SPECIFICATIONS
HP-85 Personal Computer

READ/WRITE MEMORY
STANDARD: 16,384 bytes (10% for system overhead).
OPTION: 32,764 bytes, using 16,384 byte plug-in RAM.
CRT DISPLAY
SCREEN SIZE: 127 mm (5 in) diagonal.
INTENSITY: Adjustable.
CHARACTER GENERATION: 5 × 7 dot character font in 8 × 12 dot cell.
SCREEN CAPACITY: 16 lines × 32 characters full display, 64 line scrolling.
CURSOR: Underline.
PLOTTING AREA: 256 wide by 192 high dot matrix.
THERMAL MOVING-HEAD PRINTER
INTENSITY: Adjustable.
PRINT SPEED: Two 32-character lines per second.
CHARACTERS: 5 × 7 font in 7 × 10 cell.
GRAPHICS OUTPUT: Dot for dot reproduction of CRT display.
PAPER SIZE: 122m (400 ft) long by 10.8 cm (4.3 in) wide roll.
TAPE CARTRIDGE DRIVE
PROGRAM CAPACITY: 195K bytes.
DATA CAPACITY: 210K bytes.
ACCESS: Directory, file-by-name, 42 named files, max.
ERROR RATE: 1 bit in 10^9.
SEARCH SPEED: 152 cm/s (60 in/s).
READ/WRITE SPEED: 25.4 cm/s (10 in/s).
REWIND TIME: 29 s, max.
CARTRIDGE SIZE: WIDOOA, 63.5 × 82.5 × 12.7 mm (2.5 × 3.25 × 0.5 in).
QUARTZ CRYSTAL TIMER
TIME KEEPING: Seconds since midnight with year and day-in-year.
TIME SET: Must be reset each time power is turned on.
PROGRAMMABLE TIMERS
NUMBER: Three.
OPERATION: Independently programmable interrupt.
RANGE: 1 ms to 99,999,999 ms (27.7 hours).
BEEPER
TONE: Programmable from approximately 0 to 4575 Hz.
DURATION: Programmable.
DEFAULT TONE AND DURATION: ~2000 Hz, 100 ms.
I/O SLOTS
NUMBER: Four.
INTERFACE OPTIONS: HP-IB (82937A), RS-232, GP-IO, BCD, 16K byte RAM module (82903A), ROM drawer (82936A) for up to six ROM modules.
BASIC LANGUAGE
PREDEFINED FUNCTIONS: 42, of which 10 are trigonometric.
STATEMENTS: 81, 16 of which are for graphics.
COMMANDS: 20, 10 of which are programmable.
DYNAMIC RANGE
REAL: ~9.99999999999E499 to ~1E~499, 0, 1E~499 to 9.99999999999E499.
SHORT: ~9.99999E99 to ~1E~99, 0, 1E~99 to 9.99999E99.
INTEGER: ~99999 to 99999.
TEMPERATURE ENVIRONMENT
OPERATING: 5° to 40°C (41° to 104°F).
STORAGE: ~40° to 65°C (~40 to 149°F).
POWER REQUIREMENTS
SIZE: HWD 15.8 cm × 41.9 cm × 45.2 cm (6.3 in × 16.5 in × 17.8 in).
WEIGHT: 9.06 kg (20 lb).
VOLTAGE: 90-127 Vac (115 Vac line) or 200-254 Vac (230 Vac line) switch selected on rear panel.
FREQUENCY: 50-60 Hz.
CONSUMPTION: 25W.
PRICE IN U.S.A.: HP-85 base price: $3,250.
MANUFACTURING DIVISION: CORVALLIS DIVISION
1000 N.E. Circle Blvd.
Corvallis, Oregon 97330 U.S.A.

Acknowledgments

A great deal of credit must be given to the management team for the HP-85. Ernst Erni was the section manager for the project. His foresight and guidance contributed enormously to the success of the product. Kent Stockwell was the project manager responsible for the mechanical designs and overall system integrity. Chung Tung's support as lab manager was very beneficial. Tim Williams and John Nairn served as project managers responsible for the plug-in modules.

Also, credit goes to Marv Higgins, who engineered the subassembly interconnect scheme as well as the EMI shielding design, and to Chuck Dodge, who did the excellent job of industrial design on the product. Norm Glaeser was responsible for product QA. Deme Clainos and Carmen West contributed their skills in the product marketing area.

Adding I/O Capability to the HP-85

With the implementation of I/O features, the capabilities of a self-contained personal computer system are expandable to control instruments, add on more powerful peripherals, and even talk to other computers.

by John H. Nairn, Tim I. Mikkelsen, and David J. Sweetser

THE HP-85 is an integrated, stand-alone personal computer. This means that within this single package are contained all of the elements of a small computer system. Part of Fig. 1 shows a block diagram of the HP-85 mainframe. The box labeled CPU contains the main processor, ROM and RAM memory, operating system, power supply, and all of the other elements that in a large mainframe or minicomputer would constitute the computer itself. The remaining elements provide the human interface, letting a programmer and/or operator communicate with the computer. Besides the obvious advantage of having a complete, functional computer in a single portable unit, the integrated computer architecture provides certain operational advantages.

For example, in the HP-85, a graphics display created on the CRT can be dumped to the built-in printer by executing a single COPY statement. This feature would be much more difficult if not impossible to provide if the printer were an external device with unknown operational characteristics.

