Network Working Group
Request for Comments: 190
NIC: 7135
P. Deutsch
Xerox Research Center, Palo Alto, CA
13 July 1971

DEC PDP-10 -- IMLAC COMMUNICATION SYSTEM

This report describes an operational system for communicating textual display information between a main-site computer and a remote-display processor.

The main site machine is a DEC PDP-10 with the BBN paging hardware (henceforth TENEX).

The remote machine is a IMLAC PDS-1 (henceforth IMLAC).

Section (1) briefly describes the IMLAC hardware configurations.

Section (II) describes the display facilities presented to the user.

Section (III) describes the system calls (JSYS calls) implemented in the TENEX monitor to provide these facilities.

Section (IV) describes the formats of the messages used for communication between TENEX and the IMLAC.

Section (V) analyzes the division of responsibility between the two machines.

(1) Hardware configurations.

The standard IMLAC is a 16-bit minicomputer with 4K of 2 microsecond core, a cycle-stealing display, an input keyboard, and an asynchronous serial communication interface.

The display is normally programmed to draw characters using very short vectors.

The display comes in two major configurations, depending on the presence or absence of hardware for drawing long vectors.

In the sequel, specifications depending on the configuration will be flagged LVH or non-LVH respectively.

The I/O system normally does not provide for interrupts when characters arrive from the serial interface.

However, the IMLAC is barely able to keep up with the PDP-10 without this feature, so we were able to persuade the manufacturer to implement it.

While no special hardware is required for the software described here, the character interrupt and the SRI-ARC "mouse" and "keyset" are highly recommended, and the software is oriented towards their use.

(II) Facilities.

Each display console in the system may be in "display mode" or "teletype simulation mode".

In display mode, the information displayed consists of text strings at arbitrary positions on the display face.

In teletype simulation mode, the display shows the last 20-30 lines which would appear on a teletype listing.

A given console may switch between these modes, under program control, without losing any information.

Regardless of mode, each display has a cursor string which follows the position of the available pointing device,

The implemented system uses a "mouse" generally, but one console uses a tablet.

The cursor string may be set by a program, for example to indicate at what sort of object the user is expected to point.

In display mode, the screen of a given console is allocated to users in rectangular blocks called "display areas".

This allows users to communicate via a single display split into multiple areas.

When a user (program) requests a display area, he specifies how many text strings he will want to display in it.

Each string has its own X-Y location, character size, font (italic, underline, ...), and an arbitrary number of characters.

Each of these components is settable without disturbing the others.

Each string may be manipulated without affecting the others.

(III) JSYS calls.

(ADA) Assign a display area.

JSYS ada: [440B] allocate a display area

Accepts:

r1:

upper-left-y-cord[10], upper-left-x-cord[10], max-no-string[11]

r2:

lower-right-y-cord[10], lower-right-x-cord[10], default-c-size[2], default-n-inc[6], default-font[5]

Returns:

+1: Unsuccessful

r1: error code

+2: Successful

r1: da-id[18]

Function:

This jsys allocates a display area given the coordinates of the diagonal, the maximum number of strings to be displayed, and the default setting for the character size, font, and horizontal increment. An 18-bit da-id is returned which should subsequently be used to refer to this display area.

(DDA) Deallocate a display area.

JSYS dda; [441B] deallocate a display area

Accepts:

r1: da-id

Returns:

+1: Unsuccessful

r1: error code

+2: Successful

Function:

This jsys deallocates a display area given the associated da-id.

(STRDA) STRing display: add, delete, or change.

JSYS strda; [442B] Manipulate (move, write, replace, delete) a string in a display area

Accepts:

r1: string-id[18], da-id[18]

r2: first byte pointer or 0 or -1

r2: second byte pointer or 0

r4: y-cord[10], x-cord[10], font[6], c-size[3], h-inc[7]

Returns:

+1: Unsuccessful

r1: error code

+2: Successful

r1: string-id[18]

Function:

This jsys writes a new string, replaces, deletes, or moves (optionally replacing) an extant string within a display area. In addition, the font, character size, and horizontal increment may be specified for the string.

The string may be specified by two byte pointers or by one byte pointer with the string terminating with a zero character.

If first byte pointer is zero then

if a new string is being written then

an error code illstr is returned.

Otherwise, the string already exists so

delete the string from the display area.

If the first byte pointer is -1 then

if a new string is being written then

an error code illstr is returned.

Otherwise, use the old string.

If the first character of the string is a zero character, the string to be displayed is null, but the string is not deleted.

The coordinates (optional unless the string is new or being moved) are relative to the upper leftcorner of the display area.

If the jsys is to effect an extant string, a zero coordinate means use the old value.

For the font, c-size, and h-inc fields a field of all one's indicates that the display area default value (set in the ada jsys) is to be used. A 0 means use the value which was previously used for the (extant) string.

If the string is new, then an 18 bit string identifier is returned.

(SCSR) Set the Cursor StRing.

