EMAS 2900 FEP NOTE

No: 1 (revised)
Date: 2/5/79

# The EMAS 2900 Front End Processor

## Contents



## 1. Introduction

This note describes the system software running in the EMAS 2900 Front End Processor, which is a DEC PDP11/34. It also describes operating procedures.

Section 2 summarises each software component in turn; subsection 2.1 then described the AM1 handler in greater detail. It is intended that further subsections will be added in due course, each likewise dealing with a single component in detail.

Section 3 is taken largely from the EMAS 2900 Operator's Manual. It describes the operational aspects of the FEP.

Figure 1.1 below indicates how the software components of the FEP are interlinked, and, for the ERCC 2970 installation, how the FEP is connected to the EMAS 2900 mainframe on the one hand and the Regional Computing Organisation (RCO) Network on the other. The protocol hierarchy for interactive and non-interactive operation is shown diagrammatically in Figures 1.2 and 1.3, and for completeness the formats of the various protocol packets are also given (Figure 1.4).

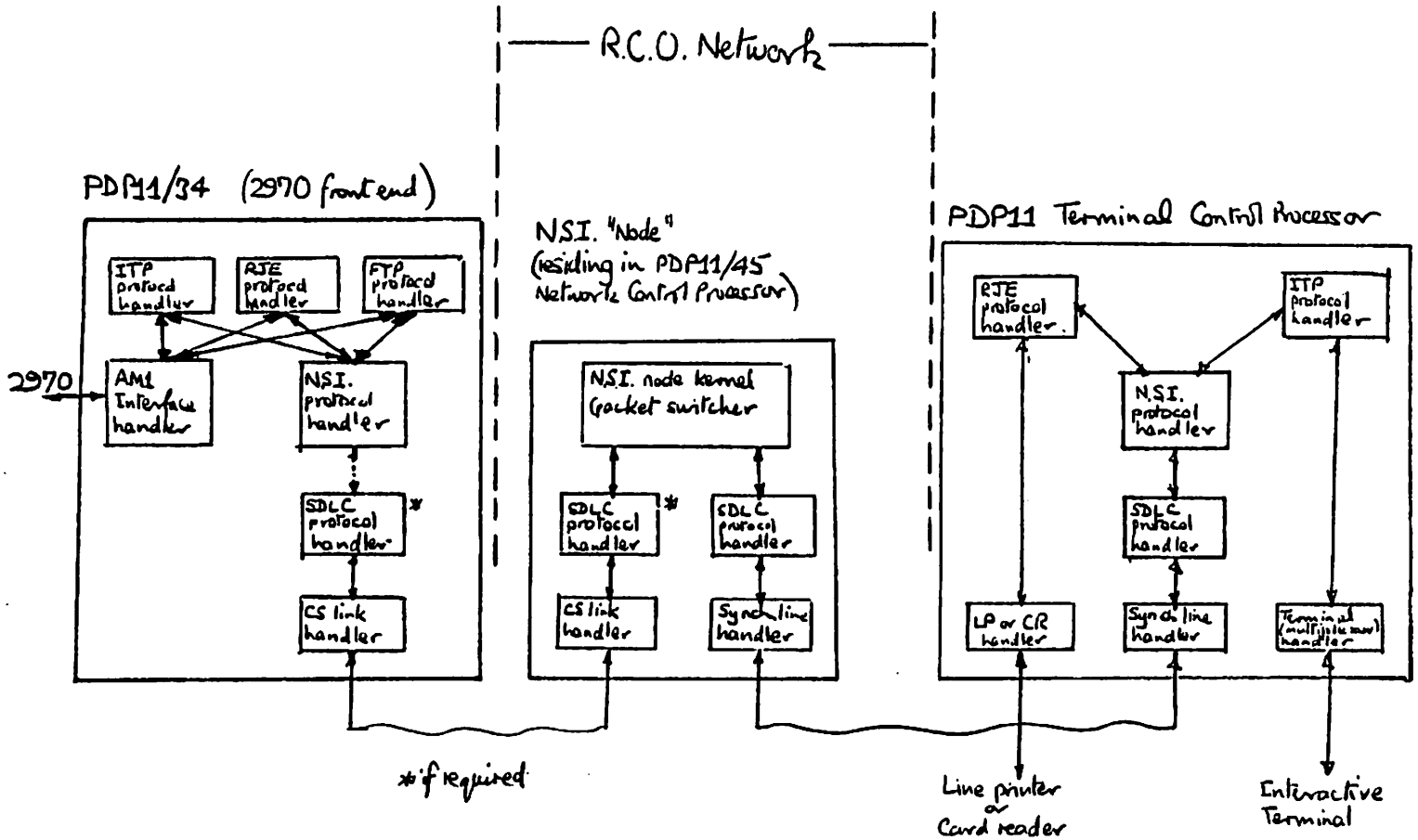



Figure 1.1: Hardware and Software Configuration for ERCC 2970

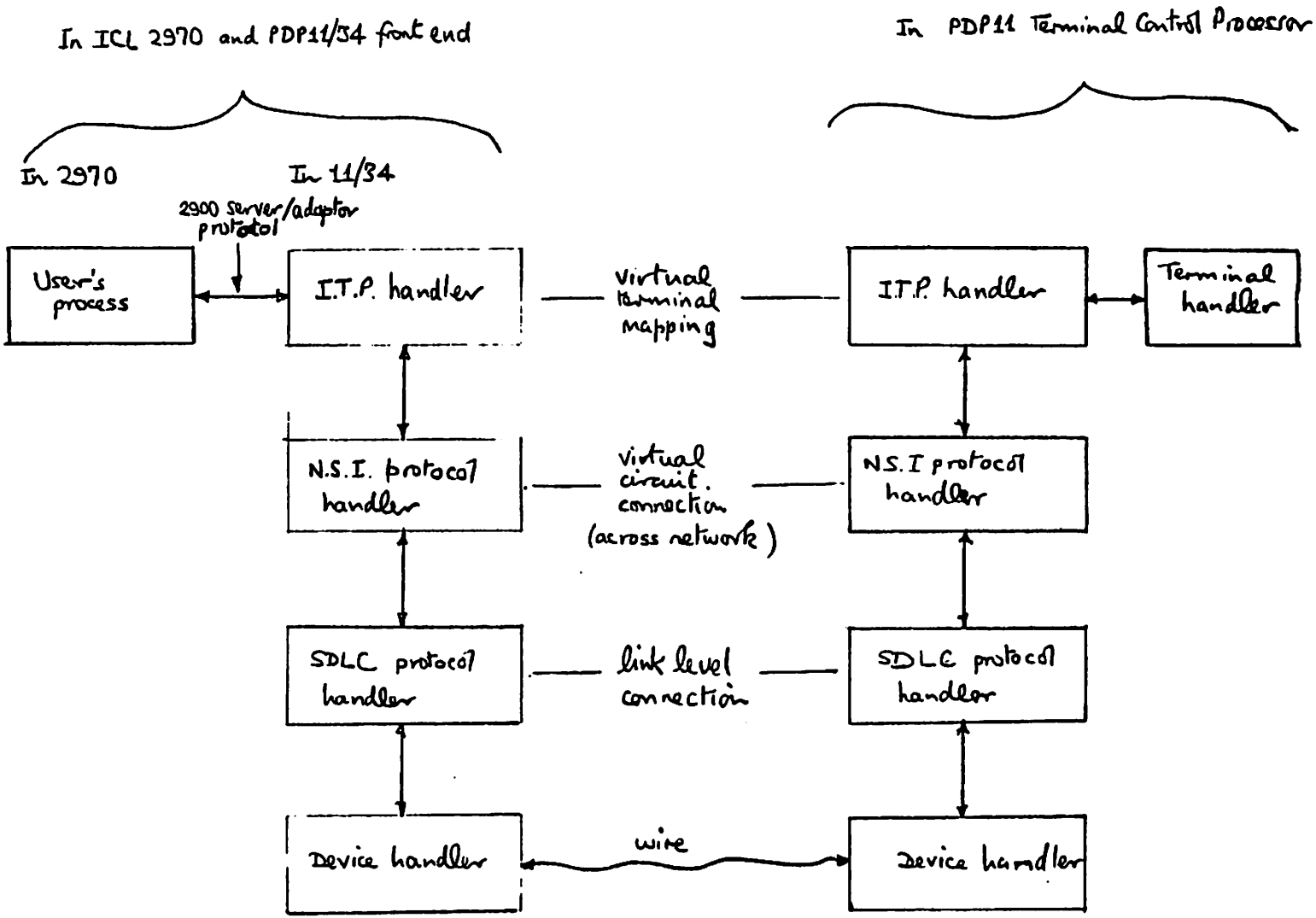
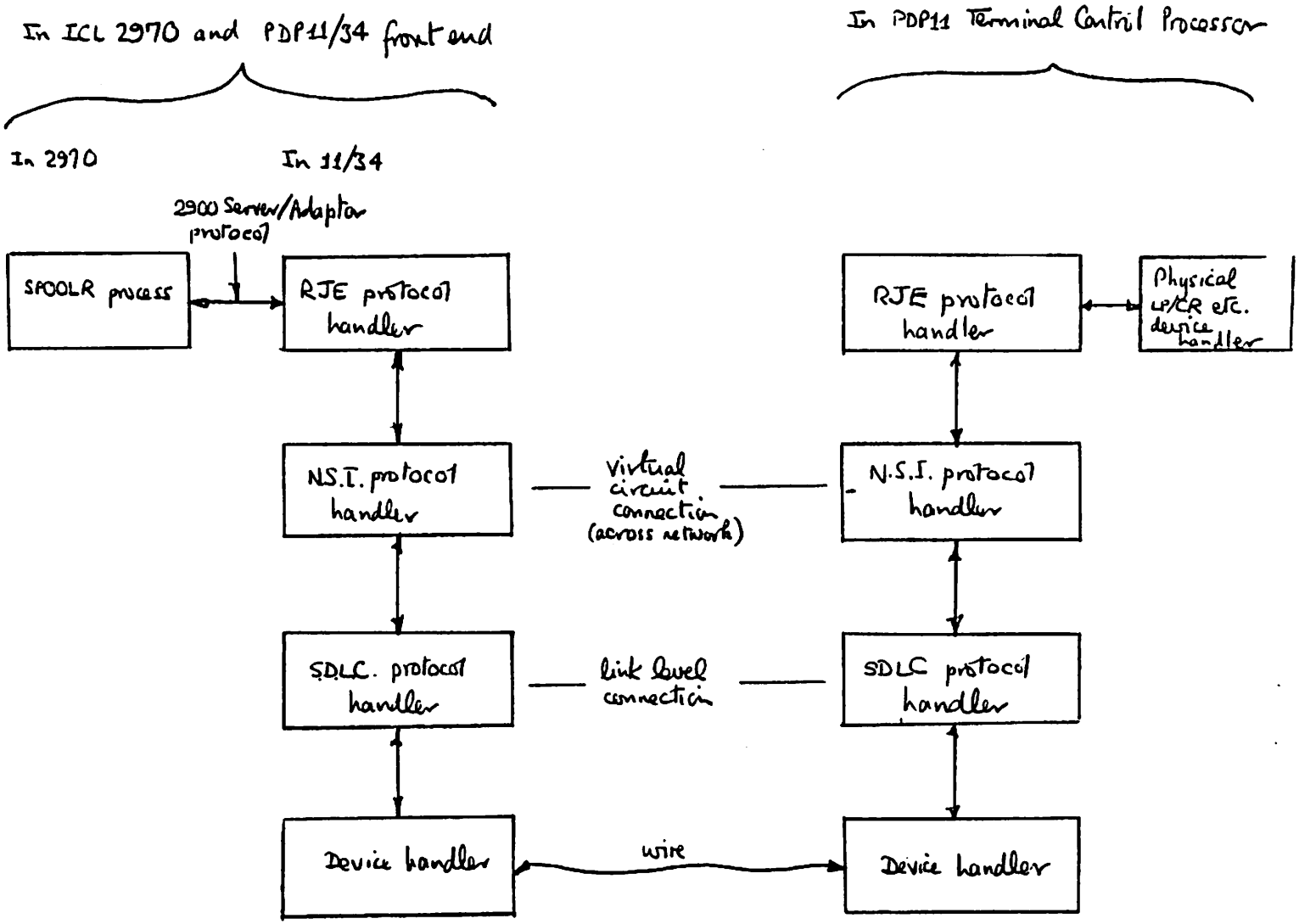


