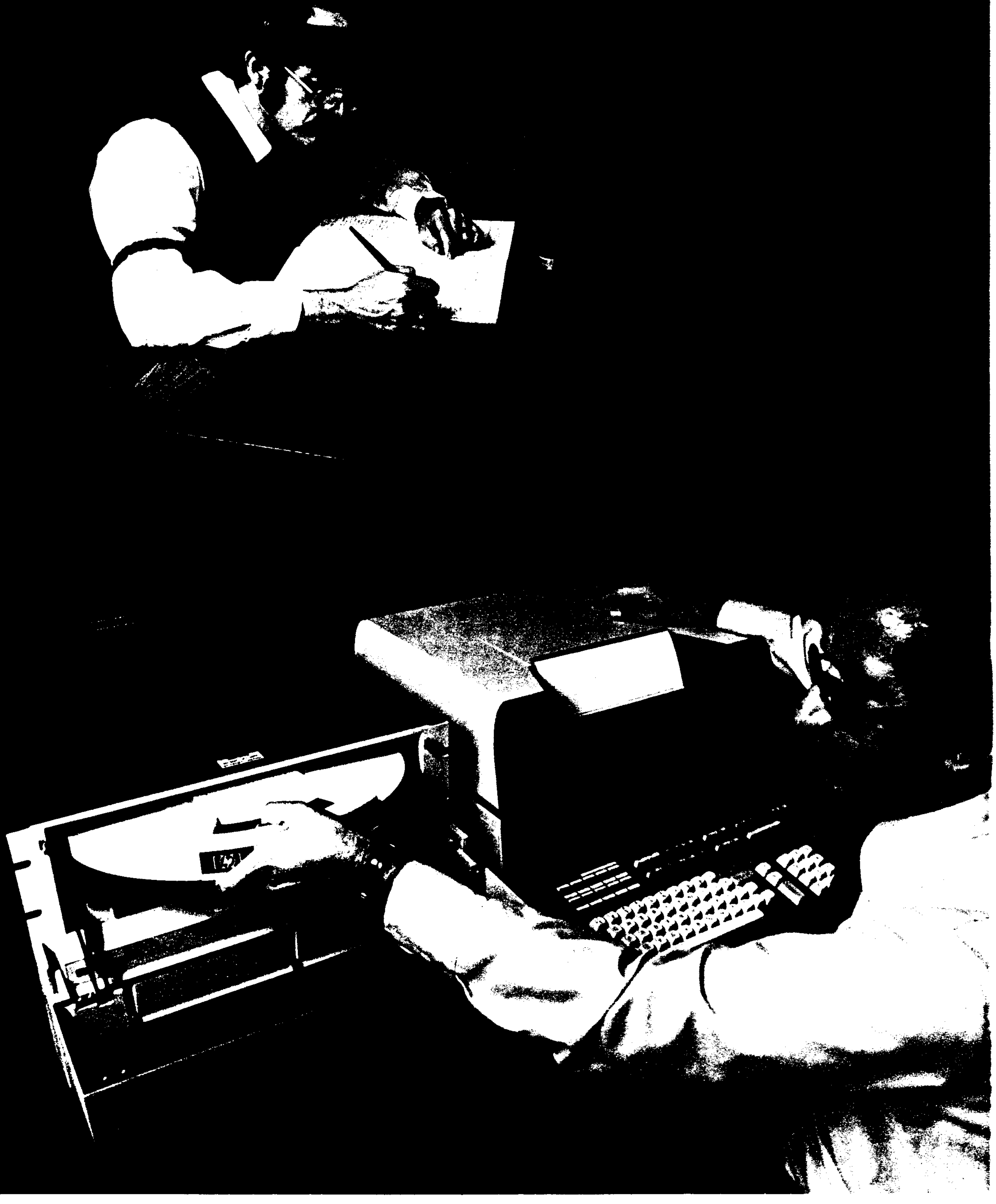
HEWLET



Mass Memory System Broadens Calculator Applications

Inventory control, payroll, order processing, and other large-data-base applications are now within the capability of HP's most powerful desktop calculator, the BASIC-language Model 30.

by Havyn E. Bradley and Chris J. Christopher

MODEL 30 OF HEWLETT-PACKARD'S 9800 Series Calculators, or Model 9830A if you prefer, ranks among the world's most powerful desktop calculators.[1] Its BASIC-language programming, along with alphanumeric string and matrix manipulation, give it tremendous data handling capability. As a result, many users have developed applications that require storage capacity much larger than Model 30's standard tape cassette can provide. Typically, these applications also call for random data access, and therefore are not conveniently run on any magnetic tape system. Such applications include inventory control, payroll processing, order processing, account maintenance and others.

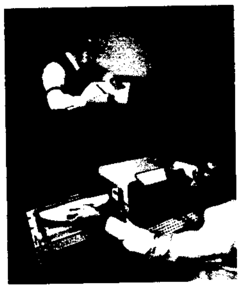
The most obvious answer is a disc drive, and two are now available as peripherals to the 9830A. The drives are Hewlett-Packard 7900 Series Disc Drives[2] that have been slightly modified and given new model numbers. Model 9867A has one removable disc cartridge. Model 9867B has one fixed disc and one removable cartridge. Each fixed disc or cartridge can store 2.4 million 8-bit bytes of data. Data access is rapid and, most important, the drives have proven high data-handling reliability.[3]

On the assumption that a calculator-based system should have a command set that is flexible and powerful, yet easy to use, the FILES commands of HP 2000C time-shared BASIC were implemented for the new system. This means that users can take advantage of the large library of programs for HP 2000 Time-Shared Computer Systems. Only minor modifications are needed to run these programs on the new calculator-based system.

Included with the new system is a comprehensive package of support software to simplify system turn-on and data backup. Close cooperation between hardware and software development teams was maintained to arrive at a reliable and versatile system at minimum cost, and to make the hardware as unobtrusive to the user as possible.

Mass Memory System

Besides the disc drive, the new mass memory system includes the elements shown in Fig. 1. A plug-in read-only memory (ROM) block is installed in one of the 9830's eight ROM slots to expand the calculator's instruction repertoire. An interface cable assembly containing interface circuitry and a small cache



Cover: *There was a time when a small business only needed one good bookkeeper. Today he usually has a staff and, more and more frequently, an automatic data processing system. A particularly painless way for a small business to get started in automated accounting, inventory control, and other large-data-base applications is the new 9880A/B Mass Memory System for the 9830A Calculator.*

In this Issue:

