HASP WORKSTATION EMULATOR

093–000116–00

Original Release - April 1975

PREFACE

The HASP Workstation Emulator consists of a Data General ECLIPSE Computer and associated peripheral devices, operating under control of the HASP Multi-Leaving Terminal Control Program (HAMLET).

HAMLET permits an ECLIPSE computer to communicate with either an IBM 360/370 system, or another ECLIPSE computer. When communicating with an IBM 360/370, HAMLET emulates an IBM HASP or ASP remote job entry (RJE) workstation.

This document describes the HAMLET program, program generation, operating procedures, and a special User Interface. Several programming examples are interspersed throughout the manual as a further aid in describing the proper use of HAMLET.

The following Data General publications will assist users of this software package:

093-000056	<u>Real Time Operating System User's Manual</u>
093-000075	<u>Real Time Disk Operating System User's Manual</u>

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION TO HAMLET

GENERAL

The HASP Multi-leaving Terminal Control Program (HAMLET) permits an ECLIPSE computer and its associated peripheral devices to communicate with either an IBM 360/370 system or another ECLIPSE workstation.

When communicating with an IBM 360/370, the program emulates an IBM HASP or ASP remote job entry (RJE) workstation. Multi-leaving is supported, with up to seven input streams, seven output streams, and an operator console stream for efficient use of multiple peripheral devices. The inter-leaving and data compression features contribute to efficient data transfer over a communication line. The ECLIPSE workstation can be configured with disk and magnetic tape peripherals, in addition to video terminals, card readers, and line printers. The communication line protocol is IBM Binary Synchronous (BISYNC) for multi-leaving terminals.

HAMLET FEATURES

Some of the important features of HAMLET are:

- Full HASP/ASP multi-leaving and inter-leaving protocol
- Full data compression
- ECLIPSE-to-ECLIPSE communications
- User interface capability
- Local file transfer
- Complete operating systems support
- HASP/ASP command format compatibility
- Transmission checking
- Disk and magnetic tape file support
- Automatic sign-on record transmission
- Automatic or dynamic device/file assignment
- Workstation console traffic logging
- Vertical forms control

INTRODUCTION TO HAMLET (Continued)

ECLIPSE-TO-ECLIPSE COMMUNICATION

HAMLET allows ECLIPSE-to-ECLIPSE communications for fast and efficient transfer of program and data records without any user programming. The multi-leaving, inter-leaving, and data compression features are also used in ECLIPSE-to-ECLIPSE data transfer.

USER INTERFACE

HAMLET also provides a special User Interface that permits a user's program to send or receive data from HASP or ASP. The interface formats data for transmission, thus eliminating any special user-written communications software.

LOCAL FILE TRANSFER

Local file transfer can be performed by the ECLIPSE workstation operator (i.e., disk to printer, disk to disk, etc.), while the workstation is on-line to either HASP/ASP or another ECLIPSE workstation.

FILE ORGANIZATION

File organization can consist of: binary or ASCII sequential (created in Line, or sequential mode), user-defined, Print or Punch.

When data is transferred to HASP or ASP, the data is translated to EBCDIC, and converted to a format suitable to HASP or ASP before transmission. HAMLET receives data from HASP or ASP as either Print or Punch streams.

When data is transferred between two ECLIPSE workstations, all of the previously mentioned types of files can be transmitted and received.

OPERATING SYSTEM SUPPORT

HAMLET operates under Data General's core-based Real-Time Operating System (RTOS), disk-based Real-Time Disk Operating System (RDOS), or Mapped Real-Time Disk Operating System (MRDOS), to support a wide variety of computer processing applications.

An ECLIPSE workstation operating under control of HAMLET can act as a remote batch terminal in either core-based or disk-based configurations. Under RDOS, the ECLIPSE workstation operates in either foreground or background. MRDOS adds the capabilities of extended memory and memory protection.

HASP/ASP COMMANDS

The HAMLET command format allows an operator to send HASP or ASP commands from the ECLIPSE workstation for interpretation by the HASP or ASP system. The results of the command's execution are displayed back at the ECLIPSE workstation's console where appropriate.

TRANSMISSION CHECKING WITH EXTENDED RETRY

The HAMLET extended retry feature checks validity of all data received by the terminal. If invalid, the sending station re-transmits the data block.

DISK AND MAGNETIC TAPE CAPABILITY

HAMLET supports both disk and magnetic tape storage, letting users transmit data files to disk and tape for storage and processing.

SIGN-ON RECORD

HAMLET automatically transmits the sign-on record to the host computer, bringing the HASP workstation on-line when the communications link is established. This eliminates the need to manually input a sign-on record each time a HASP workstation begins operation.

DATA OVERRUN PROTECTION	If a receiving peripheral device cannot accept data from the host computer, the workstation sends a function control bit ("wait-a-bit"). Transmission on that stream is suspended until the device can receive data. The data stream continues from the point of termination. Sending and receiving of a data stream can also be suspended by the operator when necessary.
DEVICE/FILE ASSIGNMENT	Device or file assignment is automatic (by default), or dynamic, through operator selection. A specific device or file is assigned to receive a specific data stream. Unless the operator changes the selection, the stream will always address that device by default, thus eliminating operator device assignment each time data is received.
CONSOLE MESSAGE LOGGING	HAMLET can automatically log console messages onto a file for operator reference. This is convenient if the console device is a video display, since a hard copy would not otherwise be available.
VERTICAL FORMS CONTROL	Standard HASP/ASP vertical forms control is fully supported. HAMLET also permits a user to generate vertical forms control formats that can be assigned to a particular print stream.

CHAPTER 2

HAMLET PROGRAM OPERATION

GENERAL

The HAMLET program provides the means for sending data from an ECLIPSE workstation to a host IBM system operating under HASP or ASP or to another ECLIPSE workstation, and of receiving and directing output from the host computer to the ECLIPSE workstation's peripheral devices (disk, magnetic tape, line printers, video display, etc.)

HAMLET performs input from and output to multiple devices concurrently. Data to be transmitted is labeled with a source identifier and is called an <u>input stream</u>. Data received by HAMLET is labeled with a destination identifier and is called an <u>output stream</u>.

An input stream is read from an ECLIPSE workstation input device using a HAMLET input routine. The data is then compressed and blocked for transmission to the remote computer. Data received from the remote computer is deblocked and decompressed. The output stream is then written to an ECLIPSE workstation peripheral device by an output routine.

Transmission and reception of data on the communications line is controlled by the line procedure and the synchronous device driver. Data flow is shown in Figure 2-1.

For the remainder of this chapter (except where noted), HAMLET is used to denote the ECLIPSE workstation, and HASP refers to either HASP or ASP.

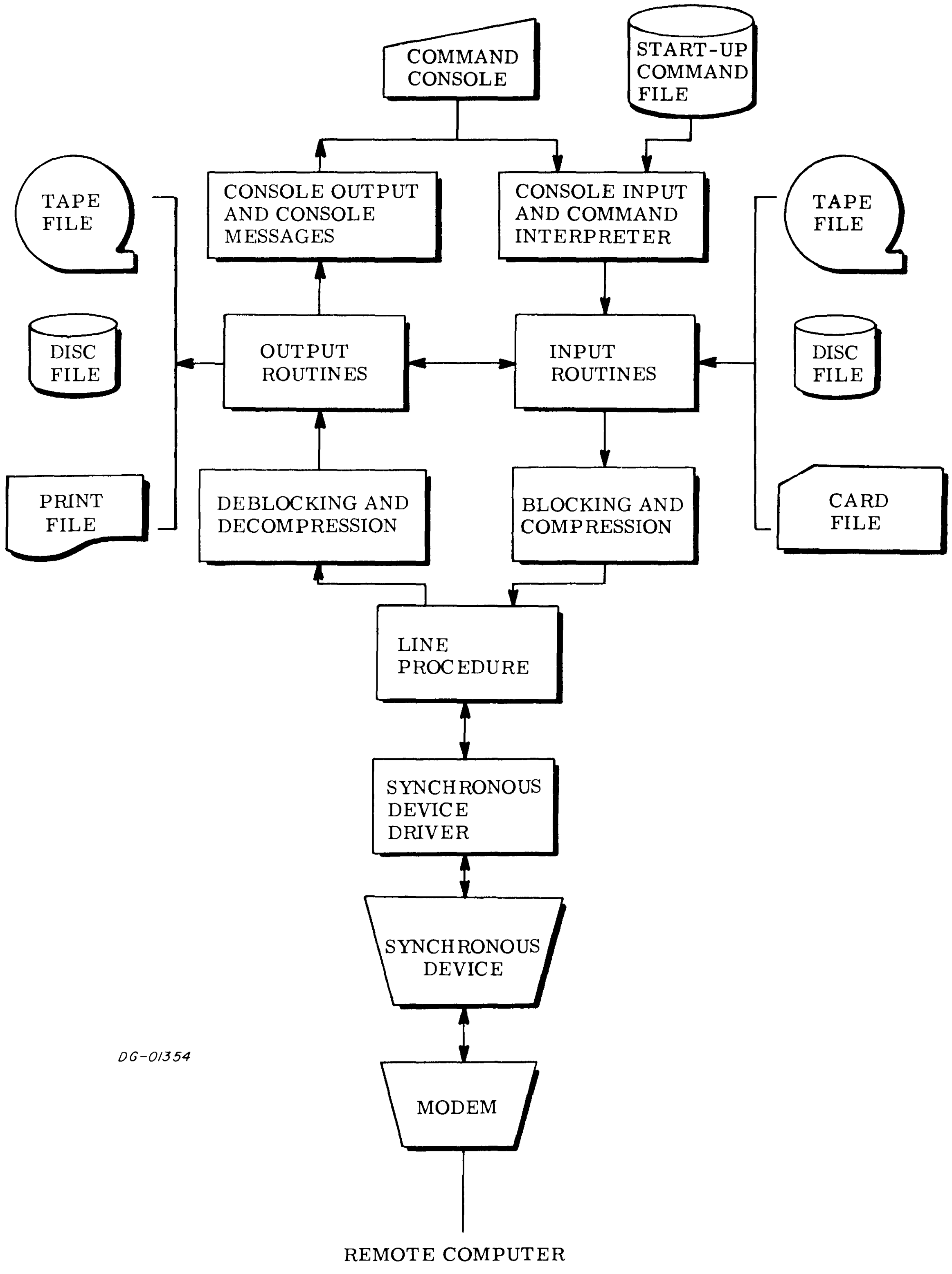


FIGURE 2-1. HAMLET Data Flow

LOCAL AND REMOTE

For purposes of clarification, the terms local and remote are defined as follows:

1. LOCAL - Refers to data transferred between HAMLET and its peripheral devices. Locally, data is either input to HAMLET, or output from HAMLET (e.g., input from a card reader to HAMLET, HAMLET output to a line printer).

2. REMOTE - Refers to the sending or receiving of data over a communication line (e.g., HAMLET to HASP, HASP to HAMLET, or HAMLET to HAMLET).

In remote data transfer, the HAMLET program utilizes the features of multi-leaving, data compression, and inter-leaving.

MULTI-LEAVING

Multi-leaving is a term that describes an efficient method of transmitting data records over a communication line. It permits multiple physical records to be input from, or output to, one or more concurrently operating input or output devices. The records are grouped together into a single transmission block before being sent or received.

For example, multiple records to be sent from HASP to HAMLET (for output to two line printers and a disk file), could be grouped into a single transmission block and sent to HAMLET. On receiving the transmission block, HAMLET deblocks the records, identifies the output device(s) assigned for each record, and then outputs the records.

DATA COMPRESSION AND DECOMPRESSION

During the formation of a transmission block, and as an integral part of the multi-leaving function, consecutive, duplicate non-blank and blank characters appearing in a data record are compressed before transmission.

A string of consecutive, duplicate non-blank characters is represented by a String Control Byte (identifies the number of characters to be duplicated) preceding the character to be duplicated. In the case of consecutive blank characters, only a String Control Byte is required to specify the type and number of blank characters.

Conversely, the data block is decompressed by the receiving station.

INTER-LEAVING

Inter-leaving refers to the sending of a transmission block without requiring a separate acknowledgment to signal a successful reception by the receiving station. A data block is sent in response to a data block received. Therefore, the ECLIPSE workstation can be transmitting data from its card readers while it is printing received data.

TRANSPARENT TRANSMISSION

All computer-to-computer transmissions employ EBCDIC code in transparent mode. Conversion to and from ASCII is performed by HAMLET automatically when required.

INPUT AND OUTPUT STREAMS

Depending on the options chosen during program generation (Chapter 3), HAMLET can control up to a maximum of 7 local input and 7 local output devices operating concurrently. Each active device is controlled by a numbered input or output stream (I1-I7 for input; O1-O7 for output). The number of input and output streams chosen need not be the same, and depends on the anticipated volume of input and output.

INPUT AND OUTPUT STREAMS (Continued)

Each input and output stream is assigned a unique storage area. The storage area contains information that controls the opening and closing of the assigned device, handles file translation, and stores status flags for controlling input or output of data records.

The following example gives an overall description of the use of input and output streams.

Assume that an ECLIPSE workstation has the following configuration:

ECLIPSE (with real-time clock)
1 Card Reader
1 Disk Drive
1 Line Printer
1 Tape Drive

After loading HAMLET (described later in Chapter 4), the operator enables the communications line between HAMLET and the remote computer. After the program verifies that a communication link exists, the operator loads a job into the card reader and issues a console command to transfer the file to the remote computer. HAMLET assigns the first available input stream (I1) and generates a "request-to-send." After the remote computer grants "permission-to-send," the data records are input to HAMLET, encoded, blocked, and sent to the remote computer.

