it into the appropriate character set, and writes the data to the circular buffer on the 6500, updating the FET pointers.

11. The stack request is returned to CPSP, indicating successful completion.

12. CPSP sets the FET and FNT complete. If the job was in auto recall, it is restarted.

4.2 Flow of control for a write to a connected file:

1-6. Identical to read above.

7. 1FP checks the port on the 7/32 to see if there is room for a line of data.

8. If there is no room, 1FP swaps the job out in WT.OUT state. The original CIO request is placed in the delay stack for the job. The request will be reissued when the job restarts.

9. When the entire output stack on the 7/32 is empty, the 7/32 will send an FP.OTBS protocol record to the MANAGER control port on the 6500. MANAGER will see the OTBS, and since the job is in WT.OUT state, MANAGER will call MAN to free up the job.

10. The job swaps in, and the CIO request is reissued. Steps 1-7 above are repeated exactly.

11. If there is room for a line of data, 1FP reads data from the circular buffer on the 6500, and translates it to ASCII. 1FP stops at logical end-of-line, end-of-buffer, or 240 characters, whichever comes first.

12. The translated data is written directly to the next buffer on the 7/32 (a small buffer if the data is 80 characters or less, otherwise a large buffer).

13. 1FP sends a HEREIS command to the 7/32, informing it that a new buffer of data is available. The 7/32 adds the buffer to the port output circular list.

14. If there is more data in the user buffer on the 6500, 1FP goes back to step 11. Otherwise, 1FP passes the stack request back to CPSP, indicating that it is complete.

15. CPSP sets the FET and FNT complete, and restarts the job if it was in AUTORECALL.

## 4.3 CPCIO

CPCIO processes I/O requests that were made by CPU programs storing a CIO call in RA+1. CPCIO gathers general information and does preliminary validity checking on it. CPCIO is fully described in 6SM 121.

CPCIO selects a driver based on the file type. For connected files (device type=61B), CPCIO sets up a TDB and initiates task CP2TT.

## 4.4 CP2TT

CP2TT processes all CIO requests on connected files. It translates the CIO request into a stack request for 1FP, and issues that request through CPSP.

### 4.4.1 CP2TT is entered by CPCIO to process any I/O code on a connected file.

| Code | Action |
|---|---|
| 10 | READ - restricted to interactive HUSTLER jobs and control ports. A stack request for order code O.FREAD is formed and issued to CPSP. The following fields are set:<br><br>STPORT - port number. For HUSTLER jobs, taken from word W.CPPORT of the control point area. For non-HUSTLER jobs, taken from word W.FETPT of the FET.<br><br>STCCC - character code. For HUSTLER jobs, taken from field FDCCC of the FNT. For non-HUSTLER jobs, taken from word W.FETCC of the FET.<br><br>STPRMT - prompt flag. For HUSTLER jobs only, taken from Q.PROMPT in the control point area. |
| 20 | READSKIP - restricted to interactive HUSTLER jobs and control ports. A stack request for order code O.FRDSKP is formed and issued to CPSP. Fields are set as for READ. |
| 14 | WRITE - restricted to interactive HUSTLER jobs or to a control port. A stack request for order code O.FWRT is formed and issued to CPSP. The following fields are set:<br><br>STPORT - as for READ.<br>STCCC - as for READ. |
| 24 | WRITER - restricted to interactive HUSTLER jobs or to a control port. A stack request for order code O.FWRTR is formed and issued to CPSP. Fields are set as for WRITE. |
| 34 | WRITEF - identical to WRITER. |
| 414 | FWTBL (write block) - restricted to console jobs. A stack request for order code O.FWTBL is formed and issued to CPSP. The following fields are set:<br><br>STFEPT - as for read<br>STCCC - as for read |
| 420 | FWBER (writer block) - restricted to console jobs. A stack request for order code O.FWRBL is formed and issued to CPSP. The following fields are set: |

STFEPT - as for read
STCCC - as for read

420 FWBER (writer block) - restricted to console jobs. A stack request for order code 0.FWRBL is formed and issued to CPSP. The following fields are set:

STFEPT - as for read
STCCC - as for read
STLEV - taken from W. FETLVL of the FET.

424 FWBEF (writef block) - restricted to console jobs. A stack request for order code 0.FWRBL is formed and issued to CPSP. The following fields are set:

STFEPT - as for read
STCCC - as for read
STLEV - level number. Set to zero if an EOR is needed before the EOF, else it is set to 17B (EOF).
STFEOF - write an EOF after an EOR. This flag is set if an EOR is needed before the EOF. If set, 1FP will write a level 17B EOR (EOF) immediately after the EOR being written.

430 FRDM (read front-end memory) - restricted to console jobs and system programmers. A stack request for order code 0.FRDM is formed and issued to CPSP. STPORT, STCC, and STPRMT are not set.

434 FWTM (write to front-end memory) - restricted to console jobs. A stack request for order code 0.FWTM is formed and issued to CPSP.

444 FHL (halt load the front-end) - restricted to console jobs. A stack request for order code 0.FHL is formed and issued to CPSP.

450 FRDNF (read native format) - restricted to console jobs. A stack request for order code 0.FRDNF is formed and issued to CPSP. The following fields are set:

STPORT - port number, from word W.FETPT of FET.
STCCC - character code, from word W.FETCC of FET.

