another operating system at times. It simply resets the timer chip parameters to the
default mode of operation that MS-DOS and other operating systems may expect
the ROM BIOS to have provided when they first start.

`Milli_delay` (line 11502) is provided for use by any task that needs very short
delays. It is written in C without any hardware-specific references, but it uses a
technique one might expect to find only in a low-level assembly language routine.
It initializes a counter to zero and then rapidly polls it until a desired value is
reached. In Chapter 2 we said that this technique of busy waiting should generally
be avoided, but the necessities of implementation can require exceptions to gener-
al rules. The initialization of the counter is done by the next function, `milli_start`
(line 11516), which simply zeroes two variables. The polling is done by calling
the last function, `milli_elapsed` (line 11529), which accesses the timer hardware.
The counter that is examined is the same one used to count down clock ticks, and
thus it can underflow and be reset to its maximum value before the desired delay
is complete. `Milli_elapsed` corrects for this.

### 3.9 TERMINALS

Every general purpose computer has one or more terminals used to communi-
cate with it. Terminals come in an extremely large number of different forms. It
is up to the terminal driver to hide all these differences, so that the device-
dependent part of the operating system and the user programs do not have to be
rewritten for each kind of terminal. In the following sections we will follow our
now-standard approach of first discussing terminal hardware and software in gen-
eral, and then discussing the `MINIX` software.

#### 3.9.1 Terminal Hardware

From the operating system's point of view, terminals can be divided into three
broad categories based on how the operating system communicates with them.
The first category consists of memory-mapped terminals, which consist of a key-
board and a display, both of which are hardwired to the computer. The second
category consists of terminals that interface via a serial communication line using
the RS-232 standard, most frequently over a modem. The third category consists
of terminals that are connected to the computer via a network. This taxonomy is
shown in Fig. 3-27.

**Memory-Mapped Terminals**

The first broad category of terminals named in Fig. 3-27 consists of memory-
mapped terminals. These are an integral part of the computers themselves.
Memory-mapped terminals are interfaced via a special memory called a video
RAM, which forms part of the computer's address space and is addressed by the CPU the same way as the rest of memory (see Fig. 3-28).

Also on the video RAM card is a chip called a **video controller**. This chip pulls character codes out of the video RAM and generates the video signal used to drive the display (monitor). The monitor generates a beam of electrons that scans horizontally across the screen, painting lines on it. Typically the screen has 480 to 1024 lines from top to bottom, with 640 to 1200 points per line. These points are called **pixels**. The video controller signal modulates the electron beam, determining whether a given pixel will be light or dark. Color monitors have three beams, for red, green, and blue, which are independently modulated.

A simple monochrome display might fit each character in a box 9 pixels wide by 14 pixels high (including the space between characters), and have 25 lines of 80 characters. The display would then have 350 scan lines of 720 pixels each. Each of these frames is redrawn 45 to 70 times a second. The video controller could be designed to fetch the first 80 characters from the video RAM, generate 14 scan lines, fetch the next 80 characters from the video RAM, generate the following 14 scan lines, and so forth. The video controller may also be set up to fetch an address, 8 bits in each byte, and output the bytes as needed. The address may be always blank, or it may be used to fetch needed patterns for the character display.

The IBM 3270 display makes use of a character-oriented video RAM (see Fig. 3-27). Each character is displayed in the RAM, and the RAM is addressed. The video controller determines the color, background, etc., of the character.
following 14 scan lines, and so on. In fact, most fetch each character once per scan line to eliminate the need for buffering in the controller. The 9-by-14 bit patterns for the characters are kept in a ROM used by the video controller. (RAM may also be used to support custom fonts.) The ROM is addressed by a 12-bit address, 8 bits from the character code and 4 bits to specify a scan line. The 8 bits in each byte of the ROM control 8 pixels; the 9th pixel between characters is always blank. Thus $14 \times 80 = 1120$ memory references to the video RAM are needed per line of text on the screen. The same number of references are made to the character generator ROM.

The IBM PC has several modes for the screen. In the simplest one, it uses a character-mapped display for the console. In Fig. 3-29(a) we see a portion of the video RAM. Each character on the screen of Fig. 3-29(b) occupies two characters in the RAM. The low-order character is the ASCII code for the character to be displayed. The high-order character is the attribute byte, which is used to specify the color, reverse video, blinking, and so on. The full screen of 25 by 80 characters requires 4000 bytes of video RAM in this mode.

![Image of video RAM and screen](image)

**Figure 3-29.** (a) A video RAM image for the IBM monochrome display. (b) The corresponding screen. The $\times$ are attribute bytes.

Bit-map terminals use the same principle, except that each pixel on the screen is individually controlled. In the simplest configuration, for a monochrome display, each pixel has a corresponding bit in the video RAM. At the other extreme, each pixel is represented by a 24-bit number, with 8 bits each for red, green, and blue. A $768 \times 1024$ color display with 24 bits per pixel requires 2 MB of RAM just to hold the image.

With a memory-mapped display, the keyboard is completely decoupled from the screen. It may be interfaced via a serial or parallel port. On every key action the CPU is interrupted, and the keyboard driver extracts the character typed by reading an I/O port.

On the IBM PC, the keyboard contains an embedded microprocessor which communicates through a specialized serial port with a controller chip on the motherboard. An interrupt is generated whenever a key is struck and also when one is
released. Furthermore, all that the keyboard hardware provides is the key number, not the ASCII code. When the A key is struck, the key code (30) is put in an I/O register. It is up to the driver to determine whether it is lower case, upper case, CTRL-A, ALT-A, CTRL-ALT-A, or some other combination. Since the driver can tell which keys have been struck but not yet released (e.g., shift), it has enough information to do the job. Although this keyboard interface puts the full burden on the software, it is extremely flexible. For example, user programs may be interested in whether a digit just typed came from the top row of keys or the numeric keypad on the side. In principle, the driver can provide this information.

