APRIL 1978

HEWLETT-PACKARD JOURNAL

A Highly Integrated Desktop Computer System

System 45, the new flagship of the HP 9800 Series, gives the user unprecedented power in a single compact unit. It offers advanced capabilities in program documentation, string and matrix operations, subprograms, program linking, tracing, formatted output, mass storage, and graphics.

by William D. Eads and Jack M. Walden

SYSTEM 45 IS A HIGHLY INTERACTIVE, highly integrated personalized desktop computer. It is designed to give the user unprecedented capabilities in input/output, computation, and storage, all in a single unit on the desk top.

System 45, which is also known as Model 9845A (Fig. 1), has the most powerful central processor and the largest built-in mass storage system ever offered in a desktop computer. It also features a 12-inch CRT display, BASIC interpretive language conforming to the latest ANSI standard, extensive applications software, and an optional graphics package with high-speed hard-copy output.

Design Philosophy

Design of the System 45 Desktop Computer was heavily influenced by interviews with many typical users of desktop computer systems, particularly the BASIC-language HP 9830A. Two characteristics were common to many users. One was the complexity of their applications, which included planetary motion modeling, life science analysis, administrative functions, and computer-aided design. The second common characteristic was the desire for a friendly, easy-to-learn, easy-to-use system. Based on these and related inputs, the decision was made to develop a totally integrated desktop computer system including not only the most popular peripherals, but also the entire operating system and user read/write memory.

The implementation of this fundamental decision did not mean simply integrating a collection of existing peripherals into a box, independent of relative performance or form factors. Instead the approach was to develop and integrate an operating system and a set of peripherals that would be balanced in performance. For example, the single-line display of past products was replaced by a multi-line CRT that incorporates the powerful feature of graphics as an option. The printer of past products was upgraded considerably to a high-speed, page-width printer/plotter that produces hard copies of anything on the CRT, including both alphanumerics and graphics. The integral thermal printer/plotter is optional. System 45 incorporates two high-performance tape cartridge units (one is optional), doubling the built-in mass

Cover: *Model 9845A Desktop Computer, also called System 45, incorporates several often-needed peripheral devices, including a CRT with both alphanumeric and graphics capability, mass storage in the form of two cartridge tape units (one optional), and an optional page-width printer that has graphics capability. Four I/O slots are provided for external peripherals and the HP Interface Bus.*

In this Issue:

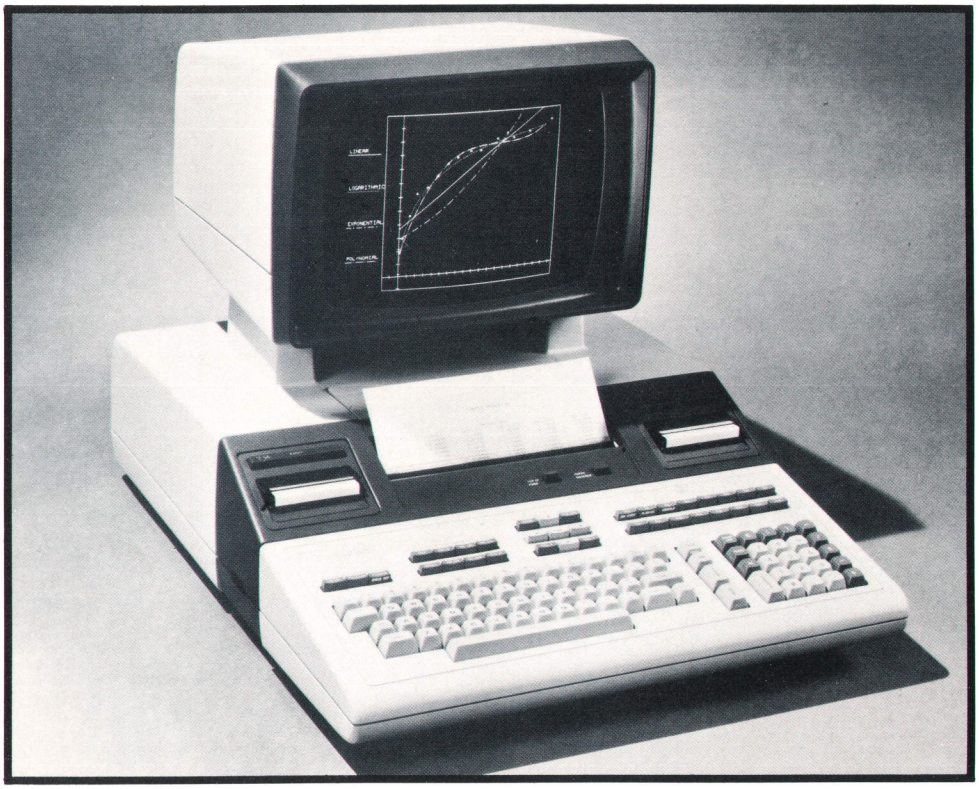


Fig. 1. *The new HP Series 9800 System 45 features the most powerful central processor and the largest built-in mass memory now available in a desktop computer. It also has a 12-inch CRT display, enhanced BASIC language, an optional graphics package with high-speed hard-copy printing, and applications software.*

memory capability of past products. The enhanced operating system of up to 150K bytes of control, language, and options is built into read-only memory (ROM) located in two user-accessible drawers. System 45 also integrates up to 64K bytes of read/write memory, along with two main system processors to optimize system operation and computation. The integral keyboard was upgraded from past products to improve its operation, efficiency, and reliability. Four input/output ports are provided for external peripherals such as disc memories, impact printers, ink plotters, paper tape punches, and so on.

The high degree of integration was made possible by the large-scale integration of some of the key components. System 45 with all of its options includes 36 NMOS LSI chips, 19 MSI chips and 75 NMOS ROM chips,* all custom developed and processed in HP's integrated-circuit facilities.[1] further integration was made possible by high packaging densities, a high degree of modularity in several assemblies, a very efficient switching power supply, and a novel dual fan parallel cooling scheme.

In the pages that follow, System 45's design is described in three parts: firmware and software in this article, hardware in the article on page 11, and the thermal printer in the article on page 22.

*NMOS=N-channel metal-oxide semiconductor integrated-circuit process (also see reference 1).
LSI=Large-scale integrated circuits.
MSI=Medium-scale integrated circuits.

Firmware/Software Objectives

System 45's personality is largely a result of the contents of the system programs that control its two major processors. These are firmware programs residing in read-only memory. The goals in the design of the firmware system were oriented heavily toward two features: straighforward, self-evident, easily-used operations and language statements for the novice or nonprogrammer, and sophisticated and powerful operations and language statements available to the advanced user who must solve complex problems. Both these capabilities had to be achieved without complex system configuration procedures or an elaborate job control language.

System 45's programming language is an extension of the new standard BASIC defined by the American National Standards Institute (ANSI). Statements such as LET, INPUT, and GOSUB are a part of this standard. Enhancements to this language have been made in the areas of program documentation, string and matrix operations, subprograms, program linking, tracing, formatted output, mass storage files, and graphics.

Display Functions

The CRT display and the keyboard are the user's interactive link with System 45. They operate together almost as a single entity, with most keyboard operations immediately reflected in some way on the display.

The display operates in several modes, each presenting a particular collection of information that is

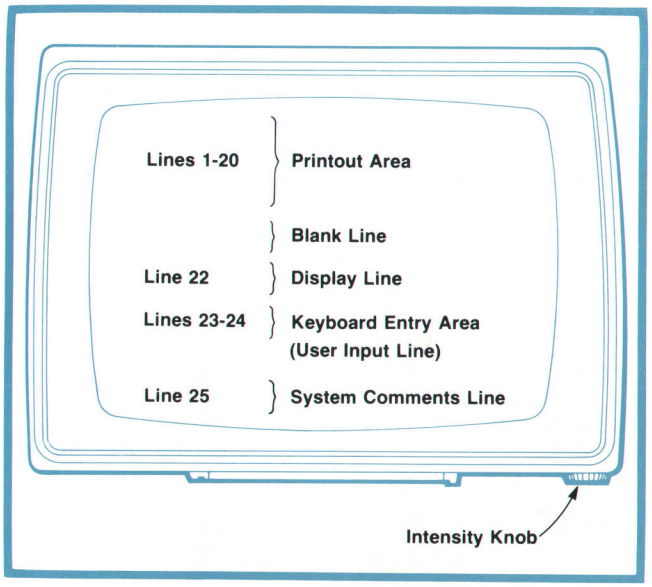


Fig. 2. *Most keyboard operations are immediately reflected in the display. In the normal mode, the display is divided into a printout area for program-generated information and an interaction area that handles communications between the user and the computer.*

pertinent and useful for the task being performed. The normal mode, in which the display operates most of the time, provides a print area for user-program-generated information and an interaction area that handles the user's communication with the computer (Fig. 2). The print area of the display consists of the top 20 lines. Display information for this area is generated by user-program statements PRINT and PRINT USING. The buffer for the display can store 3552 active characters. If more than 20 lines of output are generated, they remain in the buffer and the user may scroll up and down through them. Up to 350 short lines may be viewable.

The bottom area of the display is the user interaction area. It has three functions. The first is the presentation of messages to the operator under user program control. This occurs upon execution of the DISP statement, or as a prompt as part of the INPUT, LINPUT, or EDIT<string name> statements, which call for operator-supplied information from the keyboard.

The second function is the user input line. All user entries (data, commands, etc.) immediately appear here. A cursor is present in this double-length, 160-character line, and full editing capabilities—insert character, delete character, and cursor movement—are provided.

What happens to information keyed into this line is determined by the system control key pressed after keying in the line. If the EXECUTE key is pressed, the system treats the line as a command to be executed. This includes the evaluation of expressions, as in a keyboard controlled calculator, which can be done even though the computer is running a program.

If the STORE key is pressed, the line in the display is treated as a program line to be checked for syntax and stored as a program line in the computer.

If the computer has paused for user input of data via an INPUT, LINPUT, or EDIT<string name> statement, the user supplies the data into this line and then presses the CONT key.

The third function of the bottom area of the display is the presentation of system-generated information to the user. Error messages and information about the state of the system (busy, etc.) appear here. If the computer is being used as a calculator, the results are presented in this line of the display.

Two other modes of the display can be invoked by the user to enhance certain operations. They are the EDIT LINE mode, which is tailored for the entry and editing of programs, and the EDIT KEY mode, which is used for defining and editing the special function keys (to be described in more detail later). In these two modes, the user input lines and the system status line remain functionally the same, but are moved up to the middle of the screen.

Keyboard Functions

The keyboard is divided into functional key groups: display control, system control, special function keys, typewriter section, number pad and associated arithmetic operator keys, and some miscellaneous functions (Fig. 3). The display control keys provide for cursor movement, scrolling of the display, insertion and deletion of characters, and insertion and deletion of program lines in the EDIT LINE mode. The system control keys include STOP, RUN, PAUSE and CONT.

The typewriter section conforms to regular typewriter conventions. Included in this area is the STORE key. The number pad, which is used for calculator operations and the entry of numeric data, includes the numeric keys, the arithmetic operators and the EXECUTE key. The miscellaneous group is located in the typewriter section, and controls the setting and clearing of tabs in the input line.

The special function keys are a group of 16 user-definable keys, each of which may be shifted, providing a total of 32 user-specified special operations. There are several uses for these keys. One is to invoke the equivalent of up to an entire line of keyed input by pressing a single key. Almost any key can be included in this line, even editing and control keys. The second use of the special function keys is to provide a set of frequently used commands as single keystrokes. These are defined by the system when power is turned on and are labeled on the keyboard.

The third use of the special function keys is to provide the user with direct (interrupt) control of the

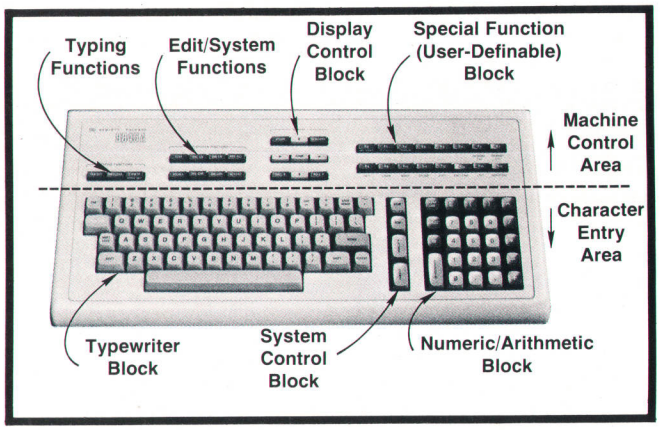


Fig. 3. *The keyboard is divided into functional groups. N-key rollover assures that no keystrokes will be missed regardless of how fast a typist the user happens to be.*

computer while a program is running. The action to be taken is incorporated as part of the program by an ON KEY # statement. When the specified key is pressed, the program is interrupted and the specified routine is executed. This allows the user to alter program operation by direct keyboard control.

A special feature of the keyboard is a RECALL operation. Each entry made and invoked by the user is pushed onto a recall stack. The entries on the stack may be recalled successively by pressing the RECALL key.

The design of the keyboard assumes that System 45 operators will have a wide range of typing skills, from novice to experienced typist. The key switches in the typewriter block were chosen to give essentially the same stroke and touch as on conventional electric typewriters. The keyboard design also features N-key rollover. When an experienced typist encounters a word such as "the", the typing rate can be as high as 250 words per minute. N-key rollover assures that no characters will be missed during these burst modes of character entry.

Mass Storage System

The mass storage system controls all mass storage devices connected to or built into System 45. In the basic machine this means the built-in tape cartridge. However, the same functional structure (statements and commands) is used to control optional external mass storage devices such as the HP 9885A Flexible Disc. Advantages of this unified command structure are two. For one, the operation of a program or group of programs designed for one device can be transferred to other devices without change, and within a single program the operation and treatment of a variety of devices is the same. The advantage to the user is the simplification of program development or redevelopment when upgrading to higher-capability devices.

The mass storage system on all devices operates on a file-by-name basis, with each medium carrying its own directory of the files on that medium. File-by-name provides the greatest simplicity in file operations, and relieves the user of the tedious detail of management of the medium. The mass storage device associated with any file is specified as a part of the file name specification. If the user program does not provide this specification, the system automatically uses the default device, which is normally the internal tape cartridge, but can be defined by the user to be any device by the MASS STORAGE IS statement.

The mass storage system provides a variety of files, including special files for storing and loading of programs, storing and loading the special function keys, or storing and reloading the entire state of the machine (data, program, keys, display, etc.). A most important function is the storage and retrieval of user data by programs. The mass storage system provides a variety of ways to print and read data of all types. The data file may be treated as a single, sequential file in a serial mode. Or it may be accessed randomly by records of specifiable size. The random-access record size is user-specifiable, and is in no way limited by the physical record size of the hardware. The system totally isolates the user from the hardware.

Programs can be stored and loaded in two forms: internal (compiled) form, or source form. The internal form provides the greatest speed. The source form is written as a data file, with the program appearing in the file as a sequence of string data items, one string per line. These source program files can be treated as string data by other programs for a variety of purposes, such as modification, cross-reference generation, and so on.

A special file type is also provided for maximum-speed transfer of data arrays. In this mode, the data is transferred directly from the value area in memory via direct memory access (DMA).

PRINT USING and IMAGE

As a BASIC-language computer, System 45 provides simplified output capability via the PRINT statement. There is little control of output format, but maximum transfer of data with little effort or concern by the user.

For more sophisticated output applications in which the format or organization of the printed result is important, the PRINT USING and IMAGE statements are provided. The primary control of output by an IMAGE statement is character-by-character, as contrasted to the field specification provided by FORMAT statements in FORTRAN. This character-by-character specification allows more flexibility and control than can be achieved in field control.

The PRINT USING/IMAGE capability is flexible and

```
10   DIM A$[40]
20   PRINT USING 30;1537.25,10.7
30   IMAGE "MONEY FORMAT:",6X,2(" $"DDCDDD.DD,2X)
40   PRINT USING 50;1537.25,10.7
50   IMAGE "EUROPEAN FORMAT:",3X,2("DM"DDPDDDRDD,2X)
60   PRINT USING 70:3.7,1732.58
70   IMAGE "CHECK PROTECTION:  ",2("$"***C***.DD,2X)
80   PRINT USING 90;25,1236
90   IMAGE "LEADING ZEROS:",6X,2(10Z,3X)
100  A$=CHR$(34)&"DYNAMIC FORMAT:"&CHR$(34)&",5X,7D.2DM,2X,7D.2DS"
110  PRINT USING A$;-1,1342.4
120  END


MONEY FORMAT:          $1,537.25         $10.70
EUROPEAN FORMAT:     DM1.537,25       DM10,70
CHECK PROTEOTION:  $*******3.70   $***1,732.58
LEADING ZEROS:       0000000025    0000001236
DYNAMIC FORMAT:             1.00-      1342.40+
```

Fig. 4. *The* PRINT USING *and* IMAGE *statements provide sophisticated output formatting. The optional thermal printer can provide hard copies of anything on the display.)*

versatile. Numeric quantities can be output in standard fixed (F) or floating-point (E, scientific) notation. Fixed-width fields with leading zeros or asterisk fill can be specified. The printing of large numbers separated into groups of three digits by commas can also be specified. Outside the United States, the comma is often used to indicate the radix point instead of the period that is common in the U.S. This can be specified using the IMAGE capability as simply as the use of a decimal point.

The IMAGE capability also allows the specification of floating literals as a part of numeric notation. For example, it is common to have the sign of a number (+ or −) or a dollar sign ($) float, in the sense that leading zeros are not printed and the symbol floats to the first position to the left of the first significant digit. In System 45 the floating literal is generalized to be any set of characters. For example, this would allow DM to be floated in printing money values in Germany.

An IMAGE specification can be provided by a separate IMAGE statement or specified by a string expression. The IMAGE statement allows fixed, unchanging definition of the format at the time the program is written. The string expression specification allows the full use of string manipulation to construct a format at run time, if fixed formatting is not acceptable.