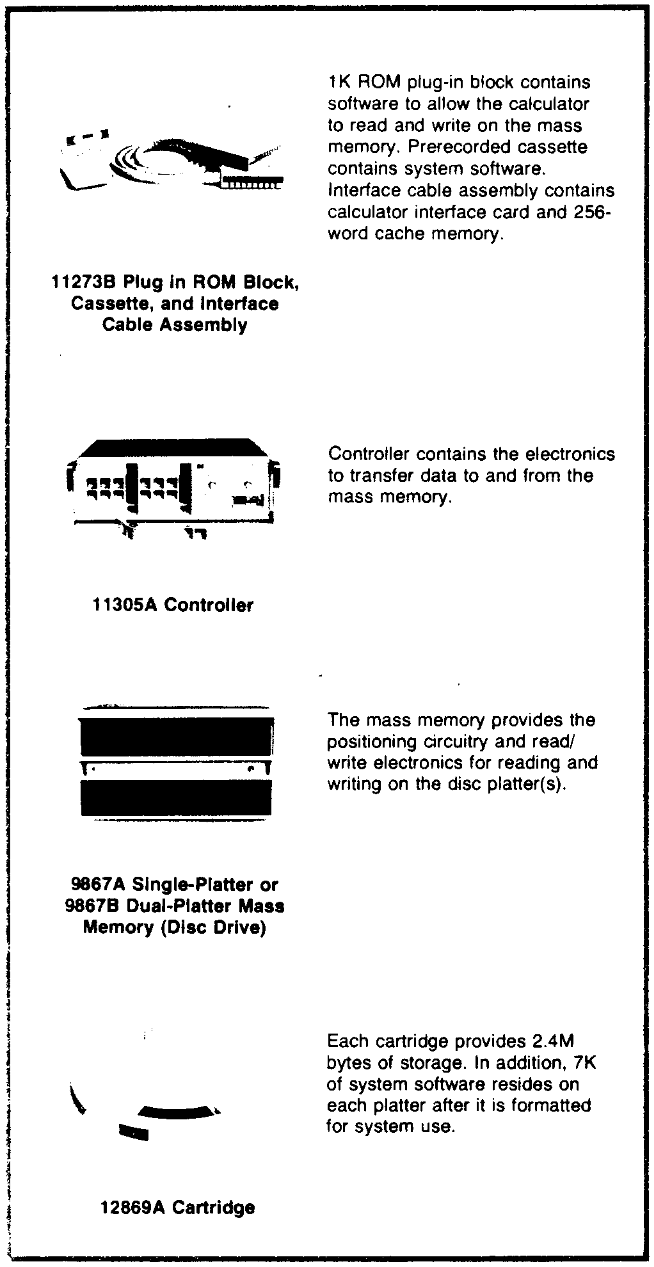

1K ROM plug-in block contains software to allow the calculator to read and write on the mass memory. Prerecorded cassette contains system software. Interface cable assembly contains calculator interface card and 256-word cache memory.
11273B Plug in ROM Block, Cassette, and Interface Cable Assembly
Controller contains the electronics to transfer data to and from the mass memory.
11305A Controller
The mass memory provides the positioning circuitry and read/write electronics for reading and writing on the disc platter(s).
9867A Single-Platter or 9867B Dual-Platter Mass Memory (Disc Drive)
Each cartridge provides 2.4M bytes of storage. In addition, 7K of system software resides on each platter after it is formatted for system use.
12869A Cartridge

Fig. 1. *Model 9880A/B Mass Memory System for Model 9830A BASIC-Language Calculator consists of the elements shown here. The FILES commands of HP 2000C time-shared BASIC are used to access the mass memory; advantages are ease of use and a large existing program library.*

memory connects the calculator to the disc drive controller. Another cable assembly connects the controller and the disc drive.

The complete system is designated Model 9880 Mass Memory System. Model 9880A includes the 9867A Mass Memory (disc drive) and Model 9880B includes the 9867B.

Fig. 2 shows the information-transfer paths in the system. All *data* transfers between the calculator and the mass memory go through the cache memory. All *addresses, commands,* and *status* information are processed through the 9830A input/output structure.[4] The basic difference is that cache-memory transfers go directly to the calculator read/write memory and are much faster than transfers through the calculator I/O structure.

The cache memory is a 256-word (512-byte) MOS read/write memory. It allows data transfer from the calculator at one rate and to the mass memory at a different rate. This avoids any synchronization or timing problems between the two components. The full 256-word contents of the cache memory are transferred during any read/write operation with the mass memory. The cache memory appears to the calculator as an extension of its own memory, but it is not accessible by the user.

Multiple Calculators and Platters

The mass memory has an extremely large storage capacity. Each disc, or platter, as it is frequently called, can store 2.4 million bytes. Therefore, one mass memory is usually enough for an individual user.

For applications that require still more storage, the 11305A Controller can handle up to four platters in the same system, giving a total storage capacity of 9.6 million bytes. This offers some important advantages to the user. For example, he can access separate data bases stored on different cartridges and process the data on another platter. It also makes duplication of important data bases very simple.

To match this storage and accessing capacity, the controller can also handle up to four 9830A Calculators in the same system (see Fig. 3). Any of the four calculators can access any of the four platters connected to the system. Each platter contains the full system software and is not dependent on any of the others for its operation.

It should be emphasized that this system is *not a time-shared system in the usual sense.* Only one calculator can access a platter at any given time. The remaining calculators in the system can access their respective cache memories during this time. The cache memory remains dedicated to the calculator, not to the controller.

It is true, however, that two or more users can be working on different applications and sharing one or more mass memories, and in most such situations, neither user would notice that someone else was using the system. An order processing system, for example, might have someone entering new orders, another preparing shipping papers and updating inventory records, and still another preparing order acknowledgments and billing customers. The new order entries could be transferred from a holding file to the main data base at the beginning of each day.

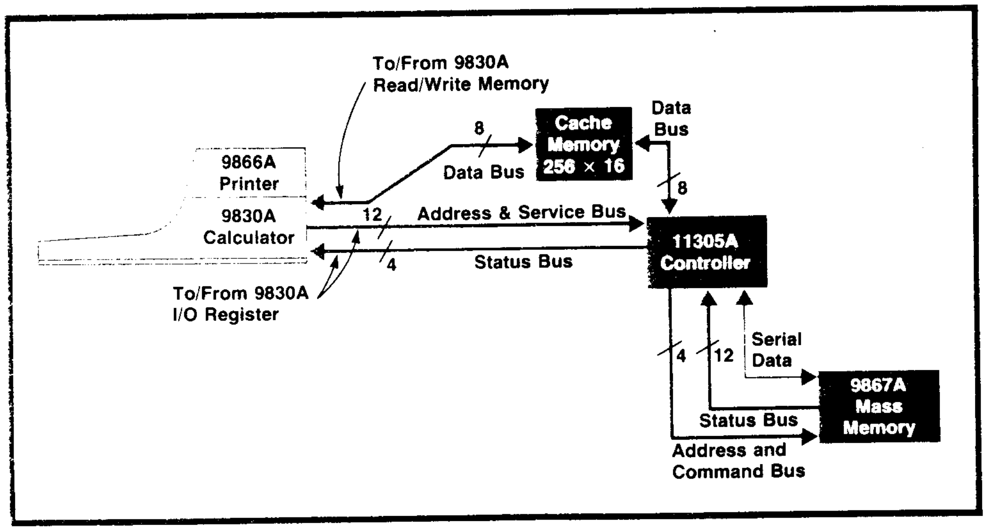


Fig. 2. *Data transfers to and from the mass memory (disc drive) go by way of a cache memory that is separate from the calculator I/O system and considerably faster.*

Software Organization

The 9830A add-on ROM structure allows the user to select specific ROM blocks to integrate a system that meets his needs.[1] The maximum number of add-on ROMs is eight. Each contains 1024 words. The total software necessary to meet the objectives of the mass memory system was estimated at approximately 7000 words. Obviously, it would be impractical to provide all this software as firmware because this would consume most of the add-on ROM capability of the calculator. This problem was solved by adapting the software organization shown in Fig. 4.