At this point, the remote computer has a job to be sent to HAMLET and output to the line printer. A "request-to-send" is transmitted to HAMLET. "Permission-to-send" is granted, and HAMLET determines if an output stream can be assigned. Output stream O1 is assigned and the line printer is designated as the output device. Output stream O1 is active until one of the following occurs:

1. Normal termination of the print job.

2. An operator command to Kill the output stream.

INPUT AND OUTPUT STREAMS (Continued)

Let us further assume that during the printing of the first output stream (01), the remote computer transmits a "request-to-send" another output stream, also to be output to the line printer. HAMLET determines if an output stream is available. In this instance output stream 02 can be assigned, however the line printer is actively printing the job assigned to output stream 01.

The operator has the following choices:

1. "Permission-to-send" is not granted, in which case no data is sent by the computer.

2. "Permission-to-send" is granted. The print job is assigned as output stream 02, and a disk file is assigned for output. This file can be used for later printing to the line printer as a local HAMLET transfer (disk to line printer).

As can be seen, depending on the workstation configuration and the number of input and output streams specified during program generation, all streams can be concurrently active. As soon as a stream is completed (or killed by the operator), its storage area can be re-assigned.

FILE TYPES

When HAMLET transmits a file to HASP, file types binary sequential, RDOS Line, RDOS Sequential, or user-defined are translated to EBCDIC and converted to a format suitable to HASP before transmission. HAMLET receives either Print or Punch files from HASP. Punch files are normally stored as binary sequential files. When Print files are output to tape or disk, they are stored as RDOS Sequential.

When transmission is between two ECLIPSE workstations, all of the previously mentioned types of file organizations can be transmitted.

The user has an additional option when transferring files locally. Files can be transferred without modification, or file types can be changed during transfer (e.g., RDOS Line to RDOS Sequential).

CHAPTER 3

HAMLET PROGRAM GENERATION

GENERAL

Before using HAMLET it must be tailored to the specific requirements of the user, and the input and output hardware configuration of the ECLIPSE workstation. This tailoring function is defined as HAMLET program generation (CGEN). Although HAMLET can be executed using the RDOS, MRDOS, or RTOS operating systems, it can only be generated under RDOS or MRDOS. For RTOS stand-alone users, see paragraph "RTOS Users" at the end of this chapter.

CGEN operates as an interactive program that asks the user a series of program generation questions. Based on the responses to these questions, CGEN generates HAMLET - which is custom built to provide an efficient executive for the specific ECLIPSE workstation configuration and its system operational requirements.

CGEN OPERATING PROCEDURES

CGEN is loaded by entering the following CLI command:

```
CGEN Δ HCON)
```

where the symbol Δ denotes a blank, and HCON is a file containing the generator questions; the blank must be entered as part of the CLI command;) denotes a carriage return.

After CGEN is successfully loaded, the program responds with the console message:

```
PROGRAM GENERATOR
```

followed by the first program generation question. Following each question, the user must either respond with a valid answer and a carriage return, or a carriage return. When a carriage return is used as a response, the program generation default value is used.

HAMLET PROGRAM GENERATION (Continued)

CGEN OPERATING PROCEDURES (Continued)

After the final question has been asked, and the response given, the linkage map is displayed on the system printer followed by the CLI prompt:

```
    R
```

which signifies that HAMLET has been generated. HAMLET can then be executed under the name HAMLET or the optional user-supplied name given during CGEN (HAMLET execution procedures are fully explained in Chapter 4).

PROGRAM GENERATION QUESTIONS

There are two types of program generation questions: (1) questions needed to create a parameter file, and (2) questions needed to create a command stream for RLDR.

Parameter questions must be answered with a numeric value. This value is checked against a range specified for each question. RLDR questions can be answered with alphanumeric characters. If the response to a question is invalid, the question is repeated. All numeric values input are assumed to be octal, unless a period follows the number, in which case the value is assumed to be decimal.

PARAMETER AND RLDR QUESTIONS AND RESPONSES

In the following presentation, each question is given, followed by an explanation of the available options. The default value for each question is also provided. All values (except where noted) are decimal.

PARAMETER QUESTIONS

PROGRAM MODE?

Identifies the HAMLET mode.

0	-	HAMLET-HASP or ASP mode.
1	-	HAMLET-to-HAMLET mode. The HAMLET program being generated will send the Sign-On record.
2	-	HAMLET-to-HAMLET mode. The HAMLET program being generated will receive the Sign-On record from the remote ECLIPSE workstation.

Default: 0
Range: 0 - 2

NUMBER INPUT STREAMS?

This value (1-7) identifies the number of input streams that can be operating concurrently. The maximum number of input streams that can be assigned is 7.

NOTE: The number of input streams concurrently transmitting data to HASP/ASP is a HASP/ASP system generation parameter, and should be taken into consideration when determining this value.

Default: 1
Range: 1 - 7

NUMBER OUTPUT STREAMS?

This value (1-7) identifies the number of output streams that can be operating concurrently. The maximum number of output streams that can be assigned is 7.

Default: 1
Range: 1 - 7

PARAMETER QUESTIONS (Continued)

PERIPHERAL DEVICE BUFFER SIZE?

Output to, and input from an ECLIPSE workstation's peripheral devices is handled through peripheral device buffers. All buffers are the same size, and the length must be the length of the longest physical record to be processed.

The minimum size depends upon the longest console message that can be expected (HASP sends up to 120 characters for console display) and the maximum must be somewhat less than the maximum transmission buffer size of 2048 words (4096 bytes). The exact value can be computed only by knowing the content of each data record, since compression affects the amount of data that can be packed into a transmission block. If no duplicate characters exist, then the maximum physical record, in bytes would be:

Max. Physical record = (CBSB-11)x63/64+R-1

where CBSB = communications device buffer size in bytes, and R = remainder from CBSB-11/64

If a card reader is used as an input device, and the Hollerith Code is to be translated to EBCDIC, then this value must be at least 81 words (162 bytes).

Default:	81 (words)
Range:	40 - 2010 (words)

NUMBER OF PERIPHERAL DEVICE BUFFERS?

The number of peripheral device buffers to be allocated affects the multi-leaving capabilities of the program, and the throughput of data during execution. A minimum of 5 buffers must be allocated. Since configurations and applications vary for different installations, the optimum number of allocated peripheral device buffers can only be determined by experimentation.

Default:	15
Range:	5 - 300

PARAMETER QUESTIONS (Continued)

COMMUNICATIONS DEVICE BUFFER SIZE?

In HAMLET-to-HASP mode, this value must be the same as the value of the &MLBFSIZ parameter required in the HASP or ASP system generation program. Typically, this value is 200 words (400 bytes).

In HAMLET-to-HAMLET mode, the value must be the same as the remote computer's communications device buffer size.

Default:	200
Range:	10 - 2048

NUMBER OF COMMUNICATIONS DEVICE BUFFERS?

The number of communications buffers affects throughput of data transmitted over a communication line. Up to ten buffers can be allocated, however the recommended number is at least 2. Since configurations and applications vary for different installations, the optimum number of communications device buffers can only be determined by experimentation.

Default:	2
Range:	1 - 10

PARAMETER QUESTIONS (Continued)

MAX BUFFERS PER STREAM?

This value limits the number of buffers in the peripheral buffer pool that can be held by an active input or output stream at one time.

When an input or output stream reaches this value, it is pended until one of the buffers within this range is freed.

The user can use the following formula to determine the number of peripheral device buffers to be allocated:

> Max. buffers per stream = (No. of peripheral device buffers - 3)/(No. of input streams + No. of output streams)

The use of this formula permits equitable buffer allocation when all streams are concurrently active. A greater number of buffers can be allocated when all streams are not active concurrently.

Default:	6
Range:	1 - 15

PARAMETER QUESTIONS (Continued)

REMOTE COMMAND ID?

1 - Signifies that a command or message sent to HASP is to be prefixed by a "$". Any message or command prefixed by "$" is not interpreted by HAMLET. Interpretation is made by HASP.

0 - Signifies that a command or message sent to ASP will be prefixed by an "*". Any message or command prefixed by an "*" is not interpreted by HAMLET. Interpretation of the message or command is made by ASP.

In HAMLET-to-HAMLET mode, a message sent to the remote ECLIPSE can be preceded by either an "*" or "$". The message will be displayed on the remote ECLIPSE console device.

Default:	1
Range:	0 - 1

SPOOLING?

This function exists in addition to spooling facilities automatically available under RDOS/MRDOS.

If 1 is entered, the HAMLET spooling function will provide for temporary storage of data that has over-run the particular output stream to which it is directed. The excess data will be dumped to disk and the affected stream only will have a "wait-a-bit" set.

If 0 is entered, then all streams will be halted in the event of over-run on any stream. Over-run occurs when the output stream performing output cannot write the data faster than it is received (the number of buffers queued to the output stream would exceed the number allowed). The default value calls for selection of this function, and it is recommended. It should be excluded only in cases where core requirements are stringent.

Default:	1
Range:	0 - 1

PARAMETER QUESTIONS (Continued)

STATIC TABLE SIZE?

This value determines the size of the static assignments table. Replacements and deletions to the table can be made using the ASSIGN (A) command (Chapter 4).

Default:	3
Range:	0 - 20

LINE PRINTER WIDTH?

In order to avoid wrap-around of data that exceeds the ECLIPSE workstation's line printer width, any print line that exceeds this line printer width entry will be truncated.

Default:	80
Range:	1 - 132

CONSOLE TEXT?

Console messages can be coded, or message text can be displayed in place of a code (see paragraph entitled "Console Message Format" in Chapter 4 for an explanation of console messages).

0 - Only coded messages will be displayed on the system console.

1 - Message text will be displayed in place of a code (there is an additional core requirement when using this option due to the size of the module (532 decimal words) containing the text).

Default:	1
Range:	0 - 1

PARAMETER QUESTIONS (Continued)

NUMBER OF SYN CHARACTERS?

The number of SYN characters preceding a transmission block can be varied.

Default:	4
Range:	2 - 31

DEVICE CODE?

Enter one of the permissable device codes for the communications device. This number must correspond to the physical device code set for the device.

Default:	30 (octal)
Range:	30, 31, 70, 71

2-4 WIRE?

This allows the synchronous communications device to transmit and receive data using synchronous data sets interfaced for four-wire as well as two-wire operation. In a four-wire operation, the modem receiver and transmitter remain on, minimizing the time delay in switching from a Transmit to a Receive mode.

0 denotes 4-wire; 1 denotes 2-wire.

Default:	1
Range:	0 - 1

Optional Command Modules

Several optional command modules exist that the user can individually include or exclude from the version of HAMLET being generated. These modules are described below. In all cases the module is included by default.

LOG COMMAND MODULE?

For users with CRT system console devices, this provides a hard copy option by including the Log Command module.

Include:	1
Exclude:	0

SIGNON COMMAND MODULE?

At start-up, this module permits the transmission of an optional Sign-On record to replace the Sign-On record chosen during generation of this version of HAMLET (see last question in this chapter). If this module is excluded, then the Sign-On record formatted in response to the last question in this chapter is the only Sign-On record that can be transmitted to the remote computer.

Include:	1
Exclude:	0

REMOTE CHARACTER MODULE?

This module permits a start-up change to the remote command identifier chosen in response to the question REMOTE COMMAND ID? discussed earlier in this chapter.

Include:	1
Exclude:	0

Optional Command Modules (Continued)

MODE CHANGE MODULE?

This module permits the program mode to be changed at start-up to any of the three possible modes.

Include:	1
Exclude:	0

PEND, KILL MODULE?

The ability to Pend, Kill, and Unpend specific input and output streams is an option. It is recommended for inclusion, but the program can operate without the functions. The three functions are either all included, or all excluded, because of the high degree of common code used by the three.

Include:	1
Exclude:	0

RLDR QUESTIONS

OPERATING SYSTEM?

Defines the operating system under which HAMLET is to be executed.

0	-	RDOS
1	-	MRDOS
2	-	RTOS

Note that an RTOS generation assumes the name of the RTOS system to be RTOS.RB.

NAME?

While all examples in this manual refer to the name HAMLET, the user can name this generation of HAMLET by supplying a legal operating system file name.

Default: HAMLET (HAMLET.SV)

RLDR QUESTIONS (Continued)

NUMBER OF TASKS?

It is the user's responsibility to specify the number of tasks which will be running concurrently. HAMLET requires 10 system tasks as a basic starting point. In addition, 1 task per input stream and 1 task per output stream is required. The user's interface requires 1 task for input and 1 task for output. The maximum number, therefore, is 26 tasks. The user must select a value between 12 (permitting 1 input and 1 output stream) and 26.

Default: 12

NUMBER OF CHANNELS?*

The system uses channel 0 for console input and channel 1 for console output. One additional channel is required for each concurrent stream running, up to a maximum of 14. The user's interface tasks are unrestricted as to their I/O requirements. Thus, the number of channels required ranges from 4 (one input and one output stream) upwards. The only fixed assignments are for console input and output. All other channels are obtained thru the operating system call, .GCHN (get a channel). When a stream completes, it returns its channel. Also the spooling and log options require one channel each if selected.

Default: 5

USER INPUT MODULE?*

The user must specify the name of the relocatable binary module which includes the entry point .HAMI (as discussed in the User Interface chapter).

Default: No user input module.

*Not asked when the operating system was specified as RTOS.

RLDR QUESTIONS (Continued)

USER OUTPUT MODULE?

The user must specify the name of the relocatable binary module which includes the entry point, .HAMO as discussed in the User Interface chapter. User input and output processing is independent. Either may be included with or without the other.

Default: No user output module.

SIGNON CARD?

The user must respond with the exact text to be used as the default SIGNON command, regardless of whether the SIGNON command module is included in this generation. The user need not enter trailing spaces. The generator will append spaces to the 80th character, and no more. Since HASP and ASP use different Sign-On record formats, the user must supply the correct format for the particular program mode being generated. In HAMLET-to-HAMLET mode HAMLET passes the Sign-On record to the remote system and displays it on the remote console.