JSYS scsr; [450B] display a string (vectors later) as the cursor

Accepts:

r1: first byte pointer or 0 or -1

r2: second byte pointer or 0

r3: font[5], c-size[2], h-inc[6]

No defaults allowed

Returns:

+1: Unsuccessful

r1: error code

+2: Successful

Function:

This jsys is used to set the cursor string. Later, a set of vectors will be allowed also. If the string length is zero or the first byte pointer is 0, nothing will be displayed for the cursor. If the first byte pointer is -1 then the old string will be used. If a cursor did not previously exist, an illcon error return will be executed.

(SDDA) Suppress the display of an area.

JSYS sdda; [444B] suppress all display in a display area

Accepts:

r1: da-id[18]

r2: 1 or 0

Returns:

+1: Unsuccessful

r1: error code

+2: Successful

Function:

The display image is removed from the display area but is not destroyed if r2 = 0.

(RDDA) Restore the display of an area.

JSYS rdda; [446B] restore all display in a display area

Accepts:

r1: da-id[18]

Returns:

+1: Unsuccessful

r1: error code

+2: Successful

Function:

The display image is restored in the display area.

(SDDA) Suppress the display of a String.

JSYS ssda [445B]; suppress display of a string in a display area

Accepts:

r1: string-id[18], da-id[18]

Returns:

+1: Unsuccessful

r1: error code

+2: Successful

Function:

The display image for the given string is suppressed in the display area

(RDSA) Restore the display of a String.

JSYS rsda; [447B] restore display of a string in a display area

Accepts:

r1: string-id[18], da-id[18]

Returns:

+1: Unsuccessful

r1: error code

+2: Successful

Function:

The display image for the given string is restored in the display area.

(TSNDA) Teletype Simulation oN.

JSYS tsnda; [451B] turn tty simlaution on

Accepts:

none

Returns:

+1: Always

r1: 1 if was in work station mode, 0 otherwise

Function:

Restores the tty simulation display area, and suppresses all others (except the cursor). Turns wsmode (work station mode flag) off for this console and returns previous value of wsmode.

(TSFDA) Teletype Simulation ofF.

JSYS tsfda; [452B] turn tty simulation off

Accepts:

none

Returns:

+1: Always

Function:

Suppresses the display of the tty simulation display area, restores the display of all other display areas, and sets wsmode on.

(RSTDA) Reset display areas.

JSYS rstda; [453B] Reset display areas

Accepts:

none

Returns:

+1: Always

Functions:

Deallocates and removes images from all display areas associated with this console except the tty simulation and cursor, the display of which is restored.

(IV) Message formats.

Messages are sequences of 8-bit characters, of which 7 contain useful information.

The higer-order (200B) bit should contain even parity on IMLAC input and is set to even parity on IMLAC output.

If the IMLAC receives an odd parity character, it halts at present.

In the remainder of this document, the parity bit will not be discussed.

A message may be either a character or a command.

Single-character messages from the PDP-10 to the IMLAC represent program output intended for the teletype.

Commands from the PDP-10 represent display information.

Commands from the IMLAC represent characters or other input information.

Every command is prefixed by an internal escape character (code 33B) and a character count.

The escape character will henceforth be referred to as ESC: is has the same code as the ASCII escape character 33B.

Messages sent from TENEX to IMLAC:

Characters 40B-177B are directed to the teletype simulation area.

Character 12B (line feed) starts a new line in the teletype simulation area.

An ESC indicates that display or control information is coming, as follows.

Every message beginning with ESC contains the number of following characters as its second character.

Certain constructs appear in several command messages.

(da) A display area identifier is a pair of characters containing 12 bits of information:

1st: bits (0:5) + 40B

2nd: bits (6:11) + 40B

(NSTRS) A string count is a single character between 0 and 177B.

(STRID) A string identifier is a single character between 1 and 177B.

(RETAIN) The retention flag, if non-zero, specifies that an existing string should be retained rather than overwritten.

(CSIZE) A character size is a single character between 0 and 3:

The character sizes are respectively x1/2, x1, x2, x3.

(HINC) A horizontal increment is a single character.

In the present implementation, HINC is ignored and a standard spacing is supplied as follows.

LVH:

0: 3 units

1: 6 units

2: 12 units

3: 18 units

non-LVH:

0: 4.5 units

1: 9 units

2: 18 units

3: 27 units

(FONTS) A font specification is a single character.

In the present implementation, the font is stored but does not affect the display.

(outxy) An output X-Y coordinate pair is encoded in four characters as follows:

1st: X,bits(0:5) + 40B

2nd: X,bits(6:11) + 40B

3rd: Y,bits(0:5) + 40B

4th: Y,bits(6:11) + 40B

(inxy) An input X-Y coordinate pair is encoded in four characters as follows:

LVH: see (outxy) above.

non-LVH: as above, except that each 12-bit coordinate is actually of the form 1400B+40B*[v/9]+(v MOD 9), where the actual coordinate is 0<=v<=719.

This means there are actually fewer points on each axis.

(string) A string is just the requisite number of characters.

Control characters will be displayed as a distinctive blot.

Each display operation has a corresponding message.

01B - ADA (assign display area)

Followed by (da) NSTRS CSIZE HINC FONT.