The internal peripherals chosen to be incorporated into the HP-85 make it a complete tool for solving a large variety of computational problems. However, there are two classes of tasks that require that the system be expandable.

The first class consists of tasks for which the human interface elements need to have more powerful characteristics than those provided by the internal peripherals. A particular application may require a printer capable of full page width printing, or of filling out standard forms in multiple copies. For graphics displays, a full-size plotter with multicolor capability might be required. Or the use of a human interface element not even provided in the basic system, such as a digitizer, may be necessary. The ability to support external as well as internal peripherals greatly expands the class of problems the system can solve.

The second class consists of tasks in which it is desirable to eliminate as much of the human interface as possible. In data acquisition applications, measurement instruments are available that can communicate their results directly to the computer, eliminating the need for manual entry of data by an operator. When the computer is capable not only of acquiring and/or performing analyses on the data, but also of making decisions based on that data and providing feed-

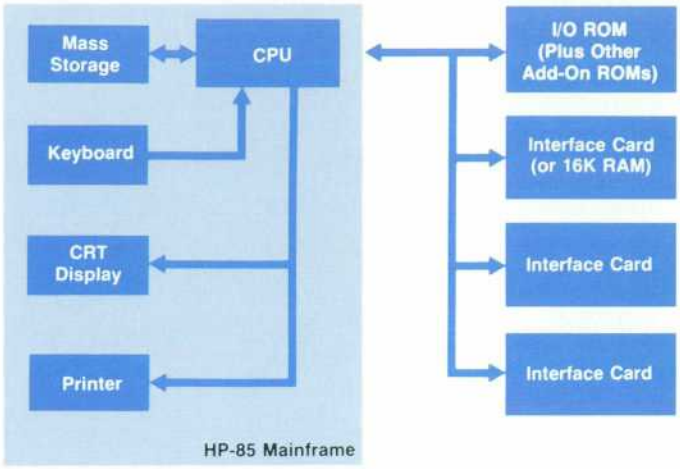

Fig. 1. *The basic functional blocks for the HP-85 personal computer are integrated into a single mainframe for a self-contained computer system. Four I/O slots are provided to expand system capability. Various combinations (such as the one shown) of the add-on ROM drawer, additional 16K RAM, and peripheral interfaces can be installed in these four slots.*

back that modifies the operation of the system to which it is connected, the computer functions as a controller.

Two elements are required to add I/O capability to the HP-85. First, a piece of firmware (I/O ROM) is required to give the operator access to the interfaced device. Second, a piece of hardware (interface card) is required to provide electrical, mechanical, and timing compatibility for the device to be interfaced to the computer.

The HP-85 is a BASIC language computer. Many of the language features are similar or identical in syntax and semantics to the interfacing statements on the 9835 and 9845 Computers.[1,2] The design of the HP-85 interfacing capability also draws heavily from the 9825 Computer's I/O architecture.[3]

The HP-85 contains both read-only memory (ROM) and read-write or random-access memory (RAM). The RAM contains the user's BASIC language program (software) and data. The ROM contains the machine language program (firmware) which recognizes and executes the statements provided by the BASIC language. Thus, the operating system ROM in the HP-85 provides such statements as PRINT, DISP, and INPUT for accessing the internal peripherals. When external peripherals are added, their wider range of capabilities requires more extensive BASIC language statements to fully use these capabilities. Additional plug-in modules, called add-on ROMs, merely enrich the BASIC language by increasing the number of statements and functions that can be recognized and executed.

Some of these add-on ROMs may be dedicated to a particular device. For example, a ROM may add statements to the BASIC language which are designed only to provide complete control of a floppy disc or of a full-size plotter. The I/O ROM, on the other hand, is designed to be as much as possible a general-purpose tool for communicating with the wide variety of peripherals and instruments available.

Almost all computers provide language extensions for outputting data to a device or entering data from a device. The OUTPUT and ENTER statements are usually sufficient for communicating with external peripherals. In controller applications, however, where timing may be critical and the speed of the external devices and instruments may not be well matched to the speed of the computer, other methods of transferring data and control information can increase the performance of the system. In many cases this can make the difference in whether the application can be done at all. For example, in taking data from very slow devices, the ability to transfer data under interrupt allows the HP-85 to perform other operations instead of waiting on the slow device, thus increasing system throughput. In the other extreme, being able to capture data in a burst transfer from a very fast device and then process the data into internal format at a later time can be a "make-or-break" capability in fast data acquisition applications.

In the back of the HP-85 are four slots that allow add-on ROMs, add-on RAM, and interface cards to be connected to the internal memory bus. A single ROM drawer containing the I/O ROM and up to five other add-on ROMs can be plugged into any one of these slots. Thus a typical interfacing configuration can include the ROM drawer and three interface cards; or the ROM drawer, the add-on RAM module and two interface cards, as shown in Fig. 1.