Default: None.

After a valid response has been given to the above question, the HAMLET program is generated. Figure 3-1 shows a sample CGEN dialogue.

RTOS USERS

An RTOS version of HAMLET has been provided as an absolute binary tape (HAMLET.AB). This program provides the stand-alone user with most of the standard capabilities found in an IBM RJE workstation.

The following parameters were used to build the RTOS.RB module, which is part of the tape:

16K words of memory
A 10 Hz clock
16 tasks
9 channels
2 magnetic tapes
1 cassette tape
1 paper tape reader
1 paper tape punch
2 line printers (132 columns each)
2 card readers
1 Plotter
1 TTY

The parameters shown in Figure 3-1 were used to generate the other modules that comprise the standard binary tape. Note that the CGEN parameters that request the number of tasks and number of channels do not appear in this dialogue, since the operating system to be used was specified as RTOS, and these values are supplied as part of the RTOS.RB generation.

```
NUMBER INPUT STREAMS?  DEFAULT = 1
3)
NUMBER OUTPUT STREAMS?  DEFAULT = 1
3)
PERIPHERAL DEVICE BUFFER SIZE?  DEFAULT = 81.
)
NUMBER OF PERIPHERAL DEVICE BUFFERS?  DEFAULT = 15.
25.)
COMMUNICATIONS DEVICE BUFFER SIZE?  DEFAULT = 200.
)
NUMBER OF COMMUNICATIONS DEVICE BUFFERS?  DEFAULT = 2
3)
MAX BUFFERS PER STREAM?  DEFAULT = 6
8.)
REMOTE COMMAND ID?  DEFAULT = 1
)
STATIC TABLE SIZE?  DEFAULT = 3
)
LINE PRINTER WIDTH?  DEFAULT = 80.
132.)
CONSOLE TEXT?  DEFAULT = 1
)
NUMBER OF SYN CHARACTERS?  DEFAULT = 4
)
DEVICE CODE?  DEFAULT = 30
)
2-4 WIRE?  DEFAULT = 1
)
LOG COMMAND MODULE?  DEFAULT = 1
)
SIGNON COMMAND MODULE? DEFAULT = 1
)
REMOTE CHARACTER MODULE?  DEFAULT = 1
)
MODE CHANGE MODULE?  DEFAULT = 1
)
PEND, KILL MODULE?  DEFAULT = 1
)
OPERATING SYSTEM  DEFAULT = 0
2)
NAME?  DEFAULT = HAMLET.SV
)
USER INPUT MODULE?  DEFAULT =
)
USER OUTPUT MODULE?  DEFAULT =
)
SIGNON CARD?
/*SIGNONΔΔΔΔΔΔΔREMOTE1
```

Figure 3-1. SAMPLE CGEN DIALOGUE

CHAPTER 4

HAMLET OPERATING PROCEDURES

GENERAL

The HAMLET program, which is created by CGEN, operates under a series of commands entered by the user during program initialization (start-up), and/or after program initialization (run-time).

Start-up commands can be used to change some of the options chosen during CGEN, and are entered through the system console device, a file, or as part of the User Interface (fully explained in Chapter 5).

Run-time commands are entered through the system console device, or as part of the User Interface, and can be entered only after program initialization.

The method of loading the HAMLET program depends on the operating system being used.

LOADING HAMLET UNDER RDOS/MRDOS

HAMLET is loaded under RDOS/MRDOS by entering one of the following CLI commands through the system console device:

CLI Command	Explanation
HAMLET)	HAMLET is loaded. CN.READY is displayed on the system console (CN). Program is ready to accept run-time commands. The symbol ( )) denotes a carriage return.

	CLI Command	Explanation
LOADING HAMLET UNDER RDOS/MRDOS (Continued)	HAMLET/S)	HAMLET is loaded. ENTER PARAM OR CR is displayed on the system console. Program is ready to accept start-up commands via the system console.
	HAMLET/SΔ<filename>)	HAMLET is loaded; start-up commands are supplied through the RDOS Line file named in <filename> . The symbol Δ denotes a blank; angle brackets <> enclose a user-supplied entry, and are not entered as part of the command. CN.READY is displayed on the system console. Program is ready to accept run-time commands.

LOADING HAMLET UNDER RTOS

After the HAMLET program has been successfully loaded under RTOS,

```
ENTER PARAM OR CR
```

is displayed on the system console. Program is ready to accept one of the following:

1. Start-up commands through the system console.

2. Carriage return.

   The message

   ```
   CN.READY
   ```

   is displayed on the system console signifying that the program is ready to accept run-time commands.

"ENTER PARAM OR CR" MESSAGE RESPONSE

Regardless of the operating system being used, the message

```
ENTER PARAM OR CR
```

displayed on the system console requires either a carriage return or a start-up command.

CARRIAGE RETURN

A carriage return (not preceded by any other character) is interpreted as the end of start-up commands.

```
CN.READY
```

is displayed on the system console. No further start-up commands can be entered. The program is ready to accept run-time commands.

CONSOLE MESSAGE FORMAT

Messages are displayed on the console device in the following format:

```
<prefix> [<stream number>].<message text>[CODE=xx]<time>
```

Any item enclosed in brackets [] indicates that the item is optional.

<prefix> identifies the source of the message. It can have the following values:

Value	Message Source
CN	Console
DC	Decompress
I	Input
KN	Control
LP	Line Procedure

CONSOLE MESSAGE FORMAT (Continued)

Value	Message Source
O	Output
UO	User Output
UI	User Input

<stream number> identifies the input or output stream number when the <prefix> is either I or O.

CODE = xx is displayed as the result of a system error. xx is the error code returned by the operating system.

For example, the following message:

```
01.  I/O ERROR CODE = 22     13:01:00
```

means an I/O error has occurred in output stream 01. The RDOS code 22 denotes that the line limit of 132 characters has been exceeded.

<time> identifies the time of day (hours:minutes: seconds).

DIAL-UP PROCEDURE

After the initial "CN.READY" message appears on the system console, the operator is ready to make the necessary connection with the remote computer.

The operator dials the remote computer, and on hearing a high-pitched tone, depresses the button labelled "DATA", and places the phone back on the cradle. If the "DATA" button doesn't light, either the user has generated HAMLET with an incorrect device code, or there is an equipment malfunction.

HAMLET OPERATING PROCEDURES (Continued)

DIAL-UP PROCEDURE (Continued)

After the communication line has been properly established, HAMLET bids for the line. If the line-bid is not answered, HAMLET will re-try in three seconds. After 10 re-tries (approximately 30 seconds), the message

```
LP.LINE BID FAIL
```

is displayed on the system console and HAMLET continues to bid for the line.

If the bid is answered, HAMLET sends the Sign-on record automatically. If accepted, then the operator is ready to communicate with the remote computer.

If HAMLET is communicating with HASP or ASP, and one of the following occurs:

1. The format of the Sign-on record is invalid;
2. The Remote ID number or password is incorrect;
3. Another user is already logged on the HASP or ASP system under the same Remote ID number; or
4. The HASP or ASP operator has not started the line,

then HASP or ASP will acknowledge the Sign-on record, but after processing the record will disconnect the line. When HAMLET recognizes the disconnect, the message

```
LP.LINE ERROR
```

is displayed on the console device and the program terminates.

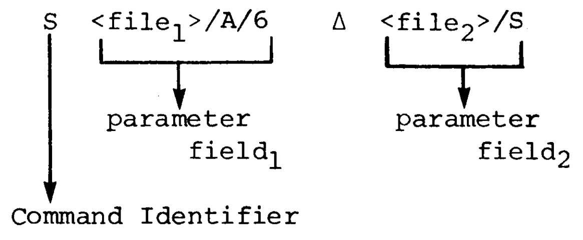
HAMLET COMMANDS

HAMLET start-up and run-time commands are shown in Table 4-1. The table specifies when the appropriate commands can be entered into the system.

COMMAND FORMAT

A command is comprised of a command identifier, followed by one or more parameter fields.

Example:



The following rules apply when entering a command:

1. Blanks are permitted between the command identifier and field$_1$.

2. Blanks are <u>not</u> permitted within a parameter field.

3. At least <u>one</u> blank must appear between parameter fields when a command has multiple fields.

4. The last parameter field must be immediately followed by a carriage return.

<u>Examples</u>:

```
XΔACCT/A/6ΔRATE/S
XΔΔACCT/A/6ΔΔRATE/S
XACCT/A/6ΔΔΔRATE/S
```

In the above example, all commands are syntactically correct, and would all be accepted by HAMLET.

ENTERING START-UP COMMANDS

As has been previously explained, the

```
ENTER PARAM OR CR
```

message signifies that HAMLET is ready to accept start-up commands via the system console. At this point, the user can either enter a command or carriage return. When a command is entered and accepted by the program, another

```
ENTER PARAM OR CR
```

is displayed on the system console. After all start-up commands have been entered, a carriage return (not preceded by any other character) is issued. The message

```
CN.READY
```

is then displayed on the system console signifying that the program is ready to accept run-time commands.

ENTER RUN-TIME COMMANDS

After CN.READY is displayed, run-time commands are entered. After each run-time command is entered and accepted by the system, another CN.READY message is displayed.

HAMLET OPERATING PROCEDURES (Continued)

COMMAND DESCRIPTIONS

The following pages describe each HAMLET command. Commands are listed in alphabetic order for ease of reference. Table 4-1 is a summary of start-up and run-time commands.

		CAN BE ENTERED		
Command Identifier	Function	Start-Up Only	Run-Time Only	Any-Time
A	ASSIGN FILE (STATIC)			X
C	REMOTE COMMAND ID	X		
H	HALT PROGRAM			X
K	KILL STREAM		X	
L	LOG FILE			X
M	PROGRAM MODE	X		
O	ASSIGN FILE (DYNAMIC)		X	
P	PEND STREAM		X	
S	SIGNON RECORD	X		
U	UNPEND STREAM		X	
X	TRANSFER FILE		X	

Table 4-1. Summary of HAMLET Start-up and Run-Time Commands

HAMLET OPERATING PROCEDURES (Continued)

Command Identifier:	A (STATIC ASSIGNMENT)
Format:	A <stream-type-id> [Δ<filename> [Δ<forms-id>]]
Function:	This command creates an entry in a table that permits HAMLET to automatically assign the supplied <filename> and optional <forms-id> to the requesting <stream-type-id>.

Description:

A <stream-type-id> is composed of a stream-type and a number between 1 and 7 inclusive. All streams transmitted and received must be specified by a <stream-type-id>. The valid stream-types are listed in table 4-2. Stream-type-ids provide information about the format of the stream they identify. For example, a "print" stream-type identifies data intended to be output to a line printer, and an "ASCII sequential" stream-type identifies data intended to be written in ASCII sequential format.

When connected to a HASP system, only "Print" or "Punch" stream-types can be received by HAMLET. These are identified as "P" and "N" type streams respectively. When connected to another ECLIPSE, all stream-types listed in Table 4-2 can be received.

Since multi-leaving permits reception of multiple streams of the same type, they are distinguished by a number. Thus, P1 and P2 can coexist and represent print-stream-1 and print-stream-2.

When a remote system wants to send an output stream, it first sends a "request-to-send" message that contains the stream-type-identifier for the particular stream that the remote wishes to send. In effect, the remote system is asking if the receiver can accept "print-stream-1".

Command Identifier: A (STATIC ASSIGNMENT) (Continued)

Description: (Continued)

When HAMLET receives this request, it determines if it has the resources required to acknowledge the remote's request. One of the required resources is a file to which the received stream will be written. This file can be assigned in two ways: One is by a static assignment (described below), and the other is by a dynamic assignment ("O" command described later in this chapter).

The A command permits the static (or automatic) assignment of output files to requesting streams. Thus, when HAMLET is collecting its resources, it scans a table of static assignments to see if there is a file to be assigned to the requesting stream. The table contains stream-type-ids and associated files. If a stream-type-id in the table matches the request, the assignment is made.

Note that this table is created prior to receiving the "request-to-send" from the remote system. When a match is made, the file associated with the stream-type-id is automatically opened for output.

Thus, the command:

```
    A  P1  $LPT)
```

says that if a request-to-send P1 is ever received from a remote system, assign the file $LPT for output. Do this each time a request-to-send P1 is received. For while only one P1 may exist at anytime, that identifier can and will be reused to send streams serially.

The <filename> assigned is not checked by HAMLET for suitability to the stream-type being received. It is the user's responsibility to make meaningful associations between stream-type and <filename>.

Command Identifier: A (STATIC ASSIGNMENT) (Continued)

Description: (Continued)

The <forms-id> parameter can be 1 or 2 characters and represents the identifier for an optional printer vertical forms control table (discussed in detail in Appendix B). When present, with stream-type "print", it becomes the table that will be used to simulate vertical forms control. When the stream-type is not print, the <forms-id> has no effect.

Thus,

```
    A  P1  $LPT  F1)
```

says to create an entry in the automatic assignment table so that whenever a "request-to-send" print-stream-1 is received, file $LPT will be assigned and data will be output using vertical forms control table, F1.

The number of entries that can be contained in the automatic assignment table is a CGEN option.

Entries may be altered, added, or deleted at any time. To alter an entry, re-enter the command with the new <filename> and/or <forms-id>. Thus assuming the table has assigned P1 as above.

```
    A  P1  FILER  F6)
```

associates FILER and forms-id F6 with P1 replacing $LPT and F1.

To remove a stream-type assignment, enter the stream-id followed immediately by a carriage return.

Thus:

```
    A  P2 )
```

removes the assignment of P2 and permits the space utilized to be reused by another assignment.