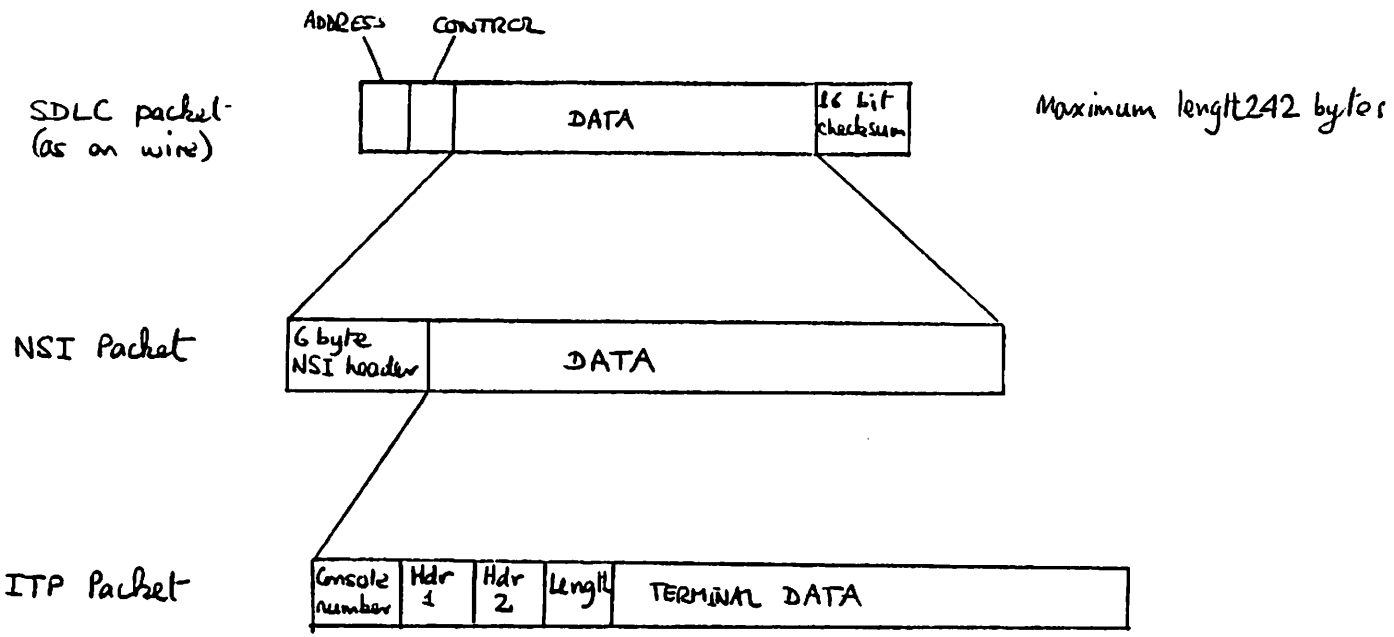

Figure 1.2: RCO Network Protocol Hierarchy - Interactive




Notes: N.S.I. — Node Standard Interface.
The structure for a File Transfer Protocol (FTP) is essentially the same, although it will normally be used for mainframe/mainframe communication.

Data flow — ⟷

Figure 1.3: RCO Network Protocol Hierarchy - Non-interactive




Notes: I.T.P. – Interactive Terminal Protocol
N.S.I. – Node Standard Interface

Figure 1.4: RCO Network - Packaging of Interactive Data

## 2. Software Components

The software consists of a number of co-operating programs, each running as a task under the operating system DEIMOS. (DEIMOS is a general purpose operating system designed for use in PDP11s with memory management.)

The left hand part of Figure 1.1 above shows the connections between the programs which drive the FEP. The arrows indicate the interfaces between the tasks. In addition all the tasks interface to the buffer manager and to the common TT output handler.

It should be obvious that nearly all the above tasks are critical; e.g. if the NSI handler crashes, then everything will stop. The only exception is the RJE handler, the failure of which will only bring down the RJE part of the service.

Each of the programs in the 2900 FEP is now summarised in turn.

1) AM1 Handler [Task name: AM1H]

The AM1 handler handles the "protocol" necessary on the AM1 interface. For example, the 2970 will inform the handler that it is prepared to make a transfer on a stream 'n'. The handler determines what use the stream is put to, and instructs the appropriate protocol handler. The ITP handler, for example, will then read (or write) characters from the 2900, enclose the ITP packet headers round it and pass the "message" to the NSI interface module. This module adds on the NSI packet header, and passes the "message" to the CS link handler for transmission into the Network.

2) Interactive Terminal Protocol (ITP) Handler [Task name: ITPS]

This program handles the protocol-dependent features of the ITP For example, after receiving user output from the 2900, it will add the protocol header to the output and, obeying the protocol flow control rules, will pass the output to the Network Standard Interface (NSI) protocol handler, for transmission into the RCO network.

In addition, the handler controls the virtual connections along which the data to and from each Terminal Control Processor (TCP) flows. Each TCP sets up one connection to the FEP along which all terminals connected to that FEP communicate.

3) RJE Protocol Handler [Task name: RJES]

This handler controls all remote file transfers to and from the EMAS 2900 mainframe.

Data formatting and flow control on each virtual connection is performed by this module. Messages from operator consoles, i.e. logons and queue enquiries, are passed through to the mainframe with the protocol headers removed. Messages to operator consoles are processed into the correct form and passed to the NSI handler.

4) Network Standard Interface (NSI) Protocol Handler [Task name: NSIH]

This handler processes the protocol-dependent features of the RCO NSI, passing the information contained within to the high-level protocol modules (ITPS and RJES), in a near network-protocol-independent manner. For example, when a high-level protocol accepts a data packet from the handler, it forms the network-dependent "SEND BLOCK RESPONSE" packet and transmits this through the network. In the case of a different protocol, e.g. X25 level 3, the handler would construct a "window change" or 'RR' packet from the same response from the higher level.

5) Computer Science Asynchronous Link Handler [Task name: CSLK]

The handler receives blocks from the NSI handler and transmits them, one character at a time, down the link. It blocks up the character record from the link and passes them in a block to the NSI handler. In addition, in contains watchdog timers to establish when the link goes down.

