Roach - IPC

To: Distribution

From: Robert S. Coren

Date: 03/29/78

Subject: High-speed Terminal Input

INTRODUCTION

This MTB discusses the introduction of a feature in MCS to allow interactive asynchronous terminals to transmit input at maximum channel speed. The principal purpose for this feature is to provide support for screen terminals that have the capability of transmitting part or all of a screen in response to a single keystroke. (The feature may also be useful for terminals such as the TermiNet 1200 that may have a cassette attachment.) It is assumed that most terminals equipped with this feature precede and follow the data thus transmitted with a recognizable pair of characters (such as STX and ETX); every terminal that we know about that has a "transmit" key does so.

Full-screen input has been casually referred to as "block mode input" or "block transfer"; henceforth, however, we will use the term "frame input" to describe this feature, in order to reserve the term "block mode" for matters relating to the use of ipc_$block.

In order to provide frame input, we have had to solve most of the internal problems associated with user-settable break characters. The user interface for this more general feature has yet to be specified; accordingly it will not be in the initial implementation, nor will it be addressed directly in this document. The plan is to provide frame input in MR6.5, and possibly to provide general user-settable break characters in MR7.0.

BREAK_CHARACTERS

The concept of a break character on Multics exists at several levels. In the FNP, a break character causes an interrupt from the channel (this interrupt is simulated by software for LSLA channels) and indicates that an interrupt is to be sent to the central system to inform it that input is on its way. In dn355, the ring-0 interrupt handler, the presence of a break character in input data indicates that a wakeup is to be sent to the process using the channel if that process previously attempted to read when there was no input available. In tty_read, the ring 0 call-side routine, a break character is used to delimit the piece of input to be translated, converted, and returned to the user ring. Finally, tty_get_line, after calling hcs_$tty_read, does not generally return to its caller until it has read a break character out of ring 0.

For asynchronous line types, the only break character recognized by current software at all these various levels is the newline character. (The FNP also interrupts on formfeed so that output can be restarted on channels using page-length checking, but these formfeeds are intercepted by dn355 and do not cause wakeups.) In order to implement frame input, it must be made possible to recognize additional break characters at all levels in a consistent fashion.

Other problems exist in the present implementation. Since every newline typed causes the FNP to interrupt the central system, and (on asynchronous HSLA channels) every newline typed causes the FNP itself to be interrupted, a user who hits the "transmit" key of his/her 1200-baud terminal when the screen contains a series of successive newlines generates an FNP interrupt approximately every 8 milliseconds, and the FNP cannot keep up. This can result in loss of connection, and occasionally results in FNP crashes.

We cannot prevent the user from generating these interrupts, but there are two things we can do: we can modify the HSLA software to detect that the FNP is falling behind, and avoid damage to the system (or lasting damage to the channel's connection); and we can provide the user with a way to send a full screen without generating excessive interrupts.

SUMMARY OF PROPOSED NEW FEATURE

For the purposes of this discussion, a frame is defined as a block of arbitrary input, starting with a pre-specified "frame-begin" character and ending with a pre-specified "frame-end" character, which may be transmitted at channel speed. A new tty_ mode, framei (for "frame input"), specifies that the user expects to generate frame input from the terminal; a new order, "set_framing_chars", specifies the frame-begin and frame-end characters (which need not be different from each other). If the user is in framei mode and has set (nonnull) framing characters, any occurrence of the frame-begin character is assumed to mark the beginning of a frame; the frame-end character becomes the only break character, so no further input interrupts are generated until a frame-end character appears. The user's process can either read the entire frame by calling iox_$get_chars, or read the input line by line (as at present) by calling iox_$get_line, as explained below.

USER INTERFACE

Specification of Framing Characters

As indicated above, the framing characters are specified by the set_framing_chars order, which causes them to be stored both in ring 0 and in the FNP. In addition, we propose to add a "framing_chars" keyword to the terminal type specification in the TTF so that reasonable defaults can be supplied for terminal types (such as the Delta Data 4000) whose framing characters are known. The framing characters associated with the terminal type will be included in the structure used by the set_terminal_data order used by tty_ and the answering service to initialize a terminal when setting its type. To allow frame input to work in rawi mode, the framing characters must always be specified in the terminal's code.

If no framing characters have been specified, either in the TTT or by explicit order, then frame input cannot be recognized and may not be accepted. In addition, frame input is only recognized if framei mode is on as well.

Use of iox_$get_line and iox_$get_chars

The get_line entry of the tty_ I/O module will continue to work the way it does today, i.e., it will not return to its caller until a newline character is read out of ring 0 (unless the caller's buffer fills up, in which case the contents of the buffer are returned along with a status code of error_table_$long_record). A change, however, is being proposed in the meaning of the get_chars entry. In the present implementation, this entry does not return to its caller until the caller's buffer is full. There is no known software that currently uses this entry, and it seems useful to make get_chars cognizant of the concept of break characters.