Table 4-2. Permissable Stream Type Identifiers

STREAM TYPE	DESCRIPTION	HASP MODE		ECLIPSE MODE	
DESIGNATOR		SEND	RECEIVE	SEND	RECEIVE
-	CONTROL RECORD	✓	✓	✓	✓
-	OPERATOR MESSAGE DISPLAY REQUEST		✓	✓	✓
-	OPERATOR COMMAND	✓		✓	✓
L	INPUT	✓		✓	✓
P	PRINT		✓	✓	✓
N	PUNCH		✓	✓	✓
S	ASCII SEQUENTIAL	✓*		✓	✓
E or H	EBCDIC SEQUENTIAL	✓*		✓	✓
U	USER DEFINED	✓*		✓	✓

*These will appear to HASP as type "INPUT".

Command Identifier: C (Remote Command ID)

Format: C <remote command character>

Function: This command allows the user to change the remote command character chosen during CGEN.

Any command or message preceded by the remote command character is passed to the remote computer without HAMLET interpretation.

In HAMLET-to-HASP program mode the remote command character is $ (dollar sign); HAMLET-to-ASP remote command character is * (asterisk). A HASP or ASP command is executed by HASP or ASP, and the result is returned on the ECLIPSE workstation's console device.

In HAMLET-to-HAMLET program mode, any desired character can be used as the remote command character. Messages preceded by the remote command character will be displayed on the remote ECLIPSE workstation's console device.

NOTE: This command can only be used if the Remote Character module was included during CGEN.

Example:

```
C$)
```

HAMLET OPERATING PROCEDURES (Continued)

Command Identifier:	H (HALT)
Format:	H
Function:	This command causes HAMLET to terminate. First, a "SIGNOFF" record is sent to the remote station, if the communication link exists. Then, the communications link is disconnected and finally, a return to the operating system is made.

Example:

```
    H )
```

Command Identifier:	K (Kill)
Format:	K <stream-type-id>

Function:

This command causes the referenced stream to be immediately terminated. When an input stream is referenced, and it has previously sent data, then an EOF is encoded and queued for transmission. If no transmission has occurred, the stream is stopped and all its resources are freed. If an output stream is referenced, a permanent wait-a-bit is set to prevent further reception, and all buffers queued are released prior to executing a PEND statement.

The <stream-type-id> may be any input stream currently running (I1, I2, ..., I7) or any output stream (O1, ..., O7).

NOTE: This command can only be used if the PEND/KILL module was included during CGEN.

Example:

```
K I2)
```

Command Identifier:	L (Log)
Format:	L [<filename>]

Function:

This command causes the old log file (if any) to be closed and the new file named in <filename> to be opened as the log file. Subsequently, all messages entered at the console, or written to the console, will be logged in the log file. To examine the log file, issue another L command, and then transfer the old file to a suitable device.

For example:

```
    L  LOG2 )
    X  LOG1  $LPT )
```

creates a new log file (LOG2), closes the previous file (say LOG1) and prints the old file on the line printer. If no file name is given then no additional logging of messages will occur until another L command is given.

NOTE: This command can only be used if the Log Command module was included during CGEN.

Command Identifier:	M (MODE)
Format:	M <program mode>

Function:

This command allows the user to change the program mode chosen during CGEN.

0	-	HAMLET TO HASP/ASP
1	-	HAMLET TO HAMLET (Sends Sign-On Record)
2	-	HAMLET TO HAMLET (Receives Sign-On Record)

This command can only be used at start-up. The mode cannot be changed during run-time.

NOTE: This command can only be used if the Mode Change module was included during CGEN.

Example:

```
M 1)
```

Command Identifier:	O (Dynamic Assignment)
Format:	O <stream #> [Δ<filename> [Δ<forms-ID>]]
Function:	Dynamically assign an output file to a requesting output stream, and automatically transmit a "permission-to-send" to the remote computer.

Description:

This command is used only when a console message is displayed requesting an output file assignment. A request for a dynamic output file assignment indicates that a static output file assignment has not been previously entered (see A command).

where:

O signifies that this is a dynamic output file assignment.

<stream #>	identifies the specific output stream that will handle the output.
<filename>	can by any output filename (equal to or less than 15 characters). If the file does not exist, HAMLET will attempt to create it.
<forms-ID>	has the same interpretation as the A command. It selects the table to be used to simulate vertical forms control (Print type streams only).

The following examples should help illustrate the methods that can be used to dynamically assign an output file.

Command Identifier: O (Dynamic Assignment) (Continued)

Description: (Continued)

Example 1:

The following console message is displayed:

```
O2. ASSIGN P2 FILE?
```

where this message is from output stream 2 (O2) requesting a dynamic assignment for a P (Print) stream-type, where the digit 2 in P2 further qualifies the stream as Print-Stream-2. (Note that in general Print stream messages will have the same output stream number and print stream number [O2 and P2 in the above example]).

The following are examples of some valid commands that could be issued by the operator in response to the above request for a dynamic file assignment:

(1) O2 Δ ACCT ⮐

where output is to a disk file call ACCT, and the file will be output later to the line printer as a local HAMLET file transfer. HAMLET's default vertical-forms-control table will be used.

(2) O2 Δ $LPT Δ F1 ⮐

where output is to the line printer. F1 is the name of the table to be used for vertical format control (see Appendix B for generation of optional vertical-forms-control tables).

Command Identifier: O (Dynamic Assignment) (Continued)

Description: (Continued)

Example 2:

The following console message is displayed:

```
O7. ASSIGN N1 FILE?
```

where this message is from output stream 7 (O7) requesting a dynamic assignment for an N (Punch) stream-type (note that in N stream-types, output stream numbers and Punch stream numbers are generally not the same). If the operator should enter

```
O7 Δ DFILE)
```

Punch stream-1 is assigned to output stream 7 and will be output to a disk file named "DFILE".

Killing a File Assignment Request

If the operator does not wish to accept the output stream, the Command Identifier (O) is entered followed by the output stream number and a carriage return. This causes the "request-to-send" to be denied by HAMLET. The remote computer will not send the data.

HAMLET OPERATING PROCEDURES (Continued)

Command Identifier:	P (PEND)
Format:	P <stream-type-id>
Function:	This command causes the referenced stream to be pended. No resources are freed in anticipation of a subsequent unpend (U command). The referenced Input or Output stream is simply pended. On output, a "wait-a-bit" is set; on input, no more device I/O is performed.

NOTE: This command can only be used if the PEND/KILL module was included during CGEN.

Example:

```
P 03 )
```

Command Identifier:	S (SIGN-ON)
Format1:	S <Sign-On record>
Format2:	S <filename>

Function: This command allows the user to change the Sign-On record chosen during CGEN.

In format 1, the <Sign-On record> can be up to 80 characters following the command identifier. It must be in the precise format required by the remote system (e.g., HASP or ASP). Only the essential data need be supplied; the record will be filled to 80 characters by HAMLET.

Example:

```
    SΔ/*SIGNONΔΔΔΔΔΔΔREMOTE 26⤸
```

If the first non-blank character after the command-id (S) is not a slash (/) then format 2 is assumed. In format 2, a <filename> is given which contains a Sign-On record of up to 80 characters in the precise format required. HAMLET will issue a read (.RDL) to the file and use the first record read as the Sign-On. The file will then be closed.

Example:

```
    S  $CDR⤸
```

where the first record (card) in the card reader ($CDR) is the Sign-On record.

Command Identifier:	U (UNPEND)
Format:	U <stream-type-id>
Function:	This command causes a previously pended stream to continue execution. It has no effect if the referenced stream is not currently pended.

NOTE: This command can only be used if the PEND/KILL module was included during CGEN.

Example:

```
    U I2)
```

Command Identifier: X (Transfer file(s))

Format:

```
X <Field 1> [<Field 2>]
```

where:

<Field 1> ≡ <filename$_1$>[/<stream-type>[/<stream #>]]

<Field 2> ≡ <filename$_2$>[/<stream-type>]

Function: To transfer files locally or remotely.

Description: A local transfer is one that does not involve the communication line. A file is copied to another local file, with some transformations permitted (see Table 4-3).

A remote transfer causes the named file to be sent over the communication line to the remote system.

The command can be considered as two fields. Field 1 is composed of <filename$_1$>, <stream-type>, and <stream #>; Field 2 is composed of <filename$_2$> and <stream-type>. Field 2 is optional. If absent, a remote transfer is implied. If present, a local transfer is implied.

<stream-type> represents the organization of the data (Table 4-4). To specify a stream-type, its designator (L, S, E, U, H, N, P) is entered.

Thus,

```
X FILEK/S )
```

states that FILEK is to be read by HAMLET as an ASCII sequential file and sent to the remote system.

Command Identifier:	X (Transfer file(s)) (Continued)
Description: (Continued)	If a <stream #> is specified in Field 1, then the named file is sent as the indicated stream-type and stream-number.

Thus,

```
    X  FILEK/S/3 )
```

states that FILEK is to be sent as ASCII-sequential-stream #3. If the <stream #> is not given, the system picks the lowest inactive stream # starting at 1.

When transmitting to HASP, keep the following in mind: HASP assumes it is receiving data from one (or more) card readers (regardless of whether data is being sent by HAMLET from disk, tape, etc.). In order to transmit data to HASP using two or more concurrently operating input streams, the HASP system must have been generated with as many remote card readers specified for the ECLIPSE workstation as the number of input streams that can concurrently transmit data to HASP.

The use of the stream # has more application to HAMLET-HAMLET transmission.

Also, HASP only receives stream-type "input". Therefore, when the X command specifies ASCII sequential (S) and the remote system is HASP, the stream-type is changed to type "INPUT" by HAMLET automatically to conform to HASP conventions. In HAMLET-HAMLET mode, the stream-type remains ASCII sequential, and thus provides information at the receiving system about the organization of the data to be received.

Command Identifier: X (Transfer file(s)) (Continued)

Description: (Continued)

If no <stream-type> is specified in the command, then the file is assumed to exist in "LINE" mode (.RDL/.WRL issued).

Thus,

```
X  JOBA
```

causes file JOBA to be opened and its contents transmitted. The file is read using a .RDL system call.

For other <stream-types>, HAMLET performs the following actions:

Print (P) - Data is output to the line printer using .WRS to simulate forms control.

Punch (N) - Data is output as though the type were Binary sequential.

ASCII Sequential - Data is input and output using .RDS/.WRS with a byte count of 80. It is assumed to be in ASCII code.

EBCDIC Sequential - Data is input and output using .RDS/.WRS with a byte count of 80. No translation is performed. EBCDIC is used here to represent non-ASCII format.

Hollerith (H) - When referencing the card reader, ($CDR) a .RDL will be issued that converts the Hollerith code to ASCII. For EBCDIC conversion, the "H" must be specified as in:

```
X  $CDR/H )
```

Command Identifier:	X (Transfer file(s)) (Continued)

Description: (Continued)

User Defined (U) - This type is not interpreted by HAMLET. It assumes that the user will provide a task for input and output functions. Since a user must specify a task, the command must come from the user interface (Chapter 5).

In Field 2, a <stream-type> can be selected. If different from <stream-type> in Field 1 then conversion to the output stream type occurs (if permitted by Table 4-4). Note that no stream # can be assigned in Field 2. Local transfers are all assigned a stream # of 0 on the output side.

The following examples summarize this discussion:

```
    X ALPHA BETA )
```

Input ALPHA using .RDL and output BETA using .WRL.

```
    X ALPHA/2 BETA/S )
```

Input ALPHA using .RDL and denote as stream 2. Output BETA using .WRS with no translation. The stream # is assigned as 2.

```
    X $CDR/H )
```

Read the card file in the reader ($CDR) and send it to the remote system.

```
    X ALPHA $LPT )
```

Input ALPHA using .RDL and output to the line printer ($LPT) using .WRL.

Command Identifier: X (Transfer file(s)) (Continued)

Description: (Continued)

```
    X ALPHA $LPT/P )
```

Input ALPHA using .RDL and output to $LPT as a print job. This will not work well because the Forms control information is specially encoded by HASP and the local transfer does not create this special character. (For those interested, it is called an SRCB and is defined in IBM's HASP manual.)

```
    X ALPHA/S/2 )
```

Send ASCII sequential stream #2 to the remote system.

To transfer a print-stream from a disk file to the local printer,

```
    X SPRINT/S $LPT/S )
```

must be used since file SPRINT contains multiple "LINE-FEED" characters that simulate vertical forms control. This information was added when the print stream was originally output to file SPRINT.

Table 4-3. Permissable Transfer Types

FROM		TO			
		ASCII LINE	ASCII SEQUEN.	EBCDIC SEQUEN.	PRINT
ASCII LINE	(L)	√	√	√	
ASCII SEQ	(S)	√	√	√	
EBCDIC SEQ	(E)	√	√	√	
USER	(U)	√	√	√	√*
HOLLERITH	(H)	√	√	√	
PUNCH	(N)**	√	√	√	

*User must provide SRCB.