6) File Transfer Protocol (FTP) Handler

This module is still to be written; its function will be to transmit and receive files from the network in the EPSS Study Group 3 FTP protocol.

7) Buffer Manager [Task name: BUFF]

This program controls all the data buffers used in the FEP system. All other tasks send requests for buffers to the buffer manager and return buffers to it after they have been used.

8) HDLC Protocol Handler [Task name: PROT]

On systems which communicate to the network via a synchrous line an HDLC protocol handler is used instead of the CS link handler. This module performs the same functions as CSLK, but ensures an error-free path by implementing HDLC at the link-to-link level.

For details of how these tasks communicate with each other, and other features of the FEP system, consult the DEIMOS User Manual, by B.A.C. Gilmore (ERCC).

## 2.1 The AM1 Handler

The AM1 handler controls the PDP11 hardware interface with the 2900 mainframe. In the case of the 2900 P series the interface is via the GPC, with 2900 S series via the DCU. The handler also controls the low-level functions of the software "protocol" that is used to communicate between a 2900 and a PDP11.

The handler has three distinct functions:

1) To handle "internal" transactions with the 2900

2) To handle data transactions with the 2900

3) To interface to the higher-level protocol tasks

These are described in turn below (subsections 2.1.1 - 2.1.3).

---

### 2.1.1 "Internal" transactions with the 2900

There are a number of control signals identified by bit 0 set in the receive status register sent out by the 2900 which are completely handled by this task. The response to each is defined either by the 2900 interface hardware or software.

The functions, and their responses, are:

a) Send Property Codes

This is sent to the PDP11 on every IPL of the 2900, and is used by the 2900 hardware to establish how many front ends (if any) are currently available.

The response is 4 data bytes, followed by a normal primary status:

14,0,0,0, 16 + control bit.

The '14' defines a front end to EMAS 2900.

b) Identify

Identify is produced by the GPC microprogram (in a P series 2900) to determine which of the possible mechanisms on the interface has set an "attention bit" in primary status. This is not actually used in the FEP case, as there is only one mechanism, but nevertheless the identify must be replied to.

This response is two data bytes and a normal primary status:

0,X'80', 16+control bit.

c) Sense

This command is normally generated by EMAS 2900 software to obtain the secondary and tertiary status bytes after certain conditions, e.g. send property codes. Currently no "real" information is actually transferred to the mainframe. Response: 2 data bytes and primary status:

0,0, 16+control bit.

d) Limit

Limit is sent by the 2900 when it has read the number of bytes specified in its current command. The action taken depends on the state of the handler. If it is handling one of the above control signals, e.g. identify only expects two data characters (so that a limit is seen before the primary status can be transmitted), then a primary status is sent and earlier, still pending, actions are ignored.

In the second case, the interface will be in a "user pending or writing to the 2900" state (see below). The action in this case is still to send a primary status.

There are other possible control signals, e.g. initialise, autoload, which are not used by EMAS 2900. If any is actually received, the response is a primary status with "unsuccessful" set.

---

### 2.1.2 Data transactions with the 2900

All data transactions to or from a 2900 consists of two parts. First, a four-byte part describing the logical stream which the transfer is to to be on (this also determines the direction), and the maximum length of that particular transfer. This four-byte "header" always goes from the 2900 to the FEP. The second part of the transaction is the actual data; this can be transferred in either direction.

From the point of view of the handler, the actual transaction is as follows:

a) A write control command signal, followed by four data bytes - two for the stream number and two for the length of transfer, followed by a limit command - to cate the end of that part.

b) The handler must now send a primary status to the 2900, to signal the completion of the command.

c) The 2900 will now send either a READ command or a WRITE command, depending on the direction of the transfer. A number of data bytes is thus transferred in the relevant direction and the command again terminates by a primary status.

There are two distinct types of data transfer: a control stream transfer, and a transfer on any other streams. The distinction is only made for the convienience of handling control stream transfers (see below) which are done by the AM1 handler itself.

A control stream transfer (stream numbers -1 and -2) is done in units of 24 bytes. In the case of an outward transfer, the 2900 sends up to 8 units simultaneously. These are read by the handler, the stream number to which they refer is determined, and the entire message is then passed to the higher level protocol which "owns" that particular stream. In the case of inward transfers, the higher level protocol tasks send messages containing 24 bytes to the handler which queues them and, when permitted, sends up to 8 of them into the 2900 in any one transfer.

### Control of transfers

If the link is idle, the 2900 may at any time start up a transfer to the FEP. If the FEP wishes to initiate a transfer to the 2900, it must send an ATTENTION to the 2900, which should then respond with a request to do an inward control transfer. If a transfer is in progress, the FEP may indicate its desire for a transfer by setting the ATTENTION bit in the next (and subsequent) primary statuses that it sends to the 2900.

Note: All transfers from the FEP to the 2900 are initiated with a control transfer on a stream that requests the 2900 to initiate an actual data transfer on the particular stream. Several of these may be outstanding at any time, and it is the prerogative of the 2900 to decide which it honours first.

### Stream transfer

When a transfer is initiated on a stream other than the two control streams, the handler passes physical control of the link over to the higher level protocol task and lets it do the actual transfer. When it is finished, or when the other task senses a command on the link (e.g. a LIMIT), it passes control back to the handler, which performs the primary status.

### 2.1.3 Interface to the high-level protocol tasks

There are six messages from the handler to a higher task (parameters in brackets):

| | | |
|---|---|---|
| 0 | Pass interface address (physical address of interface) | |
| 1 | Do input (stream, max trf) | do data transfer from 2900 |
| 2 | Do output (stream, max trf) | do data transfer to 2900 |
| 3 | Message (address of message) | control message from 2900 |
| 4 | Mainframe up | state of mainframe |
| 5 | Mainframe down | state of mainframe |

There are five messages from high level protocol tasks to the AM1 handler (parameters in brackets):

| | | |
|---|---|---|
| 0 | Send high level control message (address of message) | to be sent on inward control stream |
| 1 | Send low level control message (address of message) | to be sent on inward control stream |
| 2 | Here I am (stream no) | claim of stream no |
| 3 | Return control (action) | pass control of link back to the handler |
| 4 | Stop | tells the handler to close down |

There are a number of actions that can be performed on "return control". The values and actions are:

| | | |
|---|---|---|
| 0 | Send normal primary status | |
| 1 | Send normal primary status + short block | significant information transferred, e.g. EOF |
| 2 | Send normal primary status + long block | transaction not satisfied - 2900 should put another one on |
| 3 | Send unsuccesful primary status | |
| 4 | Send primary status + condition | |
| 5 | Same as 0 | |

In addition, if the top bit of the flag is set, then the handler should accept the test characters sent from the 2900; otherwise it should not.

### 2.1.4 AM1 handler fault messages

The AM1 handler produces error messages (on the FEP console) of the following form:

AM1: FAULT x n1,n2

where: x is a character indicating the class of fault
n1,n2 specify a subclass and/or give extra information

Note: except where stated otherwise, "input", "output", "receiver" and "transmitter" are with respect to the PDP 11/34.

Classes

x= T - Timeout (character level) - a character from the 2900 was expected (i.e. in the middle of a sequence) but did not arrive within 20 ms (approx).

n1= state of receiver side of AM1
n2= state of transmitter side of AM1

x= I - Input fault

n1= 1 - after initial WRITE CONTROL (w.r.t. 2970), four characters should be received before a limit; n2 gives the number of characters actually received.

2 - a transfer request has been made by the 2970, on a stream that has not been CLAIMED by a higher level task; n2 is the stream number.

3 - a READ (w.r.t. 2970) has been issued on a stream >0 but a number of non-control characters have also been received; n2 is the number of characters in error.

4 - same as 3 except a WRITE (w.r.t. 2970) has been issued.

5 - an unexpected control character has been received from the 2970; n2 is the control character.

6 - non-control characters have been received before a control character; n2 is the number of characters.

x= S - A CONTROL message (high or low level) has been issued to a stream that has not been CLAIMED by a higher level task.

n1= number of offending stream
n2= offset into the entire block received from the 2970

x= O - Unexpected OUTPUT interrupt. An interrupt has been generated on the transmitter side of the AM1 - this should not have occurred.

n1= 0
n2= output state of AM1

x= E - Transmitter state fault. The transmitter side of the AM1 has got into an error state.

n1= output state of AM1
n2= previous output state of AM1

x= R - Return control fault. A RETURN CONTROL has been received from a higher level task at a time when the AM1 handler has not released control.

n1= input state of AM1 (w.r.t. 11/34)
n2= output state of AM1 (w.r.t. 11/34)

x= C - Clock timeout (or major timeout). A request for transfer on the control stream, or an output interrupt, has been outstanding for 15 seconds. This will generate a message to the higher level tasks informing them that the 2970 has gone down.

n1= 2 - transfer outstanding
n2= 1 - interrupt outstanding

## 3. Operating the EMAS 2900 Front End

The following description relates specifically to the operation of the FEP on the ERCC 2970.

The EMAS 2900 Front End Processor, a DEC PDP 11/34, is connected to both GPCs on the 2970 mainframe by a locally produced NRPI link. Figure 3.1 illustrates this connection.