Essentially, we propose that get_chars work in analogous fashion to get_line, except that it uses any currently-defined break character as its delimiter. Thus in the normal default case, where newline is the only break character, it would behave exactly the same as get_line; in framei mode, it would return an entire frame if frame input was present.

The table below summarizes the behavior of iox_$get_chars under the proposed scheme. N is the character position of the first break character in the (converted) input; B is the size in characters of the caller's buffer.

no break present	break present, N <= B	break present, N > B
call ipc_$block	return N characters	return B characters and error_table_$long_record

Blocking for Input

As a future extension, we propose a mechanism that will make it possible for a user to call iox_$get_chars and be certain of not going blocked in tty_ (without having to use the read_status order). A new mode, blocki ("block for input"), would specify that tty_ should call ipc_$block when it has no input to return. For compatibility with current implementations, the normal default would be for blocki mode to be on. The effect of turning blocki mode off would be to cause tty_get_chars and tty_get_line to return 0 characters in the circumstances in which they currently call ipc_$block. It would then be up to the caller to

decide when or if to go blocked for input.

It should be noted that blocki mode is different from all modes defined so far for tty_ in that it would be known in the user ring only, and would never be passed on to ring 0. This would require some modifications to the way modes are set and reported.

RING_0_INTERFACE

The only existing entry into ring 0 to read terminal input is hcs_$tty_read. If a newline is present in ring 0, this entry returns all characters up to and including that newline (assuming that the caller's buffer is big enough to hold them); if no newline is present, however, it returns whatever input is present (again, up to the length of the caller's buffer). Furthermore, it ignores newlines in rawi mode, and returns whatever is available.

We propose to retain the tty_read entry with its present behavior for purposes of compatibility, while adding two new entries: hcs_$tty_get_line and hcs_$tty_get_chars. These two entries are intended to be called by the corresponding entries in tty_, and to behave the way the tty_ entries do in "blocki mode. In other words, hcs_$tty_get_line returns 0 characters if there is no newline in ring 0, even in rawi mode; otherwise it returns all characters up to the first (non-escaped) newline. Similarly, hcs_$tty_get_chars returns 0 characters if there is no break character (however defined) in ring 0; otherwise it returns all characters up to the first break character. Both entries return error_table_$long_record if the appropriate break character is present but the caller's buffer is too small to include it.

It is in hcs_$tty_get_chars that the Multics side of frame input is implemented. Specifically, whenever a frame-begin character is seen, the normal break character (newline) is ignored, and the frame-end character is used as the sole break character. When a frame is completed, the normal break character is reinstated until another frame begins.

Several things should be noted about this scheme. It eliminates the necessity for a kludge that presently exists in the get_line entry of tty_, whereby it checks the input provided by hcs_$tty_read to see if it ends in ASCII newline unless the terminal is in rawi mode; this allows APL to read EBCDIC input in

rawi mode. Under the proposed scheme, tty_ can trust any input returned to it by hcs_$tty_get_line to end in newline, since otherwise hcs_$tty_get_line would have returned 0 characters. The second point is that, in order for the scheme to work in rawi mode, break characters must be known in ring 0 in the terminal's (untranslated) code. This is already true of newline (assuming the terminal's code for newline based on line type) because it is used to determine how much input to translate and convert at once, and the specification of the set_framing_chars order makes it true for the frame-end character as well.

The retention of hcs_$tty_read serves the purposes of special-purpose I/O modules (such as bisync_) that wish to do all interpretation of input in the user ring. By calling hcs_$tty_read in rawi mode, they can use ring 0 as a completely transparent data pipe. Finally, the answering service (i.e., astty_) can be changed at a later date to call hcs_$tty_get_line rather than hcs_$tty_read, since it is really interested in complete lines, and currently believes that that is what ring 0 is sending it (which is usually true).

It should be pointed out that under present FNP implementation it is very rare that input not ending in a break character finds its way into Multics memory, so that the current ring 0 software usually does what is wanted. We are not prepared, however, to guarantee that such input will never appear, and in fact we are considering modifications, described later in this MTB, that will increase the frequency with which such "partial" input appears in ring 0.

FNP CONSIDERATIONS

Excessive Interrupts

The issue of generating excessive HSLA interrupts by inputting a screen full of newlines has been mentioned above. By using framei mode, a careful user can avoid this problem. To cover the case of a careless or malicious user who hits the transmit key without entering framei mode and specifying framing chars, the HSLA software will be modified to detect that it is not keeping up with the channel, and drop receive mode on the channel right away. When this happens, a few BEL characters are sent to the terminal to warn the user that the input is not being received properly. Receive mode is restored if the user hits QUIT; if this has not happened within 10 seconds, a QUIT is signalled automatically, and receive mode is restored.