454 FWTNF (write native format) - restricted to console jobs. A stack request for order code 0.FWTNF is formed and issued to CPSP. Fields are set as for 0.FRDNF.

All other IO codes are ignored. The FET/FNT are set complete, and no error status is returned.

4.4.2 CP2TT expects the following fields in the TDB to be set by CPCIO:

```
CS      (code/status)
D.FA    (FNT address)
LENGTH  (FET length)
FETA    (FET address)
CPNUM   (control point number)
FIRST   (FET first)
LIMIT   (FET limit)
```

CP2TT uses the following additional fields in the TDB:

```
SCR1  (holds the IOTAB entry for the CIO code)
STK1  (holds 1st word of stack request)
STK2  (holds 2nd word of stack request)
```

4.4.3 CP2TT stack request setup.

CP2TT sets up the stack request as follows:

FEPT set for IO.READ, IO.RDSKP, IO.WRITE, IO.WRITF, IO.FRDNF, IO.FWTNF, IO.FWTBL, IO.FWBER, IO.FWBEF.
For HUSTLER jobs, set from word W.CPPNBR in the control point area.
For non-HUSTLER jobs, from word W.FETPT in the FET.

FSTA passed to it by CPCIO in the TDB.

FEC never set

RES never set

NTA never set

PRMPT on IO.READ, IO.RDSKP, set if a HUSTLER job, and prompt is on (W.PROMPT in the control point area)

CCC set on IO.READ, IO.RDSKP, IO.WRITE, IO.WRITR, IO.WRITEF, IO.FRDNF, IO.FWTNF, IO.FWTBL, IO.FWBER, IO.FWBEF.
For HUSTLER jobs, set from field FDCCC in the FNT.
For non-HUSTLER jobs, set from field FETCCC in the FET.

EP set from the EP bit in the FET.

O set based on the CIO code:

```
IO.READ  - O.FREAD
IO.RDSKP - O.FRDSK
IO.WRITE - O.FWRT
IO.WRITR - O.FWRTR
IO.WRITF - O.FWRTR
IO.FRDM  - O.FRDM
IO.FWTM  - O.FWTM
IO.FRDNF - O.FRDNF
```

IO.FWTNF - O.FWTNF
IO.FWTBL - O.FWTBL
IO.FWBER - O.FWRBL
IO.FWBEF - O.FWRBL

FETA FET address as passed by CPCIO

FIR FIRST from the FET W.FETFRS field

LIM LIMIT from the FET W.FETLIM field

LEV level from the FET FETLVL field (IO.FWBER,IO.FWBEF)

4.4.4 FNT/FET field specifications:

The only relevant field in the FNT is FDCCC (character code) for IO codes 10, 14, 20, 24, 34, 414, 420, 424, for HUSTLER jobs.

FET fields are only relevant for IO codes 10, 14, 20, 24, 34, 414, 420, 424, 450 and 454 for control ports. The fields are: FETCCC (character code) and FETPT (port number).

Note that other fields do have relevance for CPCIO, CPSP, and 1FP.

4.4.5 Error Aborts

If an error is detected, 6DM is called (via 1IO) to dayfile a message to the user. The FET/FNT is then completed with the error bits (FCSERR = 9-13 of FET word 0) = 22B. If the error processing bit is clear, the job is aborted. This is fully described in Section 2.8 of 6SM 121.

4.4.6 CP2TT is a very simple and straightforward routine.

CP2TT is a main loop which calls a set of subroutines. All subroutines return to their caller. (In general, X3 indicates an error code on return.) The main flow of CP2TT is:

1. Check the CIO code for a valid CP2TT operation.
2. Verify that the requestor is allowed to make this request.
3. Insert the port number into the stack request.
4. Insert the character set into the stack request.
5. Process any input timeout condition.
6. Set the prompt flag in the stack request.

7. Process EOR requests.

8. Issue the stack request to CPSP.

4.4.7 CP2TT Organization.

The CP2TT main loop controls the flow through CP2TT. The major subroutines are described below.

INST — Checks that the CIO code is for a valid CP2TT operation, and that the requestor is allowed to make the request. Sets up a skeleton stack request.

PORT — Sets the front-end port number into the stack request. The port number is picked up from the control point area or the FET.

CHARSET — Sets the connected file type into the stack request from the FNT or the FET.

INPTO — For a read request when the "input timeout" flag is set (Q.CPINTO) in the control point area, returns a carriage return (null line) to the job. Manager signals an input timeout by setting bit Q.CPINTO in the job control point area, and restarting the job.

PROMPT — For a read request, sets the prompt flag in the stack request if the prompt flag (Q.PROMPT) is set in the control point area.

ISSUE — Completes the stack request by inserting the order code, FST and FET addresses, and the control point number. Subroutine ESTK2 is called to actually pass the request to CPSP.

COMPLET — Sets the FET and FNT complete for any request which does not require a stack request.

PROCEOR — sets the STLEV and STFEOF flags in the stack request. If the I/O code requires a level number, the level number is extracted from the FET and inserted into the stack request. If the I/O code is an EOF, the level number is set to 17B if the last operation was a writer; otherwise the level number is set to zero and the STFEOF flag is set to tell 1FP to write an EOF after the EOR00.

