NAME
console — bitmap display and keyboard

DESCRIPTION
PARTIAL ANSI EMULATOR
The console of the LISA running UniPlus+ is the built-in bitmap display and keyboard. To be used as the console terminal it has some features of a Digital VT100 terminal and includes certain additional features. The termcap entry is vtm. The size of the screen is 38 lines by 90 columns. The following sequences apply:

- `<esc>M` reverse scroll
- `<esc>H` set horizontal tab stop at cursor position

The optional parameter `<n1>` in the following is a numeric value given by a string of zero or more decimal characters in ASCII representation. A value of one (1) is assumed for missing or zero-value arguments.

- `<esc>l<n1>A` move cursor up `<n1>` rows, or to the top if `<n1>` is > current row
- `<esc>l<n1>B` move cursor down `<n1>` rows, or to the bottom if current row + `<n1>` is > maximum row
- `<esc>l<n1>C` move cursor right `<n1>` columns, or to the right margin if current column + `<n1>` is > maximum column
- `<esc>l<n1>D` move cursor left `<n1>` columns, or to the left margin if `<n1>` is < current column
- `<esc>l<n1>L` insert `<n1>` line(s)
- `<esc>l<n1>M` delete `<n1>` line(s)
- `<esc>l<n1>P` delete `<n1>` character(s)

The optional parameter `<n1>` in the following is a numeric value given by a string of zero or more decimal characters in ASCII representation. A value of zero (0) is assumed for missing arguments.

- `<esc>l<n1>J` clear screen
  if `<n1>` is 0, clear from cursor to end of display
  if `<n1>` is 1, clear from beginning of display to cursor
  if `<n1>` is 2, clear entire screen
- `<esc>l<n1>K` clear line
  if `<n1>` is 0, clear from cursor to end of line
  if `<n1>` is 1, clear from beginning of line to cursor
  if `<n1>` is 2, clear entire line
- `<esc>l<n1>g` clear tab stops

October 1984 - 1 -
CONSOLE (5L)

\[ <\text{esc}>|<n1>|<n2> \]
if \(<n1>\) is 0, clear tab stop at the cursor position
if \(<n1>\) is 3, clear all tab stops
set/reset reverse video or underlining all characters written to the screen after this are written according to the new settings
if \(<n1>\) is 0, set normal video and no underlining
if \(<n1>\) is 1, set reverse video
if \(<n1>\) is 4, set underlining
if \(<n1>\) is 7, set reverse video
Note that more than one attribute can be specified at a time;
\(<\text{esc}>|<n1>|<n2>\) is identical to \(<\text{esc}>|<n1>|<\text{esc}>|<n2>\). A maximum of 10 such numeric parameters is allowed.

The optional parameter(s) \(<n1>, <n2>\) are numeric values given by a string of zero or more decimal characters in ASCII representation. A value of zero (0) is assumed for missing arguments. A value of zero (0) is assumed for the second argument if the semicolon (;) is missing.

\[ <\text{esc}>|<n1>|<n2>|H \]
move cursor to row \(<n1>\), column \(<n2>\) or the appropriate edge of the screen if \(<n1>\) or \(<n2>\) is out of bounds. Row and column values are 0 origin (i.e., \(0 \leq \text{value} < \text{maximum}\))

\[ <\text{esc}>|<n1>|<n2>|f \]
move cursor to row \(<n1>\), column \(<n2>\) or the appropriate edge of the screen if \(<n1>\) or \(<n2>\) is out of bounds. Row and column values are 0 origin (i.e., \(0 \leq \text{value} < \text{maximum}\))

KEYBOARD
Some keys on the Lisa console keyboard require explanation. This table shows how UNIX uses these keys.