**Treated as binary sequential.

Table 4-4. Permissable Stream Type Identifiers

STREAM TYPE	DESCRIPTION	HASP MODE		HAMLET MODE	
DESIGNATOR		SEND	RECEIVE	SEND	RECEIVE
-	CONTROL RECORD	√	√	√	√
-	OPERATOR MESSAGE DISPLAY REQUEST		√	√	√
-	OPERATOR COMMAND	√		√	√
L	INPUT	√		√	√
P	PRINT		√	√	√
N	PUNCH		√	√	√
S	ASCII SEQUENTIAL	√*		√	√
E or H	EBCDIC SEQUENTIAL	√*		√	√
U	USER DEFINED	√*		√	√

*These will appear to HASP as type "INPUT".

```
A P1 $LPT                          -       COMMANDS OBTAINED FROM
A P2 X1                                    STARTUP FILE VIA:
A P3 X2                                    HAMLET/S <FILENAME>
S /*SIGNON       REMOTE1                   NOTE THAT THE L COMMAND TO CREATE
M 0                                        THIS LOG FILE DOES NOT APPEAR
C $                                        IN THE FILE.
CN.READY 13:36:50                  -       OPERATOR MAY DIAL ANYTIME NOW
O1.STARTED  13:37:19               -       HASP SENDS A PRINT JOB IMMEDIATELY
O1.ENDED  13:38:02                         FROM A PREVIOUS SESSION
$DQ                                -       HASP DISPLAY QUEUES COMMAND
CN.READY 13:38:08
$*13.45.08  17 XEQ A               -       RESULT OF $DQ COMMAND
$*13.45.08   4 XEQ B
$*13.45.08   1 XEQ I
$*13.45.08   7 PRT   3
$*13.45.08   1 PRT  16
$*13.45.08   2 PRT  22
$*13.45.08   1 PRT  24
$*13.45.08   1 PUN   0
$*13.45.08  11 HOLD
$*13.45.08  15 PERCENT SPOOL UTILIZATION
X JOBA                             -       TRANSFER FILE JOBA TO HASP
CN.READY 13:38:18
I1.STARTED  13:38:19               -       INPUT STREAM 1 STARTED (JOBA)
$*13.45.20 JOB 321 ON RM1.RD1     -- DGI01
PI1                                -       PEND INPUT STREAM 1 (#321)
CN.READY 13:38:42
UI1                                -       UNPEND STREAM 1
CN.READY 13:38:57
I1.ENDED  13:39:19                 -       INPUT STREAM 1 TRANSFER COMPLETED
$DJ 321
CN.READY 13:39:34
$*13.46.33 JOB 321 DGI01     AWAITING EXEC     A PRIO  2
$CJ 321                            -       COMMAND TO HASP TO CANCEL 321
CN.READY 13:39:42
$*13.46.41 JOB 321 DGI01     AWAITING PRINT    1 PRIO  2
O1.STARTED  13:39:44               -       OUTPUT PRINT STREAM 1
O1.ENDED  13:39:54                         CONTAINS JOB 321 OUTPUT
X JOBB                             -       TRANSFER FILE JOBB
```

Figure 4-1. Sample Command Dialogue

```
CN.READY 13:40:04
I1.STARTED  13:40:05
I1.ENDED  13:40:16
$*13.47.11 JOB 324 ON RM1.RD1  -- DGCJOB03
X SPRINT/S $LPT/S                 -       START LOCAL TRANSFER
CN.READY 13:40:19
I1.STARTED  13:40:19
I1.ENDED  13:40:34
O0.ENDED  13:40:35                -       LOCAL OUTPUT STREAMS ASSIGNED # 0
A P1                              -       REMOVE STATIC ASSIGNMENT
CN.READY 13:40:46
$CJ324
CN.READY 13:40:54
$*13.47.53 JOB 324 DGCJOB03 AWAITING PRINT   1 PRIO  2
O1. ASSIGN P1 FILE ? 13:40:56     -       DYNAMIC ASSIGN
O1 $LPT F1                        -       ASSIGN PRINT STREAM 1 TO $LPT
CN.READY 13:41:05                         USING FORMS CONTROL, F1
O1.STARTED  13:41:06
O1.ENDED  13:41:16
X JOBA
CN.READY 13:41:23
I1.STARTED  13:41:24
$*13.48.24 JOB 325 ON RM1.RD1  -- DGI01
X SPRINT/S $LPT/S
CN.READY 13:41:41
I2.STARTED  13:41:42              -       LOCAL TRANSFER
I2.ENDED  13:41:58
O0.ENDED  13:41:58
I1.ENDED  13:42:22                -       REMOTE TRANSFER COMPLETES
$CJ325
CN.READY 13:42:51
$*13.49.51 JOB 325 DGI01    AWAITING PRINT   1 PRIO  2
O1. ASSIGN P1 FILE ? 13:42:54
O1 $LPT F2                        -       UNKNOWN FORMS ID ASSIGNED
CN.READY 13:43:16
O1.STARTED  13:43:17
O1.BAD FORMS ID, USING STANDARD 13:43:17
O1.ENDED  13:43:28
H                                 -       SIGNOFF SENT, PROGRAM TERMINATED
```

Figure 4-1. Sample Command Dialogue (Continued)

CHAPTER 5

USER INTERFACE

GENERAL

The HAMLET User Interface provides a programmed link to the HAMLET program's facilities. Using modules provided in this chapter, the user can queue data for transmission and/or dequeue data received from HASP and ASP, or another ECLIPSE.

The functions provided by the user interface are optional and in addition to those provided by the standard program.

The following steps are necessary to utilize the option:

1. Create interface modules which define the input and output controller and associated tasks.
2. Assemble these modules using the MLP.SR parameter tape to create the proper .RB files.
3. Specify the .RB file names to the CGEN program.

The following pages describe the facilities available to the user. These are listed in alphabetical order for ease of reference. A sample user interface using many of these functions can be found at the end of the chapter.

Abbreviation and Notes

SSA represents the caller's stream slot address. This value is stored in the user stack point (USP) for each user task by .KMII or .KMOI at initialization. It must not be changed by the user. The stream slot is an area of memory reserved for each user task permitting storage of the stream's state.

There is a set of functions for use by input tasks starting with the letters .KMI. Another set for use by output tasks starts with the letters .KMO. Most other functions are for use of either type of task except where noted.

Registers are destroyed on all function returns with the exception of AC2 which contains the SSA.

.EROR

Function:

Provides a mechanism for displaying error status on the system console device.

Call:

```
AC2 = SSA
[ACO = system error code]

.EROR
<VALUE> [+1B0]
Normal Return
```

<value> is an error code displayed either as a number, or equivalent text if the text option is selected at CGEN. If the optional Bit 0 is set to 1, the value passed in ACO is also displayed following "CODE=". AC2 must contain the stream slot address that can be obtained from USP.

Return:

Always returns.

<u>USER INTERFACE (Continued)</u>

<u>.FATL</u>

<u>Function</u>:	Display calling function's program counter (PC) and terminate HAMLET due to an unrecoverable error condition.
<u>Call</u>:	.FATL No Return
<u>Return</u>:	Never returns.

<u>.HAMI</u>

<u>Function</u>:	This name must be declared as an entry point in the user input module named during CGEN. When HAMLET discovers an address for .HAMI, it starts a task (input controller) with priority = 3 and task ID = 43 (octal) at .HAMI. This task can start either program or user tasks to control input streams.

<u>.HAMO</u>

<u>Function</u>:	This name must be declared as an entry point in the user output module named during CGEN. When HAMLET discovers an address for .HAMO, it starts a task (output controller) with priority = 2 and task ID = 42 (octal) at .HAMO. The task thus started uses task communication word .UTCW to effect communications with HAMLET. This task can start either program or user tasks to control output streams.

.HIFI

Function: Passes commands to HAMLET's command interpreter. This module acts like an input command console for the User Interface.

Call:

```
AC0 = Byte pointer to command text
AC1 = Byte count
AC2 = Starting task address, or
AC2 = 0 (No task to start)

.HIFI
Exception return
Normal return
```

The text must be a valid command (as described in Chapter 3). When the user starts a user defined task as part of the command string's interpretation (X command), this address must be passed in AC2. If no task address is indicated (AC2 = 0), a program task is started. However, if the stream type is "U" a user task address must be provided. The user must check AC0 on an exception return.

Exception Return: AC0 = Codes as listed in Appendix A.

Normal Return: The command was interpreted without error.

USER INTERFACE (Continued)

.KMIA

Function:	Provides for abnormal input task termination. Caller's resources are released. If data has been previously queued via a call to .KMIQ, an end-of-file is queued for transmission, otherwise the task is killed immediately.

Call:	AC1 = Address of Buffer to Free, or
	AC1 = 0 (No Buffer to Free)
	AC2 = SSA
	
	.KMIA
	No Return

Return:	Never returns.
Note:	It is the user's responsibility to free any buffer acquired via a .KMIB call. The mechanisms available to do this are .KMIQ, .KMIA or .KMIE.

.KMIB

Function:	Gets a buffer from the peripheral device buffer pool and sets-up a Byte-pointer to the start-of-data. That is, to the point where the user may begin to store data into the buffer. No required accumulators on Call.
Call:	.KMIB Exception Return (Not Used) Normal Return USP must contain the caller's stream slot address.

Normal Return:	AC0 = Byte pointer to start of data area
	AC1 = Unspecified
	AC2 = SSA
	AC3 = Address of buffer obtained

Note:	.KMIB calls for a buffer and waits if either there are no more buffers in the pool, or the caller has too many buffers charged to him. Each time a .KMIB call is successful, the caller is charged for one buffer. Other functions in HAMLET (like .KMIQ) subsequently free these buffers and credit the user. The maximum number of buffers that a user can charge at any time is a CGEN function. It prevents any one task from securing all buffers in the pool.

.KMIE

Function:	Enqueues an End-of-file and awaits a .KILL from some other HAMLET function (which will also insure that all resources held by the caller are released). This is the normal input task termination call.

Call:	AC1 = Buffer address
	AC2 = SSA
	
	.KMIE
	No Return

Return:	Never returns.

.KMII

Function:	Provides standard initial logic for input tasks. A channel is obtained (stored at S.CHN), file opened, and a "request-to-send" generated where required. This function should be the first code executed by a starting task. The stream slot address reserved for the caller is assumed to be in AC2 (as it is when the task is first started by HAMLET). AC2 is stored in USP by this function.
Call:	AC2 = SSA .KMII Exception return (Not Used) Normal return If the task cannot be started due to lack of resources or other error, an abnormal termination results and there is no return to the caller.
Normal Return:	Task has been initialized.

.KMIQ

Function:	Queues input records for compression and transfer. The buffer containing data supplied by the user is queued to the compression queue. From that point HAMLET encodes and sends the data to its destination.
Call:	AC1 = Address of buffer to queue AC2 = SSA .KMIQ Exception return (Not Used) Normal return
Normal Return:	Buffer is queued for compression.

.KMOA

Function:	Terminates an output task abnormally. If the task is receiving communications data, a permanent wait-a-bit is set to prevent further reception. This is required since there is no method for a stream to send a "KILL" condition to the remote computer. If it is a local task, then all resources are freed and the input task's "KILL" flag is set which will ultimately cause that task to encode an EOF.
Call:	AC1 = Address of buffer to free, or AC1 = 0 (No buffer to free) AC2 = SSA .KMOA No Return
Return:	Never returns.

.KMOD

Function:	Performs dequeue functions for output tasks. Each stream slot has its own queue from which buffers are dequeued for output to peripheral devices.
Call:	No parameters are required on input. .KMOD Exception Return Normal Return

Exception Return:	AC1 = 0
	AC2 = SSA
	AC3 = Buffer address
	
	This return is taken only when an end-of-file condition occurs. The buffer returned contains no data but the user must free this buffer before terminating the task either by a call to .KMOF, or .KMOE.

Normal Return:	AC0 = Byte pointer to data start
	AC1 = Byte count
	AC2 = SSA
	AC3 = Address of buffer dequeued
	Carry = 0 - ASCII coded data
	= 1 - EBCDIC coded data

.KMOE

Function: Provides normal output task termination. Resources are freed and caller is KILLed.

Call:

```
AC1 = Address of buffer to free, or
AC1 = 0 (No buffer to free)
AC2 = SSA

.KMOE
No Return
```

If AC1 = 0, then HAMLET assumes all buffers have been freed. Otherwise, AC1 is assumed to be a buffer address.

Return: Never returns.

.KMOF

Function:	Provides for return of buffers obtained through calls to .KMOD. Buffers are obtained from and returned to a pool created by HAMLET at program start-up. When buffers are obtained by a task the task is charged. When they are returned, the task is credited. A task can be pended by the program if it has too many buffers charged to it. (See CGEN parameter, "Max Buffers per Stream").
Call:	AC1 = Buffer address to be freed AC2 = SSA .KMOF Exception Return (Not Used) Normal Return
Normal Return:	The buffer has been returned to the buffer pool.

USER INTERFACE (Continued)

.KMOI

Function: Provides initialization of output tasks. Acquires resources, and opens an output file after making a request for name, if necessary. It uses the device inhibit mask passed by the caller in AC0. This is the mask used in the RDOS system call .OPEN. The channel number obtained is stored at S.CHN in the stream slot, and the address of the stream slot is stored in USP. If an abnormal condition results from this initialization, the task is aborted.

Call:

```
AC0 = Device Inhibit Mask
AC2 = SSA

.KMOI
Exception Return (Not Used)
Normal Return
```

Return: The output task has been initialized.

.TRCD

Function: Provides for translation of a text string from ASCII to EBCDIC or EBCDIC to ASCII.

Call:

```
AC0 = Byte pointer to start of text string
AC1 = Byte count
AC2 = SSA
Carry = 0 - ASCII to EBCDIC
      = 1 - EBCDIC to ASCII

.TRCD
Exception Return (Not Used)
Normal Return
```

Normal Return: AC0, AC1, AC2, unchanged. The text string has been translated in place.

<u>USER INTERFACE (Continued)</u>

<u>.TRWN</u>

<u>Function</u>:	Provides for translation of a text string from ASCII to EBCDIC or EBCDIC to ASCII. Additionally, nulls in the source text are converted to spaces on output.

<u>Call</u>:

```
AC0 = Byte pointer to start of text string
AC1 = Byte Count
AC2 = SSA
Carry = 0 - ASCII to EBCDIC
      = 1 - EBCDIC to ASCII

.TRWN
Exception Return (Not Used)
Normal Return
```

<u>Normal Return</u>:	AC0, AC1, AC2, unchanged. The text string has been translated in place.

.USRI

Function: The user's input controller must use this symbol as an SSA to identify the start of his stream storage area referred to as a pseudo-stream slot.