4.4.8 IOTAB

CP2TT is driven by the IOTAB table. For each CIO code which causes an action on a connected file, there is an IOTAB entry which specifies the stack request order code, and various CP2TT

processing options. All additional I/O codes and options should be added using this table.

## 4.5 CPSP

CPSP is the CPU stack processor. It is fully described in 6SM 122. CPSP was originally designed to support only the disk driver 1SP. In order to support the front-end driver, 1FP, several modifications were made.

### 4.5.1 Front-end Request

All requests for I/O on connected files are issued either as CIO requests (from CPU programs) or as stack requests (from PPU programs). All CIO requests are processed by CPCIO, and thence by CP2TT. If CP2TT determines that I/O is required on a file, it issues a stack request to CPSP.

CPSP processes front-end requests similarly to disk requests. However, because there are no RBTS involved, the processing is greatly simplified. Front-end requests follow the same flow through CPSP as disk requests, avoiding all code which manipulates RBRS/RBTS. The overflow flow is:

1. CPSI picks up a new request and calls COPYSR to move the request to an 8-word slot.

2. COPYSR checks the request order code. If it is a front-end request it ensures that bit Q.STFER is set and kills the system if it is not.

3. ISR calls CFET and CFNT to copy relevant fields from the FNT and FET, followed by ESR.

4. FER performs special processing only for front-end requests. Currently this is setting bit Q.STBPT in the request for order code Q.FWRR (PPU write).

5. ESR enters the stack request into the proper DST. For front-end requests, it simply calls FEDST to return the proper DST ordinal. It then links the request into that DST.

6. FEDST is a routine to simply scan the DST for a front-end device which is on in the EST.

7. 1FP picks up the request from the DST chain, and processes it. It then returns the request to CPSP via the CE.SRX ICE request.

8. CPSP completes the request in the normal fashion.

### 4.5.2 The following routines do special processing for front-end

requests:

CFNT Only the FST address and code/status are copied to the stack request. All references to disk pointers are skipped.

CMPLT When reconstructing the FNT, all disk pointer fields are skipped.

COPYSR For non-front-end requests, flag Q.STFER is cleared. For front-end requests, flag Q.STFER must be set.

POST All analysis of the DST map is skipped.

ESR FEDST is called to find the DST for the front-end. All code for RBRS is skipped.

ISR FER and ESR are called directly for front-end requests.

4.5.3 Special Notes.

1. Throughout CPSP, bit Q.STFER in the stack request is used to key special processing for front-end stack requests. Subroutine COPYSR ensures that this bit is set correctly.

2. The ORDTAB table in ISR, which has one entry for each order code, tells COPYSR whether the particular order code is valid for the front-end. Any new front-end order code must be added to this table with the FER parameter.

3. Subroutine ESR, which links any front-end stack request onto the FD DST, assumes that MTR will start 1FP to process the request. CPSP never calls 1FP directly.

4.6 1FP

4.6.1 Overview

1FP is the PPU routine responsible for all I/O between a 6000 mainframe and a 7/32 front-end. 1FP has unlimited access to all of 7/32 memory. I/O is done by transferring "lines" of data between 6000 CM (or PPU memory) and buffers in the 7/32's LMBI, or common memory.

Control of data flow is managed in the 6000 by stack requests and control ports. In the 7/32, 1FP talks to control port, data ports, and the FPCOM area, through which 1FP talks directly to FREND.

1FP is called by the 6000 monitor whenever there is a stack request for the front-end. Once called, 1FP will execute in the same PPU until MANAGER's control port is closed and there are no

remaining stack requests. 1FP is therefore a dedicated PP to MANAGER--but MANAGER cannot call 1FP directly; it must open its control port and then make an I/O request on a connected file.

What 1FP does can be seen readily in the main loop, located at about address 1000 in the PPU (MAINLOOP). This is the flow:

1. See if FREND is alive and well.

2. Service MANAGER's Control Port, moving as many lines as possible in both directions; set the Control Port complete if FREND is dead.

3. Service the ARGUS and STIMULATOR Control Ports. Set them complete if FREND is dead or if MANAGER's Control Port is not open.

4. Process stack requests for up to 1/4 second or until none remain.

5. If stack requests remain or there is at least one Control Port open, go to 1.

6. Drop out until called again for stack request activity.

As well as doing I/O, 1FP is MANAGER's watchdog on FREND. FREND clears a flag in the FPCOM area (H.FEDEAD) at regular intervals. 1FP sets this flag (after checking its value) on every main loop pass. 1FP kills the 7/32 if H.FEDEAD stays set for too long. In addition, there is a flag in FPCOM that FREND can set to tell 1FP it has crashed. This allows 1FP to stop immediately when the 7/32 crashes.

In a crash, 1FP dumps itself into the 7/32, in a low-core buffer reserved for the purpose. 1FP then sets MANAGER's Control Port complete, which causes MANAGER to die and be dumped.

### 4.6.2 7/32 Interface I/O Subroutines

I/O to the 7/32 interface (6000 Channel Adapter, or 6CA) is uncomplicated. There are three operations that 1FP does routinely; these are Read, Write, and Test-set. In each, the PP must send functions to set the address register in the 6CA and initiate the operation, then transfer data over the channel. Read and Write simply access 7/32 memory at the specified address. Test-set is a special case of Read, in which one halfword (16 bits) is read, and the top bit of the halfword is set in the same memory cycle. The PP receives the previous value of the halfword. Except for dumping, data is always transferred 8-bits-in-12, one PP word for each 7/32 byte.