<table>
<thead>
<tr>
<th>Key</th>
<th>Use in UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>control</td>
</tr>
<tr>
<td>clear</td>
<td>delete</td>
</tr>
<tr>
<td>left option</td>
<td>escape</td>
</tr>
<tr>
<td>right option</td>
<td>escape</td>
</tr>
<tr>
<td>enter</td>
<td>line feed</td>
</tr>
<tr>
<td>—</td>
<td>move cursor one character left</td>
</tr>
<tr>
<td>→</td>
<td>move cursor one character right</td>
</tr>
<tr>
<td>↑</td>
<td>move cursor one character up</td>
</tr>
<tr>
<td>↓</td>
<td>move cursor one character down</td>
</tr>
</tbody>
</table>

October 1984
The eighteen keypad keys require further explanation. The key labelled `clear` is always the delete key. The arrow keys always cause cursor motion in the appropriate direction, by transmitting the correct escape sequence. Normally a shifted arrow key causes the corresponding character (+ * / or ,) to be transmitted. The `enter` key causes a newline to be transmitted. The remaining twelve (12) keys correspond to these characters (1 2 3 4 5 6 7 8 9 0 - and .).

<table>
<thead>
<tr>
<th>Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;esc&gt;</code></td>
<td>enter alternate keypad mode</td>
</tr>
<tr>
<td><code>&lt;esc&gt;</code></td>
<td>exit alternate keypad mode</td>
</tr>
</tbody>
</table>

In alternate keypad mode the seventeen (17) keypad keys (all except the clear key which is used for delete) cause a special code to be transmitted, according to the following table:

<table>
<thead>
<tr>
<th>Key</th>
<th>Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><code>&lt;esc&gt;</code>O0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><code>&lt;esc&gt;</code>O1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><code>&lt;esc&gt;</code>O2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>&lt;esc&gt;</code>O3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><code>&lt;esc&gt;</code>O4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><code>&lt;esc&gt;</code>O5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>&lt;esc&gt;</code>O6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><code>&lt;esc&gt;</code>O7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><code>&lt;esc&gt;</code>O8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td><code>&lt;esc&gt;</code>O9</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td><code>&lt;esc&gt;</code>Om</td>
<td></td>
</tr>
<tr>
<td>,</td>
<td><code>&lt;esc&gt;</code>Ol</td>
<td>(shifted down arrow)</td>
</tr>
<tr>
<td><code>enter</code></td>
<td><code>&lt;esc&gt;</code>OM</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td><code>&lt;esc&gt;</code>OP</td>
<td>(shifted left arrow, function key 1)</td>
</tr>
<tr>
<td>*</td>
<td><code>&lt;esc&gt;</code>OQ</td>
<td>(shifted right arrow, function key 2)</td>
</tr>
<tr>
<td>/</td>
<td><code>&lt;esc&gt;</code>OR</td>
<td>(shifted up arrow, function key 3)</td>
</tr>
</tbody>
</table>

If the codes are echoed back to the screen the last character of the sequence will be displayed.

**I/O CONTROLS**

The console driver for the LISA provides the following I/O controls to allow the user to modify parameters on the system. The following values are defined in `/usr/include/sys/al_ioctl.h`:

To set parameter values:

- `ioctl (fp, AL_SBVOL, value)` set bell volume (0-7)
- `ioctl (fp, AL_SBPITCH, value)` set bell pitch (0-8191)
- `ioctl (fp, AL_SBTIME, value)` set bell time (in clock tics, limited to 10 seconds)
- `ioctl (fp, AL_SDIMTIME, value)` set time before dimming screen (in clock tics)
ioctl (fp, AL_SDIMCONT, value) set screen contrast when dim (0-255)
ioctl (fp, AL_SDIMRATE, value) set rate of dimming (in clock tics)
ioctl (fp, AL_SCONTRAST, value) set screen contrast when bright (0-255)
ioctl (fp, AL_SREPWAIT, value) set wait before repeating keys (in clock tics)
ioctl (fp, AL_SREPDELAY, value) set delay between key repeat (in clock tics)

To get parameter values:
ioctl (fp, AL_GBVOL, value) get bell volume
ioctl (fp, AL_GBPITCH, value) get bell pitch
ioctl (fp, ALGBTIME, value) get bell time
ioctl (fp, AL_GDIMTIME, value) get time before dimming screen
ioctl (fp, AL_GDIMCONT, value) get screen contrast when dim
ioctl (fp, AL_GDIMRATE, value) get rate of dimming
ioctl (fp, AL_GCONTRAST, value) get screen contrast when bright
ioctl (fp, AL_GREPWAIT, value) get wait before repeating keys
ioctl (fp, AL_GREPDELAY, value) get delay between key repeat

Other parameters:
ioctl (fp, AL_REVVIDEO, f) control the display mode
        if f is greater than zero, screen background is set to bright
        if f is zero, screen background is black
        if f is negative, screen background is complemented
ioctl (fp, AL_GBMADDR, &addr) get physical address of beginning of bitmap display
NAME

c, rc — Corvus hard disk interface

DESCRIPTION

This section describes the special files corresponding to the parallel-port Corvus disk drives.