The single 1K Mass Memory ROM provides the controlling firmware and all the speed-sensitive mass memory commands. Additional mass memory commands that are to be executed from the keyboard or infrequently in a program are stored on each mass memory platter during initial turn-on. When these commands are encountered in a program or executed from the keyboard, the mass memory ROM fetches the corresponding bootstrap—i.e., software needed to accomplish the particular function—from the mass memory, transfers it to the calculator read/write memory (RWM), and executes it. A bootstrap fetch operation is accomplished within 50 milliseconds.

How are the bootstraps executed in the RWM of the calculator without destroying the user information currently residing there? During power turn-on, if the mass memory ROM block is present, the calculator allocates 300 words of RWM for mass memory operations. The 300 words are used for bootstrap execution and special mass memory buffers that facilitate the execution of the data transfer commands.

The support software is comprised of a number of binary programs stored on a magnetic tape cassette that every mass memory user will receive. These programs accomplish such functions as initial system turn-on, bootstrap verification, platter duplication, repack of mass memory files, and so on.

Platter Organization

Each platter, removable or fixed, represents a separate mass memory system. As such, each platter is independent and must have its own directory area, bootstraps, and so on. The platter organization is as follows.

Mass Memory for Business Applications

To help users put the new Mass Memory System to work as soon as it's installed, three applications program packages are now available. They are for inventory control, accounts payable, and accounts receivable.

In each package are all the programs a typical user needs for each type of application. Thus the programming effort required of the user is greatly reduced. While some users may elect to modify the programs, for example, by changing the format of a report, many will find the programs usable as is.

The Accounts Receivable and Billing package provides for orderly follow-up of receivables and fast preparation of statements. Statements and reports are generated at a single keystroke. The system can be operated by any clerk who understands accounts receivable—no computer expertise or special training are needed.

The Inventory Control package provides reports to help management make inventory decisions. Complete reports, such as status, activity, and reorder, are called for by a single keystroke. This package also provides for immediate inquiry into the status of any item in the inventory.

The Accounts Payable package features easy operation by normal office staff. It provides management reports to help anticipate future cash requirements, the ability to reflect discounts when available, and check writing capability. It can charge payments to various general ledger accounts and cost centers, and it can handle partial payments to invoices for balancing available cash with cash requirements.

All of the packages are designed for ease of operation, easy data entry, and expandability.

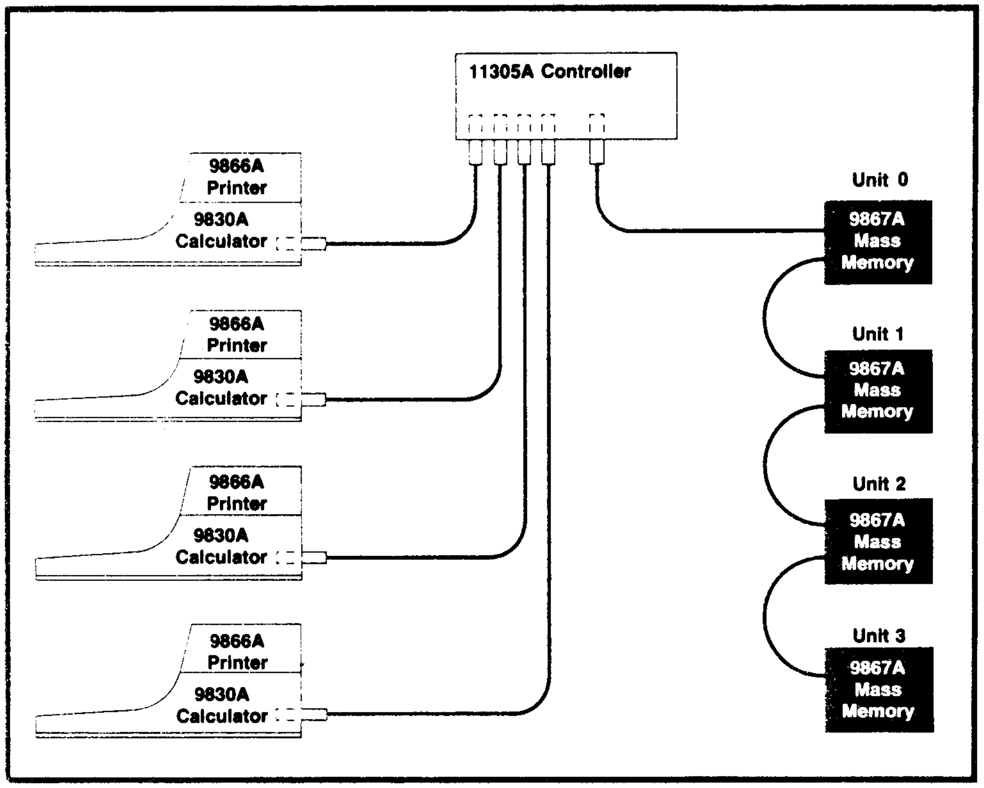


Fig. 3. *Controller can handle up to four platters and four calculators. Operation is not time-shared: only one calculator at a time can access a mass memory. At least two platters are recommended so backup copies of data can be made easily.*

Any location on the platter is defined by three variables: head, cylinder, and sector.[6] CYLINDER defines the radial position. There are 203 cylinders, 0 through 202, numbered from the outside of the platter towards the center. SECTOR defines the angular position around the platter. There are 24 sectors, numbered 0 through 23. A thin metal skirt on the hub of the platter has 24 slots cut into it to define precisely the start of each sector. An index slot provides a reference for sector 0. HEAD 0 and HEAD 1 specify the upper and lower platter surfaces, respectively.

Each platter is divided into two areas, the system area and the user area. The system area is used by the mass memory system and is occupied by the following items.