These three operations are done by the subroutines READ732,

WRIT732, and TEST732. Each of these routines has, as entry parameters, a 7/32 address, an offset to be added to the address, a byte count, and a PP buffer address. assumes FREND is dead. 1FP will then process only loading and dumping (TEST732 always reads 2 bytes, and returns the result in A, so it does not use the byte count or PP address.)

As an example, here is a typical sequence of PP instructions to read the 32-bit word W.PTIN from a data port:

```
SETK    PORTFWA,FEADLOC     POINTER TO 7/32 ADDRESS
SETK    PTIN,FEDATA         PP BUFFER ADDRESS IS PTIN
SETK    4,FEBYTES           NUMBER OF BYTES TO READ
CALLK   READ732,W.PTIN      READ TO PTIN
EXITIF  MI                  7/32 HARDWARE ERROR
```

The address at PORTFWA occupies three PP words:

```
VFD  8/0,4/bits 20-17 of fwa
VFD  4/0,8/bits 16-9 of fwa
VFD  4/0,8/bits 8-1 of fwa
```

Since the 6CA accesses 7/32 memory in 16-bit words, the address is effectively divided by 2 before it is sent to the 7/32. In 1FP, this is known as "6CA address format." This format is chosen because the address is sent to the 7/32 this way--in 3 functions, each containing a byte of the address. The last function also specifies the operation.

All 7/32 hardware driver subroutines exit with A=-1 if there is a hardware error. Any routine which calls a hardware driver subroutine, either directly or indirectly, must check the A register upon return--either to exit with A still negative, or to do some kind of error processing.

### 4.6.3 1FP/FREND Handshaking

1FP manipulates tables and buffers in the 7/32's memory, and it relies on FREND to describe these correctly. 1FP could be badly fooled if a malfunctioning FREND gave the wrong addresses for buffers or Port tables. For this reason, 1FP must be sure to start with that a good FREND program is running, and it must recheck this continually. This is implemented in the following way:

1. Whenever MANAGER initializes, it reloads the 7/32, and waits for it to initialize before opening its Control Port.

2. During initialization, 1FP checks to make sure MANAGER exists and has a Control Port open. If this is not the case, 1FP reads no addresses from the 7/32, and will do only loading and dumping (O.FRDM, O.FWTM, O.FHL) Stack requests. 1FP will

then drop out after completing all stack requests, because MANAGER's Control Port is not open.

3. When, during initialization, 1FP finds MANAGER's Control Port open, it assumes FREND has been loaded. 1FP then waits (for up to 1 second) for H.FEDEAD to clear. This tells 1FP that FREND is loaded and running.

3. During initialization, after 1FP has determined that FREND is running, 1FP reads the base addresses of the FPCOM area and the Port table. These addresses are never re-read, since they are never supposed to change.

5. After initialization, 1FP reads buffer addresses from FPCOM and the Ports. Every buffer address is checked to make sure it is a valid LMBI address, before it is used.

6. At least every 1/4 second, 1FP (subroutine FESTAT) test-sets the H.FEDEAD flag, which FREND must clear at least twice a second. If this flag remains set for one second, 1FP kills the 7/32 by halt-loading it. This stops FREND. 1FP then reloads itself, and during initialization it will determine that FREND is dead.

7. Subroutine FESTAT also monitors H.INICMP in the LMBI. This is the flag that FREND sets to tell MANAGER that initialization is complete. FREND also can put a code in this cell to tell 1FP it has crashed. In this way, 1FP can stop immediately if FREND dies other than by hanging.

8. The moment MANAGER's Control Port becomes complete, for any reason, 1FP assumes FREND is dead. 1FP will then process only loading and dumping Stack requests, and assume nothing about any LMBI tables. The same is true if MANAGER's error flag is set, or if there is no MANAGER.

9. Any time 1FP detects any kind of a problem with FREND, it kills the 7/32 (subroutine KILLER) by halt-loading it into a wait state (unless FREND has crashed), then dumps the PP memory into the 7/32 in the high end of low core. 1FP then reloads itself before setting the 6000 Control Ports complete.

### 4.6.4 Ports and How 1FP Uses Them

Port Table

(For a complete description of Ports, see FREND 6SM No. 124.)

The 7/32 has one Port in its common memory for each connection to the 6500. The Port fields used by 1FP are:

H.PTCN1 First connection number. 1FP checks this field to see if a data port is connected. If it is not, reads from the Port receive end-of-file, and writes are completely satisfied without transferring anything to the Port. This allows 6500 programs to complete while a user is being disconnected.

H.PTWTBF This halfword is a flag: 1FP sets the top bit whenever it causes a job to swap out waiting for output because the 7/32 had insufficient buffers available. This flag tells the 7/32 that it must send manager an output buffer status message when buffers become available, so the job can be swapped back in.

W.PTIN Input buffer address. This is the address of the next line of input from a Port, supplied by the 7/32 continuously. If 1FP sees this field zero, it means the connection has no input data. This address is divided by 2 (by the 7/32) for ease in addressing the interface to read the data.