The LISA has one built-in parallel port and up to six expansion ports. Any combination of the six expansion ports and the built-in port may be dedicated to Corvus disks.

A hard disk is accessed through a block special or character special file. See
mknod(1M) for a description of how to create such a file. The major device number is 2 for block interface or 11 for character (raw) interface. The minor device number specifies the physical unit number and the logical partition through which the disk will be accessed. This number is calculated by:

\[(16 \times \text{physical unit number}) + \text{(logical partition number)}\]

Valid physical unit numbers are 0, 1, 2, 4, 5, 7 and 8. (Units 3 and 6 are skipped for historical reasons.)

<table>
<thead>
<tr>
<th>Unit</th>
<th>Unit\times16</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>&quot;Built-in&quot; port</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>Slot 1 Bottom Port</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>Slot 1 Top Port</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>Slot 2 Bottom Port</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>Slot 2 Top Port</td>
</tr>
<tr>
<td>7</td>
<td>112</td>
<td>Slot 3 Bottom Port</td>
</tr>
<tr>
<td>8</td>
<td>128</td>
<td>Slot 3 Top Port</td>
</tr>
</tbody>
</table>

Logical partitions indicate, by block offset and size, which 512-byte blocks of the disk are to be accessed. The following table shows the available logical partitions:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Name</th>
<th>Block Offset</th>
<th>Size</th>
<th>Intended Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td>4060</td>
<td>32420</td>
<td>root filesystem</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>101</td>
<td>3959</td>
<td>swap area</td>
</tr>
<tr>
<td>2-6</td>
<td>c-g</td>
<td>n/a</td>
<td>n/a</td>
<td>unused</td>
</tr>
<tr>
<td>7</td>
<td>h</td>
<td>101</td>
<td>36379</td>
<td>entire disk</td>
</tr>
</tbody>
</table>

UNIX does not allow access to the first 101 blocks.

The conventional names for the special files are:

- first character of a block device is c (corvus),
- first two characters of a character device are rc (raw corvus),
- next a digit, the physical unit number (0,1,2,4,5,7,8), and
- last a character, the logical partition (a,b,h)
NAME
  eject — eject the Sony microdiskette

SYNOPSIS
  eject [ device ]

DESCRIPTION
  Eject is used to eject the Sony microdiskette. If the Sony is not in use, it
  will be ejected from its drive. If it is open for any reason, the eject will fail.

  A character-special device must be specified. If the major device number is
  6, then the standard character interface for the Sony disk is used. A major
  device number of 7 is used for a special character interface for the Sony
  disk which allows only the eject function. Either device can be specified. If
  none is given explicitly, then /dev/rs0a is used. It is assumed that
  /dev/rs0a is set up with major device number 6 and minor device number
  0.

FILES
  /dev/rs0a          default character-special device
  /dev/eject         character-special file only used for ejecting
NAME
mouse — general mouse interface

DESCRIPTION
This section describes the special file corresponding to the mouse.

The file /dev/mouse is the only interface to the data generated by the
mouse. A request to open the mouse will fail under any of the following
circumstances:

The mouse is not plugged in. (ENODEV)
The mouse is already open. (EBUSY)
The requesting process does not have the console as its controlling ter-

nal. (EPERM)
The open mode includes writing. (EINVAL)
The minor device number is not zero. (ENXIO)

Mouse data is generated and added to the input queue in records. The
record format (as defined in the include file sys/mouse.h) is as follows:

```
struct msrecord {
    ushort   mr_reset;    /* driver reset */
    ushort   mr_but;      /* button down */
    ushort   mr_ctl;      /* ctrl key down */
    ushort   mr_sft;      /* shift key down */
    ushort   mr_time;     /* time generated */
    char     mr_row;      /* row offset */
    char     mr_col;      /* column offset */
};
```

If the mr_reset bit is set, then the mouse driver was recently reset. The
mr_but bit indicates the reason for the reset. A 0 means that a new mouse
was plugged in. A 1 means that the flush time elapsed and all the mouse
records were discarded. None of the other fields are valid for reset records.

