2900 EMAS

COMMUNICATIONS

DESIGN AND IMPLEMENTATION NOTES

N. H. SHELNESS
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF EDINBURGH

24 JUNE, 1977

INTRODUCTION

It may seem rather premature to specify the long term structure and philosophy of communications facilities to be provided by 2900 EMAS.

In the past we have successfully implemented communications sub-systems and facilities without possessing a long term view. The results, I believe, speak for themselves. In many ERCC and RCO communications products an excellent implementation satisfying initial requirements has later proved a barrier to the provision of additional facilities. This experience is not limited to ourselves, nor is it always avoidable. In some forefront areas, such as distributed computing or data base design, it may be impossible to perceive the future clearly enough to take a long term view. I do not believe that main-frame communications qualifies as such an area.

There is a second reason for desiring such a specification. The communications sub-system will be worked on both in parallel and sequentially by different individuals. It is important that they share a common view of that, of which their own efforts are part. If they do not, the code that they produce or modify is unlikely to mesh correctly. This in turn is likely to result at best in inefficiency and more likely in complex bugs, that are both difficult to isolate and to correct.

I could state other arguments justifying the need for this document. I do not believe they are neccessary.

DESIGN AIMS

It is easy when evolving the design aims of a system or sub-system to degenerate into stating the obvious or proposing the desirable even though its achievement may be fundamentally impossible or outrageously expensive. I have attempted to avoid these dangers though the temptation is a great one. I have also chosen to consider design goals under a number of general headings, rather than specifically aim by aim. There is one additional comment that needs to be made. These are design aims and not a specification of a finished product. It is to be hoped that the the finished product will meet its design aims, but this cannot be guaranteed.

1). commonality

All communications traffic (ie. interactive, inter-machine, RJE and transaction) should be transported between a front-end processor and an ICL 2900 main-frame using a common logical interface, independent of the make or model of front-end processor or the physical interface by which it is attached to an ICL 2900 main-frame.

Within an ICL 2900 main-frame system there should exist a single common communications software interface. This is specified so as to avoid the use of different internal transport mechanisms for each type of communications traffic. It does not imply a single user interface for all forms of communications traffic, but we do expect that there be only one functional user interface for each type of communications traffic (ie. interactive, inter-machine, RJE, and transaction).

These aims draw heavily from experience of heterogeneous computer and terminal networks over the last eight years. If these aims are couched in networking terms they appear as follows:

a). That there be a single transport protocol for communication between front-end and main-frame processors.

b). That there be a single transport station within each main-frame regardless of the number of front-end processors attached and interfaces employed.

c). That there be a single virtual terminal, ie. embedded applications level protocol, associated with each type of communications traffic, and that this protocol be made available as the user interface for that type of traffic.

2). flexibility

The structure of the communications sub-system should cater for as wide a range of terminal types, interconnection topologies, front-end processors, and main-frame connections as possible. This is not a meaningless goal but rather a constraint on the commonality requirements specified in the preceding section. It would be possible to satisfy the preceding requirements by limiting the transport functions allowed in the attached communications network or networks to a minimal subset and by using a single existing terminal as the virtual terminal for each type of communications traffic. ie. A teletype for interactive traffic, an IBM 2780 for RJE traffic, and an ICL 7181 for transaction traffic. This is most certainly not our intent. Thus we require:

a). That the front-end to main-frame transport protocol be capable of mapping into a wide range of existing transport protocols (ie. NSI, EPSS, X25, SNA and DECNET to name a few).

b). That the virtual terminal for each type of communications traffic be capable of being mapped into a large number of real terminals. ie. into teletypes, IBM 2741s and various interactive video terminals for interactive traffic, IBM 2780s and 1130s, ICL 7020s and CDC 200 USER TERMINALS for RJE traffic, and ICL 7181s and IBM 3720s and their emulators for transaction traffic.

3). efficiency

This is again a constraint on requirements specified in preceding sections, though efficiency aims are rather more difficult to specify than were those aims specified in preceding sections.

The first requirement is that the effects of the inherent data unit mismatch between variable length messages, which are the unit of transfer between front-end and main-frame processors, and pages, which are the unit of storage allocation and transfer within the main-frame, should be minimized.

The second is that a paged virtual process should be invoked to handle communications traffic only when enough information has been transfered to trigger computational activity in the virtual process. Virtual processes should not be invoked for book-keeping operations.

