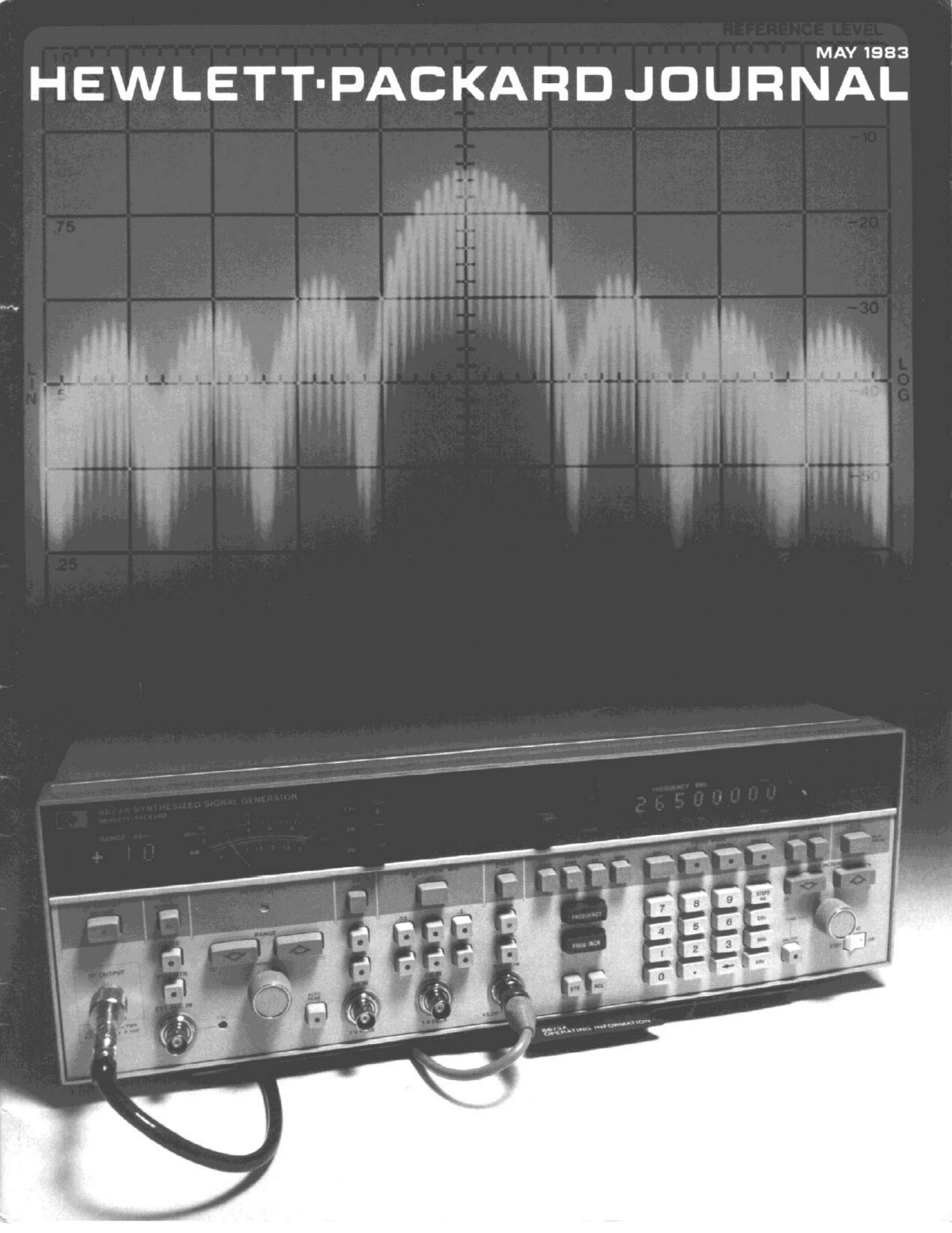
MAY 1983

HEWLETT·PACKARD JOURNAL



HEWLETT-PACKARD JOURNAL
Technical Information from the Laboratories of Hewlett-Packard Company

MAY 1983 Volume 34 • Number 5

Contents:

In this Issue:

For the design, production, and maintenance of radar and communications systems, sources of pulsed high-frequency signals are always in demand. The more versatile the source, the more jobs it can do and the more cost-effective it is (they're never cheap). Desirable attributes include broad frequency coverage, a good mix of modulation capabilities for simulating real-world signals, accurate output power controllable over a wide range, low noise, programmability for use in automatic systems, and of course, reliability.