H.PTINIK Input buffer address interlock. 1FP sets the top bit of this halfword to tell the 7/32 it is in the process of reading the input line. After the line is read, 1FP sends an ITOOK command to the 7/32, which then clears H.PTINIK, and supplies another input buffer address when available.

H.PTOTIK Output buffer address interlock. 1FP sets the top bit of this halfword when it is writing data to the 7/32. After the data is written, 1FP sends a HEREIS command to the 7/32, which causes the 7/32 to take the buffer and clear the interlock.

H.PTOTNE The number of empty output "slots" in the Port. This is the number of output lines that 1FP may write at any time. If this field becomes zero during a 1FP write operation, 1FP swaps the job out, WT.OUT.

H.PTNDIK The need-data interlock. Interlocks H.PTNDDT (J.PTOTBS and J.PTXFER). This interlock is held by 1FP whenever these fields are to be modified.

H.PTNDDT The need-data bits. 1FP clears J.PTOTBS and sets J.PTXFER when beginning a write or rewrite request, and then clears J.PTXFER at the end of the request. The PTOTBS bit says that a request for data has been initiated by FREND. The PTXFER bit says that a data transfer is active on the port.

4.6.5 FPCOM 1FP - 7/32 communication area

FPCOM — is a table in the 7/32 which is used for general communications. The following fields pertain to Port processing:

H.FCMDIK — 1FP command interlock. 1FP sets the top bit of this halfword before writing a command into FPCOM. When the command is processed, the 7/32 clears it.

W.FCMD — 1FP command. The first halfword of this word holds the 1FP command code; the second holds a port number. 1FP sends a command by setting the interlock, writing this word, then interrupting the 7/32.

There are currently 2 1FP commands: ITOOK, which tells the 7/32 an input line has been taken, and HEREIS, which gives the 7/32 an output line.

HEREIS comes in two types: FC.HI80, and FC.HI240. The difference indicates which size buffer 1FP used for the output data.

W.NBF80
W.NBF240 — These words contain the addresses of a free 80- and 240-character buffer, respectively. The 7/32 keeps these words filled, renewing them on each HEREIS command from 1FP.

H.NBUFIK — Next-buffer address interlock. When the top bit of this halfword is clear, the address in W.NBUF is valid. 1FP sets this bit before using the buffer, and the 7/32 clears it when it puts a new address in W.NBUF.

H.NOBUF — 7/32 short on buffers. The 7/32 sets this halfword nonzero when the number of free buffers in its buffer list drops below a certain threshold. Before starting on an output stack request, 1FP checks this flag. If it is non-zero, 1FP behaves as if the Port were full, and swaps the job out, WT.OUT. 1FP also sets H.PTWTBF in the Port. When the 7/32 regains its composure, it clears H.NOBUF, then searches the ports for any that 1FP swapped jobs out for. The 7/32 then sends output buffer status messages (FP.OTBS) to manager for these ports, in case the jobs should be swapped back in.

### 4.6.6 General 1FP/732 protocol

Almost all 1FP/732 interchange is based on the FPCOM table. There are two basic sequences:

1. Read (1FP reads data from 7/32)
2. Write (1FP writes data to the 7/32)

Read:

On a read, the 7/32 places the address of a buffer containing data, into word W.PTIN of the associated data port. When 1FP sees this word non-zero, it requests the PTINIK interlock, which tells the 7/32 that it is using this word. It then reads the data from the buffer. When it is done, it sends an ITOOK command to the 7/32, specifying the Port number. The 7/32 then releases the buffer pointed to by W.PTIN, and attempts to refill the PTIN word from the circular input list of the Port. The last thing the 7/32 does is release the PTINIK interlock, which tells 1FP that he can look at W.PTIN again.

Write:

On a write, 1FP first interlocks the output side of the Port by getting the PTOTIK interlock. 1FP then checks H.PTOTNE, which is the count of avalable cells on the output circular stack. If non-zero, 1FP gets a 7/32 buffer by reading a buffer address from W.NBF80 or W.NBF240, depending on how big a buffer it needs. It then writes data to this buffer. Finally, it sends a HEREIS command to the 7/32 specifying the Port number. This tells the 7/32 to refill NBUF with a new buffer, and to move the old address to the Port circular output stack. H.PTOTIK is then released by the 7/32.

### 4.6.7 Interlocks

There are 5 interlocks, all set by 1FP and cleared by the 7/32. These interlocks all have the same meaning:

1FP is altering or has altered the fields, and the only 7/32 routine which may process these fields is the 6500 ISR.

Thus, 1FP will not process a field until it can get the interlock, and the 7/32 ensures that all such interlocks have been set.

Interlocks:

| | |
|---|---|
| PTINIK | Port input interlock. Interlocks W.PTIN |
| PTOTIK | Port output interlock. Interlocks W.PTOTNE |
| PTNDIK | Need-data interlock. Interlocks H.PTNDDT. |
| NBUFIK | FPCOM next buffer interlock. Interlocks W.NBF80 and W.NBF240. |
| FCMDIK | FPCOM command interlock. Interlocks W.FPCMD |

### 4.6.8 Commands

1FP can send 2 commands to the 7/32:

1. ITOOK - 1FP read a buffer full of data

2. HEREIS - 1FP wrote a buffer full of data
   HEREIS comes in two types: FC.HI80 and FC.HI240, depending on which buffer size 1FP used.