The mr_but bit reflects the state of the mouse button (1 means pressed
down, 0 means released.)

The mr_ctl bit reflects the state of the keyboard command (APPLE) key.

The mr_sft bit is 1 if either or both shift keys are down, 0 if both are up.

The mr_time field contains the low order 12 bits of the clock ticks counter
at the time this mouse record was generated. It is useful for determining
the elapsed time between different mouse events.

The mr_row field contains the vertical offset in hundredths of an inch.
Negative means away from the user, positive means towards the user
(down the screen). The range is -128 to +127.

The mr_col field contains the horizontal offset in hundredths of an inch.
Negative means to the left; positive means to the right. The range is -128
to +127.

When the mouse is moving, records are generated at the programmed
interrupt rate (initially 28 milliseconds). In addition, records are generated
whenever the mouse button is pressed down or released, and when the
mouse device is plugged in or unplugged. Depending on certain control
parameters, mouse records may be generated for either or both keyboard
APPLE and SHIFT keys or for synchronization with the screen refresh rate. Read requests for the mouse device will fail if any of the following are true:

The mouse has been unplugged. (ENODEV)

The buffer address is odd or the input byte count is not a multiple of the record size; sizeof(struct msrecord). (EINVAL)

The mouse has been disabled by an ioctl request (see below). (EIO)

Two ioctl(2) system calls apply to the mouse device. AL_GMOUSE fills the following structure in user space with the appropriate data from the mouse driver. AL_SMOUSE “programs” the mouse driver according to the structure in user space. The structure is also defined in the include file sys/mouse.h:

```c
struct msparms {
    ushort mp_fdelay;   /* flush delay */
    ushort mp_irate;    /* interrupt rate */
    ushort mp_qlen;     /* records available */
    ushort mp_flags;    /* control flags */
    ushort mp_otime;    /* oldest records age */
    ushort mp_ytime;    /* youngest records age */
};
```

The `mp_fdelay` field is the number of clock ticks (60ths of a second) before all the mouse data is flushed if not read. If set to zero, then mouse data is never flushed automatically. If set to a value less than 20 ticks the mouse driver buffer is flushed but the flush delay remains unaffected and no reset record is generated. The value of `mp_fdelay` is set to 300 (5 seconds) when the mouse is opened. The maximum value for this field is 4095 and indicates a time of about 68 seconds.

The `mp_irate` field is the rate at which mouse records are generated when the mouse is moving. If zero then the mouse is disabled and no new records will be added to the queue until the interrupt rate is reset. This field is reset to 28 every time the mouse is opened. The range of values is 0 to 28 in increments of 4 milliseconds.

The `mp_qlen` field is the number of records currently available. The driver’s internal buffer can hold up to 128 mouse records. When there are already 128 records available, new mouse information is thrown away.

The `mp_flags` field is composed of the following flags (when the mouse is open, the MF_BUT bit is set to one and the rest are set to zero):

- **MF_VRATE** 00000007: Vertical Retrace Interrupt rate.
- **MF_BLK** 00000010: Block until data is available.
- **MF_SIG** 00000020: Signal (SIGMOUS) when data becomes available.
- **MF_BUT** 00000040: Enable/Disable mouse button events.
- **MF_CTL** 00000100: Enable/Disable command key events.
- **MF_SFT** 00000200: Enable/Disable shift key events.
- **MF_VRT** 00000400: Enable/Disable vertical retrace interrupts.

If MF_BLK is not set, then reads return EOF when no mouse data is present. If set, then the requesting process is suspended until at least one mouse record becomes available. If MF_VRT is also set, then EOF will be returned after the next vertical retrace interrupt.
If MF_SIG is set, then a signal (SIGMOUS) is sent to the process group of the console when the first record is generated.