Fig. 4 shows an example of the use of the PRINT USING and IMAGE statements.

Graphics

System 45's optional graphics capability conforms to an HP standard language for plotting and graphics. This language provides a common functional structure for a variety of HP plotting devices. In the System 45 graphics option, provision for control of a number of devices is made, including the CRT graphics option, the HP 9872A X-Y Plotter,[2] incremental plotters, and graphics output on the internal thermal printer.

The language and command structure in the graphics option provides a high degree of flexibility and device independence. For example, it is simple to generate identical plots on the CRT, on the internal thermal printer, and on a 9872A Plotter. The same commands control all devices, and a program that generates a plot on the CRT can create that same plot on the 9872A by changing a single statement, the PLOTTER IS command, to specify which plotting device is active.

A major objective of the HP plotting language is to provide the ability to use simple programs for simple jobs, along with maximum capability and flexibility at the expense of program complexity when that is required. This is accomplished by the careful definition of default conditions that are logical and straightforward in the sense that they are what the user would expect them to be.

Three ways to specify plotting units are provided: GDUs, user units, or absolute physical units. GDUs are, in effect, percentages. One GDU is defined as 1/100th of the minimum side (X or Y) of the available plotter space. User units are specified by the user to fit the application—for example, seconds and feet or microseconds and volts. Physical unit specifications are in millimeters, and are used for fixed plots in drafting, mapping, and similar applications.

A full repertoire of plotting commands is provided, including absolute PLOT, incremental PLOT, MOVE, and DRAW. These operate in any of the three unit systems. Relative and incremental PLOT commands can include rotation specifications.

The physical limits of whatever plotter is active define hard clip limits, that is, the area outside of which plotting can never occur. The user can redefine these limits to specify a sub-area of the total physical space. This is useful, for example, in plotting on a small sheet of paper on the 9872A Plotter. The limits of this hard clip area can be specified by digitizing the corners of the area. In addition to hard clip limits, the

user frequently wants to specify a sub-area of the total plotter space onto which user units will be mapped, and which provides a soft clip area beyond which normal plotting will not occur. Typically, this might be a pre-ruled grid on a graph sheet placed on the 9872A Plotter bed. This soft clip area can be specified by digitizing the corners of the area. The soft clip region can be moved about as desired by the user. In simple cases, when the total plotter area is to be used, no specification need be made. This defaults the hard and soft clip areas and the user unit mapping to the total plotter space.

Besides normal user unit scaling, which allows full flexibility, System 45 provides a special form of scaling that results in isotropic plotting. In this mode, plotting units are the same size on both the X and Y axes. Thus geometric integrity is preserved, that is, a circle remains a circle even though its size may be changed. Commands to generate axes or grids, with major and minor ticks, are also provided.

Full labeling capability is provided through LABEL and LABEL USING statements. These are analogous to PRINT and PRINT USING, generating the same form of output. Label direction and character size can be specified by the user.

The internal thermal printer can be used to obtain rapid hard copy of an image present in the CRT graphics memory. This is a dot-for-dot "dump", and takes from five to fifteen seconds. Such hard copy is useful for debugging, notebooks, quick copies, and permanent records.

Enhanced BASIC

To aid in program readability and maintainability, several features have been incorporated into System 45 BASIC, the most important of which is the definition of names within programs. ANSI BASIC allows only a single upper-case letter followed by an optional digit for numeric variables and user-defined functions. System 45 allows an upper-case character followed by 0 to 14 lower-case letters, digits, and/or underscores. Thus T may become Time_of_day or V7 could be mnemonically renamed V_base_emitter. Also, for ease of readability, lines may be labeled using this same naming definition. References to these lines, such as in GOTO statements, may use these names rather than the line number. Thus, one may write

```
100 GOTO Next_try
```

as a valid System 45 statement.

The IF statement, in addition to allowing a line number or name following THEN, also allows a single statement instead. Thus, to print the index and value of all array elements less than 0, one could write:

```
100 FOR Count = 1 to Limit      ! Loop to limit of Array
110     IF Array(Count)<0 THEN PRINT Count, Array(Count)
120 NEXT Count
```

In the above example, two additional documentation features are demonstrated. One is that comments can be added after any line by the inclusion of the ! character. This is listed in the same column position as it was typed. Also, statement keywords are listed in the same columns they were originally entered in. This allows the programmer to offset blocks of code lines to aid in the visual perception of algorithm flow.

Strings, both simple and array, are defined by the same naming conventions as numeric variables, followed by the symbol $; examples are Name$ and Line_of_data$. Strings may contain as many as 32,767 characters, as defined in the DIM statement. String arrays and numeric arrays may be up to six-dimensional, with up to 32,767 elements in any dimension. Parts of strings may be accessed or replaced by using a bracket notation for substrings, and strings may be joined using the concatenation operator. For example, A$ = B$[3,5] & "xy" will create a string of five characters in the variable A$ by using the third through fifth characters of B$ and the literal xy.

Arrays of up to six dimensions are allowed in System 45. No limit, except available memory, is placed on the number of elements in the array. Array elements are identified by integer indices such as Array(1978,4). ANSI BASIC specifies that the lower bound of a lower array index is either 0 or 1, with 0 being the default value. The limits of these System 45 indexes, however, are not limited to 0 or 1, but may be specified by the user in the DIM statement. For example,

```
100 DIM Array1(1971:1980,4),Array2(-100,100)
```

specifies a 10 × 5 matrix and a vector of 201 elements. The programmer or user is thus allowed the convenience of working with array indexes that correspond to the application, such as a calendar year or a percent deviation, instead of less obvious transformed units beginning with 0 or 1.

Most BASIC implementations allow the multiplication of each element of an array by a scalar quantity. This capability has been extended to allow addition of a scalar quantity to each element of an array, subtraction of a scalar quantity from each element of an array, and division of each element of an array by a scalar quantity. For example,

```
MAT First = Second/(7)
```

divides each element of matrix Second by 7 and places it in the corresponding element of First. Relational operations, such as <,> =,<>, and =, are allowed between elements of matrices or between matrices and scalar values. For example,

```
MAT Unequal_a_b = A<>B
```

produces a 1 in each element of Unequal_a_b for which the corresponding elements of A and B are not equal, and 0 in elements of Unequal_a_b for which corresponding elements of A and B are identical.

Another matrix capability is that of functions of matrices. All single-operand numeric functions are allowed as operators in MAT statements. For example, to take the sine of each element of the matrix Angle and place the result in matrix Sin, the statement

```
MAT Sin = SIN(Angle)
```

is used, with a significant speed advantage over the use of FOR/NEXT loops to compute the sine of each individual element.

System 45 BASIC allows the use of subprograms, much like the FORTRAN subroutine subprogram capability, within which variables are handled independently from variables of the same name in other subprograms or the main program.

The capability of subprograms has two important benefits. First, the fact that subprograms are independent of one another means that libraries of subprograms can be used to build large software systems out of many smaller debugged modules. Similarly, a large programming effort can be broken down into several smaller tasks, and independent subprogram modules can be designed by several programmers with a minimum of interaction and interference. The second benefit of subprograms results from the strategy of allocating user read/write memory to each subprogram as execution of that subprogram begins and deallocating that memory, or returning it to the pool of system available memory, when the subprogram completes execution. This means that the size and shape of arrays can be determined at the time the program is run, instead of having to be specified when the program is written. The following program fragment demonstrates the ability of a program to fit in available memory, as long as arrays Admittance, Values, Amps, and Volts do not exceed available user read/write memory.

```
10   INPUT "How many nodes and elements in your circuit?", Nodes, Elements
20   CALL Input_circuit (Nodes, Elements)
 .
 .
1000 SUB Input_circuit (N, Elts)
1010 DIM Admit_matrix (1:N,1:N), Values (1:Elts)
1020 DIM Amps(1:Elts), Volts(1:N)
 .
 .
 .
```

As shown in this example, data may be passed to and results returned from subprograms by a list of variables, called the parameter list, which is enclosed in parentheses and follows the subprogram name. In addition to simple numeric variables, simple strings, numeric arrays, string arrays, and mass storage file numbers are allowed in the subprogram parameter list. Array names are distinguished by (*) trailing the name, while file numbers are identified by a leading # symbol. Numeric and string data may also be made available to the main program and all subprograms by use of a common memory area identified by the COM statement, which is similar in form to the DIM statement and must exist somewhere in each subprogram module that accesses the common data area.

Programs that are too large to fit entirely in System 45's memory may be broken into segments and manipulated by use of several mass storage statements. User programs may be placed on a mass storage file in ASCII character format with the SAVE statement. These ASCII files may overlay or add to programs in memory by using the LINK or GET statements. LINK leaves all variables intact when executed. GET deletes all variables except for those in common storage. The internal form of programs may be manipulated by similar statements named STORE and LOAD. These statements execute much more quickly than SAVE/LINK/GET, but the latter are more flexible since they deal with string data, which programs may manipulate.

Binary programs, supplied from the factory as mass storage files, may be put together in any combination required by the user with repeated use of the LOADBIN statement. Then the entire collection of binary programs in memory at any given time may be written to a file by use of the STOREBIN statement. Finally, an entire program, consisting of binary programs, BASIC main and subprograms, data, special function key definitions, and even the contents of the CRT screen, may be put on a mass storage file by use of the STOREALL statement. At a later time this file can be placed in memory, and on the screen, with the LOADALL statement. This feature is especially useful for checkpointing programs to minimize the cost of restarting long programs if power or other failure should occur during program execution.*

To aid in program debugging, several statements have been defined to help the user monitor program flow. For example,

```
TRACE 1000,1050
```

states that when program execution of line 1000 occurs, all program branches, such as GOTO, GOSUB, and CALL will be noted. This tracing of logic flow ceases when line 1050 is executed. Up to five variables, including numeric and string as well as simple and array variables, may be monitored for changes in values. The statement

```
TRACE VARIABLES A(*),B$
```

causes a message to be printed any time any element of array A changes or any time the string B$ changes value. Whenever possible, the new value is printed within one line of printed output. Tracing all variable changes is done with the statement TRACE ALL VARI-

*"Checkpointing" means storing the state of the machine at various points (checkpoints) in a program. Should a failure occur, the program can be restarted from the most recent checkpoint.

ABLES, while tracing of variables and program branching is done with the TRACE ALL statement. These facilities are turned off with the execution of the NORMAL statement.

Internal Program Storage

System 45's execution of user programs is interpretive, in the sense that the form of the program in read-write memory, the internal form, is not directly executable machine code, and the program can be listed essentially as the user typed it in. However, the internal form of a user's program is not at all similar to that of most interpreters, which retain the character form of the user's program and check operator precedences and variable names each time a statement is executed. Instead, the program is stored as pointers to the proper execution code for each language operation, such as LET, *, or COS, as pointers to a symbol table for variables and constants, and as pointers to a scratchpad location for temporary results. Fig. 5 shows the internal form for the line

100 LET Time = Old + N* Delta

Each box in the figure represents a two-byte word (16 bits). Enough information is retained to allow the line to be listed as typed (with extra spaces and redundant parentheses removed).

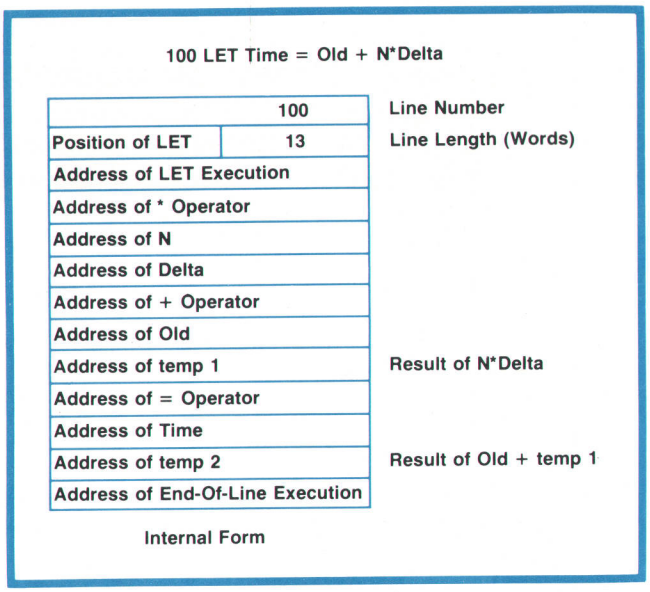


Fig. 5. *System 45's execution of user programs is interpretive, but the internal form of a program is different from that of most interpreters. Programs are stored as pointers to execution codes, symbol tables, or scratchpad memory. Shown here is the internal form for one program line. Each box represents a two-byte word.*

Operating System Details

As explained in the article beginning on page 11, System 45's architecture is based on two microprocessors, the language processor unit (LPU) and the peripheral processing unit (PPU). The PPU functions are master control over computer operations, control of input/output data transfers, and most formatting of data. The LPU is the language translator and is responsible for interpreting and executing program statements and keyboard commands. It is also responsible for managing program execution and initiating I/O operations by passing them to the PPU.

Communication between the LPU and PPU is accomplished by storing and reading information in a common area of read/write memory. Two general methods of communication are used, messages and flags. The messages are stored in two dedicated buffers in block 0 read/write memory. The sending processor stores a message in the buffer only when it is empty (flag is 0). The receiving processor looks for a message in its buffer by examining the flags associated with the buffer. It acknowledges receipt of a message by setting the associated flag to zero.

PPU Functional Description

The PPU is responsible for managing all system resources except for block 0 read/write memory, which is managed by the LPU. The resources managed by the PPU are block 1 read/write memory, I/O devices, and the LPU. The PPU performs all transfers of data and programs between the computer and I/O devices.

The PPU also establishes and controls the keyboard entry protocol. When the user makes a complete keyboard record entry the PPU disables the keyboard only until the record has been interpreted, whether or not execution of the record is complete. This allows concurrent command execution (see next paragraph). By waiting until the record is interpreted before enabling the keyboard, the PPU prevents additional entries before the computer is able to accept them. The PPU also allows keyboard entries to be made while a program is being executed. These entries are referred to as live keyboard entries.

Many device-related keyboard commands may be executed concurrently under control of the PPU. Concurrent execution does not require any action from the user, apart from entering the commands from the keyboard. Concurrent execution automatically results when a command is entered from the keyboard before the execution of any previously entered commands is complete. Commands that may execute concurrently are CAT, LIST, INITIALIZE, COPY, PRINT, PRINT #, READ #, DISP, Implied DISP, RUN, CONT, STEP, CREATE, MAT PRINT, MAT PRINT #, MAT READ #, PRINT USING, PURGE, REWIND.

PPU Process Definition

Except for PPU initialization and the process scheduler (idle loop), all PPU tasks are considered

processes. Except for the user process and the keyboard process, which are created during initialization, all processes are dynamically created and destroyed.

Each process that exists has at least one process control block (PCB) associated with it. A PCB is a 10-word read/write entity that contains, either directly or indirectly, all the information necessary for the PPU to execute the associated process.

A process may be in one of several states during its existence. The state of the process indicates whether the process is ready for the PPU to execute it, waiting for a resource or an event, or complete. If the process is in the hold state, it is waiting to reach the head of a queue or waiting for some resource. The state transitions that a process will undergo depend upon what kind of process it is and upon the circumstances of its execution.

A process will not be executed by the PPU unless it is at the head of a queue. One queue exists for each active peripheral address (1-12). Queues also exist for the optional built-in thermal printer, graphics, and other internal peripherals. There are also some system queues. Processes are queued, executed, and dequeued on a first-in-first-out basis. This ensures that I/O operations on a particular device that are generated during program line execution will be executed in the order that the program lines are executed by the LPU.

A device buffer is associated with each queue and is used by the process at the head of the queue. The buffer's primary use is for the transfer of data to/from the I/O device, but it also provides temporary storage for data internal to the process (pointers, etc.).

The process scheduler is the PPU idle loop. When there are no processes that require the PPU, the PPU executes only the process scheduler. In general, device transfers are carried out under interrupt control so that the interrupt service routine is the process executor.

The user process is the PPU executive. It is created during PPU initialization and always exists as an active process. All other PPU processes, except the keyboard process, are linked to it as descendant processes.

During program execution, all program statements executed requiring an I/O operation result in a message from the LPU to the PPU, so the PPU will execute the I/O operation.

When the RUN process receives one of these messages, it creates a process as a descendant and puts it on the bottom of the specified queue. It then resets the LPU-to-PPU communication flag to acknowledge receipt of the message. When this occurs, the LPU may continue program execution and possibly send more I/O operations to the PPU for execution. This is called overlapped mode, since execution of several I/O processes and program execution can overlap. In general, this provides greater execution speed, since several parts of program execution can occur simultaneously.

When the computer is not in the overlapped mode, it is in the serial mode. In serial mode, the execution of each statement of the program is completed by both the LPU and the PPU before execution of the next statement is begun. When the computer is initialized at power-on or by the SCRATCHA command, it is put in the serial mode. Whenever the computer is in serial mode, the user can change the mode to overlapped mode by executing the OVERLAP statement. Conversely, the mode of I/O statement execution can be changed from overlapped to serial by causing the SERIAL statement to be executed.

The LPU keeps track of the current execution mode, either serial or overlapped. When I/O statements are executed, this mode is examined. If it is OVERLAP, the LPU proceeds to the next program line. If the mode is SERIAL, the LPU waits for a message from the PPU that the I/O operation is complete, and then proceeds to the next line.

Acknowledgments

The firmware dealing with the syntax and execution of the various language statements was written by a group of six designers. John Bidwell developed the internal program form, Dave Landers wrote the string variable code, Jeff Eastman and Jeff Osborne were responsible for PPU/LPU communications and I/O statement execution, Jack Cooley developed subprogram and tracing capability, and John Schmidt designed the math and matrix packages. The coding for the PPU firmware was done by a group of four. Gene Burmeister did the mass storage system, Frank Cada was responsible for the I/O scheduler and process control, Ken Heun did the drivers for the tape cartridge and PRINT and PRINT USING, and Bob Jewett designed and implemented the display and keyboard procedures and the graphics option. The many ideas and contributions of these groups were responsible for a really high-performance system, and cannot be overemphasized. J. L. Marsh, of the disc-interface hardware group, programmed the drivers for the large external storage devices, and Leo Miller programmed the drivers for the incremental plotter interface. Brent DeWitt was responsible for the keyboard layout and electronics.