All commands consist of the command ordinal, and the Port number to which the command applies.

1FP will set the FCMDIK interlock before writing a command to W.FPCMD. This interlock is cleared by the 7/32 only when it has finished the command, and is ready for another one. Interlock clear means ready for command.

Buffers:

W.NBF80 and W.NBF240 in FPCOM contain the addresses of an 80 character and a 240 character buffer, respectively, which 1FP may write into. 1FP always sets the NBUFIK interlock before using these buffer addresses. The 7/32 refills the buffer address used by 1FP, and clears the interlock when 1FP sends a HEREIS command. Whenever H.NBUFIK is clear, the buffer addresses are valid.

4.6.9 Interlock Management

Because of the 1FP - 7/32 interlock scheme, there is obviously great potential for 1FP to finish an operation without releasing all the interlocks it owns. For this reason, there is a direct cell called NILS, which always contains the number of interlocks that 1FP owns.

NILS is incremented only by subroutine GETIL, which is the only place interlocks are gotten. NILS is decremented by DROPIL, by FECMD, and by those subroutines which call FECMD, since a command will release more than 1 interlock.

At the end of processing a control Port or stack request, NILS is checked to make sure it is zero, and 1FP kills the front-end system if it is not. The only exception to this is in the presence of hardware errors, in which case there may have been a hardware error while attempting to reserve an interlock, and the number of interlocks 1FP holds is not known.

4.6.10 Port processing subroutines:

This is a brief description of what the routines in this section do.

CKPORT — Checks the Port for connected and enabled. Also this routine computes the Port address from its number. This is particularly important-- this routine must be called before starting an operation on a Port.

| | |
|---|---|
| TFL732 | This routine performs validity checking on addresses in the 7/32. It should be used on any addresses that aren't assembled into 1FP, since a reference to an invalid 7/32 address will appear as a hardware error. |
| GETIL | This is a general purpose routine, used to get hold of all 7/32 interlocks. It test-sets the interlock until it is found free, and kills the 7/32 if it has to wait too long. |
| DROPIL | Performs the reverse functin to GETIL, by writing 1 byte to the 7/32. |
| GETOBUF | Get output buffer address. This routine reserves the interlock H.NBUFIK, and reads the address of a free buffer from the 7/32. It also reads the first word of the buffer, to make sure it is a free buffer. The buffer size gotten is based on the data length to be written (RECORD+C.DHBCT). |
| SPORTI | Sense Port input status. This routine obtains the interlock H.PTINIK, and reads the address of the input buffer, if any. On exit, it tells whether or not input is available. |
| SPORTO | Sense Port output status. This routine gets the interlock H.PTOTIK, and checks the number of free output slots, H.PTOTNE. On exit, it indicates how many lines (if any) may be written to the Port. |
| WRTPORT | This routine writes a data record to a Port. It gets a buffer, writes the data to it, sends the HEREIS command (FC.HI80 or FC.HI240) to the 7/32. |
| RDPREC | Read a record from a Port. This routine assumes that SPORTI has been called. It reads the data buffer. |
| DOITOOK | Sends an ITOOK command to the 7/32. This is not done by RDPREC directly, because RDPREC is used by the control Port processor to read the byte count of an input record which may not fit in the control port circular buffer. |
| FECMD | Send front-end command. This routine gets the H.FCMDIK interlock, writes the command word, and interrupts the 7/32. |
| CKBUF | Check 7/32 buffer status. This routine reads H.NOBUF in FPCOM, and sets H.PTWTBF in the Port if H.NOBUF is non-zero. On exit, it tells whether or not output may be done. |

### 4.6.11 Control Port Processing

1FP subroutine PCP is the main control routine for processing a Control Port. It is called by MAINLOOP once for each Control Port. It does the following:

1. Make sure there is a control point with the correct Control Port number in W.CPFE; move to that control point, and get field access.

2. Examine the input FET for the Control Port. If the 7/32 is not running, set the FET complete.

3. If the FET is not complete, move lines from the 7/32 to the CM input buffer until there are no lines in the 7/32 or the next line won't fit in the CM buffer.

4. Examine the output FET. If the 7/32 is dead, set it complete.

5. If the FET is not complete, move lines from the CM buffer to the 7/32 until the CM buffer is empty or the 7/32 Control Port is full.

Control Port data is transferred verbatim in both directions. This means that in the 6500 CM, the first data byte (the first byte after the 4-byte header) will always be in byte 4 of a CM word. The header will occupy bytes 0-3 of the first CM word of the "line." The first header byte always contains the exact byte count for the line (including the header bytes). There are no EOL bytes.

Note that this is different from the CM data representation in Native Format, in which the header has a CM word all to itself, and the data starts in the next CM word.

If FREND is dead, or the front-end is turned off in the EST, or a hardware error occurs during Control Port processing, 1FP sets the Conrol Port FET complete with an appropriate error code in bits 9-13 of the code/status.

### 4.6.12 Stack Request Processing

1FP subroutine PRS is the main control routine for processing the Request Stack. It is called once every pass through the main loop, and it processes Stack requests until none remain or until 1/4 second has elapsed, whichever comes first. This time limitation ensures that the Control Ports get regular service, especially when the Stack request load is heavy.