On each interface card is a custom integrated circuit called the translator chip (TC), which translates between the timing and protocol requirements of the internal memory bus and a microcomputer bus (Fig. 2). Each card also contains an 8049 microcomputer and a set of discrete drivers that implement the electrical and protocol requirements

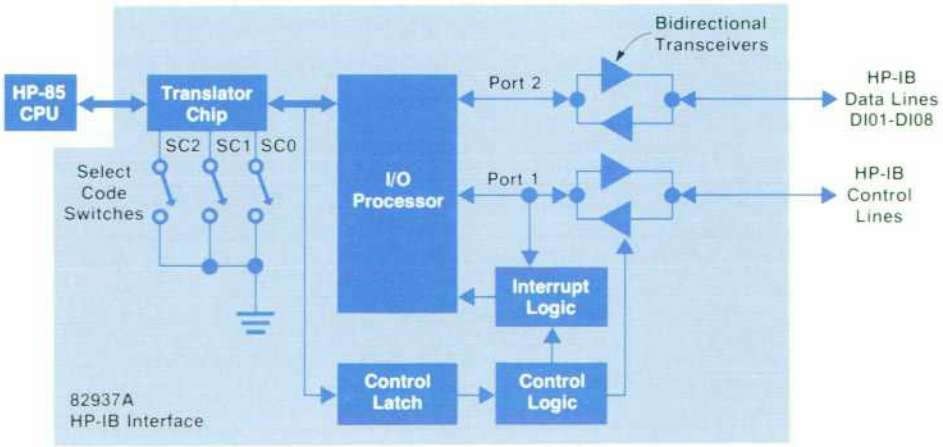

Fig. 2. *Typical interface card design. The translator chip permits communication between the system CPU and the microprocessor on board the interface. The I/O processor then implements the electrical requirements for the particular interface functions.*

for that card.

The most general-purpose interface card is the sixteen-bit parallel card (GPIO). Sixteen TTL compatible bidirectional lines and sixteen open-collector output lines, plus a variety of status, control, and handshake lines make this card a versatile workhorse. A variation of this card, with the input lines organized into four-bit nibbles, is also available for interfacing devices that operate in binary-coded-decimal (BCD) format.

A third type of interface card is designed to connect the HP-85 to serial I/O devices using either RS-232C or 20-mA current loop configurations. This card will also allow terminals to be connected to the HP-85, or allow the HP-85 to be connected to a modem or a host computer.

The fourth card is the 82937A HP-IB Interface, the HP-85's implementation of IEEE Standard 488-1978. Because of the microprocessor included on this card, a very powerful implementation of both controller and noncontroller capabilities is provided.

The HP-IB Interface Card is available now and the others (GPIO, BCD, and Serial I/O) will be available later.

I/O ROM

There are three areas that any interfacing capability in a computing system must connect—the programmer, the operating system, and the I/O hardware. On HP's interpretive computers the user interface is through keyword execution of the language extensions provided by the I/O ROM. An important aspect of the friendliness of these computers is the choice between calculator and program mode execution of the keywords. The program mode means the statement has a line number and is to be stored for later program execution (parsed,* stored in the program memory, executed at **RUN** time). The calculator mode means the statement does not have a line number and is to be immediately executed (parsed, stored in temporary memory, and executed).

In the HP-85, the I/O hardware—the interface card—is a type of channel processor. A channel processor is a computer with limited resources that performs many of the central computer's interfacing tasks. The interface to the I/O hardware is the I/O drivers. These implement the protocol that the central computer uses to tell the I/O card what it is supposed to be doing. The I/O drivers include data passage routines, command passage routines, status and control routines, and an interrupt handler.

The interface to the operating system is what makes interfacing capabilities a natural extension of the computing system. For example, I/O events can cause changes in the BASIC program flow. The interface to the operating system is achieved primarily in three ways—routine linkages, read/write memory use, and register use. An example of a routine linkage is the print driver—when the programmer specifies PRINTER IS 4, all PRINT commands cause a routine in read/write memory to be called. When the I/O ROM is in the system it replaces this read/write memory routine with a routine of its own to handle the PRINT. If the I/O ROM is not in the system the default system routine generates an error.

Read/write memory is used when the I/O ROM passes information to or from a system-defined location. For example, when an error occurs in an I/O ROM statement, the I/O ROM puts the error number in a common location that the system knows.

The register interface to the operating system consists of some CPU registers that the system reserves for system status and control. For example, register sixteen indicates the machine state—idle, running, and so forth.

There are several basic groups of interfacing and related capabilities provided by the HP-85 I/O ROM to the user.

- Formatted transfers—input and output of data with formatting and conversions
- Buffered transfers—fast handshake and interrupt data transfers through explicit buffering
- Register access—ability to set and interrogate the card configuration
- Interrupt access—ability to receive hardware interrupts
- Timeouts—ability to detect an inoperative card or peripheral
- Miscellaneous—keyboard masking, base conversion and binary operations
- HP-IB control—high-level access to HP-IB capabilities.

The ability to input and output data (*formatted transfers*) is probably the most commonly used of all interfacing capabilities. In some systems input and output are the only interface functions provided. The HP-85 allows for free-field or formatted input (via ENTER) and output (via OUTPUT). These statements are the tools that a programmer uses to get arbitrary string and numeric data into and out of the computer. Many times the peripheral has a non-ASCII character set, so a character conversion capability is provided to allow the programmer to set up a conversion table and deal with data as if it were normal ASCII data.