Special thanks go to Ed Olander and Chuck Near for their help in defining and managing the development of System 45, including all phases of hardware and software design.

References

1. J.E. DeWeese and T.R. Ligon, "An NMOS Process for High-Performance LSI Circuits," Hewlett-Packard Journal, November 1977.
2. L.G. Brunetti, "A New Family of Intelligent Multi-Color X-Y Plotters," Hewlett-Packard Journal, September 1977.

William D. Eads

Bill Eads received his BA and PhD degrees in electrical engineering from Rice University, completing his doctoral program there in 1970. He joined HP that same year, working in Santa Clara, California on design and management of a computer-assisted artwork system. He later joined the Loveland, Colorado Division contributing again in design and management, this time on NMOS II processor chips. One patent resulted from those efforts. His most recent responsibility is management of System 45 language development. Bill was born in Dumas, Texas, and he and his wife now live in Loveland and have a daughter and a son. Free time for Bill means biking, hiking, gardening, and cross country skiing. Sundays find him teaching Sunday School.

Jack M. Walden

As group leader and project manager for software development in HP's Loveland, Colorado, facility, Jack Walden applies a wealth of experience acquired from a varied and interesting background. After Jack received his BS degree in engineering physics at South Dakota School of Mines and Technology in 1944, he headed north to Alaska seeking adventure and fortune. He spent 15 years in our 50th state, designing, building, and operating radio and television stations. In 1960 he enrolled at Oklahoma State University and earned his MSEE and PhD degrees in 1962 and 1965. He spent the next few years at OSU as an associate professor of electrical engineering and computer science, teaching a variety of graduate and undergraduate courses, then joined HP in 1969 to work on the 9810A Desktop Computer. Jack has a wide range of interests, including music, computer programming, and the game of GO. He has a special interest in restoring organs and has a two-manual 13-rank Kimball theatre pipe organ in his basement. To satisfy his creative urge, he's building an electronic/computer switching system to control it, to replace the traditional electropneumatic switching. Jack lives in Loveland with his wife, Nancy. The Waldens have five children, two of whom are working at HP.

System 45 Hardware Design

by John C. Keith, Louis T. Schulte, and Ansel K. Vogen

THE KEY CONCEPT underlying the development of the System 45 hardware was to provide in one attractive and convenient package all the normal performance features that would be required to solve a typical problem. To achieve the same functional capability with previous-generation desktop computers would have required three separate instruments in addition to the computer: a CRT display terminal, an external tape memory or mass storage device, and a stand-alone printer. With all these features in an integrated physical package, the firmware can integrate them into a total system having capabilities that would be difficult if not impossible to achieve with separate instruments.

Hardware Organization

Fig. 1 is a block diagram of the hardware organization of System 45. The basic architecture is similar to that of previous 9800 Series Desktop Computers with the notable exception that System 45 has two processors. One processor is called the language processor unit (LPU). Its main responsibility is the execution of the user's BASIC-language program, which is stored in read/write memory. The second processor is called the peripheral processing unit (PPU). As its name implies, it is responsible for managing the internal and external peripherals. This processor also does all of the formatting of data transferred during I/O operations and is the master controller of the whole system, with the LPU counted among the resources that the PPU can call upon to complete a given task.

The processors used in System 45 are similar to the one used in the HP 9825A,[1,2] but the LSI chips have been slightly redesigned to include some new features. The most significant change was to increase the addressing range from 64K bytes to 128K bytes (32,768 words to 65,536 words). This was accomplished by increasing the length of all the regis-

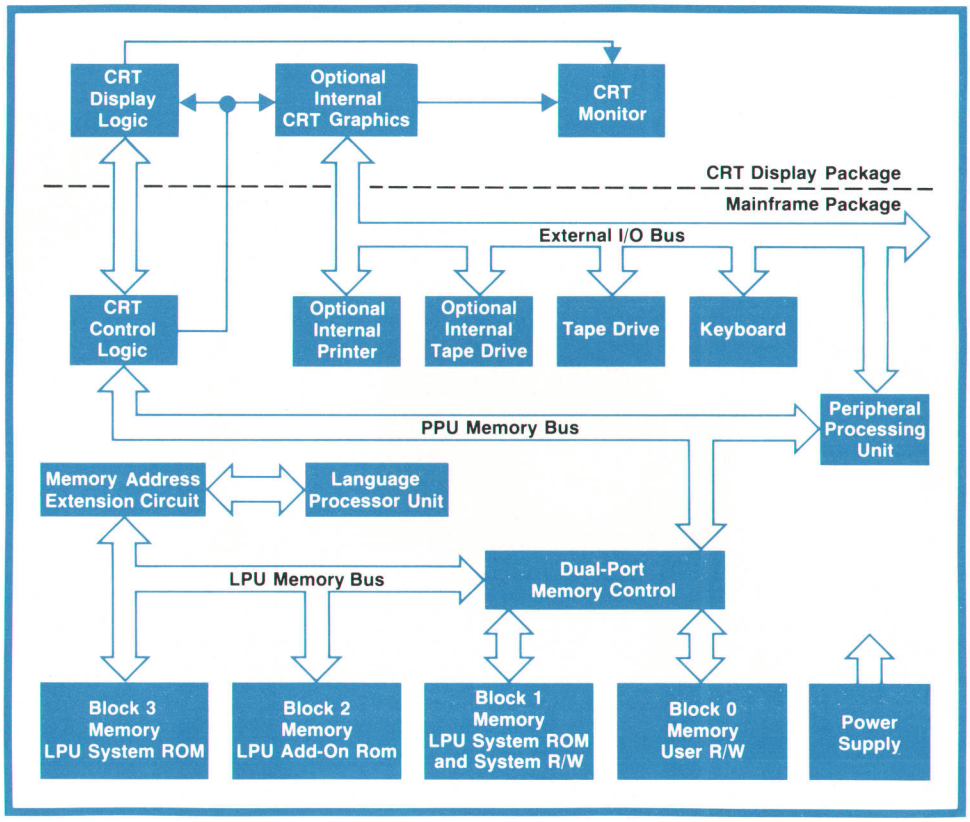


Fig. 1. *System 45 architecture is similar to that of previous Series 9800 Desktop Computers except that System 45 has two microprocessors, the language processor unit (LPU) and the peripheral processing unit (PPU). The PPU is the master controller of the system.*

ters associated with memory addressing (program counter, stack pointer, and so on) from 15 bits to 16 bits. This feature came at the expense of eliminating multilevel indirect memory cycles. Previously during an indirect memory cycle, 15 bits of a referenced memory location would be used to address the next location in memory while the 16th bit would be used to indicate whether the contents of that new location should be used as an operand or whether they should be used again as an address. This sequence could be carried on indefinitely until the 16th bit of one location indicated that the next memory cycle would access the desired final operand. With the new processor, all 16 bits are required to address a single memory location so there is no indicator available to tell the processor whether the next memory location is the final operand or another address. Thus the processor is forced to accept the first memory fetch of an indirect cycle as the final operand. This was not a great loss, because multilevel indirect cycles are rarely used, and in cases where they might be used, there are usually other techniques available to access the desired memory locations.

Another feature that was implemented on the revised binary processor chip (BPC) is a two-phase clock generator circuit that has enough capacity to meet the clock requirements of all three NMOS LSI chips in the processor. This circuit requires only a single-phase clock input at twice the desired frequency and produces the two-phase, nonoverlapping, high-voltage clock signals. Integrating this circuit on the BPC chip eliminated a large number of components that otherwise would have been necessary.

Memory Organization and Components

System 45 has 256K bytes of memory. This memory is partitioned into four blocks of 64K bytes each, as shown in Fig. 2.

Block 0 memory is exclusively reserved for read/write memory. This is the memory in which BASIC-language programs and data arrays are stored. The basic machine comes with 16K bytes. Approximately 3K bytes of this memory are used by the system and are not available to the user. This block can be expanded to a full 64K bytes by adding memory options. 4K dynamic NMOS RAMs* are used for this memory.

Block 1 memory is used primarily for the operating ROM needed to program the PPU, which consists of 40K bytes. There are also 8K bytes of read/write memory in this block that are used for I/O buffers and by the CRT for the alphanumeric display buffer. The remainder of the space in block 1 is reserved for option ROMs, primarily those associated with I/O operations.

Block 2 memory is reserved for future option ROMs, and in particular, those options more closely

*RAM = Random-access memory (read/write memory).

associated with expansion of the BASIC language.

Block 3 memory contains all the ROM for interpreting and executing a BASIC-language program. This requires 58K bytes of ROM. The remainder of block 3 can also be used for option ROMs.

The ROMs used in System 45 are similar to the 16K-bit ROM used in the HP 9825A, but they have a new feature. The new ROM is completely self-contained. It does not require an external switching network to supply power to the memory array when it is being addressed. The elimination of these external components was a significant contribution towards implementing the large amount of ROM required. In addition, a thin-film hybrid circuit was designed to accept up to eight ROM chips, further reducing the space requirements for this memory.

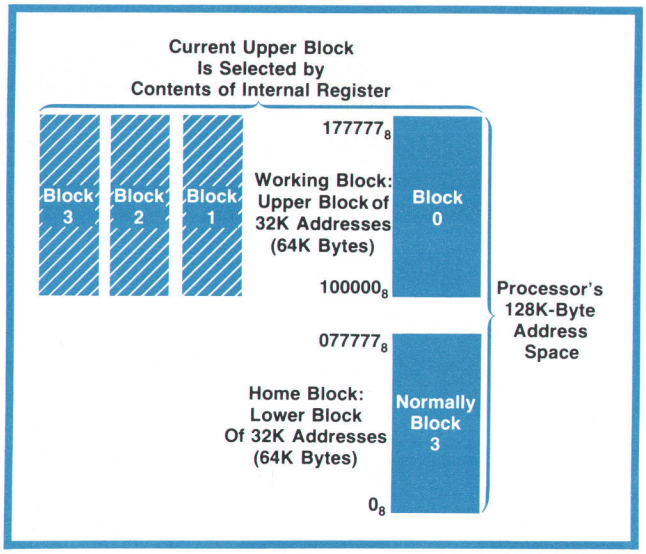


Fig. 3. *System 45's modified microprocessors can address 128K bytes of memory. Since there are 256K bytes, a memory address extension scheme was devised. When the most significant bit (MSB) of the memory address is a zero, the home block is accessed. When the MSB is a one, the working block is accessed. The working block, which can be any of System 45's four blocks, is specified by a two-bit register accessible to the system programmers.*

Memory Address Extension

Modifying the processors to address twice as much memory, or 128K bytes, as explained above, was not sufficient to meet the system's needs, since System 45 can have up to 256K bytes of memory. Therefore, an additional circuit, the memory address extension (MAE) circuit, was developed to further extend the addressing range. The design objectives for this circuit, besides extending the addressing range, were to minimize the amount of hardware required, not to let that hardware limit the system speed, and to make operation easy and convenient for the system programmers. The circuit that finally evolved provides a good balance between the last two objectives, which are, of course, incompatible.

The scheme is to divide the 128K-byte address space provided by the processor into two blocks of memory, each containing 64K bytes, as shown in Fig. 3. The lower block, which is called the home block, is always accessed when the most significant bit (MSB) of the program counter, or any other register used to specify a memory location, is a zero. When the MSB is a one, then the upper block, which is called the working block, is selected. But System 45 actually has four 64K-byte blocks, and any of these can be used as the upper working block. (The working block can be the same as the home block, although there is no practical use for this configuration.) When the MSB is a one, the contents of a two-bit register determines which one of the four blocks will actually be used. The system programmer knows in advance what working block to use and simply stores the appropriate binary code in the two-bit register.

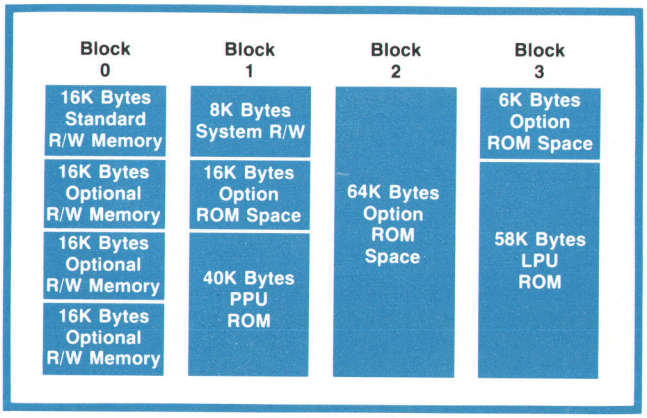


Fig. 2. *System 45's 256K bytes of memory are partitioned into four blocks of 64K bytes each.*

There are actually three different addressing modes: instruction fetches, DMA cycles, and indirect data memory references. Each has a slightly different home/working block convention and associated two-bit register to select the correct working block. The MAE hardware distinguishes between these different modes by decoding all of the various types of memory cycles that can be initiated by the processor.

Memory Address Extension is used only for the language processor unit (LPU). The functions of both processors and the resulting microcode were carefully partitioned so the peripheral processing unit (PPU) requires access to only two blocks of memory. Therefore, modifying the processors to double their addressing range was sufficient for the PPU.

Dual-Port Memory Control

The two processors communicate with each other through two blocks of memory that can be accessed by both processors. This requires a great deal of

System 45 Product Design

by Ray J. Cozzens

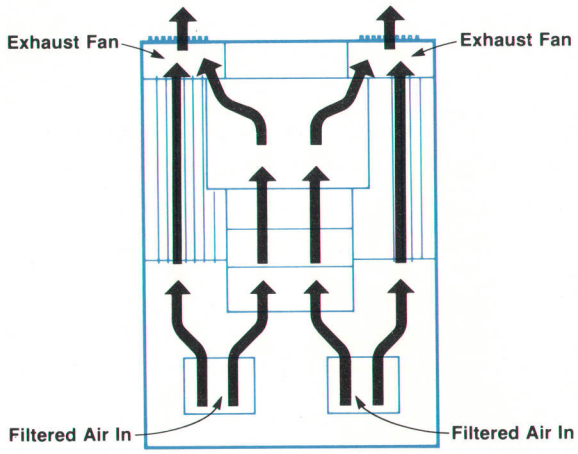


Fig. 1. *Dual-fan parallel cooling scheme contributes to System 45's high degree of integration, as well as to reliability.*

System 45's high degree of integration is made possible to a large extent by contributions made in the area of product design. One of these is a novel dual-fan parallel cooling system, which also makes a major contribution to reliability (see Fig. 1). Cool air is pulled in through two filtered air ports beneath the keyboard. This cool air first comes into contact with the keyboard logic, the tape transport drive electronics, the tape drive motors, and the ROM drawers. At this point the air separates into three nearly equal flow paths. One path is over the read/write memory, the dual-port memory control, and the CRT control logic board. A central air stream is directed across the thermal print head and paper drive motor and through the computer power supply. The third air path is over the printer drive electronics, the memory address extension board, and the system processors. The three air paths then converge and exit via two exhaust fans in the rear. This scheme helped decrease some component temperatures up to 40°C. A common rule of thumb for electric components is that the reliability doubles for each 10°C decrease in component temperature, so many System 45 components may be enjoying up to 16 times the reliability of previously considered alternatives. This air flow scheme also plays a key role in allowing the computer to operate in ambient temperatures of 5°C to 40°C.

The CRT cabinet is convectively cooled. To make this possible, the power transistors, representing roughly one-third the CRT power dissipation, are mounted on the external surface of a large finned die-cast aluminum back panel. Also, the CRT control logic board is not in the CRT cabinet; it is housed in the mainframe and is cooled by forced air. This not only moves power out of the CRT cabinet but also allows wide spacing of the remaining assemblies, eliminating the need for a costly, space consuming, and electrically noisy fan.

Serviceability

For a complex computer, System 45 is very serviceable. Trials have confirmed that a service person can abtain access to any of 90% of the assemblies in less than 10 minutes. This is made possible by the judicious use of structural foam plastic case parts and the modular design of several major assemblies, such as the power supply, the keyboard/tape units, the ROM drawers, and the printer/plotter (see Fig. 2).

Serviceability is further enhanced by system test firmware stored in a special ROM. The ROM was developed not only to aid service engineers in troubleshooting and system checkout, but also to be used on the production lines as the newly assembled computer goes through a series of pre-shipment checkouts.