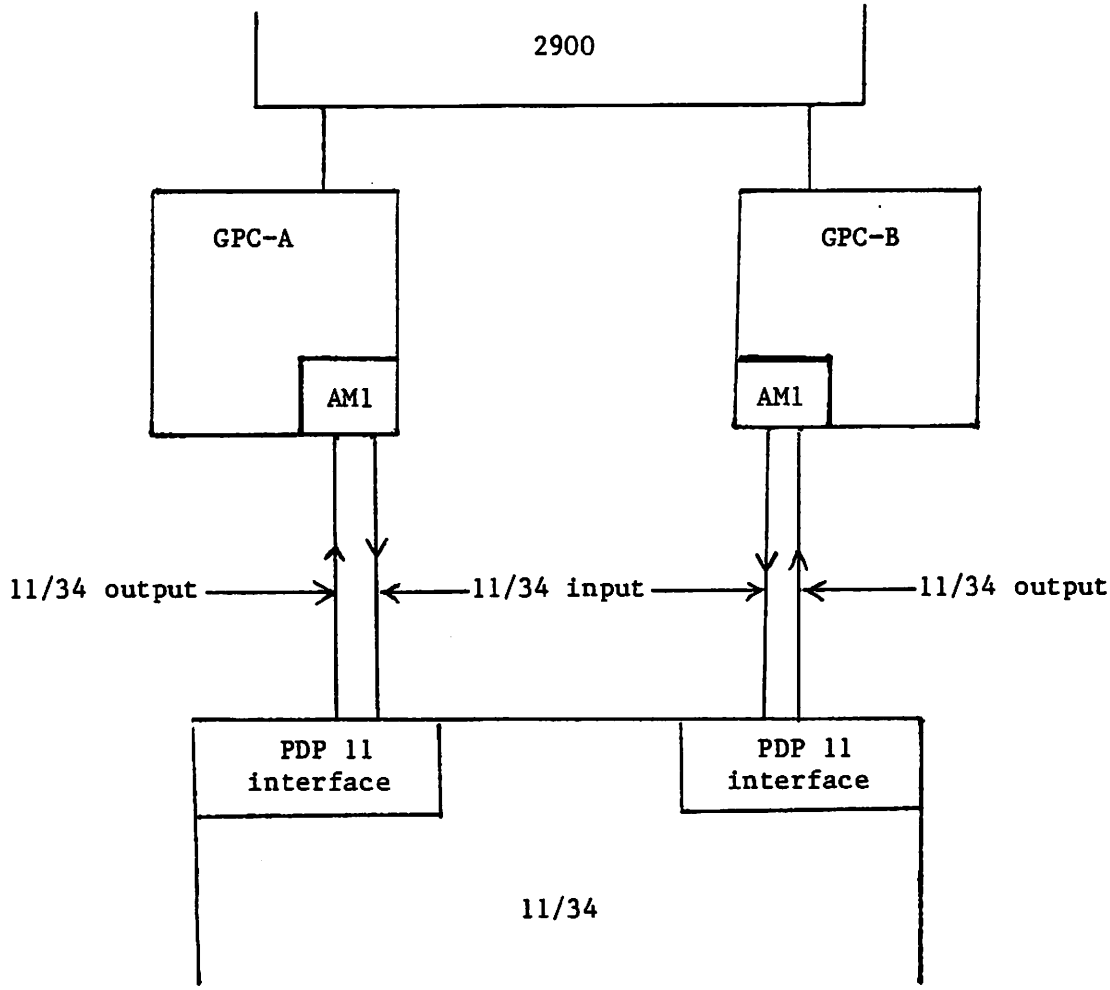



Fig. 3.1: Connection of FEP to Mainframe

As explained above, the FEP is linked to the 2970 via the AM1 interfaces. It uses only one of these - the other is for backup. The FEP is also linked to the PDP 11/45 Network Control Processor via a 1 Megabit Computer Science asynchronous link. All interactive and RJE traffic to the 2970 uses this link.

## 3.1 Loading Procedure

The procedure is as follows:

* If the RUN light is on, hold the CTRL button down and press HALT, then release the CTRL button.
* Boot the machine: hold the CTRL button down and press BOOT, then release the CTRL button.
* The bootstrap should now identify itself with something like the following (the actual numbers do not matter):

  010000 160200 170000 000002

To load the system from disc drive 0:

* Ensure the LOAD/RUN switch is at RUN.
* Type DK (cr).
* After a pause the system identifier "DEIMOS VSN 6" appears, and after a further pause of a few seconds the system prompt "%" appears.

The programs that drive the link etc. must now be loaded, by typing:

LOGON 15 (cr)

When the prompt reappears, type either

GPCA (cr) or GPCB (cr)

depending on which connection to the mainframe is currently in use.

The Front End is now ready for use.

Notes:

* In this section: (esc) = ESCAPE, (cr) = RETURN, _ = space.
* Ensure that both GPCs are switched on.
* "ATTACHED OK" indicates that the NCP (11/45) has recognised the 11/34.
* Each TCP, as it "connects" to the 11/34, produces a message of the form:

  TERMINAL_x_CONNECTED

* After a reload of the NCP (11/45), connection of the 11/34 must be re-established by typing

  (esc) INT_R_CSLK (cr)

## 3.2 Powering Off Procedure

1) The Front End should be closed down by typing on its console:

   (esc) INT_D_GATE (cr)

   The reply

   GATE: REMOVED OK

   should be received.

2) The Load/Run switch on the disc should be switched to LOAD.

3) The machine should be halted by holding down Ctrl and pressing Halt.

4) Finally, power the machine off.

## 3.3 Dumping the System

A dump should be taken if the 11/34 stops; that is, if the run light goes out. The sequence for taking a Front End dump is:

* Holding CTRL down, press HLT.
* Press CLR.
* Press 1 then 4 then 0.
* Press LAD.
* Holding CTRL down, press START; the disc lights on unit 0 should blink and the run light will go on then off again.
* Fill in the front end dump log and process the dump (See 3.4).

## 3.4 Processing a Front End Dump

A front-end dump always occupies the same site on the RK05 system disc and it is important to process this dump immediately before reloading the service system, as any future dump would overwrite the one already on the dump site. Three other sites are reserved on the system disc to hold processed dumps and these should be used cyclicly and their use recorded in the Front End dump log. The three sites are referred to as DMP001, DMP002 and DMP003.

The procedure is as follows:

* Load the Deimos operating system, as in 3.1, until the first system prompt appears.

* Log on to user number 16 rather than user 15:

```
%LOGON_16
```

* Process the dump using the program DPAL and the following replies to its prompts:

```
% DPAL_/DMPnnn        where nnn specifies the next dump site

DPAL: T
DPAL: M
DPAL: Q_47
DPAL: Q_51
DPAL: Q_52
DPAL: S
```

* Continue with the normal load as described in 3.1 by logging on to user number 15.

## 3.5 Responding to crashes

The 2970 FEP will normally run without the need for operator intervention. However, under certain circumstances action is necessary.

1. GPC TIMEOUT + FEO DOWN on the 2970 Oper

   a) If this message is accompanied by

      AM1: Fault C 2, 0

      EMAS 2900 DOWN

   on the FEP console, then a dump of the FEP is <u>not</u> necessary, but

      (esc)INT_F EAM5

   should be typed on the FEP console, to obtain monitoring information. Once it finished, a reload of the 2970 will be necessary, but not a reload of the FEP.

   b) If a message of the form:

      xxxx: BAD SEGMENT - followed by two numbers or
      xxxx: ADDRESS ERROR or
      BUFF: NO SMALL BUFFERS *** or
      BUFF: NO BIG BUFFERS ***

   has been printed on the FEP then a <u>dump</u> (see 3.3) <u>is</u> necessary.

   N.B. The last two messages above can appear without the system crashing; a dump should only be taken if the FEP is not responding to users.

2. If no users can log on to the 2970 and the 11/45 front end to the 4/75 link is up

   In this case, the 2970 FEP should be dumped and then reloaded.

B.A.C. Gilmore
J.M. Murison

# 2970 FEP - COMPILING NEW VERSIONS (1)

All the Fep modules are on file system 15, so a LOGON 15 should be done before editing or compiling a module.

To compile a module, the command is:-

IMPS source,.TT/object

If the 1st pass of the compiler succeeds, - n statements compiled, the compiler proceeds to the 2nd and 3rd pass, after the end of the 3rd pass, - code = n bytes etc., the LINKING stage will prompt for data:-

NAME:

STACK:

STREAMS:

The date typed is different for each module and is given below in the form:- GATE,300,1 - each bit, i.e. 'Gate' is typed in on a line by itself.

Modules (name of source and object file is given)

A) AM1 Handler

There are two versions,

1) EAM5/EAM5Y - this is the unmodified interface handler
2) EAM5X/EAM5XY - this is the modified interface handler

in both cases the linking parameters are:- EAM5,300,1

B) ITP protocol handler

ITPSR/ITPSRY,parameters ITPS,500,1

C) RJE protocol handler

RJESR3/RJESRY parameters RJES,300,1

D) NSI Protocol handler (Gate)

EGATE/EGATEY parameters GATE,300,1

E) C.S. Links handler

CSLK3S/CSLK3Y parameters CSLK,200,Ø

F)/

F) Buffer Manager

BUFFR/EBUFFY parameters BUFF,200,Ø

G) Common Output Handler

COMMS/COMMY parameter COMM,200,Ø

Note: For security, the original object file should be transferred to file system Ø before you compile the new file

T RJESRY/RJESRY(Ø)

and if the new object fails, the old object file can be recovered by:

T RJESRY(Ø)/RJESRY

EMAS 2900 2970-FEP LISTING SOURCE MODULES

The best way of obtaining a listing of a source module after it has been compiled is to produce an 'ALIST' (see USER manual) of the program and list it on the Computer Science line printer.