The MF_BUT flag enables/disables the generation of mouse records from mouse button events. When this bit is set, pressing the mouse button will generate two events: one when the button goes down and one when it comes up. Turning off this bit means that events will not be generated for activity by the mouse button but its state will continue to be reported in mouse records.

The MF_CTL flag enables/disables the generation of mouse records from the command key on the keyboard. This key is considered the second mouse button. Its semantics are similar to that of MF_BUT.

The MF_SFT flag enables/disables the generation of mouse records from either of the SHIFT keys on the keyboard. Either SHIFT key is considered the third mouse button. The semantics are similar to that of MF_BUT.

The MF_VRT flag enables/disables the generation of SIGMOUS interrupts each time the hardware has finished repainting the screen. Synchronizing screen updates with this signal will reduce screen flicker because the program and hardware will not be working on the same area of memory at the same time. It is set by an AL_GMOUSE call. If no records are available, this field is set to 0.

The mp_otime field contains the age in clock ticks of the oldest available mouse record (i.e., the next one to be read). The vertical retrace interrupt rate can be used to slow down the generation of SIGMOUS interrupts due to MF_VRT. It has no meaning if MF_VRT is not set. MF_VRATE is a number between 0 and 7 which specifies how many vertical retrace interrupts will be skipped between those which cause SIGMOUS interrupts.

The mp_ytime field contains the age in clock ticks of the youngest available mouse record (i.e., the most recently generated). It is set by an AL_GMOUSE call. If no records are available, this field is set to 0.
NAME

pm, rpm — Priam datatower disk drive

DESCRIPTION

This section describes the special files corresponding to the Priam disk drives.

The LISA has three expansion slots, any combination of which may be dedicated to Priam devices.

A hard disk is accessed through a block special or character special file. See mknod(1M) for a description of how to create such a file. The major device number is 3 for block interface or 12 for character (raw) interface. The minor device number specifies the physical unit number and logical partition through which the disk will be accessed. It is calculated by:

\[(16 \times \text{physical unit number}) + \text{(logical partition number)}\]

Valid physical unit numbers are 0, 1, and 2.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Unit*16</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Slot 1</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>Slot 2</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>Slot 3</td>
</tr>
</tbody>
</table>

Logical partitions indicate, by block offset and size, which 512-byte blocks of the disk are to be accessed. The following table shows the available logical partitions:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Name</th>
<th>Block Offset</th>
<th>Size</th>
<th>Intended Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td>4101</td>
<td>to end of disk</td>
<td>root filesystem</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>101</td>
<td>4000</td>
<td>swap area</td>
</tr>
<tr>
<td>2-6</td>
<td>c-g</td>
<td>n/a</td>
<td>n/a</td>
<td>unused</td>
</tr>
<tr>
<td>7</td>
<td>h</td>
<td>101</td>
<td>to end of disk</td>
<td>entire disk</td>
</tr>
</tbody>
</table>

UNIX does not allow access to the first 101 blocks.

Drives of different sizes are available for the Priam Datatower. They are currently available with 35 MB, 70 MB, and 86 MB (unformatted). When the disk is first opened, UNIX determines its size and adjusts the sizes of partitions 0 and 7 accordingly. An I/O control, UIOCSIZE, is available to get the size of a disk (see ioctl(2)). This call will get the number of blocks on the disk:

```c
#include "sys/uioctl.h"
int fp, pmsize;
ioctl(fp, UIOCSIZE, (caddr_t)&pmsize);
```

The size of partition 0 is (pmsize - 4101) blocks, and the size of partition 7 is (pmsize - 101) blocks.

As with any other hardware, UNIX does not expect a Priam disk to be replaced while UNIX is running. A disk should not be turned off or
changed without first rebooting the system.

The conventional names for the special files are:

- first two characters of a block device are `pm` (priam),
- first three characters of a character device are `rpm` (raw priam),
- next a digit, the physical unit number (0,1,2), and
- last a character, the logical partition (a,b,h)
NAME

p, rp - profile hard disk interface

DESCRIPTION

This section describes the special files corresponding to the parallel-port profile disk drives.