02B - DDA (delete display area)

Followed by (da).

04B - STRDA (string display)

Followed by (da) STRID RETAIN (xy) FORMAT [CSIZE] [HINC] [FONT] (string).

Format specifies whether each of CSIZE, HINC, and FONT is to come from the display area default, the current value for the string, or the message.

The bits are: 0 0 STF STI STC RDF RDI RDC.

RDF=1 means read the FONT from the message.

RDF=0, STF=1 means use the old value from the string.

RDF=0, STF=0 means use the display area default.

The pairs RDI-STI and RDC-STC specify HINC and CSIZE in the same way.

05B - SCSR (set cursor string)

Followed by RETAIN CSIZE HINC FONT (string).

06B - SDDA (suppress display of da)

Followed by (da) KILL.

KILL#0 means delete all strings in this display area.

KILL=0 means retain the strings.

07B - RDDA (restore display of da)

Followed by (da).

10B - SDDA (suppress display of string)

Followed by (da) STRID KILL.

KILL#0 means delete the string.

KILL=0 means retain the string.

11B - RSLA (restore display of string)

Followed by (da) STRID.

12B - TSNDA (turn teletype simulation on)

13B - TSFDA (turn teletype simulation off)

14B - Long input mode

Puts the IMLAC into the mode where it sends coordinate information in a message with every character.

This is the normal operating mode for the IMLAC.

15B - Short input mode

Puts the IMLAC into the mode where it outputs characters literally, just like a teletype.

The IMLAC starts out in this mode when turned on.

A string of 10 ESC characters, followed by a non-ESC, indicates an emergency - the IMLAC reinitializes itself and goes into short input and teletype simulation modes.

All other (control) characters are ignored.

Messages sent from IMLAC to TENEX:

Short input mode:

Every character typed on the keyboard is transmitted literally.

Long input mode:

Every message begins with ESC and a count of subsequent characters.

Codes 40B-177B represent keyboard input.

Note that the IMLAC does not echo these characters on the display.

Codes 00B-37B, except ESC, represent typed-in control characters.

The present implementation allows the user to generate all of these codes from the keyboard.

ESC may be followed by a keyset-mouse code or a control character.

Code 00B represents an ESC typed on the keyboard.

Otherwise, a code 40B-77B and a code 100B-107B follow.

This type of message is sent whenever the mouse buttons change or a character has been typed on the keyset, and the IMLAC cannot convert this to an ordinary character.

The IMLAC converts recognizable keyset chords and mouse changes to characters; see SRI-ARC documentation for a full discussion of this hardware.

The codes 40B-77B represent accumulated keyset chords.

40B means no complete chord has been struck.

The codes 100B-107B represent the state of the mouse buttons after a change: a 1-bit corresponds to a depressed button.

Other codes should not appear.

All codes are followed by the (inxy) coordinates of the mouse.

This means 7 or 8 characters are sent for each character typed.

(V) Division of responsibility

The first criterion in design of the system just described was to allocate sufficient validity checking to the PDP-10 to make it unnecessary for the IMLAC to send a response for each command.

Thus, the PDP-10 allocates and checks display area identifiers and string numbers.

In the present implementation, display areas are numbered system-wide whereas strings are numbered from 1 to N within a display area.

The only errors not detectable by the PDP-10 are transmission parity errors and overflow of the IMLAC's memory.

The former are presently not corrected, but could be handled by any standard technique.

The latter are in principle detectable by the PDP-10, since the amount of space required to store a given display is fairly simply computable.

If the IMLAC runs out of space, it deletes lines from the top of the teletype simulation display, until only three are left, before giving up.

A secondary criterion was to hold down the number of characters required to represent a display command.

We have found two problem areas and two areas in which we expect to expand the IMLAC's capability.

Echoing was relegated to TENEX since we desired to avoid the well-known complications associated with remote echoing.

The question of identifying a device as an IMLAC to TENEX gave us a great deal of trouble.

We settled on the convention of a TENEX Executive command which causes TENEX to send the "Long input mode" message.

The IMLAC starts out in short input mode.

The TENEX character input routines also may be set into either long or short mode.

Their state is changed by the TSNDA and TSFDA system calls.

In short mode, (TSNDA last), short input is passed literally, and only the character is passed from long input.

In long mode (TSFDA last), short input is padded with all-zero coordinates, and long input is passed literally.

This arrangement allows TENEX programs which do not use the special features of the IMLAC to operate correctly with either an IMLAC or a teletype.

The user may restore the IMLAC to short mode with another command or with a special key on the IMLAC keyboard.

Code is ready to allow the IMLAC to collect entire literal strings with some internal editing before sending them to TENEX.

We have not resolved the disposition of characters typed by the user between the typed command initiating literal input and the receipt by the IMLAC of the "collect literal" message.

With 8K of core, the IMLAC can handle a significant fraction of the command parsing and feedback functions of the SRI-ARC On-Line System (NLS), for which this effort is principally intended.

[This RFC was put into machine readable form for entry]
[into the online RFC archives by Lorrie Shiota, 10/01]