Another possible source of trouble is frame input on a channel in echoplex mode. The nature of the FNP implementation of frame recognition is such that when a frame-begin character appears in framei mode, echoplex is automatically disabled for the duration of the frame. In the case of a user who, as above, neglects (inadvertently or otherwise) to leave echoplex or enter framei mode before hitting the transmit key, the software can be modified to recognize excessive echoplex interrupts (which are distinguishable from break-character interrupts) and take the channel out of echoplex mode.

FNP Buffer Space

There is a rather crude mechanism in the current FNP software to prevent a runaway channel, or a foolish or clever user, from devouring all the FNP buffer space by filling the FNP with input. This is done by taking the channel out of receive mode, as described above, whenever it has more than a certain number, known as the "exhaust limit", of input buffers allocated. The present limit is rather too small for most known screen sizes; in any case, it is not clear that it is safe to allow a channel to have the entire contents of some arbitrary terminal's memory in FNP memory all at once.

We propose to get around this problem by greatly increasing the exhaust limit, and establishing a smaller limit called the "pre-exhaust" limit. When the pre-exhaust limit is reached, the FNP interrupts the central system and sends it the accumulated input, while continuing to accept further input from the channel. This will allow large frames to be input without choking the FNP. It will also result in arbitrary "partial" input being sent to Multics memory. As indicated above, we intend to modify the central system software so as not to process this input until it is "complete."

DRAFT DOCUMENTATION

The remainder of the MTB consists of draft documentation describing the new features of the tty_ I/O module and the TTF specification.

Get_chars_Operation

The get_chars operation returns characters only if a break character has been sent from the terminal, in which case it returns all characters up to and including the first break character. The break character is normally newline unless the channel is in framei mode and the frame-begin character has been input, in which case the break character is the frame-end character (see the descriptions of framei mode and the set_framing_chars order, below). If the caller's buffer is too small to contain all the characters up to and including the break character, as many characters as can fit in the buffer are returned, along with a status code of error_table_$long_record.

****

(to be added under "Global Orders")

set_framing_chars
specifies the pair of characters that the terminal generates surrounding input transmitted as a block or "frame". These characters must be specified in the character code used by the terminal. This order must be used for framei mode (see below) to be effective. The info_ptr must point to a structure with the following format:

```
dcl 1 framing_chars aligned,
    2 frame_begin char (1) unaligned,
    2 frame_end char (1) unaligned;
```

get_framing_chars
causes the framing characters currently in use to be returned (see the set_framing_chars order, above). If no framing characters have been supplied, NUL characters are returned. The info_ptr must point to a structure like the one described for the set_framing_chars order; this structure is filled in as a result of the call.

****

(to be added under "Modes Operation")

framei, ^framei
specifies that the user's terminal is capable of transmitting a block or "frame" of input all at once in response to a single keystroke. The system may not handle such input correctly unless framei mode is on and the set_framing_chars order has been issued.

----
set_tty

(additional control arguments)

-frame framing_chars, -fr framing_chars
changes the framing characters used in framei mode to those specified by framing_chars. The framing_chars argument is a 2-character string consisting of the frame-begin character and the frame-end character, in that order. These characters must be specified in the character code of the terminal. They may be entered as octal escapes if necessary. If the frame-begin character is specified as a blank or as a NUL character, the frame-begin character is not changed; if the frame-end character is omitted, specified as a blank, or specified as a NUL character, the frame-end character is not changed. The framing characters have no effect unless framei mode is on. It is an error to set one of the framing characters to NUL unless the other one is also NUL.

-print_frame, -pr_fr
prints the framing characters for the terminal.

(Insert under Notes between steps 5 and 6)

If the -frame control argument is specified, set the framing characters.

(Insert under Notes between steps 8 and 9)

If the print_frame control argument is specified, print the framing characters on the terminal.

(add under Examples)

The command line:

set_tty -frame \002\003

sets the frame-begin and frame-end characters to the ASCII STX and ETX characters respectively.

(to be added to MAM System, Section 6, description of TTF syntax)

framing_chars: <frame_begin> <frame_end> ;

The framing_chars statement is optional. If present, it specifies the framing characters generated by the terminal when sending frame input at channel speed. <frame_begin> and <frame_end> are octal numbers of up to 3 digits representing the frame-begin and frame-characters, respectively, in the terminal's character code (i.e., without translation). These characters are recognized in framei mode only.