OUTPUT and ENTER are medium-performance data transfer mechanisms that consume the machine (program execution stops until the current statement is completed). *Buffered transfers* provide two additional methods to get data into and out of the HP-85—interrupt and fast handshake. The interrupt mode of data transfer is lower in performance than ENTER and OUTPUT, but does not tie up the computer, because it can be doing many other things in addition to the interrupt transfer. For example, while transferring data in the interrupt mode, the HP-85 can plot to the CRT, do arithmetic computation, print to the internal printer, or transfer data through other interrupt transfers. The fast handshake mode is much better in performance than ENTER and OUTPUT, but still ties up the computer. Neither of these buffered transfer modes performs any formatting. The data is placed into or taken out of an I/O buffer—hence the name "buffered" transfers. The buffer is a string variable that has been modified for use by the I/O ROM (via IOBUFFER). The buffered transfers are performed via the TRANSFER statement.

The programmer needs to access specific attributes of each interface card (*register access*). These attributes vary from card to card. For example, an RS-232 user would like to get at the clear-to-send bit and change its state; a sixteen-bit parallel interface user would like to change the logic sense from positive-true to negative-true logic. These features would become cumbersome if all of them were provided at the BASIC language level. Hundreds of keywords

*Parsing is the operation of taking a statement line and decomposing it into its elementary parts.

would be required. Instead, there can be up to sixteen card status registers and twenty-four card control registers on any interface card (accessed via STATUS and CONTROL). These registers allow the programmer access to the various low-level capabilities of the interface: parity, end-of-transmission character sequences, interface control and data lines, error indicators, and so on.

A programmer often needs to perform some operation when an interface condition occurs (*interrupt access*). Rather than constantly checking for these conditions, the interface card can check for the programmer (via ENABLE INTR), and when the condition happens, the end-of-line branch is taken to a BASIC service routine. The location of the service routine is set up with the ON INTR statement.

The *timeout* capability lets a programmer set an arbitrary length of time (from 1 to 32767 milliseconds) to wait for a response from the I/O hardware. Once a timeout has occurred, there is an end-of-line branch taken to a BASIC service routine specified by the programmer.

The *miscellaneous* statements and functions do not perform any interfacing capabilities. They are included to make the job of the interface programmer easier. There is the capability to mask out sections of the keyboard so that the operator does not inadvertently disrupt program flow. Base conversions for decimal numbers to and from binary, octal, and hexadecimal allow the programmer to display interfacing information in a convenient form. Boolean functions (and, or, exclusive or, complement, bit test) let the programmer test and modify interfacing information easily.

HP-IB is a mnemonic for Hewlett-Packard's implementation of IEEE Standard 488-1978. Since HP-IB is such a pervasive interfacing standard for instrumentation, printers, plotters and peripherals of all descriptions, there are several high-level HP-IB control statements in the HP-85 I/O ROM. There are many commands that send multiline messages on the HP-IB (such as TRIGGER, PASS CONTROL,