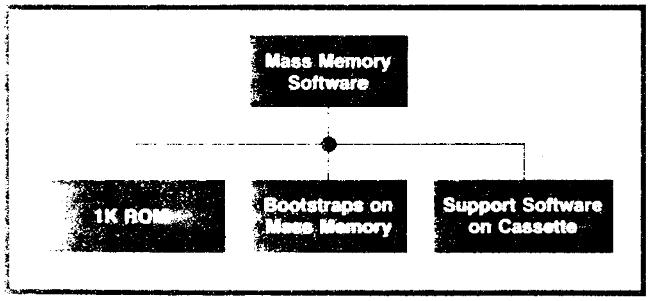


Fig. 4. *Mass memory software organization. Control firmware and speed-sensitive mass memory commands are stored in the read-only memory. A prerecorded support cassette is used to record bootstraps (routines to execute other mass memory commands) on the platter when the system is turned on.*

Item	Location
Directory	Head 0, Cylinders 0 and 1
Availability Table and Defective Track Table	Head 0, Cylinder 2
Bootstraps	Head 0, Cylinders 3 to 7

The directory serves as the file index of the mass memory system. All file names, along with other pertinent information such as relative location and size, are entered in the directory. The availability table contains a list of all vacant mass memory segments that are available for user files. The directory and the availability table are automatically updated by the system every time a file is created or deleted. The defective track table contains a list of the defective tracks present on the platter. A maximum of six defective tracks are allowed in the user area.

The remaining platter area is available for user data, program files, and key files. This area has a capacity of 4752 512-byte records.

Files

The mass memory system accepts three types of files: data, program, and key files. The minimum file size is one 512-byte record. The maximum file size is 4752 records.

Data files are created via the OPEN command. This command is both keyboard-executable and programmable, and has the following syntax:

```
[Line No.]  OPEN "DATA1", N
[Line No.]  OPEN A$, N
```

N specifies the length of the file in records, DATA1 is a file name, and A$ is a string variable that represents a file name to be assigned later. The line numbers are used only when the OPEN command is in a BASIC-language program.

Data files can be deleted via the KILL command.

[Line No.] KILL "DATA1"
[Line No.] KILL A$

Like OPEN, the KILL command is both programmable and keyboard-executable. Thus a program can determine the necessary number of data files and their sizes, create these files, and dynamically delete them when their contents are no longer needed.

Program and key files are automatically created by the SAVE and SAVE KEY commands, which store in the mass memory the present main-line programs and key definitions, respectively. The system calculates the minimum file size, opens the file, and stores the specified information. Programs can be read from the mass memory and transferred to the calculator via the GET and CHAIN commands. The GET command destroys the variable values established by any previously executed program whereas CHAIN retains them. Like data files, program and key files can be deleted via the KILL command.

Data Handling

The mass memory system enhances the data handling capacity of a 9830A Calculator. Numeric data in integer form (±32767), split precision form (±9.99999E+63), and full precision form (±9.99999999999E+99) can be stored in mass memory files. Integers and split precision numbers occupy four mass memory bytes. Each full precision number occupies eight mass memory bytes. When numeric data is retrieved from the mass memory, type conversion is allowed and is automatically accomplished by the mass memory system, provided that numeric overflow is not encountered.

If a String ROM block is present in the 9830A, alphanumeric strings of characters can also be stored in the mass memory. The maximum string size is 255 characters. Strings and numeric items of any precision can be stored in mass memory files in any order.

Data can be stored in mass memory files in serial or random modes. In the serial case, the complete file is considered a serial access storage device. In the random case, a file can be used as a random access storage device in which logical records (256-word subfiles) can be modified independently.

The data files that will be accessed by a given program must be declared by name in a FILES statement before data storage or retrieval is attempted. A maximum number of ten files can be declared in a FILES statement and be active at any one time.

Upon execution of the FILES statement, it will be verified that the declared files exist in the directory as data files. In addition, the "files" buffer will be created. This buffer will contain address, size, and other file information. Once this buffer is created, all subsequent data transfer commands will refer to the declared files by numbers corresponding to their relative positions in the FILES statement. This technique allows greater flexibility and higher data transfer rates.

Data is stored in the mass memory via the PRINT # command and retrieved via the READ # command. Entire arrays can be transferred to and from the mass memory via the MAT PRINT # and MAT READ # commands when a Matrix ROM block is available in the system.

The PRINT #n; END command prints an "end of file" marker in the next available position of the nth data file, where n = 1,2,3,4,5,6,7,8,9,10. This marker indicates the end of available data and can be detected by the READ # commands.

The TYP(n) function can be used effectively to determine the type of the next data item in a data file, and thus avoid terminal execution errors caused by type mismatch. The value of the function is an integer between 0 and 6, depending on whether the next item is a full-precision number, a string, an end-

```
10 REM THIS PROGRAM ALLOWS THE USER TO ENTER SALES INFORMATION.
20 REM THE INPUT INFORMATION (CUSTOMER NAME AND ADDRESS, NO. OF UNITS PURCHASED
30 REM AND TOTAL COST OF TRANSACTION) IS STORED IN THE DATA FILE "PORDER".
40 FILES PORDER,TEMP,*,*
50 DIM A$[70],B[2],D$[10]
60 PRINT
70 PRINT "CUSTOMER'S NAME ?"
80 INPUT A$[1,20]
90 PRINT "CUSTOMER'S ADDRESS ?"
100 INPUT A$[21,70]
110 PRINT "NO. OF UNITS ?"
120 INPUT B[1]
130 PRINT "UNIT COST ?"
140 INPUT C
150 PRINT "DONE ?"
160 INPUT D$
170 PRINT #1;A$,B[1],C*B[1]
180 IF D$="NO" THEN 60
190 PRINT #1;END
200 END
```

Fig. 5. *Up to ten files can be active at any time. Files must be named in a FILES statement (line 40). *'s may be used to reserve storage for files to be assigned later.*

```
10 REM THIS PROGRAM PRINTS THE SALES INFORMATION
20 REM PRESENT IN THE REQUESTED DATA FILE.
30 DIM A$[6],B$[70]
40 PRINT
50 PRINT "FILE NAME ?"
60 INPUT A$
70 FILES *
80 ASSIGN A$,1,R
90 IF R=0 THEN 120
100 PRINT "ERROR--";A$;"NOT PRESENT"
110 STOP
120 IF END#1 THEN 190
130 PRINT
140 FIXED 2
150 READ #1;B$,Y,X
160 PRINT "$";X;B$
170 PRINT
180 GOTO 150
190 END
```

Fig. 6. *This program is written so it accepts a file name to be supplied by the user during execution. Data in the selected file is then printed. Line 120 ends the program when the end of the selected file is reached.*

of-file marker, an end-of-record marker, a split-precision number, or an integer.

The example in Fig. 5 illustrates the file and serial data print concepts by collecting the input data and storing it serially in file "PORDER". The PRINT # command of line 170 will store the data items listed to the right of the semicolon in the file corresponding to position one of the FILES statement, that is, the file PORDER. The PRINT # command of line 190 will place an "end of file" marker in the first location of the next available record. The *'s are used in the FILES statement to reserve buffer storage for files that will be assigned later.

File Protection

To provide restricted access to important data files and to protect files from accidental deletion, the user may protect his files via the PROTECT command:

PROTECT "FILE NAME", "PROTECTION KEY"

The protection key should be a secret string of one to six characters.

The only way the user can gain access to a particular protected data file is through the ASSIGN command:

ASSIGN "FILE NAME", N, X, "PROTECTION KEY"

N is the FILES statement position to which the specified file is to be assigned. X is a return variable that provides status information about the attempted operation. The file name and protection key may also be given as string variables.