The computer can be serviced while it is fully operational. A set of brackets and extenders was developed to allow access

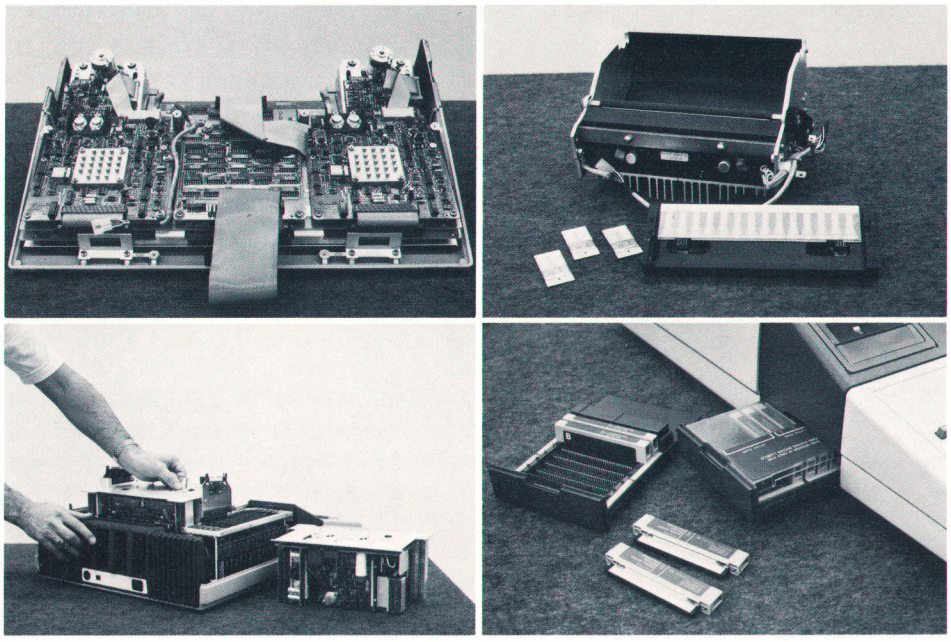


Fig. 2. *Modular design provides access to any of 90% of the assemblies in ten minutes.*

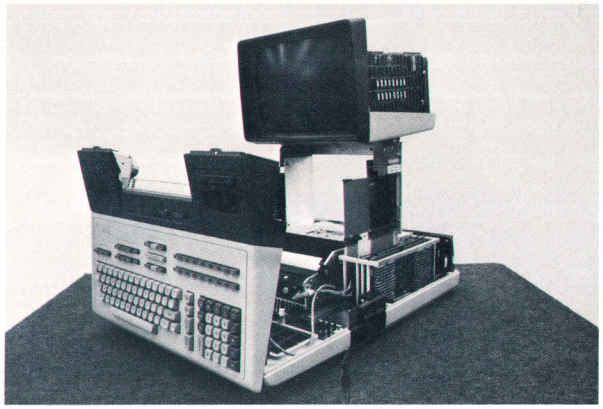


Fig. 3. *Using a specially developed set of brackets and extenders, the computer can be serviced while fully operational.*

to nearly all of the subassemblies while stepping through a troubleshooting tree or a series of tests from the test ROM (see Fig. 3).

All of the ROM in the computer is customer accessible (see Fig. 2). Option ROMs are field installable. Further, if there is ever a need to change a ROM pack, the customer can perform this simple operation in just a few minutes, eliminating the time and expense associated with a service call.

The desktop computer is made portable by removing the CRT. Only one simple motion is needed to unlatch and lift the CRT off the mainframe, disconnecting it both mechanically and electrically in a safe and reliable manner. Hard and soft carrying cases for the mainframe and the CRT have been designed for air or surface transportation of the computer.

Convenience Features

Consistent with the concept of a friendly interactive computer, several user convenience features were incorporated. Just beneath the CRT are three user "help" cards (with room to add others if the need arises). These cards include a brief description of system and language program errors, a short set of system operating instructions, such as how to remove and clean the air filters, and a quick-reference set of program statements. The reels in the tape cartridges are visible to the user, who can see whether the reels are rotating, which way they are going, and how much tape is left on either reel. The tape units are oriented to maximize cooling of the motor and tape and to minimize the chances of dirt falling onto the magnetic read/write head.

Environmental Considerations

In the normal operating mode the computer is extremely quiet for its size and capability. Its operating noise is only 46 dBA (barely a whisper). While printing normal text the noise level increases only to 55 dBA.

The system is also rugged. After exposure to the 30-g package drop test it continues to operate.

Safety and low electromagnetic interference (EMI) were primary design objectives. Safety reviews and EMI measurements were conducted at each development phase. Design changes all along have been implemented to comply with the major regulatory agencies, including UL, and regulatory agencies in Europe, such as VDE.

hardware coordination, which is provided by the dual-port memory control (DPMC). The DPMC is basically a double-pole, double-throw switch (Fig. 4). One side of the DPMC interfaces to the two independent memory buses from the processors and the other side to two memory buses from independent memory blocks. The DPMC can support simultaneous memory cycles from both processors as long as each is communicating with a different block. As soon as both processors must communicate with the same block of memory the DPMC switches at a 50% duty cycle, allowing both processors to access memory, but at a reduced speed. The locations of the firmware routines in memory were selected to reduce the need for this mode of operation to a very low percentage of the time.

The DPMC expects to see memory timing signals from each processor during a particular time relative to the internal TTL clock signal that is derived from one phase of the PPU's two-phase clock. However, each processor has its own clock generator, which divides the master oscillator frequency by two, down to the final operating frequency. Thus it is possible for the LPU clock to be out of phase relative to the final operating frequency by one-half clock time at power on. A circuit (Fig. 5) on the processor board senses this illegal turn-on condition and corrects it before any memory accesses occur. The AND gate G1 detects when ϕ_1 of one processor is true while ϕ_2 of the other processor is true (out-of-phase condition). This output sets the D flip-flop during its next clock time and holds the clock input to one processor steady until the other processor has changed states. The AND gate then detects the in-phase condition and the flip-flop is reset, allowing the held-off processor to be clocked again but now in phase with the other processor. NOR gate G2 equalizes the delay paths from the master oscillator to the final clock outputs.

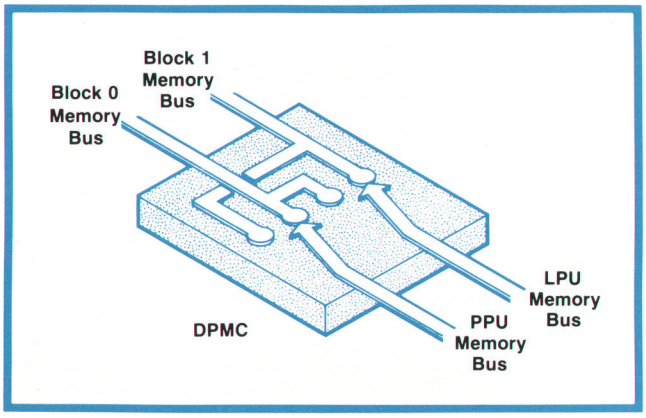


Fig. 4. *Dual-port memory control (DPMC) coordinates memory accesses by the two processors. When the LPU and the PPU must communicate with the same block of memory the DPMC switches between them at a 50% duty cycle.*

High-Resolution CRT Display

Dual raster scan circuitry was developed for the CRT display so the size and aspect ratio of the display information could be optimized for both the alphanumeric and graphic requirements. In the alphanumeric mode, the important requirements were the character size, the character font, and the ability to display 24 lines of 80 characters. A 7×9 character font (in a 9×15 matrix) was selected to give aesthetically pleasing characters for the ASCII set and the language options. A raster of 720 dots by 375 scans (Fig. 6) produces 25 lines of 80 characters each.

For graphics applications this raster was not ideal. In the graphics mode, the important requirements were to maximize the resolution in both dimensions, to be able to dump the plots directly on the internal printer (560 dots per line resolution), to stay within 16K words of refresh memory, and to maintain the same character quality as in the alphanumeric mode. A raster of 560 dots by 455 scans (Fig. 6) was chosen.

Development of a horizontal sweep circuit that could produce a linear display at two sweep frequencies, 23.4 kHz and 28.7 kHz, was essential to the dual raster scan concept. Most horizontal sweep circuits correct for nonlinearities with passive components. The horizontal sweep circuit in the System 45 display uses active linearity correction. Active linearity correction allows the use of two different horizontal frequencies for the alphanumeric and graphics rasters because the correction is frequency independent.

Fig. 7 shows the major blocks in the horizontal sweep circuit. Fig. 8 shows the yoke current and correction voltage waveforms in relation to the horizontal sync pulses.

The heart of the active correction circuit is the horizontal reference waveform generator, which supplies an ideal waveform to which the actual current waveform can be compared. The waveform of the actual yoke current is supplied by a current-sensing transformer in series with the horizontal yoke. This signal is compared with the reference waveform and the difference is amplified by the error amplifier. During the display portion of a sweep cycle, the power amplifier uses this error signal to put out a correction voltage to correct the yoke current. The automatic level controller regulates the amplitude of the reference waveform by adjusting the input level to the horizontal reference waveform generator. This compensates for component drift and reduces the dynamic range required in the correction circuit.

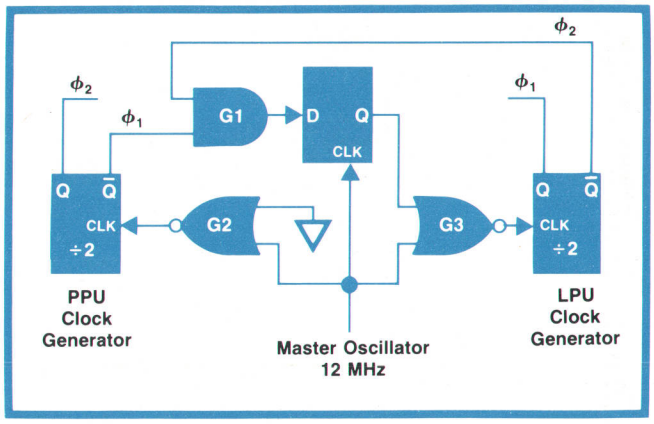


Fig. 5. *Since each processor has its own clock generator it is possible for the LPU clock to be out of the proper phase by one-half clock time when power is turned on. This circuit senses this illegal condition and corrects it.*

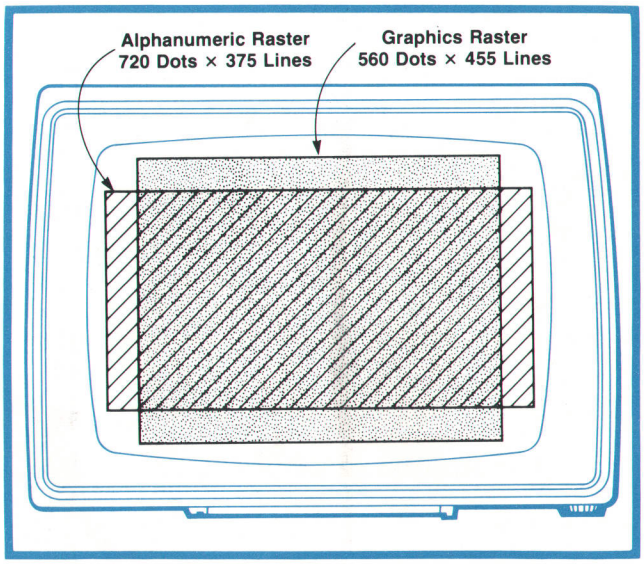


Fig. 6. *Dual raster scan circuitry was developed for the CRT display so the size and aspect ratio of the displayed information could be optimized for both alphanumerics and graphics.*

During the retrace portion of a sweep cycle, the error input to the power amplifier is disabled by the FET switch driver, which simultaneously closes a feedback path from the error amplifier to the horizontal reference waveform generator. Thus the power amplifier is not driven into saturation by a large error signal, and the integrator is reset because the reference waveform is corrected to match the sensed yoke waveform.

The 21-MHz display clock rate made it necessary to design a high-speed video circuit to assure uniform brightness for both single dots and horizontal lines. Slow video-circuit rise and fall times would have caused dots to appear dimmer than lines. The video driver (Fig. 9) is basically an inverting level shifter that provides the large (20-30V p-p) voltage swings required at the cathode of the CRT to turn the beam on and off. The voltage (−V) can be adjusted by the user to vary the brightness from about 10 to about 30 footlamberts. Video data comes from the alphanumeric and optional graphics display sections to the V1 and V2 data inputs. Capacitors C1 and C2 provide feed-

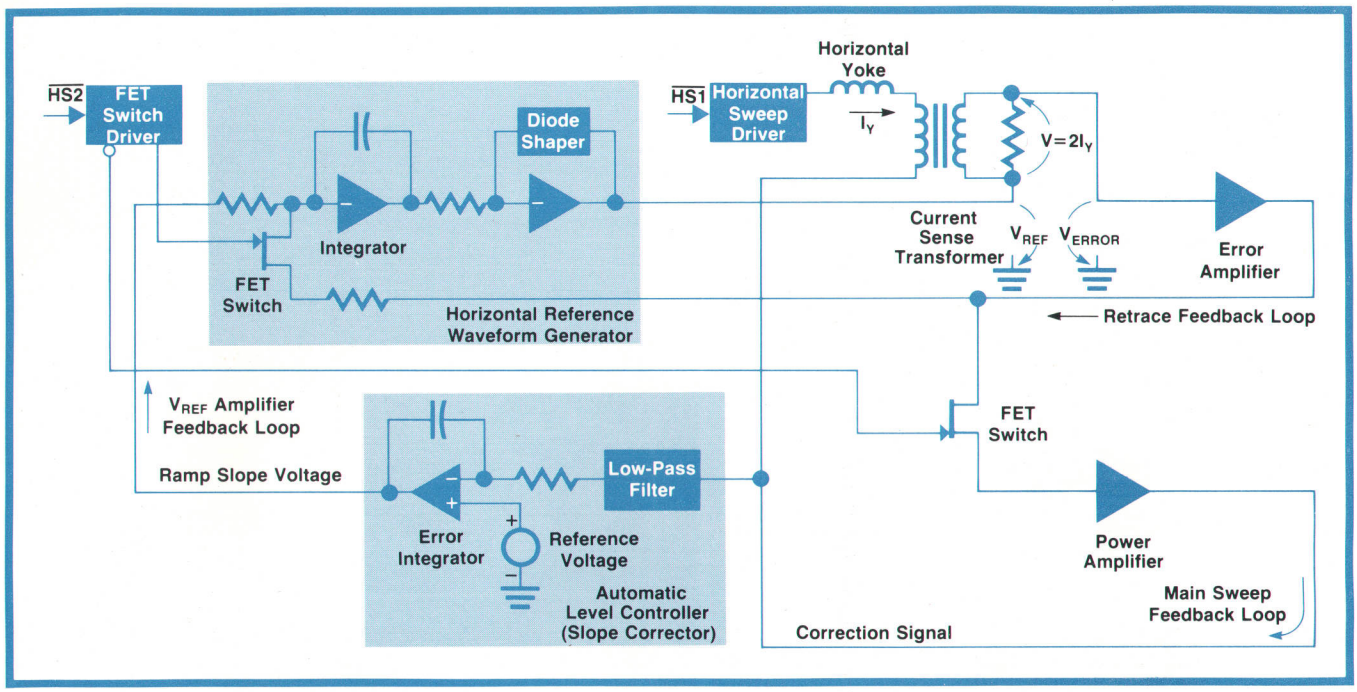


Fig. 7. *System 45 horizontal sweep circuit uses active linearity correction. The scheme is frequency independent, thus allowing the use of two different horizontal frequencies for the alphanumeric and graphics rasters. The actual yoke current is compared with an ideal waveform to generate a correction signal.*

forward to the output stage. Transistors Q2 and Q3 provide dc drive for long on or off periods. Inductor L1 provides approximately 7% overshoot in the cathode voltage, helping compensate for the finite rise time (~20ns).

Alphanumeric Display Architecture

The alphanumeric portion of the System 45 display is designed as an integral part of the system to achieve high performance at a low cost. The refresh memory for this display is located in the memory block for the mainframe peripheral processor. The data is accessed by the display using a technique called memory cycle stealing. This technique uses a feature of the processor that allows external devices to stop the processor and use its memory bus. Therefore, a simple controller in the display system can be used to fetch the data on a regular basis. Since none of the internal registers of the processor is disturbed during this process, the processor resumes operation where it left off immediately after the display memory cycle is completed. The only effect is a slight reduction in the average processing speed. The reduction in processing speed is dependent on the amount of data being displayed and averages less than 5%.

The major advantage of this architecture is that the processing of data for the display refresh memory is handled by one of the high-speed processors already available in the mainframe. The advantages to the user are high-speed listing, scrolling, and editing capability. Additional advantages include a lower overall memory cost, a simplified controller in the display system, and a display format that is controlled completely by the peripheral processor software. The design has the flexibility to accept future ROMs to tailor the display format for specific applications.

The alphanumeric display block diagram is shown in Fig. 10. A 32×16-bit PROM state machine controller generates the memory cycles to fetch the data and format it to be stored in an 80-character line buffer. Two line buffers are available. One is being loaded while the other is being displayed. The addition of the

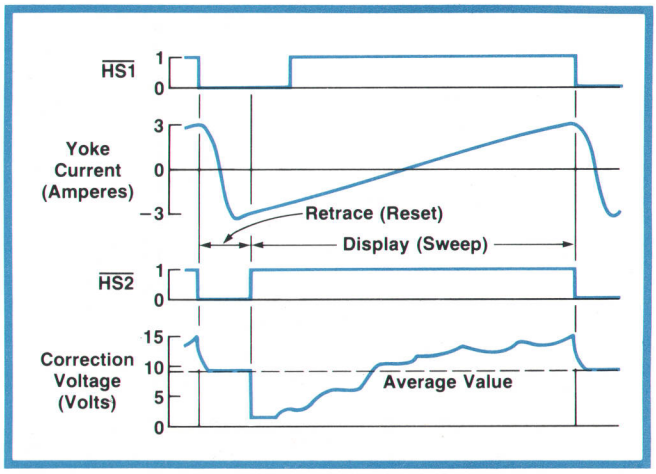


Fig. 8 *Waveforms for the horizontal sweep circuit of Fig. 7.*

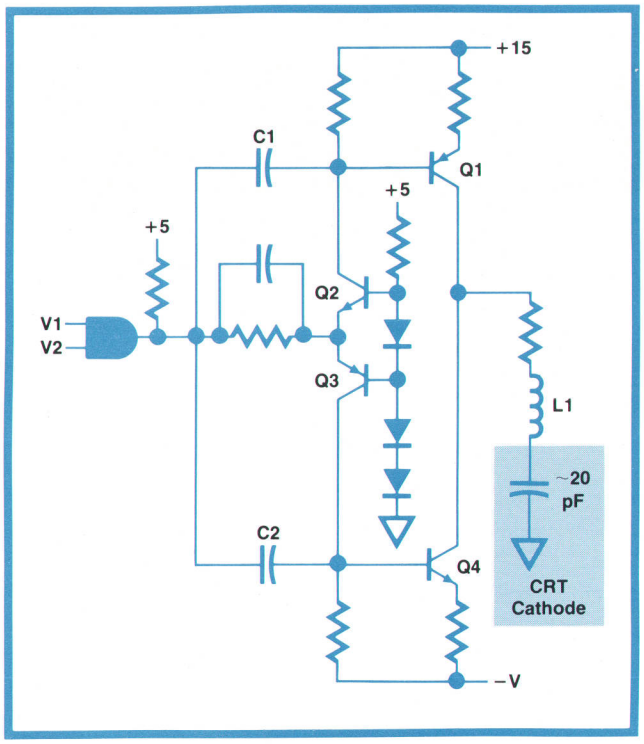


Fig. 9. *High-speed video driver has fast rise and fall times, thereby assuring uniform brightness for both single dots and horizontal lines in spite of the 21-MHz display clock rate.*

line buffers allows the number of memory cycles and the memory cycle rate to be greatly reduced so there is no period of time when the display is using the bus continuously.

