Timeshare— Turn your Minicomputer Into a Maxi!

Your TV typewriter or a similar keyboard/display type of terminal can be your access to a full-scale computer system. All you need is a modem and proper authorization to share computer time with many others

PATRICK GODDING

AN IDEAL COMPUTER OPERATION WOULD BE efficient for both the user and the computer system. A *single-user* system, where one terminal has complete control of the computer, allows interaction between user and computer. However, only a very small part of the computer's capabilities are used. *Batch* systems offer efficient computer use, but there is no interaction. A program must be loaded, executed, and printed before an error can be detected. After the program is corrected, the procedure must be repeated. This is inefficient from a user standpoint.

The disadvantages of *single-user* and *batch* modes of computer operation are greatly reduced in a *timeshare* system. Timeshare (or multi-user) systems allow interaction between a number of user terminals and a single high-speed computer (see Fig. 1).

A timeshare system

The minicomputer has made it possible to build a timeshare system that can be afforded by persons not in the computer field. Price reductions in terminals and the development of less complicated programming languages have also helped bring this situation about.

The system described in this article is typical of the trend in timeshare systems: no longer the huge centralized systems, but rather many small localized ones.

The minicomputer

Minicomputers are available in many different levels of sophistication, but all have the same standard subsystems. A CPU (*C*entral *P*rocessor *U*nit) controls all internal operations and synchronizes the input/output operations. An ALU (*A*rithmetic-*L*ogic *U*nit) processes mathematical operations. A *control memory* contains all the hardware instructions for use by the CPU and a *data storage memory* or *core memory* contains system software instructions and data storage. Peripheral devices such as line printers, card readers, magnetic tape drives and disk drives can all be supported by minicomputers, and the more sophisticated ones can support timeshare and batch operating systems simultaneously.

The multiplexor (MUX)

The MUX is a hardware device that interfaces all the user terminals to the computer. It operates much like a rotary switch—connecting each terminal line to the computer for a small amount of time, then going on to the next line.

The terminals and computer communicate via a binary code called ASCII (*A*merican *S*tandard *C*ode for *I*nformation *I*nterchange). Each character on the terminal keyboard is represented by an 8-bit ASCII number. The MUX receives and transmits these codes one bit at a time, so that each terminal user appears to have unique control over the computer. The delay that a user experiences is minimal or zero since the MUX and the computer operate at such high speeds.

The disc

When a timeshare system has no provision for mass storage, the computer's core memory must be divided into a number of sections equal to the number of users. This can greatly limit the size of a user program.

With the addition of a *disc unit,* a user program can be as large as the total amount of core memory. This is done by a software system program that swaps user programs between the disc and core memory. No swapping occurs if all the programs fit in core.

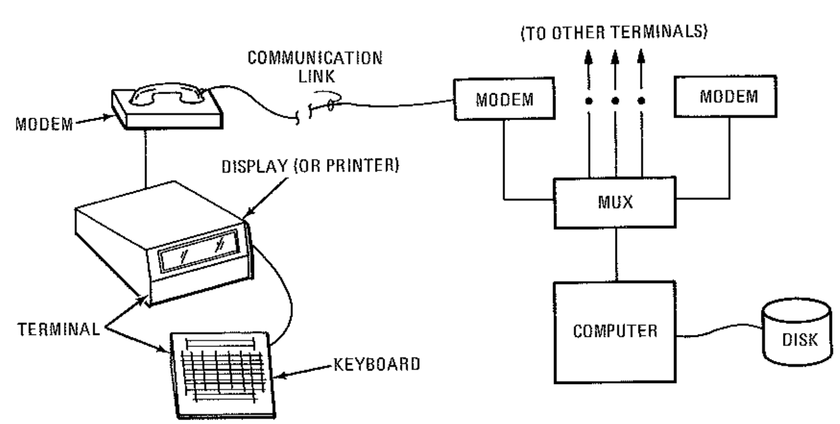

FIG. 1—TIME-SHARING TERMINAL and its connection by telephone "communications link" to a central computer. Because the computer "thinks" in terms of microseconds and the user in terms of seconds, many users can share the same computer without any observable interference or delay. The "printer" in many of these terminals is the same unit as the keyboard.

There are many types of disc units, but the concept of all is the same. The disc drive can be likened to a record player and the disc to a record, except that the drive records as well as plays. The drive also must be able to locate information anywhere on the disc, instead of simply "playing" the disc from start to finish as with a record player. Swapping programs to and from the disc does not create an unreasonable delay since many disc drives can transfer data at a rate greater than 700,000 bits-per-second.