This implies that an interactive process be invoked only upon the reception of a complete input message for which it has been waiting, but not upon the the reception of message fragments. Similarly, it implies that the spool manager be invoked only upon the completion of a file transfer rather than upon the transfer of a block.

APPROACH

The approach taken to data communications in 2900 EMAS satisfies the design aims specified in the preceding section in an extremely elegant manner. To achieve this end it employs a number of new and unique features. This would not be a beneficial characteristic unless these new features overcame problems experienced in previous virtual memory systems. They do.

The critical new feature is that communications traffic is transfered directly between non paged devices (front-ends, card readers, line printers, etc.) and files, rather than between these devices and real memory buffers.

In the case of interactive traffic, these files are also connected into the appropriate user's paged virtual memory in communications input or output mode. Input is written to a file, connected in communications input mode, by a front-end processor and is read from that file by programs running in the associated user's virtual process. In a similar fashion, output is written within a user's virtual process to a file connected in communications output mode, and is read from this file by the associated front-end processor.

In the case of RJE traffic, a sequence of spool files is associated with each RJE device stream by a system's spooling manager. The latter is a paged executive process running under the system as a paged user process, but with a higher level of system's privilege.

For I/O transfers to take place to or from a file, the relevant pages of that file must be resident in main memory. This does not imply that these pages be permanently resident in main memory, only that they be resiaent for the duration of the data transfer. As the time required for data transfer is small and the period between transfers is large, when compared to page transfer times, these pages are transfered to main memory from secondary (drum or disk) storage only as required. These transfers are performed under the control of a special paging manager called the communications controller.

In existing systems, the inclusion of a special paging manager for this purpose would be difficult, as such systems possess only a single paging manager to control the use of all memory resources. It is not a problem in 2900 EMAS, in which multiple paging managers, called local controllers, are provided on a one to one basis for each virtual process in the system. The communications controller is merely a special form of local controller. A special form is required because, unlike the virtual processes controlled by normal local controllers, the virtual process controlled by the communications controller posseses no virtual processor. As there is no virtual processor, page faults do not occur. Instead, transfer requests passed to the communications controller by interface adaptor modules generate page requests.

Local controllers request the transfer of pages between secondary and main storage via calls on a single underlying resource manager called the global controller. This seperation of function between a single global controller and multiple local controllers allows the local controllers to be responsible only for the decision to transfer a page to or from main memory and not for the mechanization of that transfer.

It also means that the sharing of pages between processes can be handled by a single global controller, and ignored by the multiple local controllers.

In addition to a capability for transferring data ~~transfers~~ to or from devices, there is a requirement for passing control information between device servers and associated communication's users. A device server is the module that controls the device to which the communication's user is connected. In the case of local peripherals such as line printers and card readers, e device servers will be embedded in the interface adaptor module for that device. In the case of devices connected through a front-end processor, such as interactive or RJE terminals, the device servers will reside in the front-end processor itself. The identity and format of this control information need only be agreed between a communication's user and a device server. In this sense it is as transparent as the data, though we intend to agree on a standard set of control messages for each type of communications traffic.

SOFTWARE ARCHITECTURE

In the preceding section our approach to the handling of communications traffic within 2900 EMAS was described. In this section we examine the software modules that mechanize the handling of this traffic within the main-frame.

Figure 1 shows the position and connectivity of communications modules within the system. It will be noted that modules possess both data and control interfaces. There are four classes of module:

1). Interface adaptor modules.

Which mechanize control and data transfers between front-end processors or local slow peripherals (card readers, line printers, etc.) and communication streams. The identity of an adaptor module depends upon the identity of the physical interface employed to connect each front-end processor or device to the main-frame. It is intended that the introduction of a new interface or device should only require the introduction of a new adaptor module, and no other changes to the system.

2). The communications controller.

Which mechanizes communication streams and handles page faults on behalf of interface adaptor modules. There is only one communications controller in the system.

3). Interactive interface modules.

Which are part of the director of each virtual process. They support the system's interactive communications interface by fielding user's interactive I/O commands and generating the appropriate calls on the communications controller and attached interactive terminal server.

4). The spooling manager

Which is an executive virtual process that accepts output files from user processes and generates the appropriate calls on the communications controller to transfer the contents to a local or remote line printer, card punch, paper tape punch or other output device. The spooling manager also generates new files into which jobs or files from local or remote card readers, paper tape readers or other input device are read under the control of the communications controller, before being transfered to the user process in which they are to be stored or executed.