RS-232 Terminals

RS-232 terminals are devices containing a keyboard and a display that communicate using a serial interface, one bit at a time (see Fig. 3-30). These terminals use a 9-pin or 25-pin connector, of which one pin is used for transmitting data, one pin is for receiving data, and one pin is ground. The other pins are for various control functions, most of which are not used. To send a character to an RS-232 terminal, the computer must transmit it 1 bit at a time, prefixed by a start bit, and followed by 1 or 2 stop bits to delimit the character. A parity bit which provides rudimentary error detection may also be inserted preceding the stop bits, although this is commonly required only for communication with mainframe systems. Common transmission rates are 9600, 19200, and 38400 bps. RS-232 terminals are commonly used to communicate with a remote computer using a modem and a telephone line.

![Diagram of RS-232 communication](image)

**Figure 3-30.** An RS-232 terminal communicates with a computer over a communication line, one bit at a time. The computer and the terminal are completely independent.

Since both computers and terminals work internally with whole characters but must communicate over a serial line a bit at a time, chips have been developed to do the character-to-serial and serial-to-character conversions. They are called UARTs (Universal Asynchronous Receiver Transmitters). UARTs are attached to the computer by plugging RS-232 interface cards into the bus as illustrated in Fig. 3-31. RS-232 terminals are gradually dying off, being replaced by PCs and X terminals.

To print this card, what happened by the the UART chip character. An error character enters the interface lines, and another character's name is detected by doing interrupt routine and the lines, taking RS-232.

RS-232 is the simplest and most primitive interface to be found on any terminal device.

Intel does not have a C character coding system, but there are terminal software senders. The X may be printed near the middle of a sentence.

**X Terminals**

The X Window System is a powerful and versatile computer and a major project of M.I.T.'s Laboratory for Computer Science.

An X terminal is a dedicated terminal used with X as one of a set of tools mapped onto a large-scale, or...
terminals, but they are still encountered on older mainframe systems, especially in banking, airline reservation, and similar applications.

To print a character, the terminal driver writes the character to the interface card, where it is buffered and then shifted out over the serial line one bit at a time by the UART. Even at 38,400 bps, it takes just over 250 microsec to send a character. As a result of this slow transmission rate, the driver generally outputs a character to the RS-232 card and blocks, waiting for the interrupt generated by the interface when the character has been transmitted and the UART is able to accept another character. The UART can simultaneously send and receive characters, as its name implies. An interrupt is also generated when a character is received, and usually a small number of input characters can be buffered. The terminal driver must check a register when an interrupt is received to determine the cause of the interrupt. Some interface cards have a CPU and memory and can handle multiple lines, taking over much of the I/O load from the main CPU.

RS-232 terminals can be subdivided into categories, as mentioned above. The simplest ones were hardcopy (printing) terminals. Characters typed on the keyboard were transmitted to the computer. Characters sent by the computer were typed on the paper. These terminals are obsolete and rarely seen any more.

Dumb CRT terminals work the same way, only with a screen instead of paper. These are often called “glass ttys” because they are functionally the same as hardcopy ttys. (The term “tty” is an abbreviation for Teletype® a former company that pioneered in the computer terminal business; “tty” has come to mean any terminal.) Glass ttys are also obsolete.

Intelligent CRT terminals are in fact miniature, specialized computers. They have a CPU and memory and contain software, usually in ROM. From the operating system’s viewpoint, the main difference between a glass tty and an intelligent terminal is that the latter understands certain escape sequences. For example, by sending the ASCII ESC character (033), followed by various other characters, it may be possible to move the cursor to any position on the screen, insert text in the middle of the screen, and so forth.

**X Terminals**

The ultimate in intelligent terminals is a terminal that contains a CPU as powerful as the main computer, along with megabytes of memory, a keyboard, and a mouse. One common terminal of this type is the X **terminal**, which runs M.I.T.’s X Window System. Usually, X terminals talk to the main computer over an Ethernet.

An X terminal is a computer that runs the X software. Some products are dedicated to running only X; others are general-purpose computers that simply run X as one program among many others. Either way, an X terminal has a large bit-mapped screen, usually 960 x 1200 or better resolution, in black and white, grayscale, or color, a full keyboard, and a mouse, normally with three buttons.
The program inside the X terminal that collects input from the keyboard or mouse and accepts commands from a remote computer is called the X server. It communicates over the network with X clients running on some remote host. It may seem strange to have the X server inside the terminal and the clients on the remote host, but the X server's job is to display bits, so it makes sense to be near the user. The arrangement of client and server is shown in Fig. 3-31.

![Figure 3-31. Clients and servers in the M.I.T. X Window System.](image)

The screen of the X terminal contains some number of windows, each in the form of a rectangular grid of pixels. Each window usually has a title bar at the top, a scroll bar on the left, and a resizing box in the upper right-hand corner. One of the X clients is a program called a window manager. Its job is to control the creation, deletion, and movement of windows on the screen. To manage windows, it sends commands to the X server telling what to do. These commands include draw point, draw line, draw rectangle, draw polygon, fill rectangle, fill polygon, and so on.

The job of the X server is to coordinate input from the mouse, keyboard, and X clients and update the display accordingly. It has to keep track of which window is currently selected (where the mouse pointer is), so it knows which client to send any new keyboard input to.

### 3.9.2 Terminal Software

The keyboard and display are almost independent devices, so we will treat them separately here. (They are not quite independent, since typed characters must be displayed on the screen.) In MINIX the keyboard and screen drivers are part of the same task; in other systems they may be split into distinct drivers.