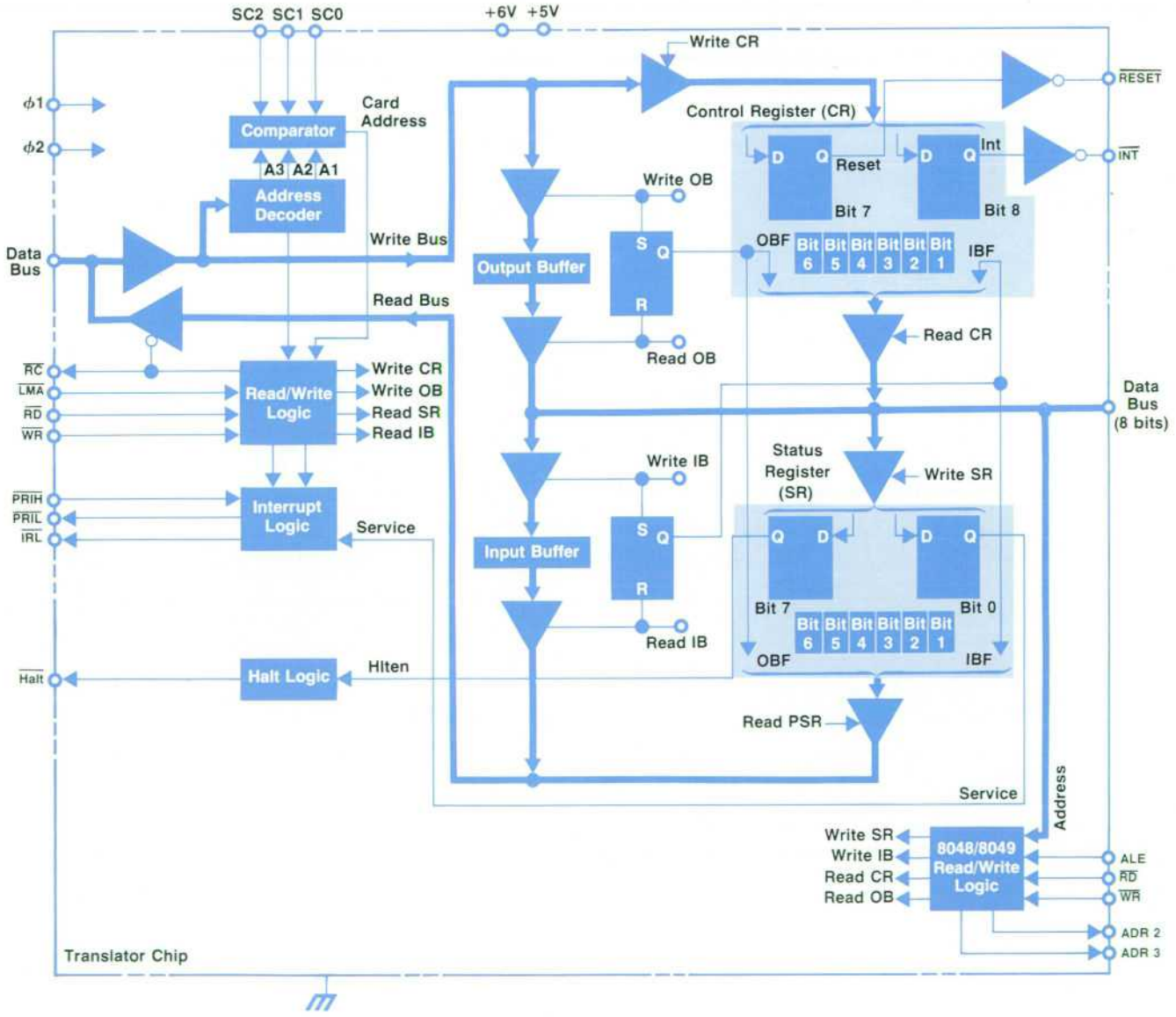

Fig. 3. *Block diagram of the translator chip that translates timing and protocol requirements between the HP-85 CPU and the interface card I/O processor.*

CLEAR, etc.). Multiline messages are those commands that the controller handshakes to the various devices on the bus in bit-parallel, byte-serial fashion. There are also commands that send uni-line messages on the HP-IB (such as ABORTIO, LOCAL, REQUEST, etc.), and there are functions that return status information (PPOLL and SPOLL). The other cards (Serial I/O, GPIO, BCD) use some of these statements for their own functions when appropriate.

Design of the 82937A HP-IB Interface

To implement all the capabilities planned for the I/O ROM, it was necessary to design intelligent I/O cards. Each I/O card contains a single-chip microcomputer to (1) communicate with the HP-85 CPU and (2) control and respond to the signals on the interface. The result of this is highly significant—the I/O ROM is not required to know what type of interface is being accessed. Communication between the I/O ROM and an I/O card is totally independent of the type of card; it is the card's responsibility to perform the appropriate I/O function over the interface.

This enhanced capability of the I/O cards permits more capability to be put in the I/O ROM than would have otherwise been possible. Also, this design approach permits overlapped processing; that is, the HP-85 CPU can be running a BASIC program while an interface card is performing I/O operations. An excellent example of this design concept is the HP-85 interface to the HP-IB. The principal elements of the HP-IB card are shown in Fig. 3. They are:

1. Translator chip: The translator chip interfaces between the HP-85 CPU and the I/O processor on the card. It provides electrical and timing compatibility between the buses and contains several registers to implement the communications protocol between the CPU and the I/O processor.
2. I/O processor: The I/O processor is a single-chip microcomputer containing 2048 bytes of program ROM and 128 bytes of RAM. It performs two tasks: it communicates with the CPU through the translator chip to determine what I/O operation is desired, and it implements the I/O operation over the bus using the bus transceivers.
3. Bus transceivers and control logic: Bidirectional transceivers are used by the I/O processor to control and to respond to signals on the bus. A latch, written to by the I/O processor, controls the direction of the transceivers and also controls I/O processor interrupts.

The translator chip (Fig. 3), hereafter called the TC, was designed to achieve several goals:

- Support eight peripheral select codes
- Let the HP-85 processor interrupt the I/O processor
- Let the I/O processor interrupt the HP-85 CPU
- Let the HP-85 CPU do a hardware reset of the I/O processor
- Provide a means for the I/O processor to halt the HP-85 CPU
- Provide general-purpose data registers for bidirectional communications between the two processors.

I/O for the HP-85 is memory-mapped. To support eight select codes, the TC's address is switch-settable to one of eight different addresses. Three switches reside in the I/O card. In the process of setting the card's select code, the user is actually setting the card's address. A mapping is done in the I/O ROM to translate from the select code specified in the program to the appropriate address.

Each TC actually occupies a pair of addresses. The lower address is used to access the control register (write-only by the HP-85 CPU and read-only by the I/O processor) and the status register (read-only by the HP-85 CPU and write-only by the I/O processor). The upper address of the pair accesses the output buffer (write-only by the HP-85 CPU and read-only by the I/O processor) and the input buffer (read-only by the HP-85 CPU and write-only by the I/O processor).

The input and output buffers are used for general-purpose communications between the two processors. Bits in the control and status registers are used to qualify data in these buffers as well as report the status of various events. To synchronize the flow of data, each processor can ascertain the condition of these buffers via flags in the status and control registers. These flags are OBF (output buffer full) and IBF (input buffer full). OBF is set when the HP-85 CPU writes to the output buffer and is cleared when the I/O processor reads the output buffer. Similarly, IBF is set when the I/O processor writes to the input buffer and is cleared when the CPU reads the input buffer.