The LISA has one built-in parallel port and up to six expansion ports. The built-in port has a 10MB hard disk which is accessed through the profile interface. Any combination of the six expansion ports may also be dedicated to profile devices.

A hard disk is accessed through a block special or character special file. See mknod(1M) for a description of how to create such a file. The major device number is 0 for block interface or 5 for character (raw) interface. The minor device number specifies the physical unit number and the logical partition through which the disk will be accessed. This number is calculated by:

\[(16 \times \text{physical unit number}) + (\text{logical partition number})\]

Valid physical unit numbers are 0, 1, 2, 4, 5, 7 and 8. (Units 3 and 6 are skipped for historical reasons.)

<table>
<thead>
<tr>
<th>Unit</th>
<th>Unit*16</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Built-in disk</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>Slot 1 Bottom Port</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>Slot 1 Top Port</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>Slot 2 Bottom Port</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>Slot 2 Top Port</td>
</tr>
<tr>
<td>7</td>
<td>112</td>
<td>Slot 3 Bottom Port</td>
</tr>
<tr>
<td>8</td>
<td>128</td>
<td>Slot 3 Top Port</td>
</tr>
</tbody>
</table>

Logical partitions indicate, by block offset and size, which 512-byte blocks of the disk are to be accessed. The following table shows the available logical partitions:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Name</th>
<th>Block Offset</th>
<th>Size</th>
<th>Intended Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td>2501</td>
<td>16955</td>
<td>root filesystem</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>101</td>
<td>2400</td>
<td>swap area</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>2501</td>
<td>7227</td>
<td>root filesystem</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(on 5MB disk)</td>
</tr>
<tr>
<td>3</td>
<td>d</td>
<td>9728</td>
<td>9728</td>
<td>extra partition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(use with 2)</td>
</tr>
<tr>
<td>4, 5, 6</td>
<td>e, f, g</td>
<td>n/a</td>
<td>n/a</td>
<td>unused</td>
</tr>
<tr>
<td>7</td>
<td>h</td>
<td>101</td>
<td>19355</td>
<td>entire disk</td>
</tr>
</tbody>
</table>

UNIX does not allow access to the first 101 blocks.

Partitions are used to divide disks into filesystems and swap space. The conventional names for the special files are:
- first character of a block device is p (profile),
- first two characters of a character device are rp (raw profile),
- next a digit, the physical unit number (0,1,2,4,5,7,8), and
- last a character, the logical partition (a,b,c,d,h)

EXAMPLE
/dev/p0a  root filesystem on the built-in disk
/dev/p0b  swap space on the built-in disk
/dev/p1h  a large filesystem using all of the disk in the bottom port of
          slot 1
/dev/p0c  small root filesystem on the built-in disk
/dev/p0d  9728-block filesystem on the built-in disk
/dev/p0b  small 2400-block filesystem on the built-in disk
/dev/p4b  swap space on the disk in the bottom port of slot 2
/dev/p4c  7227-block filesystem on the disk in the bottom port of slot 2
NAME
rtc — real time clock

DESCRIPTION
This section describes the special file corresponding to the real time clock. The program `setrtc` is available for reading and setting the real time clock. Each time UNIX is booted, the real time clock is read to set the system time.

The real time clock is accessed through a character special file `/dev/rtc`. See `mknod(1M)` for a description of how to create such a file. `/dev/rtc` has major device 13 and minor device 0. UNIX records time values in seconds since 00:00:00 GMT, January 1, 1970. A read of four bytes from the real time clock returns a long integer with its current setting. A write of four bytes resets the clock. Only reads or writes of four bytes are allowed.