Before the data is loaded into a line buffer, the data decoder determines whether it is a character byte or one of several control bytes. Efficient memory allocation is obtained by using an end-of-line (EOL) command to terminate character lines. This feature makes it possible to store several pages of data in the display refresh area of memory, on the average, and again greatly reduces the number of memory cycles needed per page. For example, approximately 130 20-character program lines, or over six pages, can be stored in the refresh memory. Scrolling speed is increased because additional processor time is not needed to list the remaining pages stored in the refresh buffer. Displaying data from non-consecutive memory addresses is done by using the new word address (NWA) command. The NWA command directs the display hardware to interpret the next word in memory as an address instead of data. The sequence of data to be displayed continues at the new address. This feature was important to allow the display to be divided into two sections, the printed section and the interactive section (see article, page 2). Several commands available to the user make it possible to use the highlighting features, to shift to an optional character set, and to shift to the optional graphics raster.

Graphics Display Architecture

The heart of the System 45 graphics display is a bit-per-element image memory implemented with 16K RAMs (Fig. 11). It is organized into 16,384 16-bit words. Each dot of the 560-dot-wide-by-455-dot-high graphics raster is represented by one of the memory bits. A graphics cursor is also provided and is independent of the image memory. Intensification of the cursor makes it easily distinguishable from the image. A TTL controller in the graphics hardware manages the cursor addresses and memory data sent by the mainframe via the I/O bus.

Vectors (straight lines) are the basic geometric elements of the graphics display. All images including

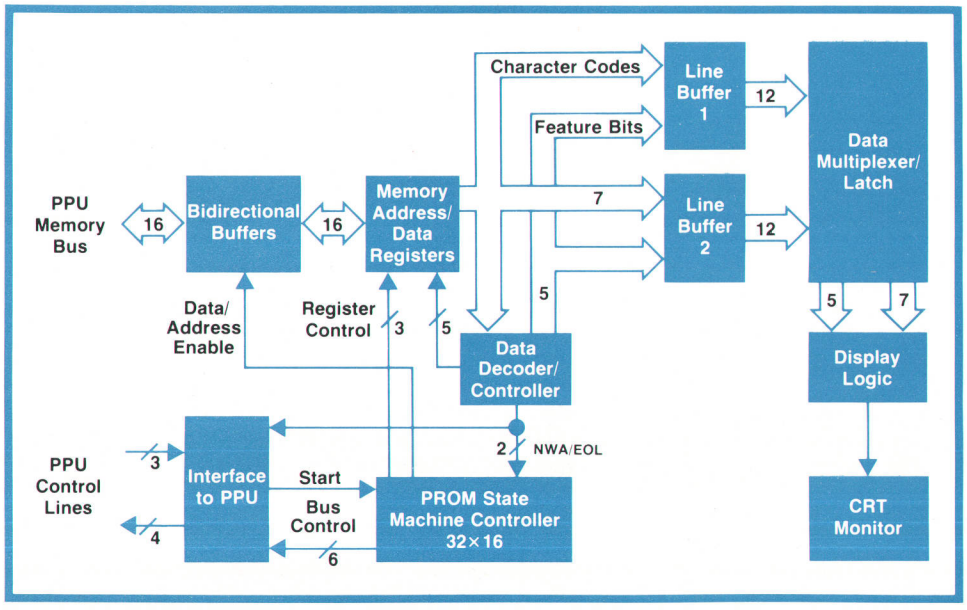


Fig. 10. *Alphanumeric display block diagram.*

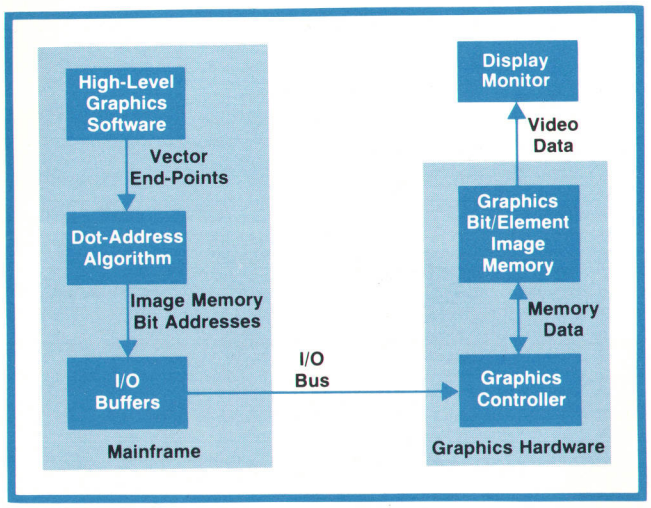


Fig. 11. *Graphics display block diagram. The image memory has one bit for each of the 560-dot-by-455-line display. It is implemented with 16K RAMs.*

curves are composed of vectors. Vectors are represented on the CRT by a series of adjacent dots that collectively give the image of a straight line. User PLOT statements furnish the vector endpoints and from these the locations of the dots between the endpoints are calculated. The graphics architecture takes advantage of the mainframe processor to do these calculations in an efficient, simple dot-address algorithm that produces a rapidly drawn vector. This simplifies the graphics hardware by eliminating the need for a hardware vector generator. The graphics hardware simply loads individual bits (dots) into its image memory.

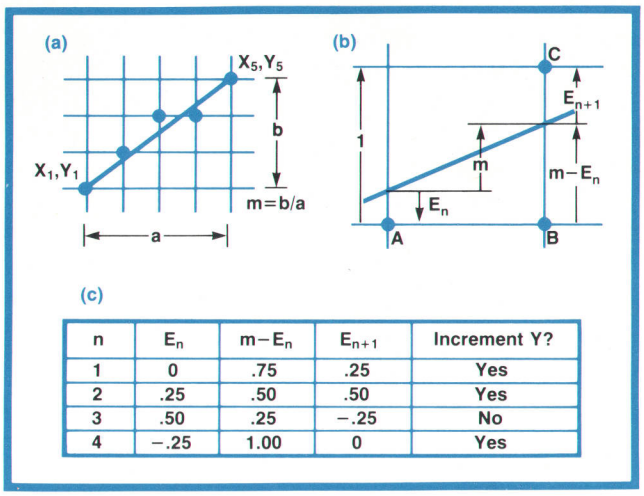
The software-hardware data format is optimized to make the dot address algorithm simple and fast. To further increase the speed, buffered I/O is used. The algorithm fills one buffer with bit addresses while the graphics hardware is unloading another buffer.

To calculate the addresses of the dots between vector endpoints the dot address algorithm takes into account the octant that the vector is in, but the basic algorithm is the same for all octants. The example of Fig. 12a is for the first octant, 0° to 45°. The first dot is placed at one of the endpoints, X_1, Y_1. For each successive dot X_n, Y_n the X value always increments by 1. The Y value may remain unchanged or may increment by 1 depending on the slope, m, and the present error, E_n. Fig. 12b shows the general situation. The last placed dot is at A. The next dot will be at either B or C depending on the value of $m-E_n$. If $m-E_n \geq \frac{1}{2}$, Y is incremented (location C) and $E_{n+1}=1+E_n-m$. If $m-E_n<\frac{1}{2}$, Y is unchanged (location B) and $E_{n+1}=E_n-m$. The calculations for the example are illustrated in Fig. 12c. The algorithm is rapid because only addition and subtraction are needed. As each new X,Y coordinate is generated it is converted into the corresponding image memory word address and bit number. After all such data has been sent to the I/O buffer it is DMA-transferred via the I/O bus and digested by the graphics controller, which loads ones into the addressed memory bits.

In addition to handling individual dot addresses, the graphics hardware can exchange 16-bit words between its image memory and the mainframe. This makes it possible to store or print the graphics image. Images stored in a mass memory device may later be rewritten to the graphics image memory to be viewed on the CRT.

Acknowledgments

Many people worked diligently to overcome significant problems during the System 45 mainframe development. Scott Bennett designed the DPMC and the I/O bus. Dan Griffin designed the MAE circuit, the revised ROM, and the ROM hybrid. Gerald Reynolds contributed the processor board, motherboard, and tape electronics. Ray Kloess and Dave Aldridge are to be thanked for their design of the power supply. Dyke Shaffer assisted late in the project with the final design changes on some of the hybrids used, and the following individuals contributed to the modified processor chips: Ken Eldredge, Dennis Peery, Paul Stoecker, Dave Uhlrich, Bill Eads, and Bill Thayer. Frans Laverman, who was the production engineer for System 45 at HP's European manufacturing facility, spent a year with the design team making contributions in numerous areas. Bob Brooks, lead engineer on the product design, had the challenge of physically integrating all the pieces. Steve Anderson



n	E_n	$m-E_n$	E_{n+1}	Increment Y?
1	0	.75	.25	Yes
2	.25	.50	.50	Yes
3	.50	.25	−.25	No
4	−.25	1.00	0	Yes

Fig. 12. *Vectors are the basic elements of the graphics display. A vector is represented by series of dots on the CRT screen. Using a simple, fast dot-address algorithm, the PPU calculates the addresses of the dots that make up a vector, based on user-supplied end points. In the first octant, for each successive dot, the X value increments by one. The Y value may or may not increment, depending on the slope m and the error E_n.*

System 45
Tape Control System

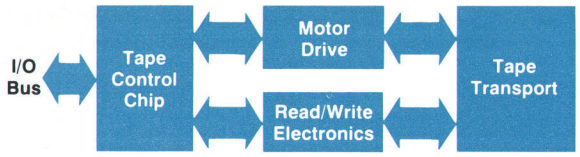


Fig. 1. *System 45 tape control system. The tape control chip is a new 14,000-transistor NMOS design.*

Fig. 1 is a block diagram of the control system for System 45's built-in mini-cartridge magnetic tape units.

The heart of the tape system is the 14,000-transistor NMOS tape control (TACO) chip (Fig. 2). TACO was designed to interface between the System 45 I/O bus and the tape transport. In addition to replacing 48 TTL MSI packages and 80 discrete components, TACO has more features than the original TTL design. Functionally, the chip is divided into five subsystems: registers, on-board digital servo, gap detector, ROM, and I/O. External circuits detect flux reversals and tach pulses and drive the motor.

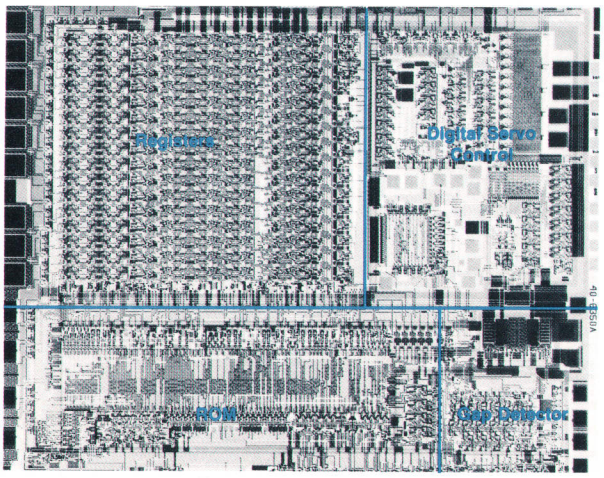


Fig. 2. *The tape control chip replaces 48 TTL MSI packages and 80 discrete components, and has more features than the previous design.*

The tape control chip with the external circuits has the following capabilities:

Digital Servo

Simple tape motion commands: fast/slow, forward/reverse, stop/go

Tape speed of 22 inches per second (ips)

Tape speed of 90 ips for rewind and search

- Controlled acceleration and deceleration.

Read/Write

- Reads and writes on a 16-word basis
- Delta-distance coding[1]
- Double buffering of data
- Software control of precompensating bits during writing.
- Automatically synchronizes to the preamble during each read
- Continuous tracking of the one/zero threshold
- Calculation of checksum during each read and write
- Reads and writes HP's standard interchange format
- Writes record, file, and end-of-valid-data gaps under software control

Other

- Measures distance on the tape by counting tachometer pulses
- Counts gaps
- Detects end-of-valid-data gaps
- Interfaces directly with the I/O bus
- Interrupts processor when an instruction is completed
- Interrupts processor when data is needed
- 32 different instructions.

The on-chip servo provides motor control for equivalent tape speeds of 22 ips and 90 ips for read/write and search operations, respectively, as well as a constant acceleration (deceleration) rate of 1200 ips^2. The servo state machine keeps track of tape direction and speed and responds dynamically to any sequence of commands from the instruction register.

The input frequency (f_t) to the servo is generated by a 1000-line optical tachometer[2] fixed to the motor shaft. A number N_t proportional to $1/f_t$ is computed by counting system clock pulses between rising edges of f_t. The difference between N_t and an internal reference number N_r is the error provided to the external servo circuitry for motor control.

There are 128 reference numbers (N_r) stored in the servo ROM. These represent tape speeds from 2 ips to 90 ips. A constant acceleration (deceleration) is effected by changing the reference at a rate equivalent to 120 ips^2. 128 references provide good linearity for a closed-loop servo bandwidth of 250 Hz.

Without additional circuitry, the system would suffer dramatic open-loop gain degradation during acceleration. To illustrate, consider two reference numbers an order of magnitude different, $N_{r1}=N_{r0}$ and $N_{r2}=10N_{r0}$. Assuming a 10% relative error was calculated with respect to each of these references, the resultant error magnitudes would be $E_1=N_{r0}/10$ and $E_2=N_{r0}$. The servo system compensates for this undesirable effect, thereby providing a nearly constant open-loop gain for all tape speeds encountered.

Overall system performance is well depicted by the conditions existing during read/write operations. Data is transferred to and from the tape at 22 ips, which corresponds to $f_t \approx 622$ kHz and $N_r(22\ \text{ips})=94$ for a 2-MHz system clock. Therefore, speed errors of 1.1% are resolved at a sample frequency (f_t) that is nearly two orders of magnitude beyond the 250-Hz servo-closed-loop bandwidth (77 dB down for a critically damped system). For System 45, a crystal-controlled oscillator provides the chip reference and affords long-term read/write speed control within better than 0.01%.

The tape control chip is packaged on a 64-pad square ceramic substrate 4 cm on a side. The leadless substrate connects to the printed circuit board through an elastomer gasket. A custom-designed heat sink is mounted to the back of the substrate with thermal compound to ensure good heat transfer.

Reference

1. 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, page 13.

2. D.M. Clifford, F.T. Hickenlooper, and A.C. Mortensen, "Mid-Range Calculator Delivers More Power at Lower Cost," Hewlett-Packard Journal, June 1976.

-Richard Kochis

was responsibile for the industrial design, including the human factors as seen by the user. Rob Beeson contributed all of the product design for the CRT display and latching mechanism. Hudson Grotzinger contributed the product design of the modular power supply and the modular keyboard. Walt Perdue developed the tape transport mounting and the ROM drawers. Ray Cozzens provided invaluable support and guidance to the product design group. Lowell Kolb designed the CRT monitor circuits and was the inventor of the video and horizontal circuits described in this article. Fred Porter designed the control logic circuits for the CRT display. Thanks also to Dick Barney, project manager for the CRT display, power supply, and keyboard, for the direction he provided. Marl Godfrey directed a team of engineers responsible for testing the final product and insuring its overall integrity—Scott Bennett, Brent DeWitt, and Fred Richart provided significant contributions to this effort. Ken Eldredge, Craig Mortensen, Rich Kochis, Bill Thayer, and John Balza contributed to the definition and development of the TACO chip.

References

1. 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.
2. W.D. Eads and D.S. Maitland, "High-Performance NMOS LSI Processor," Hewlett-Packard Journal, June 1976.

System 45 Power Supply

System 45 has a switching power supply similar to that of Hewlett-Packard 21MX Computers[1]. This power supply technology offers both small size and light weight in addition to excellent efficiency and very good regulation. Voltage regulation occurs at high voltage and low current, which is inherently more efficient than regulation at low voltage and high current. The efficiency achieved is approximately 70%. The major circuit modules are: input circuitry, a 20-kHz high-voltage transistor switching regulator, two dc-to-dc converters, and low-voltage high-current output circuits.

The output voltages are +5Vdc, +7Vdc, ±12Vdc, ±17Vdc, and ±18Vdc. Only the +5Vdc and ±12Vdc outputs are directly regulated. The transformer turns ratios in the dc-to-dc converters maintain all of the other voltages within ±1Vdc of their nominal values. The power supply delivers up to 270 watts continuously or 350 watts peak. Input requirements are 90 to 126 Vac or 198 to 252 Vac, 48-66 Hz.

The power supply consists of five printed circuit boards assembled in a compact cubic configuration (approximately 320 in^3) enclosed in a sheet metal enclosure behind the line printer assembly.

Reference

1. R.C. Van Brunt, "A Computer Power System for Severe Operating Conditions," Hewlett-Packard Journal, October 74.

-Dick B. Barney

Louis T. Schulte

After receiving a BSEE degree in 1963 from Washington University in his native St. Louis, Missouri, Lou Schulte spent a number of years designing inertial guidance systems. After joining HP in 1972, he wrote software for the 9805A Calculator, and designed the thermal printhead chip for the 9815A Desktop Computer and the graphics hardware for the 9845A. He later served as lead engineer for 9845A production engineering and is now a project manager in the desktop computer lab. Lou leans toward the dramatic, spending many of his extra hours involved in community theater. He and his wife and two daughters live in Loveland, Colorado.