In succeeding sections the structure of modules in each class as well as the interfaces between these modules will be described in greater detail.

THE COMMUNICATIONS CONTROLLER

Data transfers take place between file sections and devices on communication streams but because buffer pages need to be resident in main store for transfers to procede, the synchronization of stream transfers needs to be more complex than if the streams were modelled on simple single character sources or sinks.

Each stream is identified by a sixteen bit stream address with even ddresses identifying input streams and odd addresses output streams. A front-end will usually have a number of streams with contiguous addresses assigned to it, while a line printer, for example, will have only one. All streams operate in simplex mode with data transfers taking place in only one direction, though control information may pass in both directions. If a duplex stream is required, as for exampte with interactive traffic, it is constructed from two consectutive streams (input stream "N" and output stream "N+1"). Stream clusters required to support RJE terminals may be similarly configured from multiple streams.

A stream will always be in one of eleven states. These are:

0). Unused.
1). Disconnecting. ✓
2). Connecting. ✓
3). Connected.
4). Suspending.
5). Aborting.
6). Claiming. ✓
7). Enabling. ✓

8). Enabled. ✓

9). Queued. ✓

10). Paging in. ✓

11). Active.

A stream is unused when it has not yet been assigned to a communication's user. It is transfered from the unused to the connecting state by a connect command, and from the connecting to the connected state by a low level acknowledgement from a server. A connect mmand will normally be issued by some sort of access control process as a result of a valid logging on operation on a pre-assigned "known" stream. A stream will remain connected until a disconnect command returns it, via the disconnecting state pending a low level acknowledgement, to the unused state or until it is enabled.

An enable command will cause a stream to pass through two states before becoming enabled. These states are the claiming state in which a stream waits for the file section nominated in the enable command to be cepted by the global controller and the enabling state in which a stream waits for the attached device server to recognize the enabling operation. When the device server replies, the stream enters the enabled state. While a stream is enabled a server may transfer data on it. To do so it issues a transfer request via an interface adaptor.

A transfer request will cause the current page of the file buffer to be brought into main store. There are three states that can be entered as the result of a transfer request. If the current page is still claimed by the communications controller, the stream enters the active state directly. If there is a free page frame, the stream requests the current page and enters the paging in state. In some rare cases, all page frames allocated to the communications controller will be in use. If this is the case, the stream is queued and enters the queued state. A stream is transfered from the queued to the paging in state whenever a page frame becomes free for its use. A stream is transfered from the paging in to the active state when the requested page arrives. If the page cannot be transfered, the stream is suspended. A stream remains in the active state until a transfer has been completed by the attached interface adaptor module.

A stream will be returned from an enabled state to the connected state whenever there is a global controller or adaptor signalled error, a sequential buffer is exhausted or a disable command is received. A stream may be disabled in one of two modes: It may be suspended or aborted. The detailed effects of suspending or aborting depend upon the identity of the attached device server. A stream may be disabled while in any of the four enabled states: enabled, queued, paging in and active. In the case of an enabled or queued stream, the transition is immediate. In the case of a paging in or active stream, the stream is disabled after the page has arrived or the transfer completes.

COMMUNICATIONS CONTROLLER COMMANDS

ACTIVITY: 1

CONNECT STREAM (P1=STREAM, P2=DSNO/DACT)

Causes the stream to be associated with a particular process. This event is signalled to the attached server via an adaptor and enables it to send control information on the stream. This control information will be addressed by the communications controller to the rocess and activity identified in the P2 field.

ACTIVITY: 2

ENABLE STREAM (P1=STREAM, P2=BLOCK, P3=FLAGS/EPAGES, C
P4=MODE, P5=START, P6=LENGTH)

Causes the buffer described by the parameters to be attached to the stream. If the buffer is attached successfuly, this event is signalled to the attached server and data transfers may procede on he stream.

ACTIVITY: ~~[illegible]~~4

DISABLE STREAM (P1=STREAM, P2=MODE)

Causes the stream to be disabled in one of two modes and the event and mode to be signalled to the attached server. No data transfers are possible after a stream has been disabled.

NOTE: A sequential buffer will be automatically suspended when its end is reached.

<u>ACTIVITY</u>: Ø5

DISCONNECT STREAM (P1=STREAM)

Causes the stream to be disconnected from its owning process. This event is signalled to the attached server, thus disabling the transfer of control information on the stream.