To delete protected data, program, or key files, the user must execute the KILL command augmented by the correct protection key:

KILL "FILE NAME", "PROTECTION KEY"

The example in Fig. 6 illustrates the file assignment and serial data read concepts. The IF END # command of line 120 sets an "escape" condition for the read command of line 150. This escape condition stipulates that if an "end of file" marker is encountered in the read process, control must be transferred to line 190.

Special Commands

A set of special commands provides additional flexibility for data backup and file handling.

PLATTER-DUPLICATE transfers all the information of the specified source platter to the specified destination platter. Thus, where multiple platters are available, a backup copy of a platter's contents can be made in approximately three minutes.

DFDUMP enables the user to dump data file information onto a magnetic tape cassette. The system will dump 150 records of data on one magnetic tape cassette. If the data file is larger than 150 records and more data is present in the file, the system will request additional cassettes.

DCOPY copies the information of a specified data file into another file of a different name. The source and destination files must be of the same protective status.

DREN allows the user to change the name of a specified file. If the file is protected, the correct protection key must be given. If the proposed new name already exists in the mass memory directory, the command will be aborted.

Transfer Rates

Transfer rates for programs and data are of particular interest when the mass memory is used to GET or CHAIN many programs and transfer large amounts of

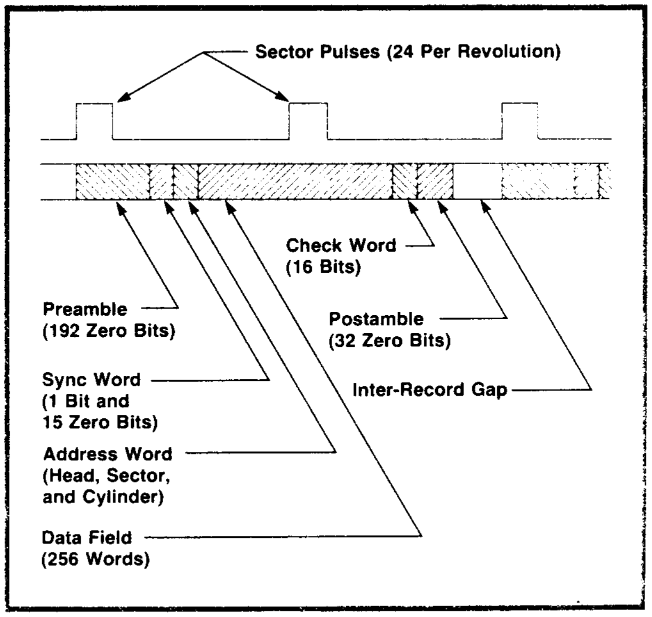

Sector Pulses (24 Per Revolution)
Check Word (16 Bits)
Preamble (192 Zero Bits)
Postamble (32 Zero Bits)
Sync Word (1 Bit and 15 Zero Bits)
Inter-Record Gap
Address Word (Head, Sector, and Cylinder)
Data Field (256 Words)

Fig. 7. *Each 512-byte (256-word) record occupies two sectors on the platter.*

data. The program transfer rate is approximately 1.2 milliseconds per word. Data transfer rates depend on the number of items present on the PRINT # or READ # list. The typical transfer rate is approximately 20 milliseconds per data item.

Record Format

A record is defined as the amount of information that is transferred to and from the mass memory in any given read or write operation. A record contains 256 words or 512 bytes of usable data. It also contains a preamble of 192 zero bits (see Fig. 7), a sync word, which is a 1 bit followed by 15 zeros, and an address word, which contains the binary equivalent of the three location variables (head, sector, and cylinder), followed by the data field of 256 words, the 16 bit checkword, and a postamble of 32 zero bits.

Each record occupies two sectors and begins only on the even numbered sectors. No address word exists for any odd numbered sector because this location is overwritten by the data field.

When information is written on the platter, a crystal controlled circuit provides a clock rate of 2.5 MHz. A clock pulse is written at the beginning of each data "cell." The data is mixed with this clock such that a 1 bit causes a pulse to occur between clock pulses, while a zero bit does not cause a pulse between clock pulses.[5] When the information is read from the platter, the presence of a pulse between clock pulses indicates a 1 bit; otherwise the cell contains a zero bit.

Small variations in the speed of the platter cause the data transfer rate to fluctuate. Pulse crowding in the magnetic medium causes further perturbations in the transfer rate. When reading, therefore, it is necessary to synchronize with the data rate rather than any absolute reference. This synchronization is started at the beginning of each record. The 192 bits of the preamble allow a phase locked loop to synchronize to the 2.5 MHz data rate from the platter. Once it is locked, the loop will track the small variations in the data rate. The absence of any 1 bits in the preamble assures that the loop becomes locked only to the clock rate. The 1 bit of the sync word establishes a reference point so the remainder of the record can be correctly interpreted.

The checkword is a 16-bit word that is uniquely determined by the address word and the data field. It enables certain errors to be detected when the record is read from the platter. The postamble provides clock pulses at the end of the record to allow the read circuitry to complete the read operation.

The entire record must be written each time any portion of the data field is to be changed. However, the address word will always be the same because it corresponds to the physical location on the platter.

Write Operations

A WRITE operation consists of four stages. First, the data to be written is placed in the cache memory. Next the desired platter location is found. Third, the location is verified to be the correct one, and finally, the record is written on the platter.

After the data has been placed in cache memory the calculator requests service from the 11305A controller. When service is granted to the calculator, the head, sector, and cylinder addresses are transmitted to the controller and mass memory. When the mass memory signals the controller that it has found the desired location, the controller commands the mass memory to read the address portion of that record. The address word that is read must compare bit for bit with the address sent by the calculator. If it does not, an error signal is returned to the calculator.

If the address is correct, the data is then written into the next sequential record on the platter (see Fig. 8). This offset allows time for the address comparison to be made and then a full record to be written, including the preamble and an updated address, as soon as the next record is reached. If this were not done, the controller would have to wait a full revolution of the platter before it could initiate the write command.

The read operation is the same except that the data

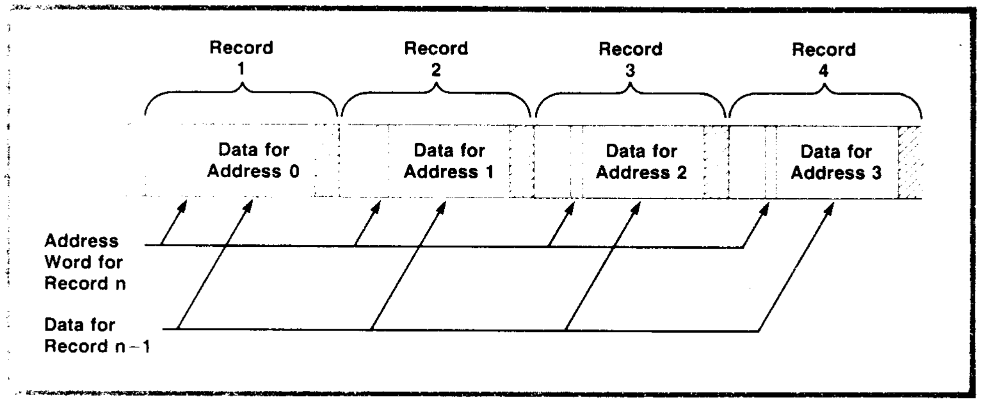


Fig. 8. *Data is offset from record addresses on the platter so data can be read or written as soon as the correct address is detected instead of waiting for another revolution of the platter.*

is read into cache memory from the platter. The calculator can then access any of the 256 words of cache memory as needed.