John C. Keith

Today John Keith is wearing his project manager's hat in HP's IC facility in Loveland, Colorado, with responsibility for IC products. Before that he was project manager for mainframe hardware on the 9845A. Other contributions to HP include the initial software definition on the 9815A Desktop Computer, initial design for a buffered keyboard on the HP-81 Calculator, work on the I/O system for the 9805A Calculator, production engineer for the 3403A Voltmeter, and participation in the design team for the 3403A. Born in Des Moines, Iowa, John earned a BSEE from Kansas University in 1969. Finally pursuing a lifelong ambition to get involved in amateur radio, he devotes any time left over to woodworking and planning an energy-efficient solar heated home for the future. John lives in Loveland with his wife and 3 children.

Ansel K. Vogen

Andy Vogen began his career at Hewlett-Packard in 1965 as a function generator circuit designer on the 3304A. He then spent 2½ years as a materials engineer, later becoming production coordinator for the 9810A and 9820A Desktop Computers. He designed the 11203A BCD Interface Card, contributed to the thin-film printhead development for the 9825A, acted as lead engineer for the alphanumeric display portion of the 9845A, and recently spent eight weeks at Böblingen, Germany, assisting in the transfer of the 9845A to production. Andy managed to find time in that busy schedule to complete work on an MSEE degree at Colorado State University, finishing up in 1970. Born in Morris, Illinois, he earned his BSEE degree in 1965 from the University of Illinois. Andy likes all types of sports, family outings, church activities, and home projects to fill his off-duty hours. A Loveland, Colorado resident, he is married and has two daughters.

Advanced Thermal Page Printer Has High-Resolution Graphics Capability

This optional System 45 built-in peripheral quietly outputs program listings or hard copies of anything on the CRT display.

by Ray J. Cozzens

SYSTEM 45'S OPTIONAL built-in page-width thermal printer/plotter is a compact, high-speed, "smart" peripheral that quietly outputs program listings or hard copies of alphanumeric and graphic data displayed on the CRT. Consistent with the overall system integration philosophy, the design objectives for the printer were not only to integrate the thermal printer/plotter into the mainframe and balance its performance with the programming and graphic capabilities inherent in the System 45, but also to enhance greatly the capabilities available to the user. Many of the key user enhancements were made possible by incorporating a dedicated microprocessor. The microprocessor also makes possible a high degree of optimization in the speed-versus-power trade off.

Another contribution that makes this printer/plotter a significant advance over its predecessors is the Hewlett-Packard-developed thin-film process used in the manufacture of the monolithic printhead

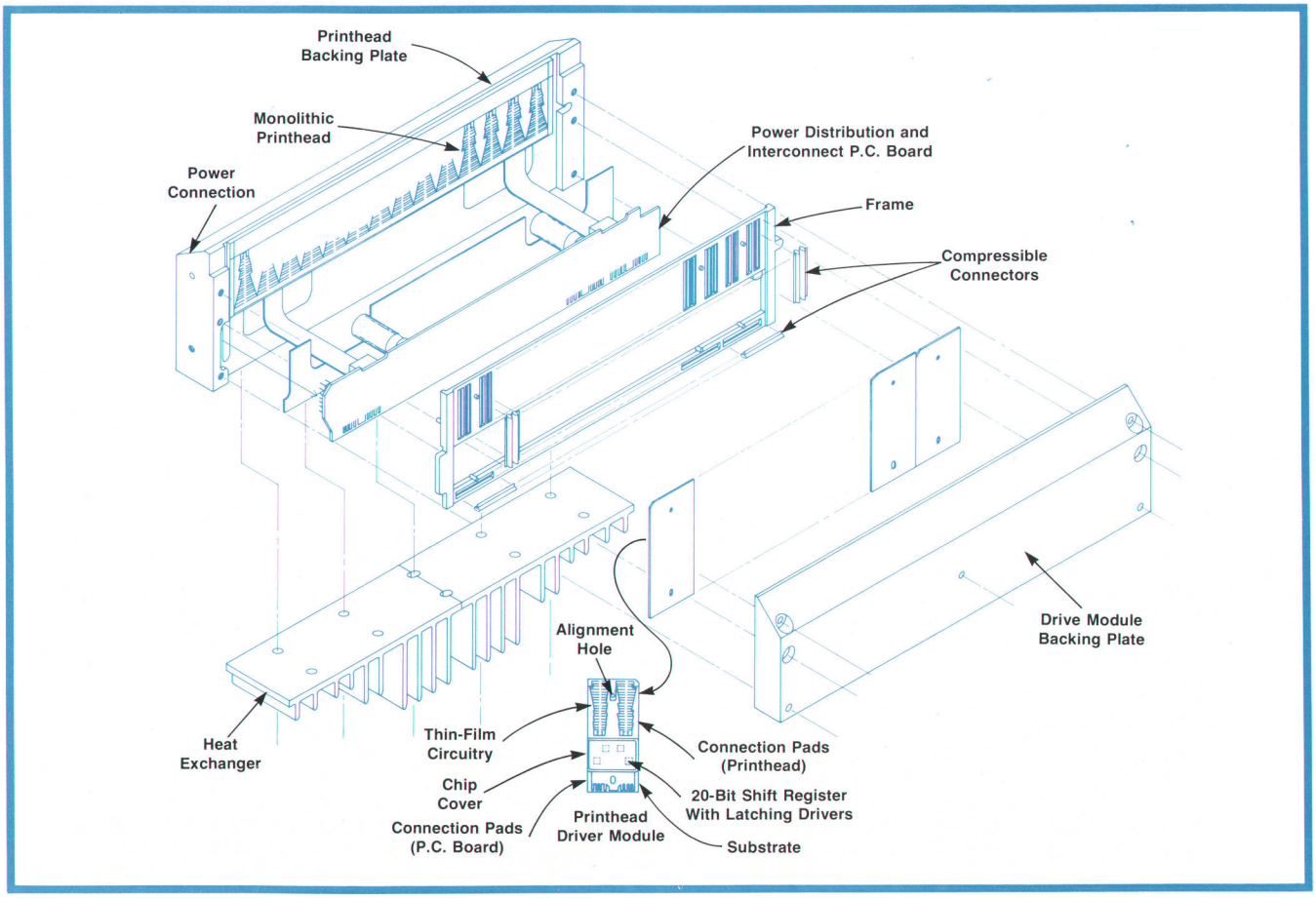


Fig. 1. *Exploded view of the printhead of System 45's optional built-in thermal printer. 560 print resistors are selectively energized to darken the thermal paper where information is to be recorded.*

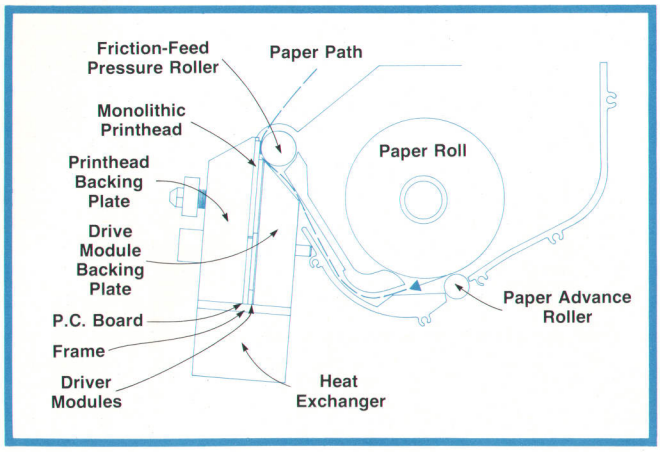


Fig. 2. *Sectional view of the printhead/paper relationship.*

(see page 25). The printhead had to have not only the ability to print alphanumeric data, but also a high-resolution plotting capability to satisfy the need for a dot-for-dot hard copy of the CRT graphic image.

Design of the Printhead Assembly

The heart of the printer/plotter is the printhead. Fig. 1 shows an exploded view of its basic elements. There are 560 equally spaced print elements (resistors) on the monolithic printhead, with a spacing of 0.330 mm (0.013 in). Printing and plotting are performed by moving thermal paper in front of the print resistors while they are selectively energized, causing the thermal paper to darken, or "burn" where information is to be recorded. Fig. 2 shows a sectional view of the printhead/paper relationship.

Power and burn information for the printhead assembly come through a single cable and a 14-pin connector onto an interconnecting printed circuit board. The burn information is carried to the printhead drive module via compressible connectors, its destination being a 20-bit shift register with latching print resistor drivers. Power is carried from the interconnecting printed circuit board to the monolithic printhead by cables wire-bonded to the back of the printhead. Power is then conducted to the front of the monolithic printhead and made available to heat the resistors selected by the driver chips (see Fig. 3).

There are seven printhead driver modules with four driver chips per module. A thermally stable plastic spacer frame holds all the compressible connectors in their proper locations. There are two backing plates, one for the printhead and one for the driver modules, for which the material and design were determined by requirements of beam strength, since all the interconnections depend upon pressure contact. These backing plates are tied together thermally and make excellent heat sinks, since their thermal capacitance is quite large. Excess heat is conducted to the finned heat exchanger below the printhead and is convectively cooled by the forced air stream. This architecture allows a high degree of serviceability. For example, if a print resistor should ever burn out, all of the driver chips can be reused, or on the other hand, if a driver chip should fail, only one driver module would have to be replaced.

Speed versus Power

The 560 print resistors are used to form 80 characters per line in a font five dots wide by seven dots high. Two dots between each character are not used in the alphanumeric mode, but are used in the graphics mode to form continuous plots. The basic 5×7 matrix is contained in a field of 7×12 dots (see Fig. 4). This allows room for ascenders, accent marks, descenders, and underlines.

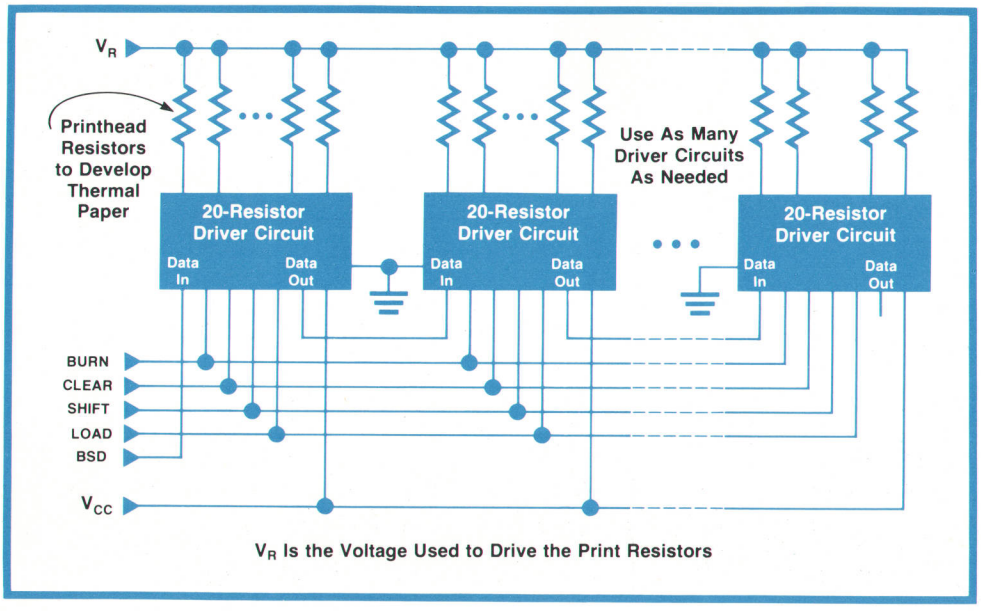


Fig. 3. *Twenty-eight 20-resistor driver chips receive burn information from the printer's internal microprocessor and energize the printhead resistors.*

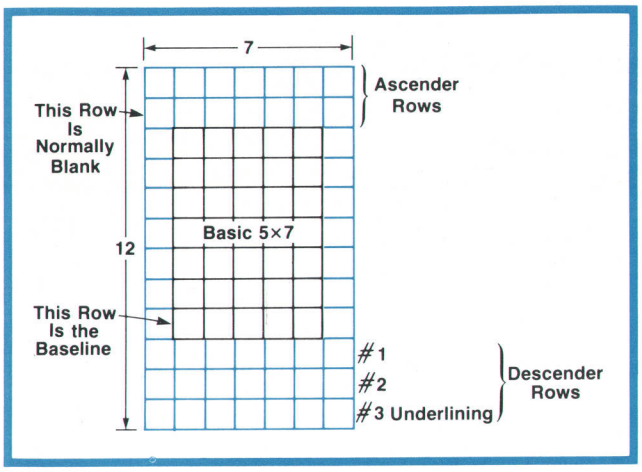


Fig. 4. *Characters are formed in a basic 5×7-dot matrix contained in a field of 7×12-dots. Two columns of dots between each character are not used in the alphanumeric mode, but are used in the graphics mode to form continuous plots.*

Close examination of a typical listing reveals that there are surprisingly few printed dots per row. Fig. 5 shows the results of such a study. It was found that 88% of the rows of a typical listing have 50 dots or fewer, and that typically a row will require only 22 dots to be printed. This information was used to optimize the trade-off between print speed and print power.

It was decided not to allow burning of all 560 dots at one time, since the peak power requirements would be excessive for an integral peripheral. Instead, only 56 dots may be burned at one time. This means that a solid row requires ten burn sequences. This provides a huge savings in peak power demand, with only a small sacrifice in print speed, since typically over 90% of the rows have 56 dots or fewer. Very few rows

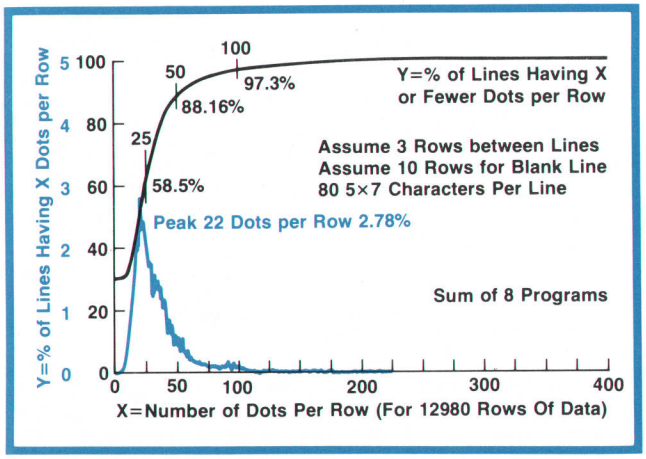


Fig. 5 *Results of a study of the number of dots per row in a typical listing shows that 88% of the rows have 50 dots or fewer. To reduce peak power demand, the printer is designed to burn at most 56 dots at a time. Thus a complete 560-dot line takes ten burn sequences.*

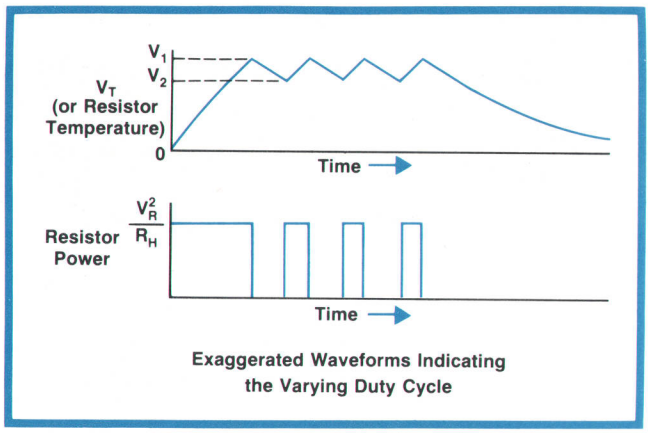


Fig. 6. *Print resistor power is controlled to heat the resistor rapidly without exceeding a reliable burn temperature. Temperature is maintained by pulse modulating the resistor power.*

require more than two burn sequences.

To implement this "variable pass" architecture, as it is called, the 20-bit shift register with latching driver chip was developed. The design allows random access to any of the 560 print resistors, up to 56 resistors per print sequence. Time is saved by loading the data for the next burn sequence into the shift registers while the current information in the latches is being printed.

Reliability of the thin-film print resistors is a function of the printing temperature, especially above 400°C. Therefore, an electrical analog circuit was designed that simulates accurately the thermal response of a thin-film resistor during a print sequence. Used in a control system, this circuit allows the resistor to heat up rapidly without exceeding a reliable burn temperature, then maintains the appropriate burn temperature through the remainder of the print sequence by pulse modulating the power to the print resistor (see Fig. 6). Also to enhance reliability, a printhead failure protection circuit senses several adverse conditions that might cause print resistor or driver chip failure and takes appropriate action to prevent damage to these components.

Microprocessor Control

The printer's intelligence resides in the hardware and firmware surrounding the microprocessor. The firmware in one HP 16K-bit program ROM provides many of the printer's features. Another HP 16K-bit character ROM provides 128 ASCII characters and control codes. This ROM can be changed to provide foreign language character sets, including German, French, Spanish, and Katakana. 256 bytes of RAM are used as a buffer to store the next line of characters to be printed and as a buffer for user-defined characters and user-defined character strings used in the character replace feature.

New Printhead Technology Developed for System 45

by Eugene R. Zeller

Design objectives for the System 45 printer required a page-width printhead that had uniformly spaced resistors along its length and was capable of a high print rate with excellent print quality.

To realize the page-width objective, several alternatives were considered, including a scanning printhead mechanism, a stationary printhead consisting of several smaller printhead segments joined end to end, and a single monolithic printhead that would span the entire page width. The single monolithic printhead approach was chosen because it is mechanically superior to the other two alternatives. The monolithic head has no moving parts, eliminating elaborate mechanisms. Furthermore, the monolithic head is very easily assembled into the final product. The segmented printhead approach, which also eliminates a complex mechanism, has the undesirable requirement of matching segments for similar resistance values, necessary for uniform print quality across the printed page, and the requirement of precise mechanical alignment of these printhead segments. The monolithic printhead approach also increases the serviceability of the printhead assembly by eliminating delicate alignment and adjustment procedures.