When a disc unit is incorporated into a timeshare system, a part of the disc storage area can be allocated to the user for program storage. This provides an area both for a user library of programs that no other users have access to and also for a common library filled with programs that can be accessed and run by any user. This is an important feature because it alleviates the need to enter a program more than one time—it can be left on the disc between uses. Computer connect time (the amount of time that a user is signed-on to the system) is greatly reduced.

The terminal

The interface between the user and the computer (or MUX) is a computer terminal. The terminal has a keyboard for sending data and some type of display or printing device for receiving data. Many terminals have their own internal memory for storing data. This can allow entering a complete program before connecting to the computer—a good feature if the user is paying for connect time. Some terminals also have external connections for cassette tape recorders, printing units, and even disc drives (usually a discette-type drive, smaller than the discs discussed earlier).

The smaller size and reduced cost of terminals makes it possible for a user to have a terminal in an office or even in the home. When the terminal is not physically close enough to the computer for direct connection, the telephone system provides an economical communications link.

A common method of data transmission via the telephone is FSK (*F*requency-*S*hift *K*eying). This type of modulation converts the serial ASCII information into two frequencies that can be transmitted readily over ordinary telephone lines. One frequency represents a logical-high level (called a "mark"), and the other a logical-low level (called a "space"). (See Fig. 2, top) The device that converts the transmitted ASCII information into FSK and the received FSK into ASCII is the MODEM (*MO*dulator/*DEM*odulator). A MODEM is required for each terminal line at the computer and at each terminal. A few terminals have built-in MODEM's—a good feature since an external MODEM means added expense.

The language—Extended BASIC

Even with the low cost of a timeshare system, many people have not considered using one because of the time involved in learning to "talk" to a computer: learning the language.

Computer languages that are very similar to English now exist. The most common, perhaps, is called BASIC. It was first created for scientists and mathematicians who were not computer-oriented, but needed the use of a computer in their work. Now, with Extended BASIC, the language has been further developed to include both powerful format statements for control of output (printed business forms, payroll checks, etc.) and file input/output capabilities for storing large amounts of data, as needed with inventories, purchase orders and the like, as well as for purely scientific purposes.

The language provides two modes of use: the immediate mode for instant calculations or program debugging and the program mode in which a user writes a group of statements to solve a specific or unique problem. The only difference between the two modes is that in the program mode, a number between 1 and 9999 must precede each statement. Examples of both types follow:

Immediate mode

```
PRINT 3/4 ↵
```

This statement tells the computer to calculate the value of 3 divided by 4 and send the result to your terminal. A curved arrow at the end of the statement represents pushing the RETURN key on the terminal keyboard. The RETURN key tells the computer to begin execution.

```
PRINT SQR ((4.2 ↑ 3) + (5.3 ↑ 3)) ↵
```

This statement says: Find the 3rd power of 4.2 and 5.3, add these results together and take the square root, then send the answer to the terminal.

Program mode

If we wanted the above problem solved for *any* two numbers, we would write a short program:

```
10 INPUT A, B ↵
20 LET C=SQR ((A ↑ 3) + (B ↑ 3)) ↵
30 PRINT C ↵
40 GOTO 10 ↵
RUN ↵
```

In this program, we have substituted variables (A and B) for the values 4.2 and 5.3.

After this program is entered into the terminal, the word RUN is entered, the RETURN key is depressed and execution begins.

When line number 10 is executed, the computer sends the terminal a question mark (?) that tells the user to enter his values. After the user enters the first value, a comma (,), the second value and depresses the RETURN key, execution continues. Line 10 has now assigned the first value that was entered to the vari-