Preliminary processing by PRS includes:

1. Abort requests without any processing if the front-end is turned off in the EST, by putting the error code in the FET and setting the BX bit in the stack request.

2. Abort requests when the 7/32 is dead, unless the requests are for loading or dumping (O.FRDM,O.FWTM,O.FHL). These requests do not require Ports, while all others do.

3. Move to the requesting control point (except O.FWRP), and get FL access.

4. See if the overlay needed to process this request is in core, and load it if not.

Next PRS calls one of several subroutines, depending on the order code, to do the real work. Each of these subroutines will process the request until it is complete, or there is an error, or the control point must be swapped out (WT.IN or WT.OUT).

These subroutines return with A=0 if there was no error, or the error code is in A. If a swap is required, the wait state is in location WAIT.

In normal request terination, PRS does the following:

1. Call subroutine SWAP, which will do an M.SWO request if the wait state in WAIT is non-zero.

2. Call subroutine CMPLT, which sets complete bits in the FET, FNT, and the stack request (direct cell copy). The FET is left busy if WAIT=0.

3. Set the BX bit in the stack request, which tells CPSP the request is not to be reissued.

4. Release FL access.

5. If the request accessed a 7/32 Port, set the flag in H.FPDONE to tell the 7/32 to send port status to MANAGER.

6. Call subroutine ENDSR, which writes the Stack request back to CM and does a CE.SRX monitor request. This gives the request to the CPU for final processing.

In the case of errors, PRS finishes the request this way:

1. Put error code in the code/status field of the Stack request. This is an internal error code, which CPSP and 1SY will use to select the dayfile message and FET error code, as defined in comdeck =DSKERR.

2. Set the BX bit so that the CPU will not reissue the request. Since the code/status field of the request is not complete the CPU will do error processing.

3. Release FL access, an set the H.FPDONE flag if the request accessed a Port.

4. Call subroutine ENDSR to write the request to CM and do the CE.SRX request, to give the Stack request back to the CPU.

### 4.6.13 Field Access

1FP, like 1SP, does not use the SCOPE 3.4 method of Field Access. The justification for this is the same as for 1SP--that is, if 1FP had to wait for a storage move during a Monitor request, it might result in system lockups.

This is clearly true in the case of 1SP. In the case of 1FP however, it turns out that FL access is not needed while any Monitor requests are done; 1FP could have been written as a SCOPE 3.4-type PP. 1FP is the way it is because the control-point assignment and FL access code was taken almost verbatim from 1SP. It was left this way after considering that it would be easier to maintain: it would not be necessary to make sure FL access was dropped when adding new Monitor requests.

FL access is released by the regular call to R.TFL. It is gotten by subroutine ASSIGN, before processing a new Control Port or Stack request. ASSIGN manipulates the PP status word directly, instead of calling R.RAFL, so as to avoid waiting for storage moves. If the storage move flag is set at a control port, processing is skipped: Conrol Port processing will be done on another main loop pass, and Stack requests are reissued.

### 4.6.14 Swapping

1FP swaps user jobs for one of two reasons:

WT.IN — There is no input data in the user's 7/32 Port. MANAGER will cause the job to swap in when the 7/32 sends an INBS message to MANAGER, indicating that the job has input.

WT.OUT — The job's port on the 7/32 is full. MANAGER will swap in when he receives an OTBS message indicating the Port is empty (or nearly empty).

To swap a job, 1FP changes his input register temporarily to CIO, while he makes the M.SWO request with recall of PPIR.

If the job is a console job, 1FP will just set the FET complete,

rather than attempting a swap. Console jobs normally have Control Ports, and they should therefore know the exact status of Data Ports which they control. MANAGER and ARGUS will know if there is input data to read, or if the Data Port has room for output, so this situation should not arise.

### 4.6.15 1FP Overlays

1FP has two levels of overlays. The first overlay level, the 6Fx routines, contain the various order code processors. These overlays load immediately following the I/O buffers.

The second overlay level, the 7Fx routines, contain the translation tables needed for the first level routines. These overlays load immediately after the 6Fx ones.

The overlays are:

- 6FC for CPU read/write orders.
- 6FP for PP read/write orders.
- 6FM for processing of O.FRDM, O.FWTM, O.FHL.
- 6FW for block read/write orders.
- 7FO, 7FA, 7FF, 7FB for translation according to file connect type (OM, AS, AF, BI respectively).
- 7FH for dayfiling hardware error messages.

1FP order codes are divided into two distinct groups--those that reference Data Ports, and those that do not. The former transfer character data and some need a translation overlay; the latter need no translation, and are all processed by overlay 6FM.

Hardware error messages are dayfiled only after Stack request processing is finished and the overlay in core is no longer needed. This is how all these overlays can occupy the same core.

Overlays (except 7FH) are loaded by subroutine ORDEROV, on the condition that the required overlay is not already in core. Thus all overlays except 7FH must be re-entrant.

### 4.6.16 7/32 Hardware Error Processing

1FP treats most of the 7/32 hardware errors it checks for as fatal. It retries only in the case where the channel never becomes full after a Read (or Test-set) function hs been sent to the 7/32. This is the only type of hardware error that has ever appeared to be recoverable. If this condition occurs, 1FP makes one retry attempt. So far, this has proven 100% successful.