Whenever a call to .WTC, .WTX, or .EROR is made, and if AC2 = .USRI, then messages displayed will have the prefix:

```
    UI
```

For example:

```
                .ENT        .USRI
                  .
                  .
                  .
                LDA     2, USRI         ;Address of PSS
                LDA     0, ECOD         ;Error Code
                .WTC                    ;Write to Console
                  .
                  .
                  .
    ECOD:       ER.BC                   ;Bad Command
    USRI:       .USRI
   .USRI:       .BLK        S.PSL       ;allocate a pseudo
                                        ;stream slot (PSS)
                                        ;of length S.PSL.
```

Produces the message:

```
    UI.BAD COMMAND    13:02:02
```

on the system console.

USER INTERFACE (Continued)

.USRO

Function: This symbol has the same meaning for the user's output controller that .USRI has for the user's input controller. Messages for output will be prefixed as:

```
UO
```

For example:

```
                .ENT        .USRO
                .TXTM   1
                  .
                  .
                  .
                LDA     2, USRO       ;Address of PSS
                LDA     0, BPTXT      ;Byte Pointer
                LDA     1, BC         ;Byte Count
                .WTX                  ;Write to Console
                  .
                  .
                  .
    BPTXT       .+1*2
                .TXT    /REQUEST TO SEND REJECTED/
    BC:         24.
    USRO:       .USRO                 ;
   .USRO:       .BLK        S.PSL     ;allocate a pseudo-
                                      ;stream slot (PSS) of
                                      ;length S.PSL
```

produces the message:

```
UO.REQUEST TO SEND REJECTED    12:06:05
```

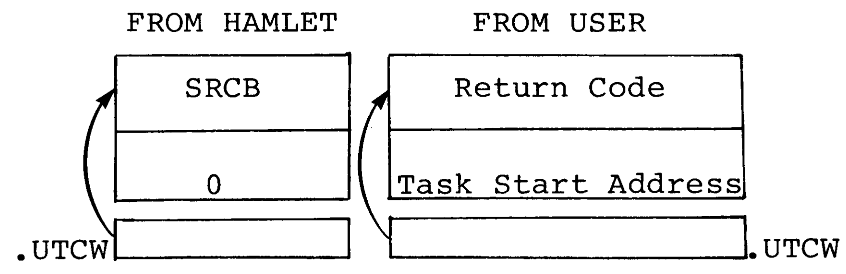
on the system console.

USER INTERFACE (Continued)

.UTCW

Function: This task communication word is used by the user's output controller and HAMLET to exchange information. Whenever a "request-to-send" is received by HAMLET, the SRCB associated with the call is passed to the user's output controller. The user then returns his decision as to what is to be done with the request. The physical mechanism is as follows:

.UTCW contains a pointer to a fixed two word table that is allocated by HAMLET and used by both HAMLET and the output controller. That is, information passed by HAMLET is overwritten by information returned by the output controller.



HAMLET issues .XMTW to the users output controller (if defined) and then performs a .REC on .UTCW. The user's output controller performs a .REC on .UTCW. The user extracts the SRCB and stores the return code and an optional task start address in the table. The return code may be one of the following:

Code = 1 - Ignore the request.
Code = 2 - Start appropriate program task.
Code = 3 - Start user specified task whose address is given following this code's location.

.WTC

Function: Provides for console output of coded error messages.

Call:

```
AC0 = Error code
AC2 = SSA

.WTC
Normal Return
```

The error code passed must be one of those listed in Figure 5-3.

Normal Return: Error message is queued for output.

.WTX

Function: Provides for console output of text specified by the user.

Call:

```
AC0 = Byte pointer to text
AC1 = Byte count
AC2 = SSA

.WTX
Normal Return
```

Normal Return: Text message is queued for output.

USER INTERFACE (Continued)

USER INTERFACE EXAMPLE

The following is an explanation of how the modules shown in Figure 5-1 are used to move data to and from HAMLET.

The user must provide a controlling task for input (input controller) and/or a controlling task for output (output controller). Entry points for the tasks must be .HAMI (input) and .HAMO (output). HAMLET starts these tasks at program initialization.

The input controller can terminate at any time, however the output controller must remain active for the duration of the program. HAMLET passes all "requests-to-send" to the output controller, and expects a response from the user.

These controlling tasks may create zero or more program or user tasks that run as input or output streams. Note that the controlling tasks do not act as input or output streams - they only initiate them and effect communications with HAMLET.

Input Controller

The user's input controller starts streams by passing command text to HAMLET's command interpreter. All commands that can be entered via the ECLIPSE workstation's console device can also be entered by the user's input controller.

In the example, a call to .HIFI is made to enter a command to transfer a user print file (UPF.SR) to the line printer ($LPT). The user's input control task then terminates. (Alternatively, the user's control task could remain active as long as required, and continue to pass commands of any type).

The user input task thus started runs independently from the input controller. The input tasks will exist only for the duration of the file transfer.

Input Controller (Continued)

Assume that in the first occurrence of the command to transfer UPF.SR to $LPT, UPF.SR has just been created and is empty. At this point the input and output will start and then immediately stop because of an end-of-file condition. The case of actual data transfer from UPF.SR to $LPT is described later in this example.

Output Controller

The user's output control task is also started by HAMLET's initializer. The task loads the address of the task communications word (.UCTW), and then waits for a message from HAMLET. When a "permission-to-send" is received by HAMLET, it determines that the user's output controller exists, and passes the request to the controller (see .UTCW description).

In the example a user output task is started (it is assumed that only "Print" stream-types will be received by HAMLET). The controller sets a flag (RFLG) that will then leave the output functions to HAMLET. That is, the output controller will return a code of 2 each time it is subsequently called.

The function of the user output task is to prefix the printer forms control byte to the data portion in the buffer (the printer forms control byte is placed in each data buffer queued by HAMLET to this task at buffer location B.SRB). The length of the record (data plus forms control byte) is also prefixed to the record and then the entire record is output to file UPF.SR via the system call .WRS. This data transfer continues until end-of-file is detected, whereupon the task terminates.

Initially, the user output task calls .KMOI to perform output stream initialization functions. HAMLET's task starter has already obtained a "stream slot" for this task in which the task's status can be maintained. The format of the "stream slot" is shown in the MLP.SR module (an extract of this module is included in Figure 5-1). The address of this slot is passed in AC2.

Output Controller (Continued)

.KMOI acquires an I/O channel and stores it at stream slot location S.CHN. .KMOI then checks to see if an output file has been provided through the entry of a previous "A" (ASSIGN) command, or requests that the operator assign a file dynamically (through the "O" command). Let us assume that this is output stream 1, and that an "A" (ASSIGN) command has not been issued. .KMOI would then issue the following console message

```
    O1.ASSIGN P1 FILE?
```

To which the operator might respond:

```
    O1ΔUPF.SR)
```

which assigns UPF.SR to print-stream-1.

If initialization is normal, the user's output task regains control at statement "GNXT". If an error occurs, .KMOI stops the task, frees the resources, and print-stream-1 is not output at this time. Another "request-to-send" will have to be received before this task, or any other task, can output print-stream-1.

At GNXT, a call to .KMOD is made to dequeue an output queue reserved by HAMLET for this task. .KMOD waits, if necessary, for buffers to be placed on this queue, and then returns to the caller. If an end-of-file is found on the queue, the exception return is taken, which results here in a call to .KMOE to end the user's output task.

Data Buffer

Each buffer returned is comprised of a header and a data portion. The header is separated from the data portion by the buffer's link word (see Figure 5-2 for buffer offsets). The header is never altered by output tasks, and only infrequently by input tasks.

As buffers are dequeued, they are acted on and output to file UPF.SR until end-of-file is encountered, at which time the file is closed, and the stream-slot and other resources are returned to HAMLET either by normal termination (.KMOE) or abnormal termination (.KMOA).

USER INTERFACE (Continued)

Data Buffer (Continued)

At this time, assume the operator halts the program. The next time HAMLET is loaded, the user's input task starts as before, but this time there is data in UPF.SR. A call is made to .KMII. .KMII has functions similar to .KMOI except that it initializes input tasks. HAMLET starts all tasks with the stream slot address in AC2.

File Data Transfer

The user must then inform HAMLET about the code (ASCII or EBCDIC) of the data as it exists in the file because HAMLET moves the code field from the status word (S.STA) in the stream slot to all buffer headers for use by other HAMLET routines.

Normally this is not necessary since the file's code is defined by the stream-type except for stream-type "U".

Next, the input task calls .KMIB which gets a device buffer from the buffer pool created by HAMLET. There is no exception return from this call. By convention, whenever an exception return cannot be used, a call to .FATL is entered which, if ever used, indicates a serious problem for the program. .FATL issues the message "FATAL CALL FROM XXX" (where XXX represents the location from which the call was made.)

After getting a buffer, a 1 byte read to UPF.SR is made. This byte indicates the length of the data record output by the user's output task. Next, the data record is read and the printer forms byte (SRCB) is moved to the header. This buffer is then given to HAMLET via the input queue interface routine .KMIQ. HAMLET functions will subsequently examine this queue and determine that a local transfer to the systems print routine is to be made. The process continues until end-of-input-file is reached whereupon the input task terminates via a call to .KMIE.

Print Streams

These routines were written to illustrate the user's interface and do not represent the normal method of outputting print streams.

Print Streams (Continued)

Print streams are normally directed to the $LPT or $LPT1, or to a disk or tape file for subsequent local transfer. Note that when a local transfer of a print job is to be made, a command of the form:

```
    X  SPRINT/S   $LPT/S
```

is entered where SPRINT is a file previously output to by the program print task. The file must now be input as an ASCII sequential stream and output as an ASCII sequential stream because of the "line-feeds" included with the data to simulate vertical forms control.

```
0001 USER  MACRO REV 03.00            09:56:21 03/20/75
                        .TITL   USER

                ; FUNCTION:
                ; --------

                ; SAMPLE USER INTERFACE TO HAMLET

                ; CALL:
                ; -----

                ;       BY PROGRAM INIT SINCE .HAMI AND .HAMO WILL
                ;       HAVE REAL ADDRESSES.

                ; RETURN:
                ;--------

                ;       USER INPUT INTERFACE & USER OUTPUT INTERFACE ARE
                ;       TASKS

                        .ENT    .HAMI, .HAMO, .USRI, .USRO

                        .EXTN   .UTCW, .HIFI, .KMII, .FATL
                        .EXTN   .KMIB, .KMIQ, .KMOI, .KMOD
                        .EXTN   .KMOF, .EROR, .KMOA, .WTC
                        .EXTN   .KMOE, .KMIE, .KMIA
                        .EXTN   .KILL, .REC, .XMTW


                        .NREL
        000001          .TXTM   1
```

Figure 5-1. User Interface Program Example

```
10002 USER
                ;       INPUT CONTROLLER TASK

00000'030455 .HAMI:  LDA     2,USRI          ;USER INP STREAM SLOT
00001'050016         STA     2,USP
00002'020472         LDA     0,BP            ;CMND BP
00003'024504         LDA     1,BCNT          ;BYTE COUNT
00004'030450         LDA     2,TASKA         ;ADDRESS OF TASK TO START
00005'077777         .HIFI                   ;CALL CMND DISPATCHER
00006'000504         JMP     WTC             ;ERROR RETURN
00007'077777 RNTR:   .KILL                   ;INP CONTROLLER CAN BE KILLED


                ;       OUTPUT CONTROLLER TASK

00010'030425 .HAMO:  LDA     2,USRO          ;USER OUTP STREAM SLOT
00011'050016         STA     2,USP
00012'020422 AWAIT:  LDA     0,UTCW          ;TCW
00013'077777         .REC                    ;AWAIT MSG FROM CNTRL
00014'135000         MOV     1,3             ;STRUCTURE POINTER IN AC1
00015'024473         LDA     1,C2            ;SET CODE = 2 (RUN SYS TSK)
00016'030473         LDA     2,RFLG
00017'101015         MOV#    0,0,SNR
00020'000404         JMP     RUNSY           ;NO USER OUTPT TASK
00021'125400         INC     1,1
00022'030411         LDA     2,UTSKA
00023'051401         STA     2,1,3           ;TASK ADDRESS

00024'045400 RUNSY:  STA     1,0,3
00025'165000         MOV     3,1             ;RESTORE FOR XMT
00026'077777         .XMTW
00027'077777         .FATL
00030'102400         SUB     0,0
00031'040460         STA     0,RFLG          ;RUN SYSTEM TASKS HENCEFORTH
00032'000760         JMP     AWAIT

00033'000206'UTSKA:  .UTSK
00034'077777 UTCW:   .UTCW
00035'000036'USRO:   .USRO
00036'000016 .USRO:  .BLK    S.PSL
00054'000115'TASKA:  UIPT
00055'000056'USRI:   .USRI
00056'000016 .USRI:  .BLK    S.PSL
00074'000172"BP:     .+1*2
00075'054040         .TXT    *X UPF.SR/U $LPT/P<15>*
      052520
      043056
      051522
      027525
      020044
      046120
      052057
      050015
      000000
00107'000014 BCNT:   12.
00110'000002 C2:     2
00111'000001 RFLG:   1

00112'030723 WTC:    LDA     2,USRO          ;SSA
00113'077777         .WTC                    ;CODE RETURNED IN AC0
00114'000673         JMP     RNTR            ;RETURN
```