A second product objective, uniformly spaced resistors across the entire printhead, was established by the need to provide hard-copy graphics output. In previous thermal printer designs, a character graphics capability was provided. Character graphics means that line segments can be formed by a combination of dots contained in a 5×7 or 7×9 character dot matrix. The limitation with character graphics is that there are gaps between characters, typically equal to the physical space of two print resistors or printed dots. The two-dot gap in previous printhead designs was used for the return current path. As Fig. 1 shows, the common conductor attachment is routed between the groups of five print resistors that provide the 5×7 dot character matrix. The 9845A graphics could not tolerate these two-dot gaps every fifth dot, so the printhead had to provide a continuous row of uniformly spaced dots (see Fig. 2).

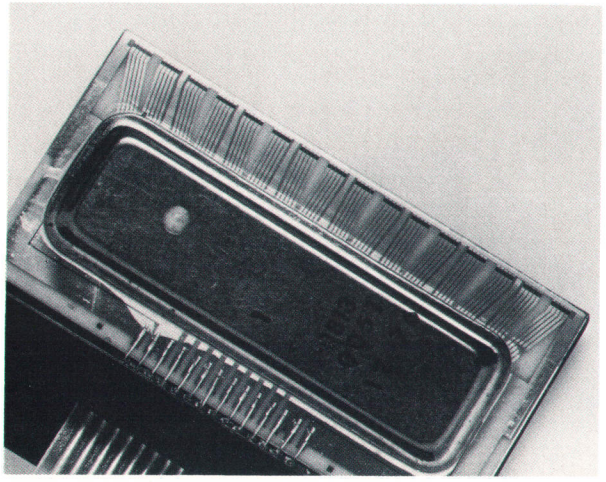


Fig. 1. *In previous printheads used only for alphanumeric printing there are spaces between groups of print resistors that correspond to the two-dot space between characters. These spaces were used for the return path for the resistor current.*

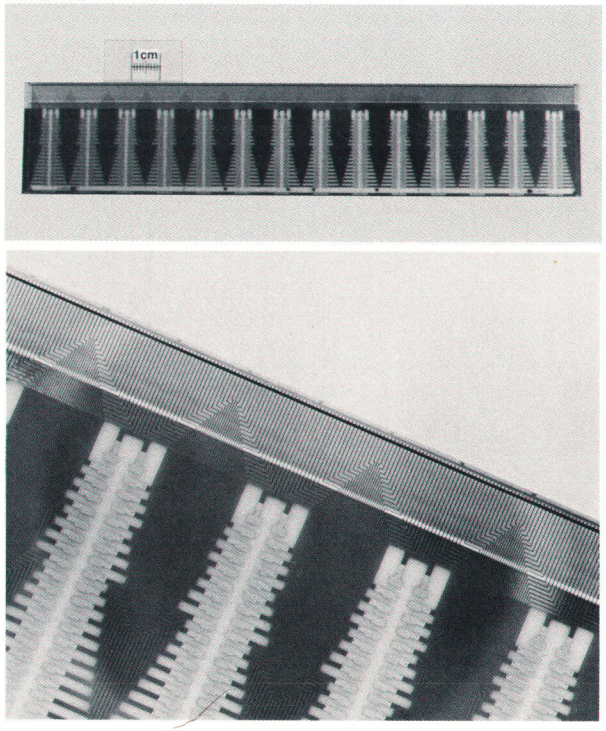


Fig. 2. *System 45's printhead has no gaps between resistors, so it can print continuous rows of dots.*

The obvious problem that arises is how to make electrical contact to the common side of the resistor dots, since the space between dots is not large enough for a conductor. To further complicate the situation, the product objectives stated that the last line printed had to be visible to the user without advancing the platen beyond the next line to be printed. This precludes designing the printhead with a wide electrical bus parallel to the row of resistor dots. The wide bus would be needed to minimize the series resistance between the power supply and the resistors being energized. The nominal print resistor resistance is 100Ω and at most 56 resistors can be energized simultaneously, so the equivalent load resistance could be as low as 1.78Ω. Therefore the equivalent series resistance must be considerably less than 1.78Ω so that the print quality remains constant independent of the number of dots simultaneously energized. Superior print quality demands thoroughly developed dots, which implies uniform dot-to-dot energy density.

These objectives and constraints called for an innovative solution. The approach chosen was to make electrical contact to the back of the printhead substrate. The common bus is deposited mostly on the side opposite the row of resistors in a three-sided deposition. The power supply is connected to the back of the substrate. The current proceeds from the power supply contact, up along the metallized back, over the top of the substrate and into the resistors, as shown in Fig. 3.

The three-sided deposition technique had to be developed for this product. In thin-film deposition, the fewer vacuum

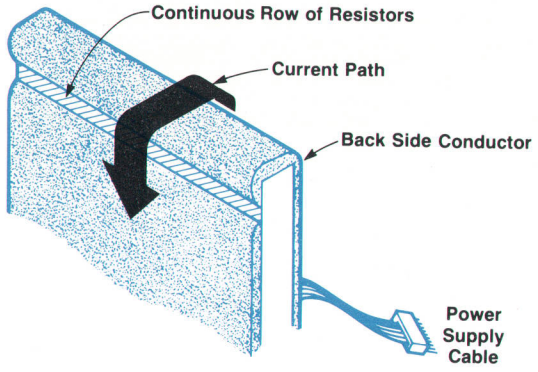


Fig. 3. *Current path in the System 45 printhead is from the back of the printhead over the edge.*

pumpdowns that are required, the less expensive the part. The approach used here is to fabricate the printhead to the point where individual resistors are formed. The printhead is then placed into the vacuum chamber. Controlling the deposition chamber gas pressure, the deposition target power, and the target-to-substrate distances causes the conductor film to be deposited onto three surfaces simultaneously. Thus, the objectives of the last line's being visible and uniformly spaced dots are achieved.

The objective of high speed made thin-film technology necessary. Thin-film resistors have a low thermal mass compared to thick-film techniques and are therefore inherently faster. Thin-film technology also provides extremely uniform resistors, which are required for uniform print quality.

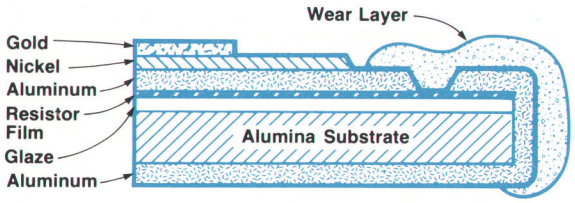


Fig. 4. *Cross-section of the thin-film System 45 printhead.*

The serviceability objective, which led to placing the driver chips onto a separate chip module that is electrically connected to the printhead via compressible connectors, required that the printhead and chip modules contain a metal that would provide a reliable pressure contact. This metal had to be metallurgically compatible with the films already present on the printhead. Since the printhead conductor films consisted of aluminum and the metal chosen for the pressure connection was gold, some form of interface layer was required to separate these two films, which have metallurgical difficulties. Nickel was selected; it is deposited along with the initial resistor conductor films. The printhead film cross-section is shown in Fig. 4.

Acknowledgments

Frank Cloutier determined the specific metallurgy of the head, while Larry Choate was responsible for the resistor film and the design of the mask aligner. Our thanks go to Lee Gilbert and his work with photolithography and mask fabrication. Ulrich Hess contributed significantly to the wear layer, reliability, and photolithography areas. Clarke Echols designed the probing system for measuring the resistance values.

The user-defined character feature allows the user to define up to nine characters at a time with any dot pattern in a 7×8 matrix. The character replace feature allows the user to replace a given character with a string of characters that may include executable control characters.

The user can select the spacing between lines of print to alter the trade-offs among legibility, print speed, and paper use. A top-of-form switch just above the keyboard allows the user to space to the top of the next form. There are two highlighting features, underline and 150% high characters. There are also hardware tabbing features in addition to the keyboard-initiated software tabbing features to assist in formatted data output. The top margin is also selectable.

An optical out-of-paper sensor causes a flag to be sent to the mainframe peripheral processing unit, halting data transmission until paper is added. Many other internal functions are controlled by the microprocessor, such as optimizing the timing between print burns and paper advance to maximize speed and minimize peak power.

Motor Drive

The motor drive circuitry uses a ministep drive concept that assures sufficient stepping torque and acceptable stepping time, while at the same time minimizing audible noise caused by high accelerations while advancing the paper.

The ministep drive breaks each 1.8° step of the step motor into eight smaller steps (0.225°) for 1600 steps per revolution. The microprocessor controls the stepping commands as necessary to accelerate, maintain speed, or stop.

A block diagram of the ministep drive is shown in Fig. 7. In a normal "one phase on" step motor drive the current would be switched in an A, B, $\overline{A}$, $\overline{B}$, A sequence. In the ministep drive the current in the A winding is slowly stepped down while the current in the B winding is slowly stepped up. The current is modulated at each current step by the modulation circuit. Electrically, the windings are at right angles to each other, so the torque is related to the square root of the sum of the squares of the A and B currents. Therefore, the current steps are discrete approximation to sine and cosine curves, as shown in Fig. 8. The

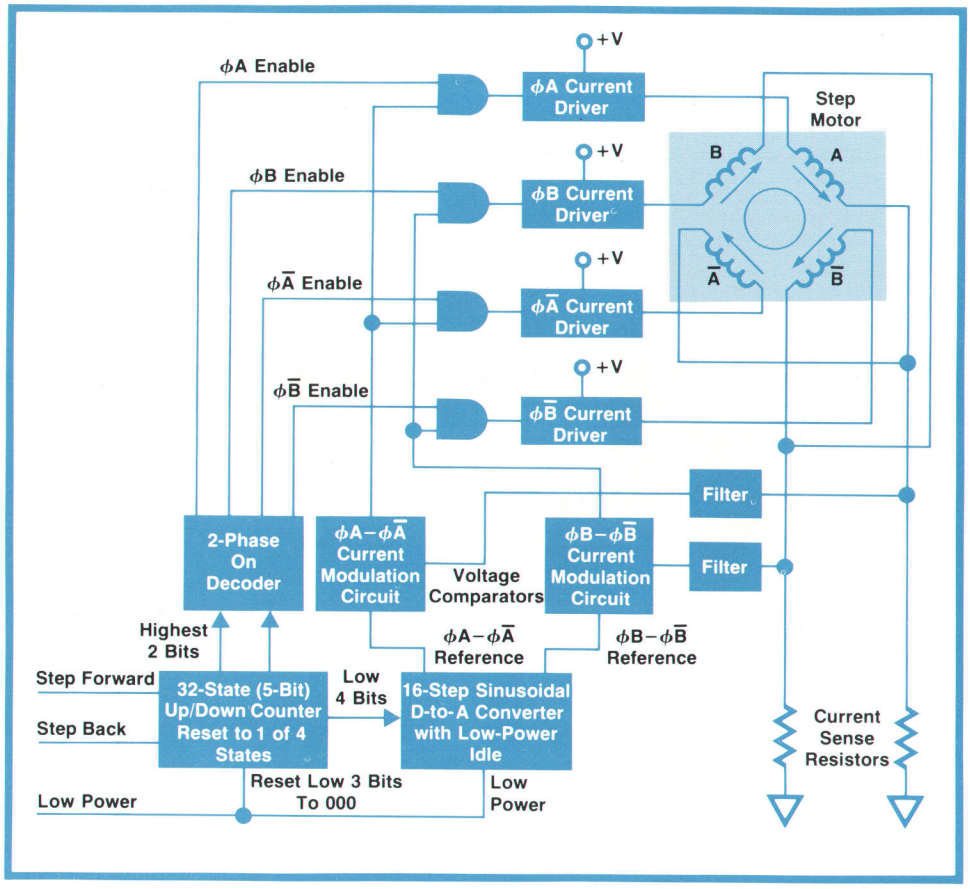


Fig. 7. *Ministep paper-advance motor drive system minimizes audible noise. Under microprocessor control, the drive system breaks each step-motor step into eight smaller steps, and slowly steps down the A-phase current while it steps up the B-phase current.*

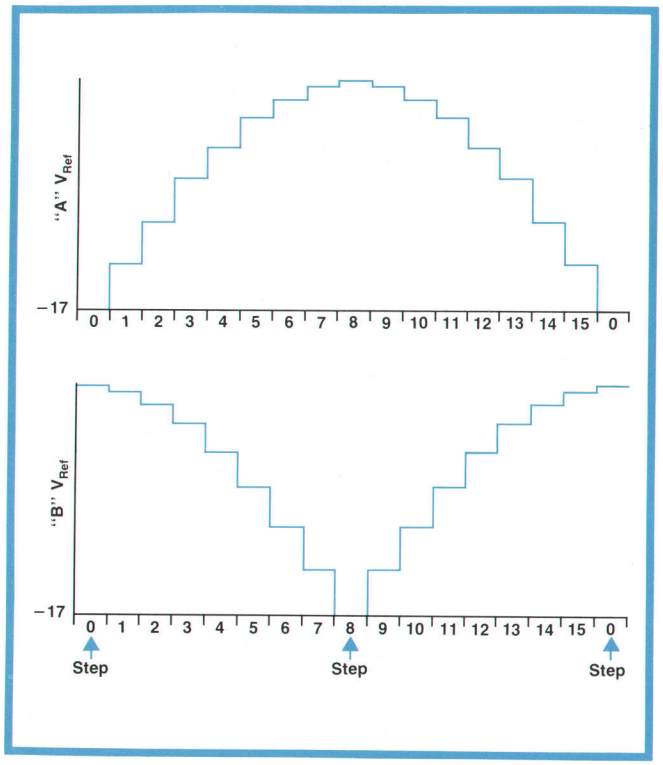


Fig. 8. *A and B step-motor current waveforms are discrete approximations to sine waves.*

sine and cosine curves correspond to the reference voltages output from the sinusoidal digital-to-analog converter to the current modulation circuit, as shown in Fig. 7.

The control algorithm for stepping between rows of printed dots is stored in ROM and appropriate step commands are given by the microprocessor at the proper times in the step. Fig. 9 shows a typical algorithm for taking a single step (eight ministeps). The sequence is as follows.

a. Four ministeps are given quickly to accelerate the rotor to the desired speed.
b. The angular velocity is maintained by successive ministeps.
c. The rotor catches up to the field position.
d. The rotor overshoots the field position and decelerates because of "spring coupling" to the field.
e. When the rotor reaches zero velocity it is locked in position with the final ministep.

One feature of the ministep drive is its ability to go to a low-power mode when not used and to reset to full step position by resetting the lower three bits of the 32-state counter.

Paper Options

A black option of the thermal paper is available in two widths and is perforated to provide U.S. standard

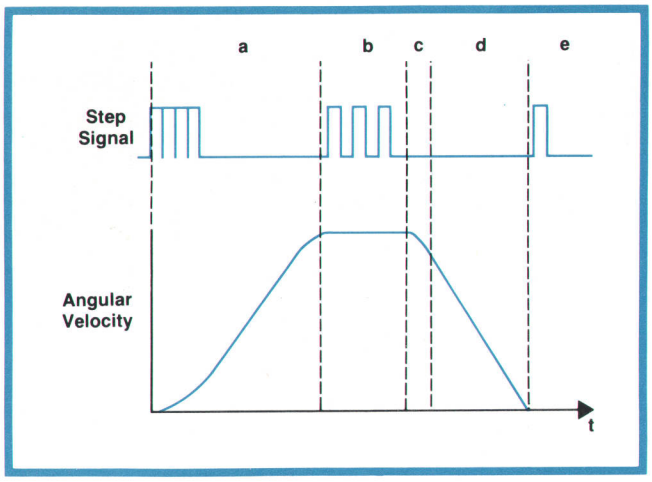


Fig. 9. *A typical microprocessor-controlled algorithm for taking a single step (eight ministeps).*

8½×11-in sheets or metric standard 21×29.7-cm sheets. The more economical blue paper option is available in the two widths but is not perforated. The printout is in a maximum field size of 18.5 cm and is formatted on the page to allow a 2-cm margin on the left side for hole punching.

Acknowledgments

Marty Wilson, lead engineer, developed the printhead architecture and coordinated the trade-offs relating to speed and power. Wally Wahlen designed the printhead driver chip and temperature control electronics. Bob Kuseski developed the printer control electronics and the related software. Dave Conner optimized the paper advance motor characteristics and head protection scheme, while Dale Harlan implemented the paper feeding and tracking mechanism.

Eugene R. Zeller

Gene Zeller, project manager in HP's Loveland, Colorado R&D lab, earned his BSEE degree in 1969 from the University of Wisconsin in Madison, joining HP that year. He also holds an MSEE from Colorado State University, Fort Collins, earned in 1972. Gene has worked on three peripheral projects for 9100A/B Calculators (9160A, 9101A, 9102A), and did memory system design for the 9810A and 9820A Desktop Computers. He was also the logic designer for the 9871A Printer. More recently, he served as project manager in NMOS II and thin-film process development, the latter for the thin-film printhead for the 9845A's thermal printer. For Gene, recreation time is spent at bicycle touring, softball, carpentry, and making hardwood cabinets. He also serves as president of his church council. A native of Rockford, Illinois, he makes his home in Loveland with his wife and two daughters.

Ray J. Cozzens

Ray Cozzens joined HP in June of 1969, shortly after graduating from Georgia Institute of Technology with a BME degree. Currently he is section manager for IC and thin-film production engineering. He has been project manager on thin-film process development for the 9815A and 9825A computer printers and for several phases of System 45 as it was developed, including the keyboard, CRT, graphics, industrial design, printer/plotter, and product design. Ray worked in the Colorado Springs Division for a short time on a special task force in thick-film process development. He was also involved in the development of the 9866A Printer, carrying the project through production. Ray is a sports enthusiast, with interests in water skiing, snow skiing, canoeing, and baseball. His latest creative endeavor is in woodworking. Ray was born in Canton, Ohio. He and his wife now live in Loveland and have two daughters, ages 7 and 5.