continued on page 25

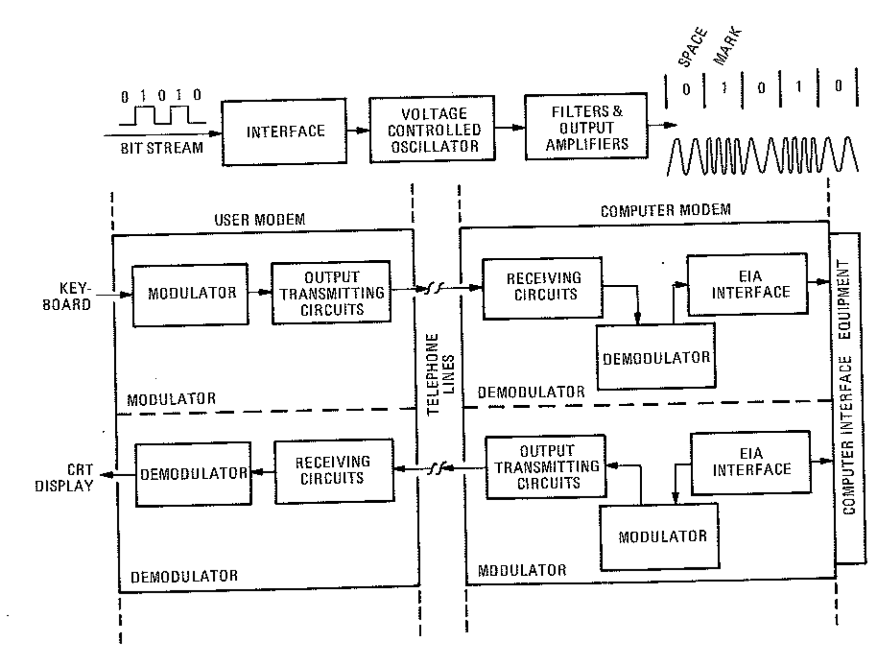

FIG. 2—HOW THE SIGNALS ARE TRANSMITTED AND RECEIVED. The marks and spaces ("bit stream" at top of diagram) are transmitted as two frequencies over the telephone lines. At the computer end, they are demodulated and sent to the computer in their original bit-and-space format. The process is reversed to return information to the terminal. Interface units prevent reaction between telephone and computer circuitry.

KOMPUTER KORNER

continued from page 22

Transmit subroutine

The transmit subroutine, shown in Table 1, for the software UART occupies twenty to twenty-five successive program steps in memory once the appropriate PUSH, POP, and RET instructions have been included. Also required is a 9.09 ms time-delay subroutine that corresponds to an asynchronous serial ASCII data transmission rate of 110 Baud, i.e., teletype speed. The program in Table 1 can be described as follows:

Register L is used as the bit counter for the 11-bit ASCII word, and is set initially to octal 013. The seven data-bits plus the parity bit, which is Bit 8, are assumed to be present in the accumulator. At the LO memory address 146, the accumulator is OR'ed to itself to clear the carry bit (shown on the far left in Fig. 2.) In Fig. 2, the least significant data bit

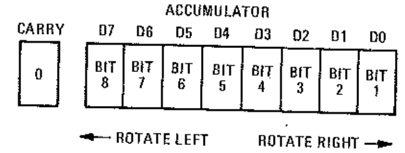

FIG. 2

is Bit 1. At address LO = 147, a RAL instruction is performed to rotate the start bit to bit position DO in the accumulator. Fig. 3 should provide you with assistance in understanding the four different rotate instructions

in the 8080A microprocessor instruction set.

At address LO = 150, the start bit is output to the SN7474 data latch. The program then goes into a 9.09 ms time-delay subroutine, after which Bit 1 is rotated into

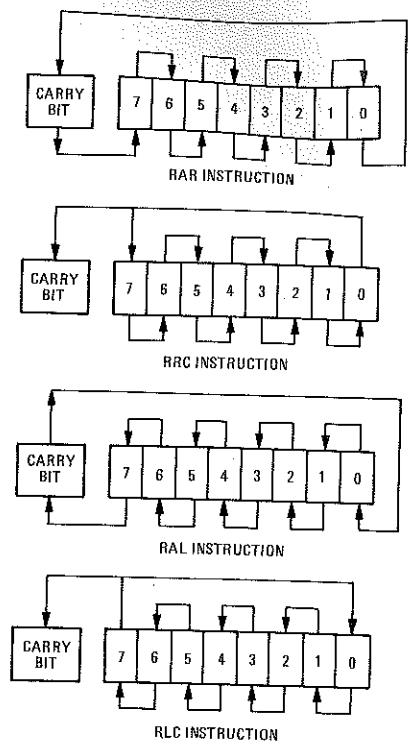

FIG. 3

the DO accumulator position and the carry bit is set to logic 1. Bit 1 is fed to the SN7474 latch, the ASCII word bit counter in register L is decremented and program control is returned to the time-delay subroutine that is called at address LO = 152. The loop from LO = 152 to LO = 164 is executed a total of eleven times, after which register L becomes zero and the JNZ instruction at address LO = 162 is ignored.