<u>ACTIVITY</u>: Ø6

SEND CONTROL (P1=STREAM, P2-6=CONTROL MESSAGE)

Forwards the control message to the appropriate adaptor for transmission to the attached device server. The format of control messages is specified in the section dealing with each type of communications traffic.

INTERACTIVE COMMUNICATIONS

The structure of the interactive communications system is perhaps best described by considering, as an example, a single interactive terminal session.

To initiate an interactive session, a user enters into a dialogue, via an interactive terminal, with an interactive terminal server. An interactive terminal needs to be connected by some means, which need not oncern us here, to an interactive terminal server resident in a front-end processor. In the course of this dialogue, the user will provide the interactive terminal server with a USER ID and PASSWORD to be used in initiating a session. The exact form of the dialogue is outside the scope of this document. Having acquired a user id and password, the server constructs and sends a high level control message on the initial connection stream (currently defined as stream 2). The format of this message is as follows:-

```
    record format LOGON REQUEST (integer INITIAL CONNECTION STREAM=2, c
                                         STREAM PAIR, c
                                 string (7) USER ID, PASSWORD)
```

The STREAM PAIR identifies the first of two consecutive streams to be used as an interactive stream if the login request is accepted. The USER ID and PASSWORD strings are those provided by the user and expected by the system.

At IPL the START-UP EXECUTIVE (process 1) will connect the initial connection stream from each front-end processor to itself. It is therefore to the start-up exectutive that all logon requests will be sent. Upon reception of a logon request, the start-up executive will attempt to validate the user id and password. If they are valid, and the system is not allready full, a start process request will be sent to the global controller. If the system is full, or either the user id or password are invalid, the logon request will be rejected.

The global controller will reply to the start process request, indicating success or failure. A failure will result in the logon ·equest being rejected. A success will result in the logon request being accepted. The format of the logon reply, which is sent as a high level control message on the initial connection stream is as follows:-

```
record format LOGON REPLY (integer DEST=X'00370006', SOURCE=0, c
                                   INITIAL CONNECTION STREAM=2, c
                                   STREAM PAIR, REPLY)
                 REPLY = 0: LOGON ACCEPTED
                       = 1: SYSTEM FULL
                       = 2: INVALID USER ID
                       = 3: INVALID PASSWORD
                       = 4: USER ALREADY LOGGED ON
                       = 5: UNABLE TO START PROCESS
```

Upon receiving the logon reply, the interactive terminal server informs the user of the result. If the logon request has been accepted, it waits for the streams to be connected and enabled.

A successful reply from the global controller will also include the identity (DSNO/DACT) of the DIRECTOR to which the INITIALIZATION message should be sent. A process may be started for a number of reasons: to run a batch job, to process transactions or to serve an interactive user. In each case the started process needs to be provided with the identity of the task for which it was started. In the case of a batch job this would be the identity of a JCL file. In the case of an interactive session it is the identity of a stream pair.

Upon receipt of an initialization message specifying an interactive session, DIRECTOR will connect the input stream to its local controller d the output stream to itself. The reason for this division is that while high level messages on the output stream are only received in response to specific requests and therefore may be waited for by director, high level control messages on the input stream arrive asynchronously, and therefore must be handled by the event driven local controller, rather than by director which is a purely sequential process. Director will in turn also inform the user level code (subsystem) when it transfers control to it of the source of its invocation.

A subsystem communicates with an interactive terminal via system calls on the interactive communications interface. (This interface is described in detail in the next section). These calls will fail unless the subsystem was invoked to service an interactive terminal.

The interactive terminal server is informed, via a low level control message which it must acknowledge, each time a communications stream changes state, though it only needs to take additional action when a stream is being enabled or disabled. For it is only once a stream has been enabled that data may be transfered on it.

A stream is enabled and disabled as a result of subsystem or user calls on the interactive terminal interface. A NOMINATE INPUT call identifies a circular input buffer, commencing at a particular virtual address and with a specified length, lying completely within a file connected in communications input mode. Upon reception of a nominate input call, director will in turn enable the input stream. A NOMINATE OUTPUT likewise defines a circular output buffer and causes director to enable the output stream. A stream may be disabled in one of two ways: it may be ABORTED or SUSPENDED. If an ABORT INPUT call is issued, all input currently held by the interactive terminal server will be discarded. If a SUSPEND INPUT call is made, the stream will be disabled without input being lost. Likewise an ABORT OUTPUT call will cause all output currently held by the interactive terminal server to be discarded, while a SUSPEND OUTPUT call will cause no output to be lost. In all four cases the reply from the interactive communications interface will specify the address of the last character transfered to or from the buffer.