SPECIFICATIONS
HP Model 9845A Computer

DYNAMIC RANGE: -10^{99} to -10^{-99}, 0, $+10^{-99}$ to $+10^{99}$.
INTERNAL CALCULATION RANGE: -10^{511} to -10^{-511}, 0, 10^{-511} to 10^{511}.
SYSTEM 45 READ/WRITE MEMORY
STANDARD: 13 498 bytes.
OPT. 201: 29 882 bytes.
OPT. 203: 62 650 bytes.
The standard read/write memory contains 16 384 bytes, 13 498 directly available to the user.
TAPE CARTRIDGE
CAPACITY: 217K bytes.
ACCESS: Directory, file-by-name.
SEARCH SPEED (bidirectional): 2 286 mm/s (90 in/s).
AVERAGE TRANSFER RATE: 1 440 bytes/s.
CARTRIDGE SIZE: 63.5 × 82.5 × 12.7 mm (2.5 × 3.25 × 0.5 in).

CRT Specifications

GENERAL
SCREEN SIZE: 261 × 193 mm (10.3 × 7.6 in).
310-mm (12.2-in) diagonal.
SCREEN BRIGHTNESS: manually adjustable from 12-30 ft-lamberts
REFRESH RATE: 60 Hz (independent of line frequency).
TUBE PHOSPHOR: P31.

ALPHANUMERIC MODE
SCREEN CAPACITY: 24 lines × 80 characters (1 920 characters).
RASTER SCAN SIZE: 236 × 122.94 mm (9.3 in. × 4.84 in).
CHARACTER GENERATION: 7 × 9 character font in a 9 × 15 character cell.
STANDARD CHARACTER SET: 128 ASCII characters.
OPTIONAL CHARACTER SETS: French, Spanish and German.
CURSOR: Blinking underline.
GRAPHICS MODE (OPTIONAL HARDWARE)
RASTER SIZE: 200 × 162.5 mm (7.9 × 6.4 in).
DOT RESOLUTION: 0.357 mm (0.014 in).
SPOT SIZE: 0.254 mm (0.010 in).
CURSOR: Full screen or blinking crosshair in plotting mode, blinking underline in letter mode.
DISPLAY SPEED: 2 032 mm (80 in.)/s.
LINEARITY: <1.5% full screen.

ENVIRONMENTAL RANGE
OPERATING TEMPERATURE: 5°C to 40°C.
STORAGE TEMPERATURE: −40°C to +65°C.
RELATIVE HUMIDITY: 5% to 80% at 40°C.

Thermal Line Printer Specifications

PRINT SPEED: up to 480 lines/min.
PLOT SPEED
NORMAL MODE: 25.4 mm/s (1 in/s).
CRT TRANSFER: 3.5-25 mm/s (0.14-1 in/s).
NOISE LEVEL: 8 hr, 100% duty cycle: 68 dBA (max).
0.5 hr, 100% duty cycle: 55 dBA (avg).
PAPER FEED: Automatic load.

System Size/Weight

SIZE: HWD 483 × 457 × 667 mm (19 × 18 × 26.3 in.).
WEIGHT: 18.6 kg (41 lb) standard mainframe.
10.43 kg (23 lb) standard CRT.
5.22 kg (11.5 lb) optional thermal printer
CUBE: 0.5 m^3 (17 ft^3).
PRICE IN U.S.A.: 9845A base price, $11,500.
MANUFACTURING DIVISION: CALCULATOR PRODUCTS DIVISION
815 Fourteenth Street, S.W.
Loveland, Colorado, 80537 U.S.A.

Personal Calculator Algorithms IV: Logarithmic Functions

A detailed description of the algorithms used in Hewlett-Packard hand-held calculators to compute logarithms.

by William E. Egbert

BEGINNING WITH THE HP-35,[1,2] all HP personal calculators have used essentially the same algorithms for computing complex mathematical functions in their BCD (binary-coded decimal) microprocessors. While improvements have been made in newer calculators,[3] the changes have affected primarily special cases and not the fundamental algorithms.

This article is the fourth in a series that examines these algorithms and their implementation.[4,5,6] Each article presents in detail the methods used to implement a common mathematical function. For simplicity, rigorous proofs are not given and special cases other than those of particular interest are omitted.

Although tailored for efficiency within the environment of a special-purpose BCD microprocessor, the basic mathematical equations and the techniques used to transform and implement them are applicable to a wide range of computing problems and devices.

The Logarithmic Function Algorithm

This article will discuss the method of generating the ln(x) and $\log_{10}(x)$ functions. To minimize program length, a single function, ln(x), is always computed first. Once ln(x) is calculated, $\log_{10}(x)$ is found by the formula

$$\log_{10}(x) = \frac{\ln(x)}{\ln(10)}.$$

ln(x) is generated using an approximation process much the same as the one used to compute trigonometric functions.[5] The fundamental equation used in this case is the logarithmic property that

$$\ln(a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_n) = \ln(a_1) + \ln(a_2) + \ln(a_3) + .. + \ln(a_n) \quad (1)$$

This algorithm simply transforms the input number x into a product of several terms whose logarithms are known. The sum of the logarithms of these various partial-product terms forms ln(x).

Exponent

Numbers in HP calculators are stored in scientific notation in the form $x = M \cdot 10^K$. M is a number whose magnitude is between 1.00 and 9.999999999 and K is an integer between −99 and +99. Using equation 1, it is easy to see that

$$\ln(M \cdot 10^K) = \ln(M) + \ln(10^K)$$

At this point, another logarithmic property becomes useful, which is

$$\ln(A^b) = b \cdot \ln(A).$$

Using this relationship

$$\ln(M \cdot 10^K) = \ln(M) + K \cdot \ln(10).$$

Thus to find the logarithm of a number in scientific notation, one calculates the logarithm of the mantissa of the number and adds that to the exponent times ln(10).

Mantissa

The problem of finding ln(x) is now reduced to finding the logarithm of its mantissa M.

Let P = 1/M. Then

$$\begin{aligned} \ln(PM) &= \ln(P) + \ln(M) \\ \ln(1) &= \ln(P) + \ln(M) \\ 0 &= \ln(P) + \ln(M) \\ -\ln(P) &= \ln(M) \end{aligned} \quad (2)$$

This may appear to be a useless exercise since at first glance −ln(P) seems to be as hard to compute as ln(M).

Suppose, however, that a new number P_n is formed by multiplying P by r which is a small number close to 1.

$$P_n = P \cdot r$$

In addition, let P_n be defined as a product of powers

of numbers a_j whose natural logarithms are known.

$$P_n = a_0^{K_0} \cdot a_1^{K_1} \cdot \ldots \cdot a_j^{K_j} \cdot \ldots \cdot a_n^{K_n}$$

Thus

$$P = P_n/r$$

$$\ln(P) = \ln(P_n) - \ln(r)$$

Using equation 2

$$\ln(M) = \ln(r) - \ln(P_n)$$

Finally

$$\ln(M) = \ln(r) - (K_0\ln(a_0) + K_1\ln(a_1) + \ldots + K_j\ln(a_j) + \ldots + K_n\ln(a_n))$$

Thus to find ln(M) one simply multiplies M by the carefully selected numbers a_j so that the product MP_n is forced to approach 1. If all the logarithms of a_j are added up along the way to form $\ln(P_n)$ then ln(M) is the logarithm of the remainder r minus this sum. Notice that the remainder r is nothing more than the final product MP_n.

Implementation

How is this algorithm implemented in a special-purpose microprocessor? First of all, the terms of P_n were chosen to reduce computation time and minimize the amount of ROM (read-only memory) needed to store a_j and its logarithm. The numbers chosen for the a_j terms are of the form $a_j = (1 + 10^{-j})$, where j = 0-4 (see Table 1).

Table 1 Values of a_j Terms

j	a_j	ln a_j
0	2	0.6931
1	1.1	0.09531
2	1.01	0.009950
3	1.001	0.0009995
4	1.0001	0.000099995

To achieve high accuracy using relatively few a_j terms, an approximation is used when $r = MP_n$ approaches 1. For numbers close to 1, $\ln(r) \approx r-1$. This yields

$$\ln M \approx (r-1) - \sum_{j=0}^{n} K_j\ln(a_j) \qquad (3)$$

Since all of the a_j terms are larger than 1, M must be between 0 and 1 if the product P_nM is to approach 1. As M is defined to be between 1 and 10, a new quantity A is formed by dividing M by 10. A is now in the proper range ($0.1 \leq A < 1$) so that using the a_j terms as defined will cause the product AP_n to approach 1 without exceeding 1.

The product P_n can now be formally defined as a series, where j goes from 0 to n. Each partial product AP_j has the form

$$A \cdot P_j = A \cdot P_{j-1}(1+10^{-j})^{K_j}, \; j = 0, 1, 2, \ldots, n$$

$P_{-1} = 1$, and K_j is the largest integer such that $P_j < 1$.

In practice, each $A \cdot P_j$ is formed by multiplying $A \cdot P_{j-1}$ by $(1 + 10^{-j})$, K_j times. There is one intermediate product, T_i, for each count of K_j, as shown below.

$$T_0 = A(1 + 10^{-0})^1$$

$$T_1 = A(1 + 10^{-0})^2$$

$$T_{K_0} = A(1 + 10^{-0})^{K_0}$$

$$T_{K_0+1} = A(1 + 10^{-0})^{K_0}(1 + 10^{-1})^1$$

$$T_m = A(1 + 10^{-0})^{K_0}(1 + 10^{-1})^{K_1} \ldots (1 + 10^{-n})^{K_n} = AP_n$$

$$m = K_0 + K_1 + \ldots + K_n$$

$$T_i = T_{i-1}(1 + 10^{-j}) \text{ for some } j \qquad (4)$$

Notice that each multiplication of the intermediate product T_{i-1} by a_j simply amounts to shifting T_{i-1} right the number of digits denoted by the current value of j and adding the shifted value to the original T_{i-1}. This very efficient multiplication method is similar to the pseudo-multiplication of the trigonometric algorithm.[5]

An Example

A numeric example to illustrate this process is now in order. Let A = 0.155. To compute ln(A), A must be multiplied by factors of a_j until AP_n approaches 1. To begin the process A = 0.155 is multiplied by $a_0 = 2$ to form the intermediate product $T_0 = 0.31$. Another multiplication by a_0 gives $T_1 = 0.62$. A third multiplication by 2 results in 1.24, which is larger than 1. Thus $K_0 = 2$ and $AP_0 = 0.62$. The process is continued in Table 2.

Table 2 Generation of ln(0.155)

j	a_j	AP_j	K_j	T_i	$\ln(a_j)$
−1		0.155		0.155	
0	2		1	0.31	0.6931
0	2	0.62	2	0.62	0.6931
					*
1	1.1		1	0.682	0.0953
1	1.1		2	0.7502	0.0953
1	1.1		3	0.82522	0.0953
1	1.1		4	0.9077	0.0953
1	1.1	0.9985	5	0.9985	0.0953
2	1.01	0.9985	0		**
3	1.001	0.9995	1	0.9995	0.00099
4	1.0001	0.9996	1	0.9996	0.00009

$$0.9996 = A \cdot P_4 = r \qquad 1.8638 = \sum \ln(a_j)$$

*Another ×2 would result in $AP_3 > 1$. Thus a_j is changed to 1.1.
**The 1.01 constant is skipped entirely.

Applying the values found in Table 2 to equation 3 results in

$$\ln(0.155) = (0.9996 - 1) - 1.8638$$
$$= -1.8642$$

This answer approximates very closely the correct 10-digit answer of −1.864330162.

This example demonstrates the simplicity of this method of logarithm generation. All that is required is a multiplication (shift and add) and a test for 1. To implement this process using only three working registers, a pseudo-quotient similar to the one generated in the trigonometric algorithm is formed.[5] Each digit represents the number of successful multiplications by a particular a_j. For the preceding example, the pseudo-quotient would be

2	5	0	1	1
↑	↑	↑	↑	↑
j = 0	j = 1	j = 2	j = 3	j = 4

With $-\ln(r) = (r - 1)$ as the first term, the appropriate logarithms of (a_j) are then summed according to the count in the pseudo-quotient digit corresponding to the proper a_j. The final sum is $-\ln(A)$.

At this point one more transformation is needed to optimize this algorithm perfectly to the microprocessor's capabilities. Recall that the factors a_j were chosen to force the product P_nA towards 1. Suppose $B_i = T_i - 1$. Forcing B_m towards 0 causes P_nA to be forced to 1. Substituting B_i into (4) and simplifying yields

$$(B_i + 1) = (B_{i-1} + 1)(1 + 10^{-j}) \text{ for some } j$$

$$B_i + 1 = B_{i-1}(1 + 10^{-j}) + 1 + 10^{-j}$$

$$B_i = B_{i-1}(1 + 10^{-j}) + 10^{-j}$$

Multiplying through by −1 results in the following equation, which is equivalent to equation 4.

$$-B_i = -B_{i-1}(1 + 10^{-j}) - 10^{-j} \text{ for some } j \qquad (5)$$

This expression is now in a very useful form, since the a_j term is the same as before, but the zero test is performed automatically when the 10^{-j} subtraction is done. A test for a borrow is all that is required. An additional benefit of this transformation is that accuracy can be increased by shifting $-B_i$ left one digit for each a_j term after it has been applied the maximum number of times possible. This increases accuracy by replacing zeros generated as B_i approaches zero with significant digits that otherwise would have been lost out of the right end of the register. This shifting, which is equivalent to a multiplication by 10^j, gives yet another benefit. Multiplying equation 5 by 10^j and simplifying,

$$-B_i \times 10^j = (-B_{i-1}(1 + 10^{-j}) - 10^{-j}) \times 10^j$$

$$-B_i \times 10^j = -B_{i-1} \times 10^j (1 + 10^{-j}) - 1 \text{ for some } j \qquad (6)$$

Notice that the 10^{-j} subtraction reduces to a simple −1 regardless of the value of j. The formation of the initial $-B_0$ is also easy since $-B_0 = -(A - 1) = 1 - A$. This is formed by taking the 10's complement of M (the original mantissa), creating 10 − M. A right shift divides this by 10 to give $1 - M/10 = 1 - A = -B_0$. A final, almost incredible, benefit of the B_i transformation is that the final remainder $-B_m \times 10^j$ is in the exact form required to be the first term of the summation process of equation 4 without further modification. The correct $\ln(a_j)$ constants are added directly to $-B_m \times 10^j$, shifting the sum right one digit after each pseudo-quotient digit to preserve accuracy and restore the proper normalized form disrupted by equation 6. The result is $-\ln(A)$.

Finally, the required ln(M) is easily found by subtracting the computed result $-\ln(A)$ from ln(10).

$$\ln(10) - (-\ln(A)) = \ln(10) + \ln(M/10)$$
$$= \ln(10 \cdot M/10)$$
$$= \ln(M)$$

Once ln(M) is computed, $K \cdot \ln(10)$ is added as previously discussed to form ln(x). At this point log(x) can be generated by dividing ln(x) by ln(10).

Summary

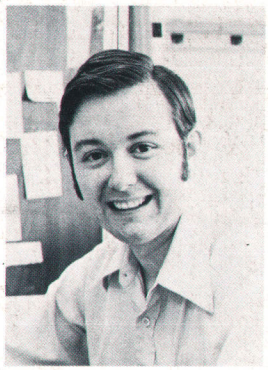
In summary, the compulation of logarithmic functions proceeds as follows:

1. Find the logarithm of 10^K using $K \cdot \ln(10)$.
2. Transform the input mantissa to the proper form required by $-B_0$.
3. Apply equation 6 repeatedly and form a pseudo-quotient representing the number of successful multiplications by each a_j.
4. Form $-\ln(A)$ by summing the $\ln(P_j)$ constants corresponding to the pseudo-quotient digits with the remainder $-B_m \times 10^j$ as the first term in the series.
5. Find ln(x) or log(x) using simple arithmetic operations.
6. Round and display the answer.

The calculator is now ready for another operation.

References

1. T.M. Whitney, F. Rodé, and C.C. Tung, "The 'Powerful Pocketful': An Electronic Calculator Challenges the Slide Rule," Hewlett-Packard Journal, June 1972.
2. D.S. Cochran, "Algorithms and Accuracy in the HP-35," Hewlett-Packard Journal, June 1972.
3. D.W. Harms, "The New Accuracy: Making $2^3 = 8$," Hewlett-Packard Journal, November 1976.
4. W.E. Egbert, "Personal Calculator Algorithms I: Square Roots," Hewlett-Packard Journal, May 1977.
5. W.E. Egbert, "Personal Calculator Algorithms II: Trigonometric Functions," Hewlett-Packard Journal, June 1977.
6. W.E. Egbert, "Personal Calculator Algorithms III: Inverse Trigonometric Functions," Hewlett-Packard Journal, November 1977.



William E. Egbert

Bill Egbert is a project manager at HP's Corvallis, Oregon Division. He produced this series of algorithm articles as part of his work on the HP-67 and HP-97 Programmable Calculators. He was project leader for the HP-67 and did microprogramming for both calculators. More recently, he was project leader for the firmware development of the HP-19C and the HP-29C. Bill received his BSEE degree from Brigham Young University in 1973 and his MSEE from Stanford University in 1976. He's been with HP since 1973. Born in Fallon, Nevada, he's married, has two small children, and lives in Corvallis.

Hewlett-Packard Company, 1501 Page Mill Road, Palo Alto, California 94304

HEWLETT-PACKARD JOURNAL

APRIL 1978 Volume 29 • Number 8

Technical information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Central Mailing Department
Van Heuven Goedhartlaan 121
Amstelveen-1134 The Netherlands
Yokogawa-Hewlett-Packard Ltd., Shibuya-Ku
Tokyo 151 Japan

Editorial Director • Howard L. Roberts
Managing Editor • Richard P. Dolan
Art Director, Photographer • Arvid A. Danielson
Illustrator • Susan E. Wright
Administrative Services, Typography • Anne S. LoPresti
European Production Manager • Dick Leeksma

HP Archive

On-line curator: John Miles, KE5FX

jmiles@pop.net