The system does not have read-after-write capability. The penalty in operating speed and software made this function costly to implement. The overall system reliability is high enough that read-after-write is not necessary.[3]

Multiple Platters

When there are two or more platters in a system the UNIT numbers shown in Fig. 3 provide platter identification for the system software. The 9867A single-platter drive has a switch on the front panel that can be set to four positions, 0 through 3. The 9867B two-platter drive has an internal setting corresponding to UNITS 0 and 1 or UNITS 2 and 3. In the 9867B, UNITS 0 and 2 correspond to the removeable cartridge and UNITS 1 and 3 to the fixed platter of the mass memory.

A UNIT n command from the calculator determines which platter in the system is to be accessed. Obviously, no two platters in the system can be assigned the same UNIT number. The UNIT number is part of the address that is sent to the mass memory. Only the drive with the selected UNIT number will respond to the controller commands.

Service Allocation for Multiple Calculators

When there are two or more calculators in a system the controller sequentially scans each of the service request lines from the calculators. As soon as a service request is detected, the controller halts the scanning process and enables the address and status lines from that calculator (see Fig. 2). These lines are common to all calculators connected to the system, so only one calculator can use them at any given time. The data lines (8 lines) between the cache memory and the controller are also common, so the controller must enable these lines to the cache memory corresponding to the calculator requesting service.

The controller remains dedicated to this calculator until it has released its service request for about 750 milliseconds. Then the controller resumes scanning the service request lines. The scan rate is about 150 kHz. The scanning circuits are designed so there is a dead zone between the time that each service request line is checked. This guarantees that there will be no transients that could cause access to be granted to the wrong calculator.

Handshake Operation

The calculator and the controller operate asynchronously. To guarantee that they do not get out of step and misinterpret data, all address and command signals from the calculator are flagged and receipt acknowledged by the controller. Each time the controller acknowledges receipt of an address byte, the calculator checks the status lines to determine whether an error occurred. The controller does not always immediately acknowledge receipt; tests may be made on the information or some other operation performed first, but all transmissions are eventually acknowledged.

If an address error is detected during the transmission or reading of addresses, three attempts are made before the operation is terminated by the calculator. If a checkword error is detected in trying to read the data, ten attempts are made before the calculator gives up and displays an error message to the user. This technique makes momentary errors recover-

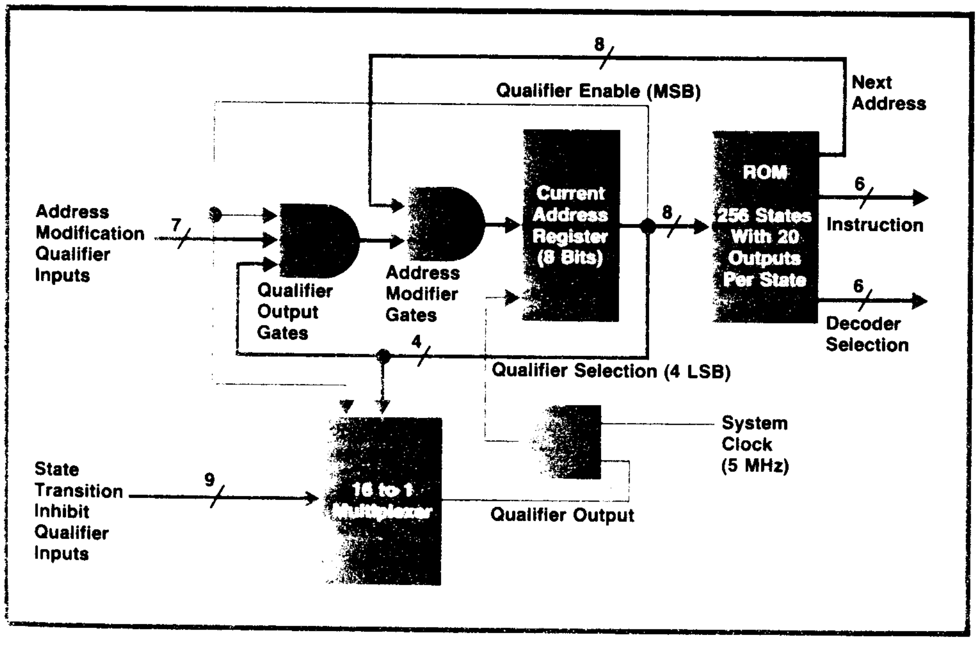

8
Qualifier Enable (MSB)
Next Address
Address Modification Qualifier Inputs
7
Current Address Register (8 Bits)
8
ROM 256 States With 20 Outputs Per State
6
Instruction
6
Decoder Selection
Qualifier Output Gates
Address Modifier Gates
4
Qualifier Selection (4 LSB)
System Clock (5 MHz)
State Transition Inhibit Qualifier Inputs
9
16 to 1 Multiplexer
Qualifier Output

Fig. 9. *Controller instructions and functions are stored in a 256-state bipolar ROM. One bit of the current ROM address determines whether the transition to the next address is qualified or unconditional.*

able so the system is not affected by them.

Controller Hardware

All of the controller functions and instructions are contained in a 256-state, 20-output ROM (see Fig. 9). The clock frequency of 5 MHz required that the ROM be bipolar rather than MOS. The 20 outputs provide information for the next address state (8 lines), the coded instruction outputs (6 lines), and the instruction decoder selection bits (6 lines).

The most significant bit of the current ROM address determines whether the transition to the next address is qualified or unconditional. For qualified transitions, two types of qualifiers are used. One type modifies the NEXT ADDRESS data that comes from the ROM. This type is generally used when an exit must be made from a microprogram loop. The second type of qualifier prevents the NEXT ADDRESS from being clocked into the address register until the qualifier is satisfied. This type is particularly useful when an asynchronous event is expected but its time of occurrence is only approximately known. There are seven modifier qualifiers and nine inhibit qualifiers. In each type, the four least significant bits of the present address determine which qualifier is to be selected.

Instructions

Six instruction decoder chips, each having four instruction bit inputs, generate the signals required to operate the system. These decoders are divided into three sets of two decoders each (see Fig. 10). Each decoder has a unique enabling bit that comes from the ROM. The six instruction bits from the ROM are decoded into 44 individual control lines for the system. The relationship between the ROM address states and qualifier selection and the interleaving of the instruction outputs made the ROM state assignments and microprogramming a very interesting and challenging task.