Once a buffer has been nominated, and hence a stream enabled, transfers may procede between the process and the interactive terminal server.

In the case of input, the interactive terminal server transmits data asynchronously on the input stream up to a limitation specified initially when the stream was enabled and subsequently by request input messages. The length and mode of an attached buffer are sent to the server as part of the low level control message when a stream is enabled. The server may transfer length characters beyond the stream position specified in the last request input message or from the start of the stream if no request input message has been received. The interactive terminal server will also send a high level control message on the input stream each time it has transfered a complete message on the stream, or when it has an interrupt message to send to the attached process. The format of this high level control message is as follows:-

```
        record format INPUT CONTROL MESSAGE (integer STREAM NO, c
                                                     INPUT POSITION, c
                                             string (15) INTERRUPT MESSAGE)
```

The INPUT POSITION will always specify the relative position (modulo buffer length) of the last character transfered on the stream. The INTERRUPT MESSAGE may be ignored except when non null. This high level control message will be routed, as allready indicated, to the local controller of the attached process. The local controller uses the INPUT POSITION field to update the INPUT ADDRESS and the INTERRUPT MESSAGE to update the INTERRUPT MESSAGE BUFFER. Both the input address and the nterrupt message buffer are part of the interface status record that may be read directly by the subsystem or user. If director is waiting for input when the message arrives, the local controller will return control to it. If the interrupt message is a member of a specified set (ie. 'A', 'Q' etc.) the local controller will force a contingency.

The INPUT ADDRESS in the interface status record specifies the extent of the input transfered to the buffer. If the subsystem or user wishes to read beyond this point in the buffer, it must wait for more input to be transferred. It achieves this by issuing a REQUEST INPUT call on the interactive communications interface. This call has two parameters: A

TRIGGER ADDRESS which specifies the address of the last character read from the buffer, and a PROMPT to be output on the user's terminal if the user has not yet typed beyond the position specified by the TRIGGER ADDRESS.

The request input call is passed immediately by director to its local controller via a special OUT. Once control has been transferred to the local controller, two possibilitie exist. Either input has now passed the trigger address or it has not. If it has, control is returned mediately to director. If it has not, a note is made and a high level control message is sent on the input stream to the interactive terminal server.

The format of this high level control message is as follows:-

```
record format INPUT REQUEST MESSAGE (integer DEST=X'00370006', c
                                             SOURCE=0, c
                                             STREAM NO, c
                                             TRIGGER POSITION, c
                                     string (15) PROMPT MESSAGE)
```

Upon reception of this message, the interactive terminal server will cause the PROMPT MESSAGE to be output on the user's terminal if and only if the user has not yet typed beyond the specified TRIGGER POSITION. It will also reset the stream capacity to BUFFER LENGTH - 1 beyonds the specified TRIGGER POSITION.

In the case of interactive output, the subsystem or user transfers data directly to the nominated output buffer as if it were a normal file, which of course it is. In order for the interactive terminal server to read from this buffer, it has to be informed of the extent of the data in the buffer. The subsystem or user achieves this via a REQUEST OUTPUT call on the interactive communications interface. This call has two parameters: an OUTPUT ADDRESS and a TRIGGER ADDRESS. The OUTPUT ADDRESS is the address of the last data item in the buffer. Upon receipt of a REQUEST OUTPUT call director issues one of two high level control messages depending upon the value of the TRIGGER ADDRESS.

If the TRIGGER ADDRESS is equal to 0 the following message is sent:-

```
record format REQUEST OUTPUT MESSAGE 1 (integer DEST=X'00370006', c
                                                SOURCE=DIRECTOR, c
                                                OUTPUT POSITION, c
                                                TRIGGER POSITION=-1)
```

This message will be replied to by the COMMUNICATIONS CONTROLLER as soon as it is accepted. If the appropriate adaptor is inhibited, there will be no reply until it is uninhibited. In this way this mechanism is self regulating and cannot swamp the system.