Figure 5-1. User Interface Program Example (Continued)

```
0003 USER
                        ;       INPUT TASK

   00115'000116'UIPT:   .+1
   00116'077777         .KMII                          ;INIT INPUT TASK (AC2 = SSA)
   00117'000027'        .FATL                          ;LOCAL TASK
   00120'021014         LDA     0,S.STA,2              ;THE USER MUST INFORM HAMLET
   00121'024464         LDA     1,EBCDC                ;OF THE CODE OF THE INPUT FILE.
   00122'104410         IOR     0,1                    ;EITHER EBCDIC OR ASCII.
   00123'045014         STA     1,S.STA,2              ;SET CODE = EBCDIC (DEFLT=ASCII)
   00124'077777 GBUF:   .KMIB                          ;GET A DEVICE BUFFER
   00125'000117'        .FATL                          ;NOT USED
   00126'041001         STA     0,S.S0,2               ;SAVE INPUT BP
   00127'055005         STA     3,S.TMP,2              ;SAVE BUFFER ADDRESS
   00130'031026         LDA     2,S.CHN,2
   00131'126520         SUBZL   1,1                    ;SET BYTE COUNT = 1
   00132'006017         .SYSTM
   00133'015077         .RDS    77
   00134'000426         JMP     RDERR                  ;CHK READ ERR
   00135'106710         LDB     0,1                    ;GET COUNT FOR DATA I/O
   00136'006017         .SYSTM
   00137'015077         .RDS    77                     ;READ SRCB + DATA
   00140'000422         JMP     RDERR                  ;CHK FOR ERROR
   00141'030016 SLAST:  LDA     2,USP
   00142'035005         LDA     3,S.TMP,2              ;RESTORE PNTR
   00143'104110         SBI     1,1                    ;DECRM BYTE COUNT
   00144'045775         STA     1,B.BC,3               ;STRIP CR, OR LF ETC
   00145'106710         LDB     0,1                    ;GET SRCB
   00146'045772         STA     1,B.SRB,3              ;SAVE IN HEADER
   00147'011774         ISZ     B.BP,3                 ;BUMP DATA BP
   00150'165000         MOV     3,1                    ;CALL SETUP
   00151'077777         .KMIQ                          ;QUEUE BUFFER FOR COMPRESSION
   00152'000125'        .FATL                          ;NOT USED
   00153'020431         LDA     0,EFLG                 ;CHK EOF SET
   00154'101015         MOV#    0,0,SNR
   00155'000747         JMP     GBUF                   ;NO, CONTINUE
   00156'102400         SUB     0,0
   00157'040425         STA     0,EFLG                 ;RESET END FLAG
   00160'041005         STA     0,S.TMP,2              ;SHOW NO BUFFER
   00161'000410         JMP     SEOF                   ;REPORT EOF

   00162'020421 RDERR:  LDA     0,EOFE
   00163'112414         SUB#    0,2,SZR                ;END OF FILE?
   00164'000410         JMP     IOERR                  ;NO, REPORT ERROR, SND EOF
   00165'125015         MOV#    1,1,SNR                ;ANY DATA ?
   00166'000403         JMP     SEOF                   ;NO
   00167'010415         ISZ     EFLG                   ;YES, SET END FLAG
   00170'000751         JMP     SLAST                  ;AND SEND LAST DATA

   00171'030016 SEOF:   LDA     2,USP
   00172'025005         LDA     1,S.TMP,2              ;BUFR POINTER
   00173'077777         .KMIE                          ;END THIS TASK

   00174'141000 IOERR:  MOV     2,0                    ;SYS ERROR CODE
   00175'030016         LDA     2,USP
   00176'077777         .EROR
   00177'100056         ER.IO+1B0                      ;ERR CODE + DSP (AC0)
   00200'025005         LDA     1,S.TMP,2
   00201'077777         .KMIA                          ;ABNORMAL END

   00202'000120 C80:    80.                            ;80. BYTE READ

0004 USER
   00203'000006 EOFE:   EREOF
   00204'000000 EFLG:   0
   00205'000001 EBCDC:  SS.EB
```

Figure 5-1. User Interface Program Example (Continued)

```
10005 USER
                  ; USER OUTPUT TASK TO EXTRACT FORMS CONTROL BYTE
                  ; WHICH IS STORED IN BUFFER HEADER AT B.SRB FOR PRINT
                  ; JOBS. THE BUFFER'S TEXT IS APPENDED
                  ; TO THIS BYTE & THE WHOLE RECORD WRITTEN AS A BINARY
                  ; SEQUENTIAL BLOCK. SUBSEQUENTLY, A USER INPUT TASK
                  ; (UIPT) WILL READ THIS DATA AND PASS IT TO THE
                  ; PRINTER TASK. THIS FACILITATES CHECKING OF VERTICAL FORMS
                  ; SIMULATION BECAUSE NOW ONLY A LOCAL TRANSFER IS REQUIRED TO GE
                  ; A HASP PRINT CONTROL RECORD  PLUS DATA.

                  ;       OUTPUT TASK

   00206'020451 .UTSK:  LDA     0,INHIB          ;DEVICE MASK (AC2 = SSA)
   00207'077777         .KMOI                    ;INIT DEVICE
   00210'000152'        .FATL                    ;NA

   00211'077777 GNXT:   .KMOD           ;DQ NXT REC: AC0=BP,AC1=BC,AC2=SSA,
                                                 ; AC3=BUFF ADDR, CARRY=CODE
   00212'077777         .KMOE                    ;END OF FILE RETURN
   00213'055006         STA     3,S.TM2,2        ;SAVE BUFF PNTR
   00214'125400         INC     1,1              ;BUMP COUNT
   00215'045002         STA     1,S.S1,2         ;SAVE BC
   00216'100110         SBI     1,0              ;DECREMENT BP BY 1
   00217'025772         LDA     1,B.SRB,3
   00220'107010         STB     0,1              ;STORE THE FORMS CONTROL BYTE
   00221'100110         SBI     1,0              ;DEC BP AGAIN
   00222'025002         LDA     1,S.S1,2         ;GET DATA COUNT
   00223'107010         STB     0,1              ;STORE THE LENGTH
   00224'125400         INC     1,1              ;ADJUST TOTAL I/O COUNT
   00225'045002         STA     1,S.S1,2         ;AND RETAIN
   00226'041001         STA     0,S.S0,2         ;SAVE BP

   00227'031026 RWRIT:  LDA     2,S.CHN,2        ;GET CHANNEL
   00230'006017         .SYSTM
   00231'016477         .WRS    77               ;WRITE BIN SEQ REC
   00232'000406         JMP     EROR             ;ERR RETURN
   00233'030016         LDA     2,USP            ;RESTORE SSA
   00234'025006         LDA     1,S.TM2,2        ;RESTORE BUFFER ADDR
   00235'077777         .KMOF
   00236'000210'        .FATL
   00237'000752         JMP     GNXT             ;FREE BUFFER AND GET NXT DATA

                  ; ERROR WHILE TRYING TO WRITE

   00240'024416 EROR:   LDA     1,.ERTO          ;TIME OUT
   00241'146404         SUB     2,1,SZR          ;EQUAL?
   00242'000405         JMP     ENDIT            ;NO,UNRECOVERABLE
   00243'030016         LDA     2,USP
   00244'021001         LDA     0,S.S0,2
   00245'025002         LDA     1,S.S1,2         ;RESTORE COUNT
   00246'000761         JMP     RWRIT            ;RETRY ON TIMEOUT

   00247'141000 ENDIT:  MOV     2,0
   00250'030016         LDA     2,USP
   00251'000176'        .EROR                    ;WRITE
   00252'100056         ER.IO+1B0
   00253'030016         LDA     2,USP
   00254'025006         LDA     1,S.TM2,2        ;GET BUFFER ADDRESS
   00255'077777         .KMOA                    ;KILL THE TASK
 0006 USER

   00256'000101 .ERTO:  ERDTO                    ;DEVICE TIMEOUT CODE
   00257'000000 INHIB:  0



                        .END
```

Figure 5-1. User Interface Program Example (Continued)

```
; STREAM SLOT PARAMETERS
; ------ ----- ----------

S.TCW = 0                   ;TASK COMMUNICATION WORD
S.S0 = S.TCW+1              ;SAVE WORD 0
S.S1 = S.S0+1               ;SAVE WORD 1
S.S2 = S.S1+1               ;SAVE WORD 2
S.S3 = S.S2+1               ;SAVE WORD 3
S.TMP = S.S3+1              ;TEMP WORD
S.TM2 = S.TMP+1             ;TEMP 2 WORD
S.LNK = S.TM2+1             ;LINK WORD
S.DR1 = S.LNK+1             ;SAVE 1 FOR DELAY
S.DR3 = S.DR1+1             ;SAVE 3 FOR DELAY
S.CNB = S.DR3+1             ;CURRNT # BUFFERS
S.TNB = S.CNB+1             ;TOTAL NUMBR BUFFERS ALLOWED
S.STA = S.TNB+1             ;STATUS OF SS (SEE SS. PARAMETERS)
S.QA = S.STA+1              ;Q HEAD ADDRESS (OUTPUT)
S.TSK = S.QA+1              ;TASK START ADDRESS
S.RCB = S.TSK+1             ;RCB
S.NRB = S.RCB+1             ;NOVA RCB
S.FCS = S.NRB+1             ;FCS
S.STN = S.FCS+1             ;STREAM NUMBER
S.TID = S.STN+1             ;TASK ID (FIXED)
S.PTO = S.TID+1             ;PR TIME OUT VALUE (INPUT)
S.SLP = S.PTO+1             ;OTHER SLOT ADDR (LOCAL XFER)
S.CHN = S.SLP+1             ;CHANNEL # ASSIGNED
S.BSQ = S.CHN+1             ;...BOQ FOR SPOOL (OUTPUT)
S.KIL = S.BSQ+1             ;KILL FLAG FOR INPUT/OUTPUT
S.NB = S.KIL+1              ;NEXT RESERVD SPOOL BLOCK
S.BSY = S.NB+1              ;BUSY INDICATOR
S.INB = S.BSY+1             ;DEVICE INHIBIT MASK
S.SUS = S.INB+1             ;SUSPEND TASK FLAG
S.NIO = S.SUS+1             ;# I/O RECORDS
S.FRM = S.NIO+1             ;FORMS ID
S.FNM = S.FRM+1             ;FILENAME  (8 WRDS)

; LOCATIONS BEGINNING AT S.FNM ARE REFDEFINED FOR
; FORMS CONTROL. THEY ARE USED AFTER THE FILE HAS
; BEEN OPENED

S.FAD = S.FNM               ;FORMS TABLE ADDR
S.CFL = S.FAD+1             ;CURRENT FORMS LINE
S.LPP = S.CFL+1             ;LINES PER PAGE THIS FORM
S.NLF = S.LPP+1             ;# LF TO WRITE NXT TIME
S.FIN = S.NLF+1             ;RUNNING INDEX
S.NFF = S.FIN+1             ;# FF

S.SIZ = S.FNM+8.            ;STREAM SLOT SIZE
S.PSL = S.QA+1              ;PSEUDO STREAM SIZE
S.QOF = S.QA                ;OFFSET TO Q HEADER
```

Figure 5-2. Buffer Offsets

```
; STREAM SLOT STATUS
; ------ ---- ------

; S.STA = -1 THEN SLOT IS FREE

SS.AS = 0B15                ;ASCII CODE
SS.EB = 1B15                ;EBCDIC CODE
SS.PG = 0B1                 ;PERMISSION GRANTED
SS.PR = 1B1                 ;PERMISSION REQUESTED
SS.WT = 1B0                 ;STREAM TASK WAITING
SS.GO = 0B0                 ;STREAM TASK RUNNING
SS.SI = 0B7                 ;SPOOL INACTIVE
SS.SA = 1B7                 ;SPOOL ACTIVE
SS.WS = 1B8                 ;WAB SET
SS.WR = 0B8                 ;WAB RESET
SS.OP = 1B14                ;FILE OPEN
SS.CL = 0B14                ;FILE CLOSED
SS.AA = 1B13                ;AWAITING ASSIGNMENT

; BUFFER POOL HEADER PARAMETERS
; ------ ---- ------ ----------

H.TCW = 0                   ;TASK COMM WORD
H.CHN = H.TCW+1             ;POOL CHAIN HEAD
H.BOQ = H.CHN+1             ;BEGINNING WAIT LIST
H.EOQ = H.BOQ+1             ;END WAIT LIST

; I/O BUFFER PARAMETERS
; --- ------ ----------

B.LNK = 0
B.TYP = B.LNK-1             ;BUFFER HEADER POINTER
B.SLT = B.LNK-2             ;STREAM SLOT POINTER (REAL OR PSEUDO)
B.BC = B.LNK-3              ;BYTE COUNT OF DATA
B.BP = B.LNK-4              ;DATA BYTE POINTER
B.STS = B.LNK-5             ;I/O BUFFER STATUS
B.SRB = B.LNK-6             ;SRCB - = X'80'--> STANDARD
B.FCS = B.LNK-7             ;CREATOR'S FCS MASK
B.RCB = B.LNK-7             ;RECVD RCB (NO CONFLICT)

B.CMP = B.LNK+1             ;CMPRSS START
B.DTA = B.LNK+5             ;DATA START
B.OFF = B.LNK+8.            ;POS LENGTH OF HEADER
B.OHD = B.OFF+B.DTA         ;TOTAL BUFFER OVERHEAD

; I/O BUFFER STATUS
; --- ------ ------

BB.DA = 0B0                 ;DATA RECORD
BB.CL = 1B0                 ;CONTROL RECORD
BB.EF = 1B0+1B1             ;END OF FILE (1B1)
BB.AS = 0B15                ;ASCII CODE
BB.EB = 1B15                ;EBCDIC CODE
BB.LD = 1B14                ;LOGGED FLAG
```