This month's cover subject, the HP 8673A Synthesized Signal Generator, is designed to satisfy the needs of some pretty demanding applications. In radar receiver testing, for example, it can simulate the radar pulses picked up by an antenna, even mimicking the variation in the strength of the pulses as the antenna rotates past a target. Its capabilities come partly from microprocessor control and partly from advanced microwave circuit design. The full design story is on pages 3 to 16. The cover shows the classic pulse-train spectrum of the pulsed microwave output from the 8673A.

A few issues ago, the subject of several of our articles was the Hewlett-Packard Interface Loop, or HP-IL, a method of interconnecting battery-powered computers, peripherals, and instruments to form portable systems that can go into the field to make measurements and record data. Currently, there's just one battery-powered HP-IL peripheral for recording data, and that's the HP 82161A Digital Cassette Drive. This book-sized device can store and retrieve 131,000 characters of data using a matchbox-sized magnetic tape cartridge. Because of its intended use, it has to be light and rugged and use little power. In the article on page 17, its designers tell us how they met these goals.

The articles on pages 25 and 36 describe a pair of unusually capable programmable pocket calculators. One of them, the HP-15C, is for engineers, scientists, and mathematicians. It has a truly remarkable repertoire of mathematical functions that one would normally expect to need a computer for. Now, if you need matrix arithmetic, complex-number operations, equation solving, or numerical integration, you can just reach in your pocket. It's a vivid demonstration, I think, of what integrated circuit and computer technologies are doing for us.

The other calculator, the HP-16C, is for computer programmers and designers of digital circuits. Besides the usual arithmetic functions, it performs many of the basic operations of digital circuits and computers. For example, it can shift, rotate, mask, set, clear, test, and count the bits in a digital word. It has the fundamental combinational logic functions (OR, AND, NOT, and XOR) and it manipulates numbers in four different bases—hexadecimal, decimal, octal, and binary. With this little marvel, the professional can simulate a computer algorithm or logic circuit to see if it works—another computer-sized job cut down to pocket size.

-R. P. Dolan

Editor, Richard P. Dolan • Associate Editor, Kenneth A. Shaw • Art Director, Photographer, Arvid A. Danielson • Illustrators, Nancy S. Vanderbloom, Susan E. Wright • Administrative Services, Typography, Anne S. LoPresti, Susan E. Wright • European Production Supervisor, Henk Van Lammeren

Compact Digital Cassette Drive for Low-Cost Mass Storage

This portable battery-operated unit uses minicassettes to store programs and data inexpensively for HP-IL systems.

by William A. Buskirk, Charles W. Gilson, and David J. Shelley

THE HP 82161A Digital Cassette Drive (Fig. 1) is a portable, programmable, mass storage peripheral for the Hewlett-Packard Interface Loop (HP-IL).[1] The storage medium is a removable minicassette that can store up to 128K bytes of information. Portability is achieved by the use of a four-cell nickel-cadmium battery pack, recharger, and power supply system similar to that used in other portable HP products. The 82161A is styled to fit in a family of compact peripheral devices such as the 82143A and 82162A Printer/Plotters, and to fit nicely in a system controlled by an HP-41 Handheld Computer or an HP-75 Portable Computer. The 82161A makes use of much of the package design of the 82143A Printer/Plotter,[2] producing a unit 178 mm wide by 133 mm deep by 57 mm high. Replacing the 82143A's printer mechanism on the top right side is a transport mechanism with a **REWIND** key and a door **OPEN** key located in front. To the left of these keys is the power switch and indicators **POWER**, **LOW BATTERY**, and **BUSY**. The top left side of the package offers a compartment to store two minicassettes. The two HP-IL cables and the recharger cable are connected to the 82161A via plug receptacles on its back panel.

Electronic System

Fig. 2 is a block diagram of the electronic system of the 82161A. An internal microcomputer controls the head and motor drive electronics for the transport assembly and interacts with the HP-IL interface logic and data buffers.