1) ALIST ITPSR,ITPSRY/.LP - to alist the ITP handler.

2) Type (esc)T LR6/LP on the Computer Science ISYS system.

# EMAS 2900 2970-FEP LISTING A PROCESSED DUMP

After a FEP is processed (see operating instructions) it can be listed on either the CS line printer or on a network line printer, this should be done before the Dump slot is overwritten.

A) Via CS Line Printer

1) LOGON 16
2) T DMP001/.LP - do list DMP001
3) on the CS ISYS machine, type (esc)T LR6/LP

These commands may be typed while the normal system is running, but 'escape' must be typed to produce the prompt '*' before each command.

B) Via the network

This must be done while the normal system is running.

1) (esc)RJELPY DMP001(16) - to list DMP001
2) the program will then prompt:-
   a) Node: - type Ø and return
   b) TERM: - type 23 and return to list on the slow devices LP
   c) FACILITY: - type 4 and return

When LP23 becomes free, it will print the dump, with delivery to B.GILMORE.

EMAS 2900 FEP

AM1 HANDLER MONITORING OUTPUT.

The AM1 handler (EAM5) outputs the list 255 transactions whenever it thinks that the 2900 has gone down (version of 10APR.79). It can also be requested to output the information by typing:-

(esc)INT F EAMS

The format of the monitoring is a single character and a number, the meanings of each character are:-

C n - a control character 'n' has been rec'd from the 2900

I n - a data character 'n' " " "

D n - a data character 'n' has been sent to the 2900

D -n - a control character 'n' has been sent to the 2900
note: this always comes as a pair with a D n,
e.g. D 16 )
D -16 ) → primary status X'20' has been transmitted

S n - 'n' is the stream number that the 2900 is going to do a transfer on, it is purely 'internal' monitoring, rather than a character transmitted or rec'd.

M n - a 'Poff' rec'd by the task, the values of n determine what
n=243 - input interrupt
=242 - output interrupt
=1&2 - control message from a higher task to go to 2900
=3 - a higher task returning control to EAM5

X n - a 'limit' has been rec'd and the current O/P sequence is terminated.
n=current state (This is a normal event on an identity).

U n - EAMS has re-asserted 'accept chr' after the 2900 has accepted the primary station, n is meaningless.

W n - In a timeout sequence, n is the timeout count.

Example of monitoring output from the AM1 link handler showing

1) an ATTN sent to the 2900 and its setting up of an inward control transfer.
2) at the end of 1), it then asychronously starts up a transfer to stream 13.
3) the handler accepts this and passes control to the relevant user program.

```
D   -32)
D    32)  attn
M     0
C    10   identify
D     0
D   128
M     0
C    12   limit
X     4
M     0
D   -16)
D    16)  terminate
M     0
C     5   write control
I   255
I   254
I     0
I   192
C    12   limit
S    -2   monitoring
D   -16)
D    16)  terminate
M     0   INT
C     2   READ (wrt 2970)
D     0
D    11
D     0   data (control stream=
D    10   24 byte)
D     0
D     1
D     0
D   186
D   255

D   255
D   255
D   255
D     0
D     0
D     8
D     0
D     0
D     0
D   150
D     0
D   129
D   129
D   129
D   129
D   -16
D    16   terminates NB is only one byte
M     0   inst (now M 243)
C     5   write control
I     0
I    13
I    11   input data
I    70
C    12
S    13   monitor of stream
D   -16)
D    16)  term
C     3   write (Mus 2970→11/34)
M    42)  return control from    (N.B. from
2     3)  10/4/79 this comes as M 3)
D   -16
D    16   terminates
U-13680   accepts last character sent
```

# 2960 FEP - NOTE ON 2970 DOCUMENTATION

1. Where '.LP' is specified - i.e. in 'LISTING SOURCE MODULES'. .TT will have to be used.

2. The source module of 'GATE' is called KGATE

# Edinburgh Regional Computing Centre

## Memorandum

To R. Eager

Please pass to Alan afterwards.

From George Howat

Date 14-10-81 Extension 2614

Ref.

Ref.

TCP Setmode Command Changes

It is acknowledged that the current implementation of setmodes (Local and Remote) contains deficiencies and irritating features. Therefore, in the next TCP Software release, I propose to alter existing and add "new" setmode functions. This will involve changes in all server and/or front-end software and documentation for mainframes which implement the ITP setmode control mechanism, (I believe that only the 2976 and INFO will remain unaffected), The modifications are not too drastic. Alterations to current practice are described below. For completeness, the appendix summarises the full RCO set.

User Terminal Interface

Three pairs of "new-format" commands are proposed:

| | | | |
|---|---|---|---|
| i ) | V, - V | : | enter, exit video delete mode |
| ii ) | G, - G | : | " " graph mode |
| iii ) | F, - F | : | " " flow control mode (XON/XOFF output) |

Note:

i) Is an unambiguous set video-delete mode, removing the present "toggle"

ii) Currently, exit from graph mode also involves entering upper-case mode and exit from binary input mode. The three functions graph, binary, upper-case are now to be decoupled.

iii) Gives the user access to XON/XOFF output mode locally at the terminal i.e. can be used on any system which does not implement ITP setmode - this has been requested by Glasgow (who cannot - will not send setmode commands) and users of micros emulating terminals.

All other user setmode commands are unaltered but their action is modified e.g. "U" imposes upper case only; binary input and graph mode flags are unaltered.

Mainframe Commands

The above changes have impact on the mainframe software: both bytes in video and graph commands (15 and 11 respectively) are now significant (see Appendix). The EMAS user interface will also have to be changed e.g. GRAPH = ON/OFF; G, - G etc.

The new command to be introduced at the request of the EMAS team, command value 28, is a "RESET DEFAULTS", see appendix.

Comments and/or suggestions to me please.

Appendix: Summary of Setmode Commands (changes marked with asterisk)

| User | Remote Command | Value(s) | |
|---|---|---|---|
| - | 0 | - | Not used |
| - | 1 | 0/1 | Echo off/on |
| H 2-255 (decimal) | 2 | 5-255 | Set page size and enter page mode |
| W 15-160 (decimal) | 3 | 15-160 | Fold line after n characters |
| - | 4 | - | Return current parameters (core map) |
| R 0-255 | 5 | 0-255 | Set data forwarding character |
| I 0-255 | 6 | 0-255 | Set escape character |
| D 0-255 | 7 | 0-255 | Set character delete character |
| C 0-255 | 8 | 0-255 | Set buffer delete character |
| - | 9 | 0/1 | Binary input mode off/on |
| T 1-160... (decimal or *) | 10 | 1-160... | Set tab position (seven values only) |
| * G,-G | 11 | 0/1 | Graph mode off/on |
| L | 12 | - | Select upper & lower case |
| * U | 13 | - | Select upper case only |
| P 0-10 (decimal) | 14 | 0-10 | Insert n pad characters after carriage return |
| * V,-V | 15 | 0/1 | Video-delete mode off/on |
| - | 16 | 0-255 | Append character to server prompt |
| * F,-F | 17 | 0/1 | XON/XOFF output mode off/on |
| - | 18 | 0-255 | Enter bulk input mode with stop char in 2nd byte |
| - | 19-22 | - | Used by Kent. |
| - | 23 | 0/1 | SCREED mode off/on |

| | | |
|---|---|---|
| 24-27 | 0-255 | 4 byte bit mask |
| 28 | - | Reset default modes |

Notes:

'-' as a 'value' means not significant. Under the 'User' column, means cannot be set locally by the user.

Default modes and parameters:

- Echo on
- Page mode off
- Fold line after 80 characters (currently 72)
- Data forwarding character = CR
- Escape character = ESC
- Character delete = DEL
- Buffer delete = DC3
- Binary mode off
- Tab values: 1, 6, 9, 12, 15, 18, 40, 80
- Graph mode off
- Upper case only
- Zero pad characters
- Video-delete off
- No append to prompts
- XON/XOFF mode off
- Bulk input mode off
- SCREED mode off

# GRAPHIC OUTPUT FILE FORMAT

Each file comprises 3 header records followed by the actual plotting data all in fixed 80 byte records.