If the TRIGGER ADDRESS is non 0 the following hish level control message will be sent:-

```
record format REQUEST OUTPUT MESSAGE 2 (integer DEST=X'00370006', c
                                                SOURCE=0, c
                                                OUTPUT POSITION, c
                                                TRIGGER POSITION)
```

This message will be replied to by the interactive terminal server, when it has transfered data beyond the TRIGGER POSITION.

When control is returned to director by the appropriate reply, it will in turn reply to the REQUEST OUTPUT call by returning the address of the last character transfered from the buffer. This information enables the the transfer of data to the buffer to be synchronized with its removal by the interactive terminal server.

When stopping, DIRECTOR must first disconnect both streams. On a normal termination the streams will allready have been disabled, but if not DIRECTOR must disable them. Having disconnected both streams it informs the START-UP process of the end of the session. The START-UP process will in turn send a LOGOFF high level control message on the initial connection stream to inform the interactive terminal server that the terminal is to be logged off.

THE INTERACTIVE COMMUNICATIONS INTERFACE

Each user's virtual process possesses an interactive communications interface. This interface consists of a set of commands implemented as system calls and three data areas. These data areas are a fixed read only interface status record, a circular input buffer mapped onto a file connected in interactive input mode, and a circular output buffer mapped onto a file connected in interactive output mode.

The interface status record is in a fixed location in the process base segment (SSN+1). It contains two fields: an input address and an interrupt message buffer. The first is an integer, while the second is a string of maximum length 15.

```
ie. record format STATUS (integer INPUT ADDRESS, c
                          string (15) INTERRUPT MESSAGE BUFFER)
```

The INPUT ADDRESS is that of the last character transfered to the rent input buffer by the communications controller. It will be set to -1 (X'FFFFFFFF') if no data has yet been transfered to the current input buffer.

The arrival of a single character interrupt message, will force an asynchronous contingency to be signalled to the user process. Multi-character interrupt messages are stored in the INTERRUPT MESSAGE BUFFER, but only if the buffer is empty. As there is no queueing mechanism, any multi-character message that arrives while the buffer is full will be discarded. The contents of the INTERRUPT MESSAGE BUFFER may be examined at any time by the user process, and cleared by a CLEAR

INTFRRUPT MESSAGE command. This facility replaces the TEST INT facility provided in EMAS at a considerable reduction in cost.

For interactive input and output to procede, the user must connect a file in interactive input mode and another file in interactive output mode. Only one file at a time may be connected in either mode. Connection in either of these two modes places certain restrictions on the size of the file and the operations that may be performed on a file (ie. EXTEND, TRIM, DISCONNECT, and DESTROY).

An input buffer is a contiguous region lying completely within the file connected in interactive input mode. The location of the buffer is specified by a NOMINATE INPUT command, which precedes an input sequence. An input sequence is terminated by a SUSPEND INPUT command or an ABORT INPUT command.

An output buffer is a contiguous region lying completely within the file connected in interactive output mode. As with the input buffer, location is specified by a NOMINATE OUTPUT command. It remains in force until disabled by a SUSPEND OUTPUT command or an ABORT OUTPUT command.

INTERACTIVE COMMUNICATIONS COMMANDS

<u>integerfn</u> NOMINATE INPUT (<u>integer</u> START ADDRESS, LENGTH)

<u>result</u> = 0: OK

<u>result</u> =-1: ILLEGAL ADDRESSES or ALLREADY NOMINATED

This command identifies the bounds of an input buffer to the director. It fails if there is allready a nominated input buffer or if the nominated buffer does not lie within the file connected in communications input mode.

<u>integerfn</u> ABORT INPUT

<u>integerfn</u> SUSPEND INPUT

<u>integerfn</u> REQUEST INPUT (<u>integer</u> TRIGGER ADDRESS, <u>string</u> (15) PROMPT)

<u>result</u> = 0: INPUT AVAILABLE

<u>result</u> =-1: ILLEGAL ADDRESS

The command fails if the TRIGGER ADDRESS lies outwith the active input buffer, or if there is no currently active input buffer.