Figure 5-2. Buffer Offsets (Continued)

```
; HAMLET ERROR AND STATUS CODES

NM.CN = 1                   ;NORMAL - DONT DISPLAY
                            ;RSVD
                            ;RSVD
NM.EF = NM.CN+3             ;ENDED
                            ;RSVD
NM.OR = NM.EF+2             ;(STREAM) STARTED
NM.IR = NM.OR               ;(STREAM) STARTED
NM.SR = NM.OR+1             ;SIGNON RECEIVED


CD.BC = 15
ER.BC = CD.BC               ;BAD CMND
ER.BS = CD.BC+1             ;BAD SYNTAX
ER.DA = ER.BS+1             ;DUPLICATE STREAM ASSIGNMENT
ER.NM = ER.DA+1             ;NO MORE SLOTS
ER.FC = ER.NM+1             ;TASK ERROR, FUNCTION CANCELED
ER.BC = ER.FC+1             ;BAD CARD CODE
ER.LE = ER.BC+1             ;LINE ERROR
ER.SU = ER.LE+1             ;STREAM UNKNOWN
ER.NR = ER.SU+1             ;PERMIS NOT REQUSTD (BUT GIVN)
ER.CU = ER.NR+1             ;CONTROL RECORD UNDEFINED
ER.UC = ER.CU+1             ;USER RETURN CODE ERROR
ER.PE = ER.UC+1             ;PERMISSION REQST ERROR - UNKNOWN STREAM TYPE
ER.UG = ER.PE+1             ;UNKNOWN GENERAL CONTROL RECORD
ER.UT = ER.UG+1             ;USER CLOCK INIT ERROR
ER.CM = ER.UT+1             ;CMS I/O ERROR
ER.AF = ER.CM+1             ;ALL STATIC SLOTS ASSIGNED
ER.BE = ER.AF+1             ;BLOCK COUNT ERROR
ER.OS = ER.BE+1             ;PERMANENTLY PENDED OUTPUT
                            ;RSVD
                            ;RSVD
ER.RC = ER.OS+3             ;RESTRICTED COMMAND
ER.TE = ER.RC+1             ;TASK START ERROR
ER.SP = ER.TE+1             ;INSUFF. SPACE FOR BUFFERS
ER.SL = ER.SP+2             ;UNKNOWN RCB RECEIVED
ER.PK = ER.SL+1             ;INSUFFICIENT ROOM IN BUFFER FRO DECOMPRESS
ER.SC = ER.PK+1             ;OPERATOR CANCEL
ER.SE = ER.SC+1             ;SPOOLING I/O ERROR
ER.BD = ER.SE+1             ;BID ERROR(RETRY COUNT EXASHTED)
ER.AE = ER.BD+1             ;ABNORMAL STREAM END
ER.IF = ER.AE+1             ;BAD FORMS ID - USING STANDARD
ER.FF = ER.IF+1             ;FORMS ERROR, CHNL NOT FOUND
ER.SO = ER.FF+1             ;SIGNOFF RECVD
ER.IO = ER.SO+1             ;I/O ERROR (LAST CODE)
```

Figure 5-2. Buffer Offsets (Continued)

APPENDIX A

CONSOLE MESSAGES

GENERAL

Depending on the option chosen during CGEN, a message will either appear as a numeric code or as message text. In the following discussion, the messages appear in sequence by numeric code. In cases where message text is displayed in place of the numeric code, a cross-reference is included on the last page of this appendix. The messages on this page appear in alphabetic order and point to the appropriate numeric code.

MESSAGES

CODE	TEXT
4	ENDED An input or output stream has terminated.
6	STARTED An input or output stream has started.
7	SIGNON RECEIVED A Sign-On record has been received from an ECLIPSE operating in slave mode. The remote number specified in the Sign-On record is not examined.
15	BAD COMMAND An unknown command was entered. It is ignored.
16	BAD SYNTAX A valid command was entered, but one or more of its parameters were invalid. It is ignored.

CONSOLE MESSAGES (Continued)

MESSAGES (Continued)	CODE	TEXT
	17	DUPLICATE STREAM ASSIGNMENT There are two possible causes for the message. 1. While entering an "X" command, the operator specified a stream number that is already running. HAMLET ignores the command. 2. A "request-to-send" has been received and it references a stream already running. This condition should not occur, and indicates a bad transmission or program error. The program continues and ignores the request-to-send.
	20	NO MORE STREAM SLOTS An attempt to start a stream has failed because the number of streams to be input or output exceeds the number of stream slots available. The request is ignored.
	21	TASK ERROR, FUNCTION CANCELED There are two causes: 1. An "X" command always causes an input task to be started to read the input file referenced in the command. It may optionally start an output task to output data to the output file referenced. If either task cannot be started then this error occurs. Normally, it means too many tasks are running concurrently (insufficient number of tasks defined). 2. A "request-to-send" has been received and an attempt to start a task to output the data that will be received has failed for the reason given above. The request is ignored.

CONSOLE MESSAGES (Continued)

MESSAGES (Continued)	CODE	TEXT
	22	BAD CARD CODE Card read error. The stream is terminated.
	23	LINE ERROR Repeated attempts to correct a communications problem with the remote computer have failed. The program terminates.
	24	STREAM UNKNOWN An "O" command has specified a stream number that is not waiting for a file assignment. The command is ignored.
	25	PERMISSION NOT REQUESTED A communications message has been received granting "permission-to-send" that was never requested. Either a bad transmission was received or there is a program error. The message is ignored.
	26	UNDEFINED CONTROL RECORD A communications message has been received and it contains an undefined control message. Either a bad transmission has been received, or there is a program error. The message is ignored.
	27	USER RETURN ERROR The user's output interface has returned a code that is not defined. The code is ignored and the program continues as though the message was never received.

CONSOLE MESSAGES (Continued)

MESSAGES (Continued)	CODE	TEXT
	30	PERMISSION REQUEST ERROR A "request-to-send" has been received but the stream-type-id specified is unknown. Either a bad transmission has been received or there is a program error. The program continues and ignores the error.
	31	UNDEFINED GENERAL CONTROL RECORD A communications message has been received that contains an undefined General Control Record. Either a bad transmission has been received, or there is a program error. The program ignores the error and continues.
	32	USER CLOCK ERROR An attempt to initiate the operating system user's clock routine has failed. The program terminates after the message has been written. This error should only occur if there is no clock in the system.
	33	COMMUNICATIONS I/O ERROR Either an illegal device code was given during CGEN, or the device code is in use. The program terminates after this message is displayed.
	34	ALL STATIC ASSIGN SLOTS IN USE An "A" command has specified a pre-assignment that cannot be made because the table containing these assignments is full. One or more of the entries must be removed. The command is ignored.

CONSOLE MESSAGES (Continued)

MESSAGES (Continued)	CODE	TEXT
	35	BLOCK COUNT ERROR In HAMLET-HAMLET mode, a communications message has been received indicating that a transmission block sequence error has occurred. All transmission blocks are numbered sequentially, and an expected block number was not received. The program continues but the data being received may be incomplete.
	36	PERMANENTLY PENDED OUTPUT An output stream cannot continue so it is permanently pended. This is required since an output stream cannot send a "KILL" command to the remote indicating that it is unable to proceed. Hence, the output stream is effectively frozen until the program is reloaded. Resources held by the stream are not released. A permanent "wait-a-bit" is set.
	41	RESTRICTED COMMAND A start-up or run-time only command has been entered when not permitted. The command is ignored.
	42	TASK START ERROR At program initialization, a required task could not be started. The program terminates. Check the program generation for an inadequate number of tasks specification.

CONSOLE MESSAGES (Continued)

MESSAGES (Continued)	CODE	TEXT
	43	INSUFFICIENT SPACE FOR BUFFERS At program initialization, a buffer pool is created from the free memory available. If the number and size of the buffers requested during CGEN is too large for memory, this error results and the program terminates. Regenerate the program and reduce the number and/or size of the buffers requested.
	45	UNKNOWN RCB RECEIVED Either a transmission error has been detected or there is a program error. The program continues, but results are unspecified for all streams currently being received.
	46	BUFFER TOO SMALL A device buffer is too small for the amount of data received. The record is truncated and written as received. Either a program error has occurred, or a transmission error, or the peripheral device buffers specified during CGEN are too small. The program continues.
	47	OPERATOR CANCEL The operator has cancelled the output stream. The permission request is ignored.
	50	SPOOLING I/O ERROR An attempt to use HAMLET's spooling function has resulted in an I/O error. Either the file to which spooling is directed (MLT.SP) was left opened previously, or it is undefined, or not randomly organized. Spooling is not used, and the program continues.

CONSOLE MESSAGES (Continued)

MESSAGES (Continued)	CODE	TEXT
	51	BID ERROR Attempts to establish a communications link have failed. The program will keep trying to make the connection, but the operator can also break the modem connection, and re-dial the remote system, perhaps obtaining a better circuit.
	52	ABNORMAL STREAM END An error has caused this stream to terminate and give up its resources.
	53	BAD FORMS ID - USING STANDARD A vertical forms id given in an "A" or "O" command cannot be found in the Forms table. The system standard will be used instead.
	54	FORMS ERROR, CHANNEL NOT FOUND A "skip-to-channel" command cannot be executed because the specified channel does not exist in the forms table. Two form feeds are executed and the stream continues.
	55	SIGNOFF RECEIVED In HAMLET-to-HAMLET mode, a Sign-Off record has been received from the remote HAMLET. No more communications with the remote system is possible, but local operations can continue.
	56	I/O ERROR A call to the operating system to perform I/O has resulted in an error condition. The code returned by the operating system is displayed as xx in "CODE=xx". The affected stream is terminated, and the program continues.

CONSOLE MESSAGES (Continued)

ALPHABETIC CONSOLE MESSAGES CROSS-REFERENCE TABLE

The following table can be used as a cross-reference for systems that use message text in place of message codes.

Message	Code
Abnormal Stream End	52
All Static Assign Slots in Use	34
Bad Card Code	22
Bad Command	15
Bad Forms ID - Using Standard	53
Bad Syntax	16
Bid Error	51
Block Count Error	35
Buffer Too Small	46
Communications I/O Error	33
Duplicate Stream Assignment	17
Ended	4
Forms Error, Channel Not Found	54
Insufficient Space for Buffers	43
I/O Error	56
Line Error	23
No More Stream Slots	20
Operator Cancel	47
Permanently Pended Output	36
Permission Not Requested	25
Permission Request Error	30
Restricted Command	41

CONSOLE MESSAGES (Continued)

ALPHABETIC CONSOLE MESSAGES CROSS-REFERENCE TABLE (Continued)

Message	Code
Signoff Received	55
Signon Received	7
Spooling I/O Error	50
Started	6
Stream Unknown	24
Task Error, Function Cancelled	21
Task Start Error	42
Undefined Control Record	26
Undefined General Control Record	31
Unknown RCB Received	45
User Clock Error	32
User Return Error	27

APPENDIX B

VERTICAL FORMS CONTROL

GENERAL

HAMLET provides selectable vertical tabbing capability. This is accomplished by allowing the user to create one or more vertical forms control tables to be loaded with the HAMLET program. The particular vertical forms control table to be used is indicated during run-time by the static assignment command (A) or Dynamic assignment command (O).

HAMLET simulates the operation of a carriage control tape for all print streams based on the associated vertical forms control table. When HAMLET receives a carriage control command in the record SRCB of the form "skip to channel X", HAMLET will insert line feed or form feed characters in the record until it encounters a line in the forms control table with channel X set.

The module, CTAB.RB, defines a standard default vertical forms control table. When no form-id is specified in an assign command, this form is used. As provided, the default form has form-id of F1 for a 63 line form, with line one having channel one set. This effectively supplies one form feed whenever a "skip to channel one" is detected.

CREATING A TABLE

The HAMLET package contains a file (CTAB.SR), from which the default table in CTAB.RB was created. If the user wants another default table or additional tables, CTAB.SR is edited to create these tables.

Two macros (CCTIT and CCTAB) are also provided in the CTAB.SR module to permit the RDOS/MRDOS user to generate vertical forms control tables. For each table to be generated, both of these macros must be used.

VERTICAL FORMS CONTROL (Continued)

CREATING A TABLE (Continued)

The first macro, CCTIT, has the format:

```
    CCTIT <form-id> <form length>
```

The <form-id> is a one or two character name used to choose a form with the A or O commands. The <form length> given in decimal, is the number of lines in the form.

An example of the CCTIT macro is:

```
    CCTIT FAΔ63
```

to create a table with the form-id FA for a form 63 lines long.

The CCTAB macro is used to set one or more channels on a particular line of a form. The format is:

```
    CCTAB<line number><channel number>...<channel number>
```

The <line number> may range from 1 to the number specified in <form length>. One or more (up to 12) <channel number> values may be included which range from 1 to 12.

All arguments are in decimal radix. For example,

```
    CCTAB  10  1  3  4  12
```

means that for line 10, set channels 1, 3, 4 and 12.

The CCTAB macro must always be called in ascending line number. The following sequence would create an erronous table:

```
    CCTAB  10   1   2
    CCTAB   3   1   2
```

since the line 3 definition occurs after the line 10 definition. The following would be correct

```
    CCTAB   3   1   2
    CCTAB  10   1   2
```

VERTICAL FORMS CONTROL (Continued)

CREATING A TABLE (Continued)

The end of the form must be noted by a macro call of CCTAB without any arguments, i.e.,

```
    CCTAB
```

A complete specification of a form would look like:

```
    CCTIT  FA  63
    CCTAB   6  10
    CCTAB  10   1   5
    CCTAB  20   1   6
    CCTAB  50   4   3
    CCTAB
```

As many different forms as desired can be specified using this format. However, the first table in the module becomes the default table and all other tables must have unique <form-ids>.

The file CTAB.SR must then be assembled by the user with the macro assembler to create CTAB.RB.

The CLI command used is as follows:

```
    MAC  CTAB.SR  $LPT
```

to generate CTAB.RB with a listing on the line printer. The resulting file CTAB.RB will be used by CGEN during the linking of HAMLET.

DataGeneral

PROGRAMMING DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title		Date
Company Name			
Address (No. & Street)	City	State	Zip Code

Form No. 10-24-004

FOLD DOWN FIRST FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

BUSINESS REPLY MAIL

No Postage Necessary If Mailed In The United States

Postage will be paid by:

Data General Corporation

Southboro, Massachusetts 01772

ATTENTION: Programming Documentation

FOLD UP SECOND FOLD UP

STAPLE