## Header Blocks

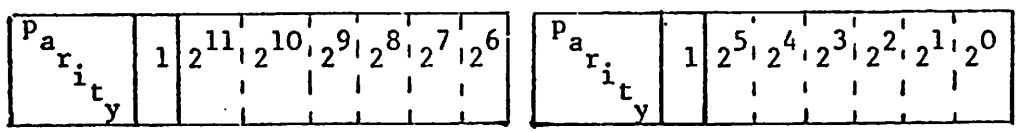
1. 80 EBCDIC 'X' characters
2. 80 ISO characters - of which the first 40 are typed on the PDP-8 teletype preceded and succeeded by CRLF

| BYTES | CONTENTS |
|---|---|
| 1-3 | spaces |
| 4-11 | jobname, eg 'ERCC05' |
| 12-18 | spaces |
| 19-26 | data, eg 10/06/74 |
| 27-30 | spaces |
| 31-38 | time, eg '14:53:54' |
| 39-40 | spaces |
| 41 | device code, 'P' = plotter<br>'B' = binary & selectric |
| 42 | plotter code, '0' when byte 41 = 'B', otherwise<br>'1' = ERCC Calcomp 936<br>'2' = Social Science 30"<br>'3' = " " 11"<br>'4' = Calcomp 564 0.1mm<br>'5' = Calcomp 564 0.005in. |
| 43-44 | 2 digit paper length requirement if >10ft or 3m, otherwise set at '00' |
| 45 | delivery information indicator<br>X'FF' ≡ no information<br>'A' - 't', ie 65-94 (N-64) characters of user's address information |
| 46-49 | spaces |
| 50-75 | spaces if byte 45 = X'FF', otherwise (N-64) delivery information characters |
| 76-80 | spaces |

3. 80 EBCDIC 'Y' characters

## Data Blocks

| BYTES | CONTENTS |
|---|---|
| 1-2 | data block count modulo 4096 - starts at 1 - in the form |

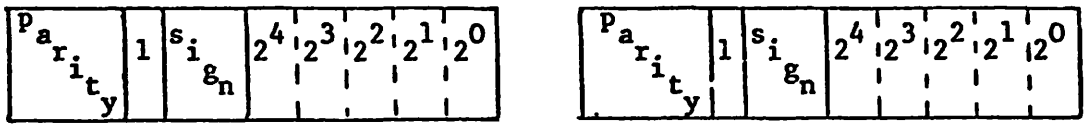



| | |
|---|---|
| 3-4 | |
| 5-6 | |
| . | |
| . | data |
| . | |
| 79-80 | |

The data may be 1, 2 or 4 bytes in length, dependent upon the next vector type.

a. Short Vector

2 bytes - which can cope with vectors <32 increments long, in the form



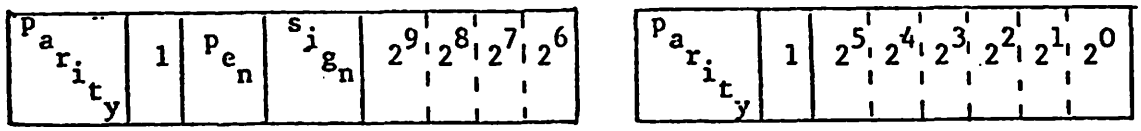

ΔX (2's complement) ΔY (2's complement)

i. possibly preceded by 1 byte (value $29_{10}$) to specify CHANGE PEN STATUS (up/down) if required prior to this move. and

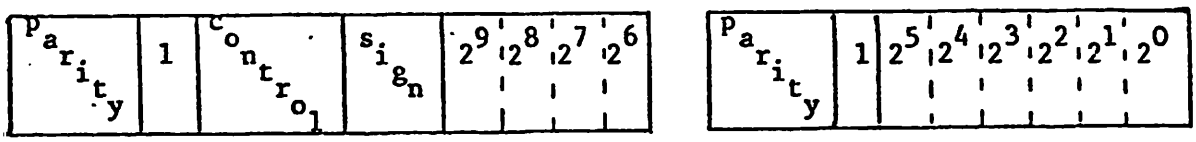
ii. also possibly preceded by 1 byte as a signal (value $31_{10}$) to specify CHANGE TO SHORT VECTORS.

b. Long Vector

4 bytes - which can cope with vectors <1ϕ23 increments long in the form




ΔX (2's complement)




ΔY (2's complement)

i. possibly preceded by 1 byte as a signal (value $30_{10}$) to specify CHANGE TO LONG VECTORS.

In ΔX the $P_{e_n}$ bit indicator is

ϕ raise pen before moving)
1 lower pen before moving) if $C_{o_{n_{t_{r_{o_1}}}}}$ below $\equiv\phi$

and in ΔY the control bit indicator is

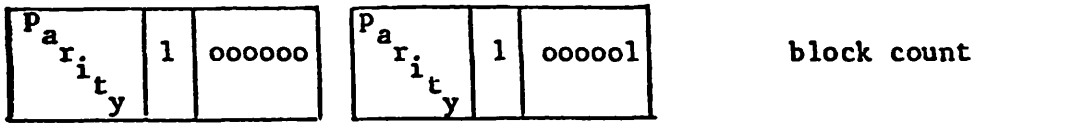
ϕ ≡ these 4 bytes are a LONG VECTOR

1 ≡ these 4 bytes are a CONTROL SEQUENCE when the ΔX value means

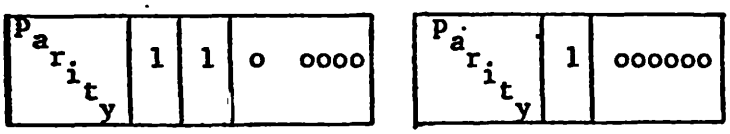
ϕ ≡ new sub-file (plotter origin change)
1 ≡ change pen to black biro
2 ≡ " " to blue "
3 ≡ " " to green "
4 ≡ " " to red "
5 ≡ " " to ϕ.1mm liquid ink
6 ≡ " " to ϕ.2mm " "
7 ≡ " " to ϕ.3mm " "
8 ≡ " " to ϕ.4mm " "
9 ≡ " " to ϕ.5mm " "
10 ≡ " " to ϕ.6mm " "
11 ≡ " " to ϕ.8mm " "

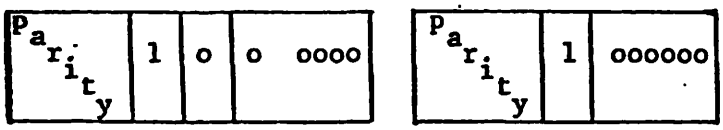
NOTES

1. Each plotter file has in the first 7 bytes of DATA BLOCK 1

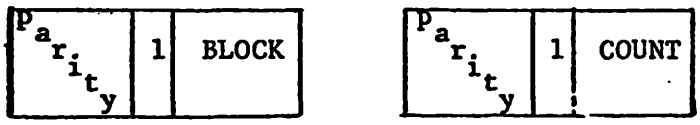




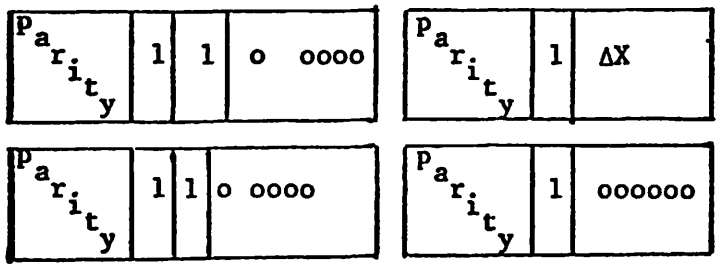



2. Each sub-file or pen colour change request starts on a fresh record with the first 7 bytes containing







3. Only complete 1, 2 or 4 byte groups will appear on any record - there will be no splitting between records

4. On a subfile, or pen change request, or when a 1, 2 or 4 byte grouping will not fit, the current record is padded out with the prevailing VECTOR TYPE INDICATOR (31 for SHORT VECTORS, 3Ø for LONG)

5. SHORT VECTORS will be used whenever possible. 2 SHORT VECTORS will be used in preference to 1 long one when the move is <63 increments long.

M.D. BROWN
16 January 1976

# REGIONAL COMPUTING ORGANISATION
# PLOTTER FILE STANDARD

# C O N T E N T S

Regional Computing Organisation
Plotter File Standard

## 1.0 INTRODUCTION

The Technical Advisory Panel of the RCO set up a working party, initially to advise on the desirability of a standard for plot files, and subsequently to produce a standard. This report gives the proposed standard. This standard is necessarily subject to continuing review as experience is gained and new facilities implemented.

The design goals were:

1. To allow packages running on any computer within the region to be plotted on any plotter.

2. To allow files intended for a particular plotter to be diverted to another plotter with a minimum of distortion.

3. To ensure that the format of the file is sufficiently flexible to allow plotters with advanced features to be incorporated later.

4. To have a consistent and easily implementable format.

A file standard is not sufficient to ensure portability; there must also be operational and programming standards. Some attempt has been made to include these, but further work will be necessary. As an example the standard does not specify what should happen if a plot exceeds the size of the plotter.

## 2.0 DATA STRUCTURE

There is a five level hierarchy of information within the plotter file. In increasing order of complexity, the five levels are:

1. Bytes

2. Commands

3. Records

4. Steps

5. The file

The plotter file consists of steps, which consists of records, which consist of commands, which consist of bytes.

## 3.0 BYTES

The basic unit of representation is the 7-bit byte. If 8-bit bytes are sent, the most significant bit will be ignored. The information bits within a byte are numbered from right to left, with bit 0 being the rightmost (least significant) bit, and bit 6 the leftmost (most significant) bit.

In byte strings representing characters the ISO standard character set is used.

## 4.0 COMMANDS

All commands consist of a control byte normally followed by data bytes. No command is split across a record boundary (see below). There are two classes of commands - vector commands and directive commands.

### 4.1 Vector Commands

These are commands to draw visible and invisible vectors. Both absolute and relative vectors are allowed. A vector command consists of a control byte followed by from 0 to 6 data bytes.