The I/O processor and HP-85 CPU exist in a master-slave relationship, with the CPU being the master. The CPU sends instructions and data to the I/O processor via the output buffer. The software protocol between the CPU and the I/O processor defines a bit in the control register as COM, for command. COM is set high by the CPU before writing a command into the output buffer; COM is set low by the CPU before writing data into the output buffer. While the I/O processor is acting on the data or command, it sets a bit in the status register to 1, which is defined as the BUSY bit. When the I/O processor finishes acting upon the command or data byte in the output buffer, it sets BUSY low, which tells the CPU that it is done.

Several commands received from the CPU require that

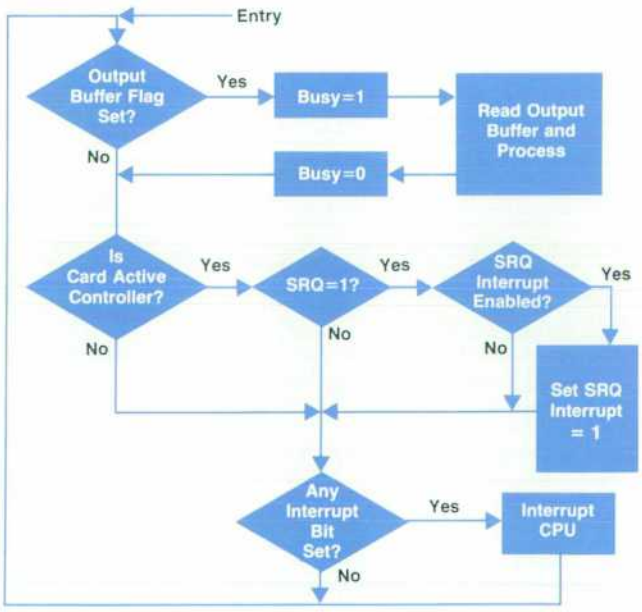

Fig. 4. *Idle loop for the 82937A HP-IB Interface card I/O processor. This loop can be exited at any point upon an interrupt from the CPU or interface bus.*

the I/O processor return data to the CPU (e.g., the command generated by the CPU to execute the user's STATUS statement). The I/O processor returns the data in the input buffer. The CPU monitors IBF to determine when the data is written to the input buffer, while the I/O processor monitors IBF to determine when the data has been read by the CPU.

Associated with each I/O ROM statement that accesses I/O is a set of rules that define the communications protocol between the HP-85 CPU and the I/O processor. The communications protocol not only defines the interaction discussed above, but also covers the interrupt protocol whereby each processor may interrupt the other.

The HP-85 CPU communicates with the I/O processor primarily to convey bus control commands to the I/O processor. The I/O processor then controls the bus as dictated by the CPU and within the bounds specified by IEEE Standard 488-1978. At power-on, the I/O processor reads the five address switches and the system controller switch located on the interface card. These switches are set by the user to configure the I/O processor's HP-IB address and to designate whether or not the I/O processor is to assume the role of system controller at power-on. The user can verify the switch settings by reading the STATUS register.

Fig. 4 shows the idle loop executed by the I/O processor, demonstrating its interaction with the CPU and the bus. In its idle loop, the I/O processor monitors OBF to see if the CPU has written any new information into it; if so, the I/O processor sets BUSY = 1, reads the output buffer and processes it as a command (COM = 1) or data (COM = 0).

If the I/O processor is the active controller of the HP-IB, it polls SRQ. If SRQ is true and the user has enabled an end-of-line interrupt branch on SRQ, then the SRQ end-of-line interrupt bit is set. The I/O processor subsequently examines all eight interrupt cause bits; if any are set, the I/O processor interrupts the CPU. While the SRQ interrupt cause bit is set as the result of polling, the other bits are set as the result of interrupts from the bus as discussed below.

Interrupts of the I/O processor originate from two sources—the CPU (via the TC), and the bus. The CPU interrupts for certain operations, such as STATUS, to guarantee timely operation regardless of the state of the I/O processor. For example, if the I/O processor is busy handshaking data on the bus to a device that is taking an indefinite length of time, then an interrupting STATUS operation guarantees an immediate return of data.

Certain HP-IB signals mandate a response within a time limit. For example, IFC (interface clear) must be responded to within 100 microseconds. To guarantee this, IFC can be enabled to interrupt the I/O processor. Likewise, REN (remote enable) must also be responded to within 100 microseconds, and thus can be used to interrupt the I/O processor. ATN (attention) imposes no timing requirements on the I/O processor but is used as an interrupt input to ensure that ongoing I/O is properly suspended. During an IFC or ATN interrupt, end-of-line interrupt bits may be set, depending upon the interrupt enable mask provided by the user. The interrupt enable bits are examined back in the idle loop; if any are set, the I/O processor interrupts the CPU. The CPU then typically executes the branch specified in the user's BASIC program.

Interface Select Codes and Device Specifiers

The HP-85 can have up to three interface cards plugged into it (see Fig. 5). It also has the internal CRT and printer that I/O ROM users may want to access. How does a programmer tell the I/O ROM which of these devices to interface with? This is done through *interface select codes*.