<u>integerfn</u> NOMINATE OUTPUT (<u>integer</u> START ADDRESS, LENGTH)

<u>result</u> = 0: OK

<u>result</u> =-1: ILLEGAL ADDRESSES or ALLREADY NOMINATED

integerfn ABORT OUTPUT

integerfn SUSPEND OUTPUT

integerfn REQUEST OUTPUT (_integer_ OUTPUT ADDRESS, TRIGGER ADDRESS)

result >= 0: ADDRESS of the last byte transfered by an interface adaptor.

result =-1: ILLEGAL ADDRESS

integerfn CLEAR INTERRUPT MESSAGE

MARK 1 FRONT-END ADAPTOR

An extremely simple protocol is operated between the communications controller and the mark 1 front-end (a PDP-11/20 connected via AM1-GPC1 interface). The purpose of this protocol is to effect the transfer of data and control information on logical communication streams assigned to the front-end.

For simplicity, all control information is multiplexed onto two pseudo streams (-1 and -2) which we call the inward and outward control streams. Because control information is transmitted as stream data, all transfers to and from the front-end are stream data transfers. All requests for data transfers other than on the inward control stream are passed as control messages on the inward control stream. Therefore, it is only to request a transfer on the inward control stream that the front-end processor needs to be able to signal the main-frame. This it does by setting the device specific X bit in the primary status byte, either on termination of a transfer, or in an asynchronous interrupt.

Control messages have the following format: They commence with a two byte stream number followed by a two byte sub-identifier that performs a function similar to the control parameters of a 2900 EMAS inter-process message. A sub-identifier of 0 indicates a high level control message. Its body contains five four byte parameters (P2-P6). All other sub-identifiers indicate low level control messages. Their bodies contain only a single four byte parameter, which when sent from the main-frame to the front-end contains three fields: The first field contains the stream state byte, the second field the stream mode byte and the third field the stream length which is a two byte quantity. If the top bit of the sub-identifier is set no reply is expected. If the top bit is not set a reply in the form of a low level control message is expected. This message should posses the same sub-identifier as the initiating message.

```
    RECORD FORMAT HIGH LEVEL CONTROL MESSAGE C
                    (SHORTINTEGER STREAM, SUB IDENTIFIER = 0, C
                     INTEGER P2, P3, P4, P5, P6)

    RECORD FORMAT LOW LEVEL CONTROL MESSAGE FROM MAIN FRAME C
                    (SHORTINTEGER STREAM, SUB IDENTIFIER # 0, C
                     BYTEINTEGER STATE, MODE, C
                     SHORTINTEGER LENGTH)

    RECORD FORMAT LOW LEVEL CONTROL MESSAGE FROM FRONT END C
                    (SHORTINTEGER STREAM, SUB IDENTIFIER # 0, C
                     INTEGER ERROR FLAGS)
```

Despite the fact that all transfer requests are initiated by the front-end, the initiation, sequencing and control of transfers to or from the front-end is performed by a control program (the MK1FE ADAPTOR) running in the main-frame. Each logical data transfer on a stream consists of two physical data transfers: An outward transfer which specifies the stream (2 bytes) and the maximum length (2 bytes) of the ensuing data transfer and the data transfer itself.

The actual length of the data transfer is under the control of the front-end which may limit the transfer at any point short of the maximum length. If it so limits the transfer it sets the short block flag in the primary status byte to so signal. If the length of the actual transfer matches the maximum length, no bit is set. If on the other hand the front-end wishes to transfer more than the maximum length, it signals this fact by setting the long block flag in the primary status byte.

```
LOCAL      -----------  <---
SEGMENT * |     +-----|----|---------> REAL BUFFER PAGE ADDRESS
TABLE     |           |    |
         - -----------     |
          |           |    |
          |     +-----|----
          |           |
RCB       |     +-----|----
          |           |    |
          |     +-----|----|----
          |           |    |    |
          |           |    |    |
         - -----------  <---    |
          |           |         |
           -----------          |
 OGIC     |           |         |
BLOCK      -----------          |
        * |           |         |
         - -----------  <--------
          |     +-----|---------
          |           |         |
           -----------          |
ADDRESS   |     +-----|----
LIST      |           |    |    |
           -----------     |    |
        * |     +-----|----|----|----> VIRTUAL BUFFER ADDRESS
        * |           |<---     |
         - -----------          |
STREAM  * |XXXXX|     |<--------
         - -----------
CONTROL   |           |
INPUT     |           |
BUFFER    |           |

          |           |
         - -----------
CONTROL   |           |
OUTPUT    |           |
BUFFER    |           |

          |           |
           -----------
```

NOTE: Words marked with a * need to be altered before each stream transfer. All other words can be settup on adaptor initialization.

CONTENTS

15th July 1980