#### 4.1.1 Control Byte -

The control byte has the following form:

| 0 | A | V | $X_1$ | $X_0$ | $Y_1$ | $Y_0$ |
|---|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Bit 6 is always zero.

A-field: This is a one bit field. The vector is absolute if this bit is one, and relative if it is zero.

V-field: This is a one bit field. The vector is visible (pen down) if this bit is one, and not visible (pen up) if it is zero.

X-field: This is a two bit field containing a count of the data bytes that follow for the X co-ordinate of the vector. Values from 0 to 3 are allowed.

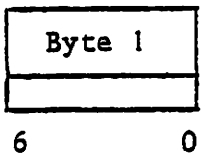
Y-field: This is a two bit field containing a count of the data bytes that follow for the Y co-ordinate of the vector. Values from 0 to 3 are allowed.

Note: Data bytes are stored following the control byte as X co-ordinate data bytes, if any, followed by Y co-ordinate data bytes, if any.

4.1.2 Data Bytes -

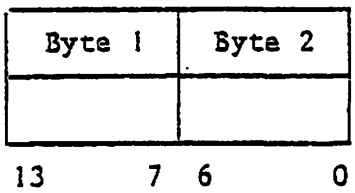
X (or Y)=0: There are no X (or Y) data bytes. For action see note below.

X (or Y)=1:



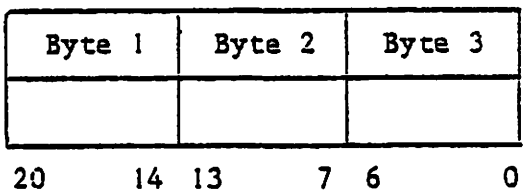

X (or Y) co-ordinates stored as sign in bit 6 and modulus in bits 5 to 0.

X (or Y)=2




X (or Y) co-ordinates stored as sign in bit 13 and modulus in bits 12 to 7 and 6 to 0.

X (or Y)=3




X (or Y) co-ordinates stored as sign in bit 20 and modulus in bits 19 to 14, 13 to 7 and 6 to 0. (see § 1.1)

4.1.3 Notes - the following byte sequences have the effects indicated.

1. 00(8) Raise the pen without changing position.
2. 40(8) Raise the pen and move to origin.
3. 20(8) Lower the pen without changing position.
4. 60(8) Lower the pen and move to origin.

## 4.2 Directive Commands

A directive command consists of a control byte, a count byte (containing the number of data bytes), and data bytes if appropriate. The only exception is the null command (4.2.2.1), which has no count or data byte.

### 4.2.1 The Control Byte

The general form of the control byte is:

| 1 | n5 | n4 | n3 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|

The control byte for a directive always has bit 6 = 1, and has a 6-bit type code N.

### 4.2.2 Commands -

#### 4.2.2.1 N=0, The Null Directive -

The null directive is used for padding out records. It is not followed by a count or data bytes.

1. control byte: 100(8)

#### 4.2.2.2 N=1, Start of File Directive -

1. control byte: 101(8)
2. count: (12 + size of account record) <76
3. data bytes:

   byte 1: override byte

   = 1 plot the file even if the required hardware facilities do not exist

   = 0 abandon plotting a file if any hardware facilities do not exist

   bytes 2,3,4: maximum x extent of plot in plotter steps.

bytes 5,6,7: maximum y extent of plot in plotter steps

bytes 8,9: number of steps/unit

byte 10: plotter units:

= 1 centimetres

= 2 inches

byte 11: number of simultaneous pens or linetypes required.

byte 12: hardware character set:

= 0 none required

> 0 required type of hardware character set (currently undefined)

bytes 13- n: (n < 76) user name/account left justified.

The start of file directive must be contained in a directive step (see § 6.2).

4.2.2.3 N=2, Start of Step Directive -

1. control byte: 102(8)
2. count byte: 8
3. data byte 1: step number (modulo 128)
4. data byte 2: overlay byte:

   = 0 non overlaid step

   > 0 overlaid step
5. data bytes 3-5: maximum x value of frame in plotter steps
6. data bytes 6-8: maximum y value of frame in plotter steps

Note: This must be the first directive on the first record of the new step.

4.2.2.4 N=3, Pen/Linetype Selection Directive -

This directive selects the pen or line type required for subsequent vector drawing. Pen/Linetypes are numbered consecutively from zero to (n-1), where n is specified in byte 11 of the start of file directive ( § 4.2.2.1).

1. control byte: 103(8)

2. count: 1

3. data byte: 0-7 selects pen 0 to pen 7

8-127 not yet defined

4.2.2.5 N=4, Pen/Line-Definition Change Directive -

This directive carries pen/line information for those devices that can utilise it. ~~It also serves to suspend plotting when operator intervention is required to change pens, etc.~~ It is a directive to modify the table of pen option entries. The previous directive specifies a physical pen change.

1. control byte: 104(8)

2. count: 4

3. data byte 1: pen type/speed

= Ø - ballpoint pen

= 1 - ink pen

= 2 - felt tip pen

4. data byte 2: colour/tone

= Ø - black

= 1 - red

= 2 - blue

= 3 - green

5. data byte 3: width

= Ø - 0.1 mm

= 1 - 0.2 mm

= 2 - 0.3 mm etc.

6. data byte 4: pen number. Pen position or line type referenced above.

The definition of this directive is subject to changes in the details of the data bytes. The directive is used to redefine pen/line characteristics. The actual implementation of the required change will be organised in various different manners depending on the actual hardware involved.

4.2.2.6 N = 5, Operator Message not Requiring Intervention -

This types a message on the operators console and continues plotting.

1. control byte: 105(8)
2. count: 1-72
3. data bytes up to 72 ISO characters to be typed

Note: No format effectors should be included in any message.

4.2.2.7 N=6, Operator Message Requiring Intervention -

Similar to 4.2.2.6. but plotting is suspended until the operator signals it to continue.

1. control byte: 106(8)
2. count: 1-72
3. data bytes: up to 72 ISO characters to be typed

Note: No format effectors should be included in any message.

4.2.2.8 N=7, Hardware Character String Directive -

This is a string of characters to be plotted, and is intended for plotters with hardware character sets. Plotters without hardware characters may or may not simulate them.

1. control byte: 107(8)
2. count: 1-72
3. data bytes: up to 72 ISO characters to be plotted

Note: See Section 8.1.9

4.2.2.9 N=8, Hardware Character String Parameters Directive -

The format of this directive has not yet been defined. It is intended for setting up parameters such as size and orientation on plotters with hardware character sets.

1. control byte: 108(8)

2. count: not defined

3. data bytes: not defined

4.2.2.10 9<=n<=31 Reserved for Future Use -

These directives have not yet been defined, and are reserved for enhancements to the regional standard.

4.2.2.11 32<=n<=62 Reserved for Local Use -

These directives are reserved for local use, and it is permissible for interpreting software to ignore them.

4.2.2.12 N=63, End of Plot Directive -

1. control byte: 177(8)

2. count: 0

This directive must be the final directive in the terminating directive step (§ 6.2).

## 5.0 RECORDS

A record consists of 80 bytes. The first byte is the step number, the second the record number within the step modulo 128. Commands may not be split across a record boundary, any unused bytes at the end of a record being padded with null directives. Records are numbered from 0 and are in strictly ascending sequence. A special catagory of record with a step number of zero and record number of Ø constitutes a directive step (see § 6.2).

## 6.0 STEPS

A step consists of a number of plotter records, and is a logical subdivision of the plotter file. For example, a plot file consisting of several graphs might be divided into steps such that each graph is a separate step. In some implementations a step may be a restart point.

A step may be overlaid on a previous step. In such cases the first step should occupy the largest frame.

## 6.1 Vector Steps

A vector step commences with a step directive (§ 4.2.2.3). Each record within a step contains the step number in the record header. Thereafter the record may contain both vector and directive commands. A vector step may be overlaid on a previous step. In such cases the first step should occupy the largest frame. Steps are numbered from 1 in strictly ascending order.

## 6.2 Directive Steps

Directive Steps are numbered Ø and do not contain a start of step directive. They are essentially records with step number Ø and record number Ø and contain only directive commands (see for instance §4.2.2.2 and § 4.2.2.12).

# 7.0 FILES

The plot file may contain header and trailer information (such as produced by HASP). The plot information starts with a record of 80 bytes with values Ø to 79. This is used for framing the file into records in stream oriented input systems. The second record consists only of (4.2.2.2) a directive step holding a start of file directive command.

# 8.0 PROGRAMMING STANDARDS

Programming standards are to allow the recording of standard practices as these are found desirable through experience of various implementations of the Plotter File Standard. No list should be assumed to be exhaustive and implementors should check the current status.

## 8.1 Generating Software

To date the following standards to be observed by software generating plotter files are established.

8.1.1 Many interpreters will only accept vectors up to 15 significant bits (+ sign bit). Hence any larger vectors should be generated as a succession of such 15 bit vectors. An absolute vector greater than 15 bits can be generated as an absolute zero (invisible) plus a succession of invisible vectors to the required point.

8.1.2 The initial default setup should be assumed by any generating software. See Appendix 1.

8.1.3 Whenever steps in a plot are to be overlaid the first step will be assumed to occupy the largest frame.

8.1.4 Each step should close by positioning forward for the next step.

8.1.5 Each file should contain a message not requiring operator intervention displaying user name, job name and delivery information on the operators console.