Every I/O ROM statement that deals with an interface specifies as part of the statement the interface select code. The range of select codes is as follows:

```
 1  :  INTERNAL CRT
 2  :  INTERNAL PRINTER
 3  :  EXTERNAL I/O
    .
    .
10  :  EXTERNAL I/O
```

Select codes 1 and 2 are accessible only through OUTPUT. An example of interface select codes is:

```
10 OUTPUT  1;  "This line goes to the CRT"
20 OUTPUT  2;  "This line goes to the printer"
30 OUTPUT  7;  "This line goes to an external
                device"
```

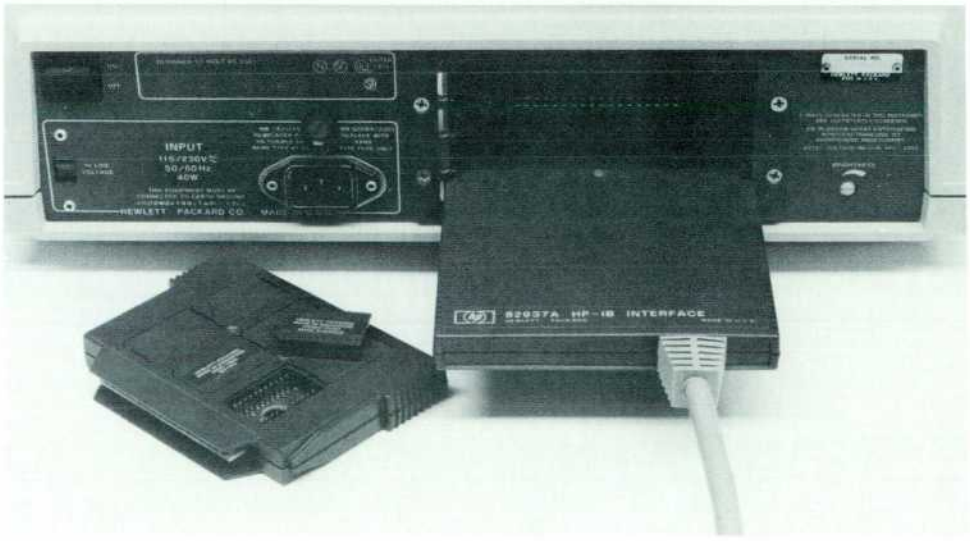

Fig. 5. *Typical HP-85 interface installation. The add-on ROM drawer and the 82937A HP-IB Interface are shown. The unused slots can be used to add two other interfaces or another interface and the 16K RAM module.*

Using HP-85 I/O Capabilities

The HP-85 I/O ROM allows for extreme ease of use. The following examples show a wide range of capabilities of the HP-85 in a simple interfacing application.

Let's assume that you, the user, have an HP-85, an I/O ROM, an HP-IB card, and a 3437A system voltmeter. Let's also assume that these components are hooked together, powered up, and working properly.

You would like to get a reading from the voltmeter. The 3437A's device address is 24 (as shipped from the factory) and the HP-IB interface card is select code 7 (as shipped from the factory). Therefore the address that you should be using is 724. You want to get a reading from the 3437A. To do that you have to execute the following statement:

```
ENTER 724;A
```

After you have executed this statement, the variable A will contain the reading from the voltmeter. That is all there is to do.

A program to get one reading and display it, would be like this:

```
20 ENTER 724;A
30 DISP A
50 END
```

Now suppose you want to take in a hundred readings and display them on the internal HP-85 CRT. The following program will do this:

```
10 FOR I=1 TO 100
20 ENTER 724;A
30 DISP A
40 NEXT I
50 END
```

Now suppose you want to plot these 100 readings. The following program does this:

```
1 SCALE 1,100, -20,20
2 GRAPH
3 GCLEAR
4 MOVE 1,0
10 FOR I=1 TO 100
20 ENTER 724;A
30 PLOT I,A
40 NEXT I
50 END
```

Now, say that this is not fast enough. Your application requires less time between voltmeter readings. First of all, you know that the 3437A is capable of much better performance. To get the HP-85 to take readings at a faster rate will require a type of data exchange known as a buffered transfer. This means that you tell the HP-85 to take the data into a buffer as fast as it can and then at a later time take the data out of the buffer and turn it into numeric data that the computer can use computationally. The following program does this fast transfer and plots the data as before.

```
1 SCALE 1,100,-20,20
2 GRAPH
3 GCLEAR
4 MOVE 1,0
5 DIM B$[709]
6 IOBUFFER B$
7 OUTPUT 724;"N100S"
8 ! THE PREVIOUS LINE CONFIGURES THE VOLTMETER
9 TRANSFER 724 TO B$ FHS;COUNT 701
10 FOR I=1 TO 100
20 ENTER B$ USING "#,K";A
30 PLOT I,A
40 NEXT I
50 END
```

-Tim Mikkelsen

The select code of an I/O card is set by switches mounted on the card. These switches are preset at the factory, depending on the type of the interface (e.g., HP-IB's select code is preset to 7). An example where the programmer might want to change the select code would be when there are two HP-IB interfaces in one HP-85. The programmer might do this to extend the number of HP-IB devices on one computer, or if the computer is to be a controller on one HP-IB and just another device on the other. In such a case, the programmer would have to change one of the select codes to prevent a hardware conflict.