A software UART transmit subroutine possesses a flexibility equivalent to the original 40-pin UART chip. With appropriate modifications to the program or the original accumulator data, you can transmit 5, 6, 7, or 8 data bits; 1 or 2 stop bits; even or odd parity; and parity or no parity. The time-delay subroutine can be modified so that you can transmit at data rates from 60 to 9600 Baud for a 750-kHz clock rate and higher for 2-MHz and 4-MHz clock rates.

The conversion from one data transmission rate to another is easily accomplished with the aid of appropriate software time-delay subroutines that replace R—C time-constant circuits. An additional advantage that is gained from the use of software is the potential to perform code conversions. For example, 5-level Baudot KSR machines are in widespread use and can still be obtained for under $50. It is not too difficult to develop software that converts ASCII to Baudot and thus produce an inexpensive hard-copy terminal for the laboratory scientist, engineer, ham or computer buff.

Receive subroutine

The software UART receive subroutine requires 50 instructions and will not be repeated here. (Copies of the transmit and receive subroutines and a description of the

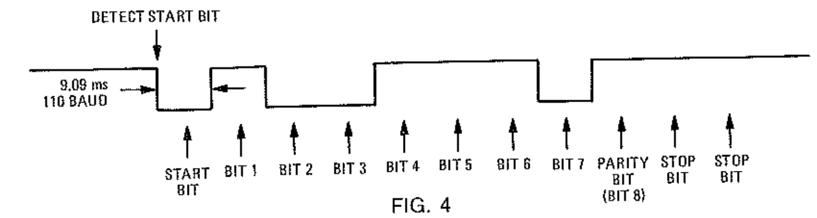

FIG. 4

smart data-entry station are available from Professor Paul Field, Department of Chemistry, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061.) The basic programming concepts associated with the receive subroutine are shown in Fig. 4, which represents an 11-bit asynchronous serial ASCII word that is being detected by the 8080A-based microcomputer with the aid of the SN74126 three-state buffer gate shown in Fig. 1. The program repeatedly tests the serial ASCII input line for a logic 0 state. Once a logic 0 state is detected, which corresponds to a start bit, the program goes into a 4.54-ms wait loop. Upon leaving the wait loop, the program again inputs the logic 0 into bit position DO in the accumulator, thus testing the validity of the start bit. The start bit is rotated to the carry bit and the program then enters a 9.09-ms wait loop, after which it inputs Bit 1 into position DO in the accumulator. Register H is used as the SAVE register that stores the growing ASCII data word. The SAVE register is rotated one position, and the 9.09-ms wait loop is again entered, after which Bit 2 (a logic 0 in Fig. 4) is input into bit position DO in the accumulator. The input of successive data and parity bits continues until the entire 8-bit data word is entered into the SAVE register. The two stop bits are also detected. With appropriate modifications, the program can detect parity or framing errors or an overrun condition. A data-ready flag signal can also be generated from software with the aid of a second SN7474 latch.

Applications

The above software UART routines were used in a "smart" remote data-entry station that was tied via a 20 mA current-loop to a PDP 8/L minicomputer in a physical chemistry laboratory. The data entry station intercepted the 20-mA teletype current-loop tied to the minicomputer. The remote data-entry station permitted students to load data into memory and then transmit it as a block to the minicomputer, which analyzed the data and provided a print-out. With the 20-mA current loop operating in the full duplex mode, ten or more remote data-entry stations could be tied to the minicomputer.

This column provides a good demonstration of the software-hardware tradeoffs that can be accomplished using microcomputers. Similar, and perhaps more comprehensive, routines have already been written for all of the popular microprocessor chips, such as the 16-bit PACE or the 8-bit 6800. The faster and less expensive that microcomputers become, the more likely that all moderate speed digital functions will be executed via software. The theme of software replacing hardware is an important one, and we will return to it many times in future columns. **R-E**

TIMESHARE

continued from page 67

able A and the second value to B. Line 20 assigns to variable C the value of the square root of the sum of the two entered values to the third power. Line 30 outputs the value of C to the terminal. Line 40 transfers execution back to line 10, and you're ready to execute the problem for two new values.

When you have all the results you require, you push the ESC key (escape) to stop the program, then sign off or enter a new program.

Notice the similarity between BASIC and English:

- INPUT—to input a value for a variable
- LET—to let a variable equal a value
- PRINT—to print the results at the terminal
- GOTO—to go to another part of the program.

This similarity exists throughout the BASIC language, which makes it ideal for people who are not computer programmers but need or desire to use a computer. **R-E**