8.1.6 Generating software should cause the same information to be plotted on the output device, if appropriate.

8.1.7 Operator messages should not contain carriage control information. It is the responsibility of the interpreter to format such messages.

8.1.8 A negative absolute vector should not be generated.

8.1.9 Every hardware dependent operation involving a pen movement should be followed by an absolute pen positioning command.

8.1.10 Vectors should be represented by the smallest number of vector data bytes possible.

## 8.2 Interpreting Software

Interpreting software will be as diverse as the devices available for plotting. The following paragraphs are intended to clarify the action of interpreters in areas where implementations may be radically different, in addition to defining standards for interpreters to follow.

### 8.2.1 Pentype Selection

The number of simultaneous pens/line types is limited by the number requested in the start of file directive. For example, if three pens are requested only pen/line types 0,1 and 2 may be used. Within this restriction any pen type may freely be selected (4.2.2.4) by the generating software. If it is necessary to use a greater number of pen types, one of the permitted pens must be redefined by the change pen directive (4.2.2.5). When implementing generating software note that most interpreters will only handle satisfactorily a small number of simultaneous pen types (see default setup in Appendix 1).

### 8.2.2 Interpreting Pen Type Selection

Normally the interpreter will accept as many pen types as are available on the hardware. However it is feasible for the interpreter to accept a greater number, when the interpreting software must organise the changing of pens as different pen types are selected. In this case, the pen select command may cause plotting to stop while the operator is instructed to change the pen. This is not feasible if pen changes are frequent!

If a greater number of pens are requested than the interpreter can handle, and it is decided to plot the file anyway, it is invisaged that all 'extra' pen types would be treated as, say, pen Ø.

### 8.2.3 Pen/Linetype Definition Change

This command redefines one of the available pens or defines pens additional to that assumed in the default setup. It is the business of interpreting software to organise the mapping of different pen/linetype definitions onto the physical pens or linetypes available on a particular device. Any operator intervention required is organised by the interpreting software.

APPENDIX 1

## DEFAULT DEVICE SETUP

Software generating files to this standard and satellite software interpreting these files to a graphic device, should both assume that the following pen/line type initial conditions apply at the start of each file, (ref. Pen/Line Change Directive).

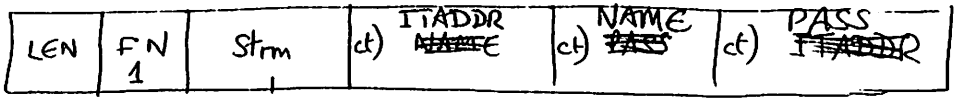
| | Output Device | |
|---|---|---|
| Pen/Line Type | Graph Plotter | Tektronix VDU |
| Ø | Black biro | Solid line |
| 1 | Red biro | Dotted line |
| 2 | Blue biro | Chained line |
| 3 | Green biro | Short dashed line |
| 4 | - | Long dashed line |

Commands creating these conditions will not appear in the file; it is the task of satellite software to reset these conditions at the start of each file if necessary.

# Messages DIRECT ↔ FEP

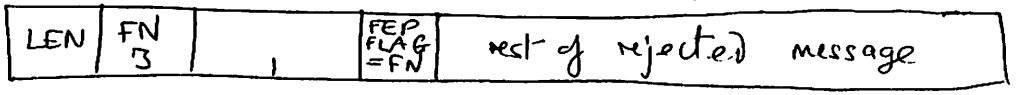
In to DIRECT

① Logon

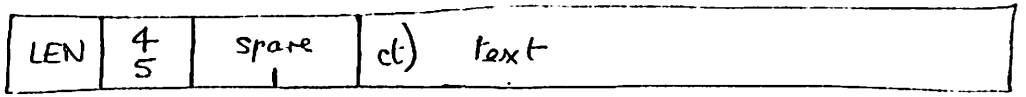



② Mode data


③ rejected message

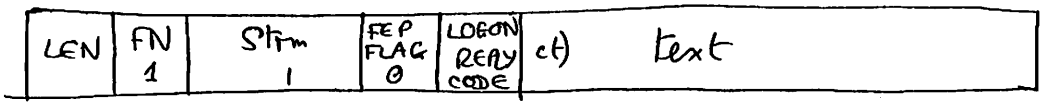



④ ⑤ } monitor msg to { Mailog / Oper

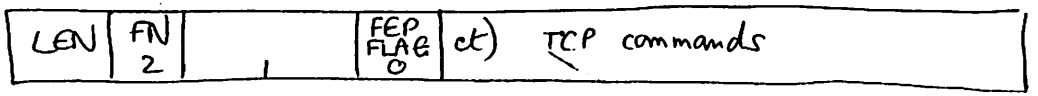



Out from DIRECT

① Logon reply




② Setmode



## TCP Action codes on a SETMODE call

| Byte 1 | |
|---|---|
| Ø or 16 | End of setmode sequence |
| 1 | If Byte 2 = Ø then echoing off, else on |
| 2 | Set page length to byte 2 lines |
| 3 | Set line length to byte 2 chars |
| 4 | Bulk 'getmode'. Returned as "ITP control + setmode" transfer |
| 5 | Set CR char to byte 2 |
| 6 | ESC |
| 7 | DEL |
| 8 | CAN |
| 9 | If byte 2 = Ø then "binary input mode" off, else on |
| 10 | Set tabstops, seven values in bytes 2 - 8 |
| 11 | Graphical o/p mode on/off |
| 12 | Lower case input allowed |
| 13 | Turn off graphmode, binary input mode, lower mode |
| 14 | Set number of pads to byte 2 (=N nulls after CR) |
| 15 | ~~Toggle~~ video mode on/off (same as 23) |
| 16 | Above |
| 17 | Flow control (1 = a, 0 = off) |
| ~~17~~,18,19 | Not used at Kent. NOP's |
| 20 | Preset input buffer. Byte 2 = count. Bytes 3 on = chars to be loaded into i/p buffer. Use for em 'open' |
| 21 | Byte 2 = Ø then tabs are expanded into spaces on i/p, else tabs are sent as themselves to host. |
| 22 | Set all user options in one go - bulk setmode |
| 23 | Reliable video mode. Byte 2 = Ø for off, else on |

N.B. setmodes can be strung end to end within one transfer if so desired. No terminator is required, since the TCP effectively appends a Ø byte to your transfer.

Bulk setmode/getmode variables

N.B. preceded by a 22 for setmode

| | | | |
|---|---|---|---|
| Byte 1 | Flags 1. | B:Ø not used<br>B:1 Echo flat 1 = don't echo<br>B:2 Graph mode 1 = on<br>B:3 Lower mode 1 = on<br>B:4 Binary i/p mode 1 = on<br>B:5:7 Not used<br>B:0 Flow cntrl (1 = on) | 'Not used' bits undefined on getmode. Masked out on setmode |
| Byte 2 | Flags 2. | B:Ø:2 Not used<br>B:3 Video mode 1 = On<br>B:4 Not used<br>B:5 Hardware tab i/p mode 1 = On<br>B:6 Raw mode 1 = on<br>B:7 Not used | |
| Byte 3 | Pad count | | |
| Byte 4 | Spec says physical line number - in fact it is undefined | | |
| Byte 5 | Linelength | | |
| Byte 6 | Pagelength | | |
| Byte 7 | Undefined (in fact it is always 1) | | |
| Bytes 8 - 14 | Tab vector 7 bytes | | |
| Byte 15 | CR char | | |
| Byte 16 | ESC char | | |
| Byte 17 | DEL char | | |
| Byte 18 | CAN char | | |
| Bytes 19 - 34 | Raw mode mask (128 bits)<br>Least sig bit byte 19 for NUL (ØØØ)<br>Most sig bit byte 34 for DEL (177)<br>bit = 1 means treat as line break char.<br>If B:6 of byte 2 = Ø, this mask may be omitted on a bulk setmode. | | |

## Use of Raw Mode

Raw mode is selected by setting B:6 of byte 2 in a bulk setmode (action code 22). Bytes 19 - 34 of the setmode define a 128 bit mask, one bit per ascii character. Setting a bit to Ø indicates that the character corresponding to that bit position is to be buffered on input and not immediately sent to the host. Raw echo will be performed by the TCP, but no translation will be performed - e.g. TABs will not be expanded into spaces. When the user types a character whose corresponding bit in the mask is set to 1, all buffered input together with that character will be sent to the host. The "line-break" character will not be echoed by the TCP.

The maximum input line which can be buffered is 120 characters. Should the user type 120 consecutive non-line-break characters, the TCP will send the buffer to the host.

To avoid problems in some ITP implementations, CR is mapped into CR+128.

The INT: character is recognised at all times. This offers the user some protection against programs which "die" in raw mode. The character can be changed via byte 16 of a bulk setmode if programs are really desperate.

Should a program rely on multiple line-break characters, it should be aware of ITP goahead restrictions. If the TCP runs out of goaheads it will reject user input and eventually force a "console failure" disconnect.

All output from the host is passed to the user with no translation. When the user inputs a line-break character, further echoing of non-line-break input is deferred until the host has responded with output. This guarantees the state of the user's VDU for programs such as screen editors.

A. L. Ibbetson
April 1981