This, however, is not enough. The HP-IB standard allows for up to 31 device addresses on a single interface (14 is the physical limit that any HP-IB interface can support). The 16-bit general-purpose interface allows the programmer access to four eight-bit ports in various ways (using a total of sixteen logical addresses). The BCD interface allows for one or two channels of information consisting of data fields, function codes, and error indications which are accessible in various combinations (using a total of seven logical addresses). How does the programmer tell the I/O ROM which device address to talk to? This is done through *device specifiers.*

Tim I. Mikkelsen

Tim Mikkelsen came to HP in 1977 after completing BS and MS degrees in computer science at Iowa State University in 1975 and 1977, respectively. Currently a product marketing engineer, he has worked on I/O ROMs for the HP-85 and the 9835 and 9845 Computers. From Missouri Valley, Iowa, Tim is a member of IEEE, and now lives in Fort Collins, Colorado. He is married with one daughter. When not involved with woodworking, downhill skiing, personal computing, and photography, Tim is busy trying to restore a 1964 TR4 sportscar.

John H. Nairn

Born in Pittsburgh, Kansas, John Nairn went to school in Denver, Colorado, earning a BS in chemistry at Regis College in 1967. He then received the MS and PhD degrees in mathematical physics from Rice University, Houston, Texas in 1971. After working as a regional analyst John joined HP in 1972 and has had various responsibilities since then. Initially he worked in marketing on the 9821 and 9830 Calculators and then did R&D work for the 9825 I/O. In 1976, John returned to marketing as interfacing product manager. Later, in 1978, he moved to his current position as R&D project manager for the HP-85 I/O. John is a member of ACM, a co-inventor for a patent related to the 9825 Calculator, and author of a regular series of articles for HP's Keyboard magazine. He lives in Fort Collins, Colorado and is interested in photography, hiking, recreational mathematics and folk guitar.

David J. Sweetser

Born in Woodland, California, Dave Sweetser attended Harvey Mudd College, where he earned BS and MS degrees in engineering in 1971 and 1972, respectively. After five years with an aerospace company, Dave joined HP in 1977 and has designed an interface for the HP-85 in addition to writing I/O software. He and his wife, who is an engineer at HP's Loveland Instrument Division, designed their own house, which is presently being built on 3 acres of land between Loveland and Fort Collins, Colorado. Dave enjoys rafting, bicycling, backpacking, cross-country skiing and snow camping.

The device specifier is like an interface select code, except that it includes both the select code and the device address. Notice that the range of device addresses is from 00 to 31. To build a device specifier—multiply the select code by 100 and add the device address. Hence select code 7, device 24 would be $7 \times 100 + 24 = 724$. Device addresses are generally preset by the factory, but can be changed in a manner similar to changing the interface select code.

References

1. S.L. Chumbley, "Extending Possibilities in Desktop Computing," Hewlett-Packard Journal, May 1979.
2. W.D. Eads and J.M. Walden, "A Highly Integrated Desktop Computer System," Hewlett-Packard Journal, April 1978.
3. D.E. Morris, C.J. Christopher, G.W. Chance, and D.B. Barney, "Third Generation Programmable Calculator Has Computer-Like Capabilities," Hewlett-Packard Journal, June 1976.

A Compact Tape Transport Subassembly Designed for Reliability and Low Cost

by Douglas J. Collins and Brian G. Spreadbury

OVER THE LAST FOUR YEARS Hewlett-Packard has developed a series of magnetic transports[1,2,3] for the 98200A data cartridge shown in Fig. 1. The use of this small data cartridge was selected because of proven reliability, large data capacity, ease of use, and low system cost.

The tape system integral to the HP-85 is in many ways a refinement of previous designs. However, the total integration of the transport with a single printed circuit board into one small package called for new techniques in electrical and mechanical design.

Transport Electronics Design

To incorporate a tape system into the compact HP-85 package required the cartridge transport to be small and to consume a minimum of power. Unfortunately, a relatively powerful motor is needed to drive the tape. For best efficiency, a pulse-width-modulated 20-kHz signal controls the motor drive transistors to keep them either cut off or fully saturated (turned on). These large power pulses can generate severe noise levels on the ten-millivolt signal lines from the tape head. To prevent this, the entire read/write circuitry is located within three centimetres of the magnetic head. Also, the motor drive circuitry has a separate power supply and is independently grounded.

The heart of the tape system electronics is the custom NMOS tape controller IC. It performs the tasks of interfacing to the HP-85 CPU, controlling motor speed and direction, and encoding/decoding data for the tape. Two Schmitt-trigger inputs provide direct sensing of the analog signals

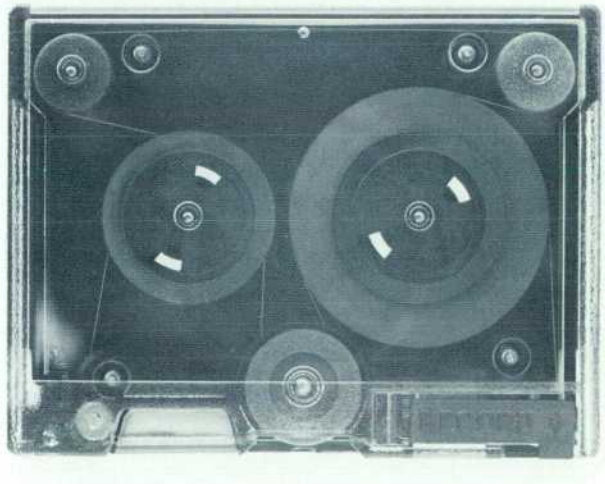

Fig. 1. *The 98200A mini data cartridge is a compact storage medium for data and programs, and has proven reliability and flexibility.*