EXAMPLES
The following C program segments describe how to access the real time clock.

```c
long rt;
if ((fp = open("/dev/rtc", oflag)) < 0)
    (void) fprintf(stderr, "setrtc: cannot open /dev/rtc");

if (read(fp, (char *)&rt, sizeof(rt)) < 0)
    (void) fprintf(stderr, "setrtc: cannot read /dev/rtc");

if (write(fp, (char *)&rt, sizeof(rt)) < 0)
    (void) fprintf(stderr, "setrtc: cannot write /dev/rtc");
```
NAME
setparams - tune changeable parameters for console screen, console keyboard, and bell

SYNOPSIS
setparams

DESCRIPTION
The parameters you can adjust are:

- bv ol   bell volume
- bpitch  bell pitch
- btime   bell duration
- dimtime time before screen saver dimming
- dimcont screen contrast when screen is dim
- dimrate rate at which screen goes from bright to dim
- contrast screen contrast when screen is bright
- repwait how long key must be held before it starts repeating
- repdelay delay between key repetitions
- revvideo select/cancel/switch reverse video

To set any of these, type the appropriate name to the question
What is your selection?

For example, type bv ol to set the volume of the bell.

Changing the background color is handled differently from the other choices (see below). For the other parameters the current setting is displayed. If a carriage return alone is typed, the setting remains the same. Otherwise, it is changed to what is entered. The parameter values are retrieved and set using the ioctl calls documented in console(5). For example, the bell volume is retrieved using AL_GBVOL and set using AL_SBVOL. See console(5) for details such as the range of values which can be used for each of the parameters.

Changing the background color is done using "revvideo." No current setting is displayed since this should be obvious. The background color can be either black or white. A black background is referred to as "normal" and a white background as "reverse video." If a carriage return alone is typed the setting remains the same. Otherwise:

+  the screen becomes black characters on a white background,
0  the screen becomes white characters on a black background, or
-  the screen becomes the reverse image of its current state.

The background color is set using the ioctl call AL_REVVIDEO documented in console(5).

To run setparams you must be able to open /dev/console for reading and writing. Normally this means you must be logged in as root or logged in at the console (the bitmap display on the LISA).

Use CONTROL-D to exit setparams.
SETPARAMS (1L)

SEE ALSO
console(5)
NAME
setrtc — read the real time clock, and optionally set it and/or the UNIX system time

SYNOPSIS
setrtc [-s] [mmddhhmm[yy]]

DESCRIPTION
Reads and prints the current time from the real time clock. If the date argument is given, the clock is set first. If the -s flag is used, the system date is set to the time read from the real time clock.

The date is specified exactly as for the date utility.

The first mm is the month number;
dd is the day of the month;
hh is the hour (24 hour system);
the second mm is the minute;
yy is the last two digits of the year and is optional.

The current year is the default if no year is mentioned. Setrtc converts to and from local standard and daylight time.

The real time clock is always accessed through the special file /dev/rtc. The permissions for this file determine which users may read or set it.

EXAMPLES
setrtc
setrtc -s
setrtc 10081700
setrtc 10081700 -s

prints real time clock value
prints real time clock value and sets system time to the same value
sets real time clock to October 8, 5:00 pm, then reads the real time clock and prints its value
sets real time clock to October 8, 5:00 pm, then reads the real time clock, prints its value, and sets the system time to the same value

DIAGNOSTICS
bad conversion
The syntax of the date argument is incorrect.

cannot open/read/write /dev/rtc
No permission to access the real time clock for the indicated function — setrtc tries to open /dev/rtc for writing only if an attempt is made to set the real time clock.

no permission for setting system time
Only the super-user is allowed to set the system time.

FILES
/dev/rtc

WARNING
It is a bad practice to change the date while the system is running multi-user.
NAME
SIO card – Tecmar Quad serial I/O card

DESCRIPTION
This section describes the special files corresponding to the Tecmar four-port card.

The LISA has three expansion slots, any combination of which may hold a Tecmar serial card.

Each serial port is accessed through a character special file. See mknod(1M) for a description of how to create such a file. The major device number is 14 for this card. The minor device number specifies the location of the port and whether modem control handling is enabled.

There is dynamic system configuration on system boot and the positioning of multiple Tecmar boards in the LISA card cage is important. The first Tecmar board (left to right looking towards the back of the LISA) has port numbers 0 through 3. The Tecmar board to its right has ports 4 through 7. A third Tecmar board has ports 8 through 11.