When 1FP detects a fatal hardware error condition, it immediately completes the Control Port transfer or Stack request it was working on, without any further action on the I/O channel. The standard signal for this to set the A register negative and exit the hardware driver subroutine. All higher level subroutines must check for A<0 and exit, until the highest level routine is returned to. This routine (PRS, PCP or MAINLOOP) must clean up its operation, then call subroutine HWMSG to do something about the error.

Subroutine HWMSG loads overlay 7FH and calls it by return jump. On return, HWMSG jumps to R.IDLE, so that the PP will be reloaded. This is done because of the possibility that the apparent hardware error is caused by something wrong in 1FP code.

Overlay 7FH formats a message for the operator, which contains all relevent information about the error. This message is flashed at the bottom of the right screen until the operator acknowledges. This is done to allow the operator to read more diagnostic information from the lights on the 7/32's front panel, before this information is destroyed by more 1FP activity. Because of space limitations, the right-screen message is severely abbreviated--these messages are described in great detail in the FREND Operator Guide. They are listed in section 7.1 below.

When 1FP is reloaded after a hardware error, it goes through its regular initialization, during which it will make some crude checks of the hardware. Any hardware errors during initialization will cause 1FP to declare the 7/32 dead, and set all Control Ports complete as they are first processed in the main loop. This is how MANAGER may be killed in the event of a catastrophic hardware error.

### 4.7 MTR

MTR was modified to call 1FP if there is at least one outstanding stack request in the 'FD" DST, and 1FP is not active. Subroutine I1FP is called by the MTR mainloop. This routine scans the entire DST for DSTs of type DT.FD. For each such DST entry, if the "1FP-active" flag is not set, 1FP is initiated by calling subroutine APPJOB. Note that this scheme will handle multiple FD DST entries.

## 5.0 Operator Communications and Procedures.

5.1 1FP generates a number of operator error messages which require operator intervention before processing can continue. These are fully described n sectin 8 of the FREND SYSTEMS OPERATOR GUIDE.

5.2 Turning off device FD in the EST will cause all front-end requests to be aborted by CPSP. This terminates all communicatin with the 7/32 by 1FP.

## 6.0 User Aspects

6.1 User aspects relate to the rules for doing input/output on connected files. These are fully described in sections 2.3 and 2.4.

## 7.0 System File Changes

7.1 1FP Hardware Error Messages.

These messages are issued to the system dayfile after a 1FP hardware error is acknowledged by the operator.

```
FD RD ERR  AAAAAA+BBB  SSSS
         Bad status after A front-end read

FD RD DCN  AAAAAA+BBB  SSSS
         Front-end read prematurely disconnected

FD RD HNG  AAAAAA
         No data from 7/32 on Front-End read

FD WT ERR  AAAAAA+BBB  SSSS
         Bad status after Front-end write

FD WT DCN  AAAAAA+BBB  SSSS
         Front-end write prematurely disconnected

FD WT HNG  AAAAAA
         Data from 1FP not taken by the 7/32

FD TS ERR  AAAAAA      SSSS
         Bad status after a Test-set operation

FD TS HNG  AAAAAA
         7/32 sends no data on test-set operation

FD HL HNG              SSSS
         Halt-load operation doesn't complete

FD LD DCN        +BBB  SSSS
         Write-interface-memory prematurely disconnected

FD LD HNG
         7/32 doesn't take data on write-interface-memory

FD FN HNG              FFFF
         7/32 doesn't accept function from 1FP

FD ST HNG
         7/32 won't send status to 1FP
```

AAAAAA = Last address sent to the 7/32.
BBB = Number of bytes transferred.
SSSS = Status after last operation.
FFFF = Last function sent to the 7/32.

7.2 I/O Error Messages

BAD CHARACTER CODE
Issued by 6SM when 2TT Detects an invalid connected file code.

ILLEGAL FRONT-END PORT
Issued by 6DM when 2TT detects an invalid (zero) front-end port number.

BAD FET BUFFER POINTERS (FRONT-END I/O)
An error in the FET on connected Read or Write

UNRECOVERABLE FRONT-END I/O ERROR
Hardware error on connected Read or Write

LINE TOO LONG FOR BUFFER (FRONT-END I/O)
Input line too long on connected Read

FRONT-END INOPERATIVE
The front-end has crashed during connected I/O

BAD ABSOLUTE FRONT-END ADDRESS
Attempt to Read or Write non-existent core in the 7/32

1FP KILLED 7/32
1FP has caused the 7/32 to crash, because of some fatal front-end system inconsistency.

1FP- DEAD 7/32
1FP has decided that the 7/32 has crashed.

8.0 REFERENCES

SMP 28 - MSU front-end.
SMP 49 - MSU front-end, phase 1.
SMP 60 - front-end command and control.
6SM 131 - Merit interactive support.
6SM 124 - FREND.
6SM 135 - MANAGER and frends.
FREND OPERATOR GUIDE.
INTERDATA 32-BIT SERIES REFERENCE MANUAL.
MODEL 7/32 REFERENCE MANUAL.

WRITTEN BY: R Bedoll John K Renwick David M. Katz
R. Bedoll, J. Renwick, D. Katz