To avoid transients caused by the unequal switching times of the ROM outputs, all of the 44 control signals from the decoders are latched into flip-flops. The outputs of these flip-flops control the rest of the system. Each signal lasts 200 nanoseconds or a multiple of 200 nanoseconds. All instructions are latched on the next clock following their generation from the ROM. Because of the complexity, careful attention was given to gate delays and switching speeds.

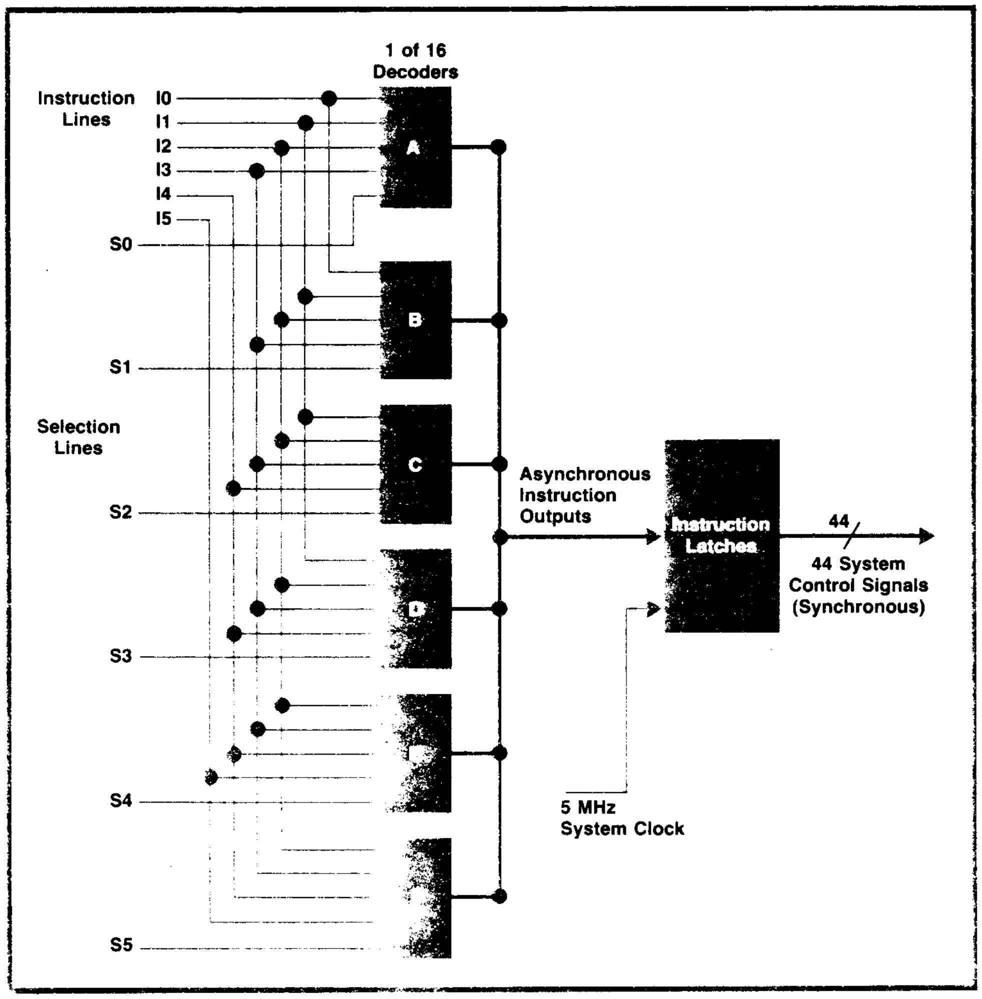


Fig. 10. *Six-line instructions from the ROM are decoded into 44 lines to control the system.*

Data Transfer

Data is transferred between the cache memory and the controller via an 8-bit data bus. Two 8-bit shift registers in the controller receive the data. The data is then transferred serially to the mass memory.

In a read operation, data from the mass memory is transferred serially into these registers, then in parallel to the cache memory. As soon as one shift register is filled, its contents are gated to cache memory while the second shift register is being filled. This requires 3.2 microseconds. Two write pulses, each one microsecond long, write 4-bit bytes into the cache memory. The remaining time is used to change the cache memory address for the second byte. The control circuits for cache memory keep count of the total number of bits transferred. When 256 words are completed, a signal is sent to the ROM qualifier and the operation is terminated.

Clock Switching

Normally, all of the system functions are controlled by the 5 MHz crystal clock. However, when data is read from the platter, the clock must be derived from this data. This poses a problem in that transfer from one clock base to the other must be accomplished without any transients that could cause false signals or cause the system to "get lost."

Two requirements must be met to avoid these problems. First the clock signal coming from the mass memory must be well established, and second, switching must take place at the beginning of a clock period to avoid any very short clock periods. Short clock periods are bad because the ROM needs a certain amount of time to respond to each clock pulse. If a clock pulse arrives too soon the ROM may jump to an incorrect address.

This problem is solved by the synchronization circuit shown in Fig. 11. This circuit allows the clock currently in effect to disable itself and the clock being acquired to enable itself. To further minimize risk, no other instructions are given in the ROM state preceding and following the clock transfer state. With this approach it is possible that an extra long clock cycle may be generated during the switching process but not any short ones.

System Reliability

Multiple-calculator/multiple-mass-memory systems have accumulated more than 7500 unit hours of testing to establish their reliability. Extensive applications development has provided further testing. Systems have transferred more than 10^{10} bits between recoverable errors.

Hardware safeguards have been implemented to reduce the possibility of loss of the user's data. Both the 9867A and the 9867B have front-panel switches which, when set, prevent any writing on their platters.

Failures can occur in any system, of course, no matter how well designed it is. For example, if primary line power fails during a write operation, that data will be lost. Improper maintenance or lack of maintenance can cause the mass memory to malfunction and data can be lost. This emphasizes the need to provide backup for any critical data. A multiple platter system makes this backup quickly and easily accomplished. Tape cassettes are another means of backup data storage and the system software also makes this easily accomplished.

Acknowledgments

The authors gratefully acknowledge the contributions of Fred Wullschleger for product design and Rich Kochis for electronic design of the controller. Also, Cal Finn made strong contributions early in the project to system definition and electronic design.