<table>
<thead>
<tr>
<th>Port</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>first card, bottom RS-232 slot connected to card cage</td>
</tr>
<tr>
<td>1</td>
<td>first card, top RS-232 slot connected to card cage</td>
</tr>
<tr>
<td>2</td>
<td>first card, bottom RS-232 slot connected to ribbon cable</td>
</tr>
<tr>
<td>3</td>
<td>first card, top RS-232 slot connected to ribbon cable</td>
</tr>
<tr>
<td>4</td>
<td>second card, bottom RS-232 slot connected to card cage</td>
</tr>
<tr>
<td>5</td>
<td>second card, top RS-232 slot connected to card cage</td>
</tr>
<tr>
<td>6</td>
<td>second card, bottom RS-232 slot connected to ribbon cable</td>
</tr>
<tr>
<td>7</td>
<td>second card, top RS-232 slot connected to ribbon cable</td>
</tr>
<tr>
<td>8</td>
<td>third card, bottom RS-232 slot connected to card cage</td>
</tr>
<tr>
<td>9</td>
<td>third card, top RS-232 slot connected to card cage</td>
</tr>
<tr>
<td>10</td>
<td>third card, bottom RS-232 slot connected to ribbon cable</td>
</tr>
<tr>
<td>11</td>
<td>third card, top RS-232 slot connected to ribbon cable</td>
</tr>
</tbody>
</table>

If no modem control is desired then the standard minor devices, 0 through 11, should be used. A modem control line is indicated by adding 128 to the port number. If modem control on some line(s) is desired then that line(s) should have a minor device number greater than 127, while other line(s) on that card can have standard device numbers. For example, if only the last two ports on the first card need modem control, then the minor devices should be 0, 1, 130, and 131.
NAME
s, rs – Sony disk interface

DESCRIPTION
This section describes the special files corresponding to the Sony microdiskette.

A Sony microdiskette is accessed through a block special or character special file. See mknod(1M) for a description of how to create such a file. The major device number is 1 for block interface, or 6 for character (raw) interface. The minor device number specifies the logical partition through which the disk will be accessed.

Logical partitions indicate, by block offset and size, which 512-byte blocks of the disk are to be accessed. The following table shows the available logical partitions:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Name</th>
<th>Block Offset</th>
<th>Size</th>
<th>Intended Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td>0</td>
<td>800</td>
<td>root filesystem</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>201</td>
<td>599</td>
<td>filesystem on</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>0</td>
<td>201</td>
<td>boot diskette</td>
</tr>
<tr>
<td>3-6</td>
<td>d-g</td>
<td>n/a</td>
<td>n/a</td>
<td>remainder of boot diskette</td>
</tr>
<tr>
<td>7</td>
<td>h</td>
<td>0</td>
<td>800</td>
<td>unused</td>
</tr>
</tbody>
</table>

UNIX does not allow access to the first 201 blocks of a boot or serialization diskette.

The conventional names for the special files are:
• first character of a block device is s (sony),
• first two characters of a character device are rs (raw sony),
• next the digit 0, the physical unit number, and
• last a character, the logical partition (a,b,c,h)

A limited character interface is also available. Character major device 7 allows only the eject function. /dev/eject can be specified as the argument to the eject utility.
NAME
speaker — interface to the LISA speaker

DESCRIPTION
This section describes the special file corresponding to the built in speaker.

The file /dev/speaker is the general purpose interface to the LISA speaker. This device is capable of producing square wave output at one of 8 volumes between 125,000 to 122 cycles per second. A request to open the speaker will fail under any of the following circumstances:

The speaker is already open. (EBUSY)
The open mode does not include writing. (EINVAL)
The minor device number is not zero. (ENXIO)

Speaker data is written in records. The record format (as defined in the include file sys/speaker.h) is as follows:

```c
struct speaker {
    ushort sk_wavlen; /* wave length */
    ushort sk_duration; /* duration */
    ushort sk_volume; /* volume */
};
```

The sk_wavlen is a measure in microseconds between transitions of the waveform. Values less than 8 are rounded up to 8 and represent the shortest wavelength (about 125,000 cycles per second) which is well above the audible range. The maximum value is 8191 and corresponds to about 122 cycles per second.

The sk_duration field is the number of clock ticks to maintain the sound. A clock tick on the LISA is 1/60th of a second.

The sk_volume field determines the volume. Only the low 3 bits are used.