Paul Morrison contributed to the support software development and Gene Burmeister to development

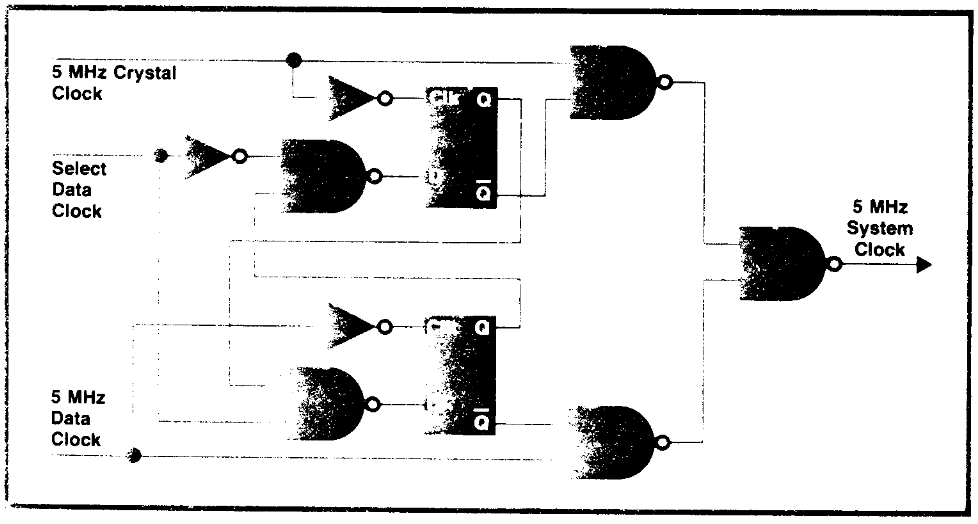


Fig. 11. *5-MHz crystal clock controls the system except during read operation, when the clock signal is derived from the data. This circuit synchronizes switching from one clock to the other.*

of the special commands. Special thanks go to Jack Walden for his software help and invaluable group leadership. Much credit also is due to the many people who contributed to software testing.

The authors would also like to express their appreciation to Chuck Near, peripherals section manager, and Chuck McAfee, group leader, for their suggestions and encouragement throughout the project.

Larry Mitchell, Joe Arata, Curt Kohanec, and Mike Bir of the Mountain View Division also deserve special thanks for their help in the sometimes trying task of bringing the 9867A and 9867B Mass Memory drives to production for the 9880 Mass Memory System.

Much of the work of transferring to production was taken over by technicians Norm Carlson and Terry Smith. A grateful acknowledgment is also made to the many people from the facility shops and production areas whose efforts have made this project a success.

References

1. R.M. Spangler, "BASIC-Language Model 30 Can Be Calculator, Computer, or Terminal," Hewlett-Packard Journal, December 1972.
2. J.E. Herlinger and J.R. Barnes, "A Faster and Tougher Disc Drive for Small Computer Systems," Hewlett-Packard Journal, May 1972.
3. J.E. Herlinger and W.J. Lloyd, "Inside the 7900 Disc Drive," Hewlett-Packard Journal, May 1972.
4. G.L. Egan, "Versatile Input/Output Structure Welcomes Peripheral Variety," Hewlett-Packard Journal, December 1972.
5. W.I. Girdner and W.H. Overton, "Reading and Writing on the Fast Disc," Hewlett-Packard Journal, May 1972.
6. D.J. Bowman, "An Efficient Disc Drive/Computer Interface," Hewlett-Packard Journal, May 1972.

SPECIFICATIONS

HP Model 9880A/B Mass Memory System

DATA CAPACITY AVAILABLE TO USER:

	9880A	9880B
BYTES	2,433,024	4,866,048
BYTES PER WORD	2	2
WORDS PER RECORD	256	256
NUMBER OF RECORDS	4,752	9,504(4752/platter)
MAXIMUM NUMBER OF FILES	768	1,536(768/platter)

SPEED:

	9880A	9880B
AVERAGE ACCESS TIME	47.5 ms	42.5 ms
DATA TRANSFER TIME	5.7 ms per 512 bytes	5.7 ms per 512 bytes

(Mass Memory to Calculator or vice versa)

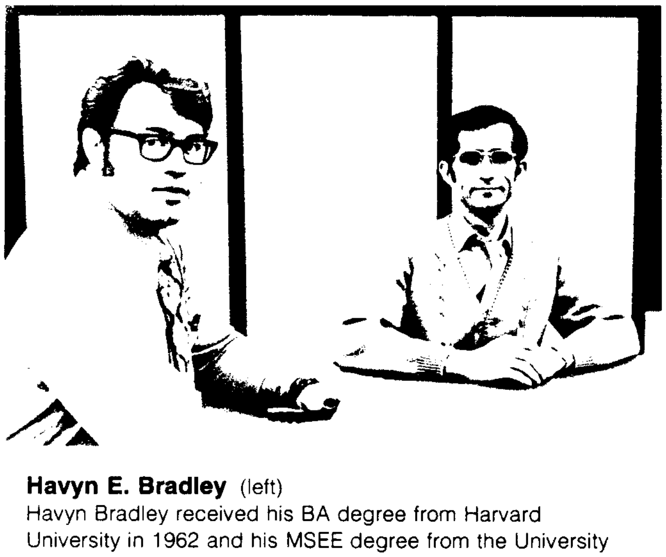
ENVIRONMENTAL (ENTIRE SYSTEM):
TEMPERATURE:
Operating: 10 to +40°C
Non-operating: −20 to +65°C
ALTITUDE: 0 to 10,000 feet
HUMIDITY: 8 to 80% non-condensing
VIBRATION: 10 to 50 Hz at 0.01 inch peak-to-peak excursion
ATTITUDE: ±30° pitch and roll

AIR FILTRATION:

	9867A Mass Memory Drive	9867B Mass Memory Drive
ABSOLUTE FILTERING	0.3 micron	0.3 micron
AIR FLOW RATE	25 CFM	75 CFM

PRICES IN U.S.A.: 9830A Calculator with 7616 bytes of read/write memory and String Variable ROM, $8495.
9880A (single platter) System, $10,945.
9880B (dual platter) System, $12,995.
(9880A/B prices do not include calculator.)
Installation (Option 101), $200.
Applications software packages, $500 each.
9866A Thermal Page Printer, $2995.

MANUFACTURING DIVISION: CALCULATOR PRODUCTS DIVISION
815 Fourteenth Street SW
Loveland, Colorado 80537 U.S.A.



Havyn E. Bradley (left)
Havyn Bradley received his BA degree from Harvard University in 1962 and his MSEE degree from the University of Wyoming in 1966. He's been with HP since 1966. Havyn has contributed to the development of the 745A AC Calibrator and the 9100A Calculator system, and has served as production engineer for the 9100A and as project leader for the magnetic card reader in the 9810A and 9820A Calculators. He was hardware project leader for the 9880A/B Mass Memory System. Havyn is married and has four children, whom he's teaching to program the 9820A Calculator. He's served as advisor to two Junior Achievement companies and once coached a women's softball team. Topping his list of favored recreational pursuits are fishing, hiking, and touring on his motorcycle, activities for which Colorado offers a superb setting.

Chris J. Christopher (right)
Chris Christopher was born in Greece, came to the U.S.A. in 1961, and graduated from Colorado State University in 1968 with a BSEE degree. At HP since 1968, Chris has been involved with computerized testing and production of 9100 series calculators and peripherals and with cassette software development for the 9830A Calculator. He was software project leader for the 9880A/B Mass Memory System. Having continued his studies while at HP, Chris received his MSEE degree just last month from Colorado State. He and his wife live in Loveland, Colorado, and frequently take advantage of the skiing in the nearby Rocky Mountains. Chris is a member of IEEE.