The criteria for microcomputer selection for the 82161A included low cost, ready availability, low power consumption, and adequate I/O. To limit the number of electrical parts in the 82161A, a microcomputer that also contained ROM, RAM, and a timer, and could generate the encoded bit timing during a write operation was needed. A 3870 microcomputer with 2K bytes of ROM and 64 bytes of RAM was selected.

The logical interface of the 82161A is a generalized mass storage driver that provides the capability to execute operations such as initializing the tape, seeking a record, reading or writing a record, and rewinding the tape (see Table I).

Fig. 1. *The HP 82161A Digital Cassette Drive is a compact battery-operated mass-storage unit designed for use in portable HP-IL systems.*

Table I
HP 82161A Digital Cassette Drive Commands

DDL0	Write buffer 0	DDT0	Read buffer 0
DDL1	Write buffer 1	DDT1	Read buffer 2
DDL2	Write	DDT2	Read
DDL3	Set byte pointer	DDT3	Read address
DDL4	Seek	DDT4	Exchange buffers
DDL5	Format	DDT5	Transfer buffer 0→1
DDL6	Partial write		
DDL7	Rewind		
DDL8	Close record		
DDL9	Transfer buffer 0→1		
DDL10	Exchange buffers		

Buffer space for two 256-byte records of data is provided. Buffer 0 is used for data transfers between the HP-IL and the minicassette tape, and buffer 1 can be used by the HP-IL controller as virtual memory. The intent is to provide space to store a page of the tape directory and thereby reduce the number of seeks to the directory at the beginning of the tape. The DDL3 (set byte pointer), DDL8 (close record), and DDL6 (partial write) commands allow a memory-limited controller such as the HP-41 Handheld Computer to modify parts of a record without having to buffer the entire record in its mainframe. The record is read into buffer 0, modified, and written back to the tape with only the modification information passing around the HP-IL.

The ability to use the 82161A for extended remote data gathering tasks has been enhanced by the addition of the power-up/down commands. When the power switch on the front-panel keyboard is in the **STANDBY** position and a loop-power-down (PWRDN) command is received, the drive's power supply is turned off. When the HP-IL controller requires the loop to be active again, it sends a string of identify message frames, which turns the drive's power supply back on.

Software

The 2K bytes of machine code in the ROM of the microcomputer can be divided into three major areas: the power-on idle routine, the HP-IL routine, and the device control routines. The power-on idle routine, which uses approximately 160 bytes of code, sets up the initial state of the 82161A at power on and then alternates between testing for a cassette to be inserted into the drive, the **REWIND** key to be pressed, and calling the HP-IL routine. This routine also executes the device-clear and power-down functions. If either command is received, the HP-IL routine flags that fact and the drive responds after it finishes its latest task and returns to the idle routine loop.

The HP-IL routine, which takes approximately 460 bytes, provides the 82161A with basic talker and listener capabilities. This routine takes care of all communication with the HP-IL interface chip and passes all necessary information to the device control routines, primarily through a set of flag registers and one data register. This polled solution to HP-IL, in contrast to an interrupt-driven solution, is required because most of the device control routines need exclusive use of the microcomputer and can only give up control at specific times.

The device control routines, which take the remaining 1420 bytes of ROM, can be further divided. One part is the command decode portion. The device control is done with device-dependent commands (DDCs). When the HP-IL routine receives a DDC that it decides is of interest to the 82161A, it passes the DDC on to the command decode routine. Either the command is executed immediately, as in the case of a read or exchange buffer operation, or flags are set to control future actions, such as write and set byte pointer where the flags control where data bytes are put. A one-byte command buffer is used to hold a DDC received when the drive is busy. The HP-IL ready-for-command (RFC) message frame following this command is not retransmitted until the present task has been completed and the new command has been decoded.

The DDL5 (format) command initializes the record positions on the tape by recording all 512 records on both tracks. Each record contains a sync byte, a byte for the record number, a second sync byte, 256 bytes of data (each data byte initialized to 255), a checksum, and a final sync byte. Only during initialization is the first sync byte and the record number written. In all following write operations, the record number is read before the remaining part of the

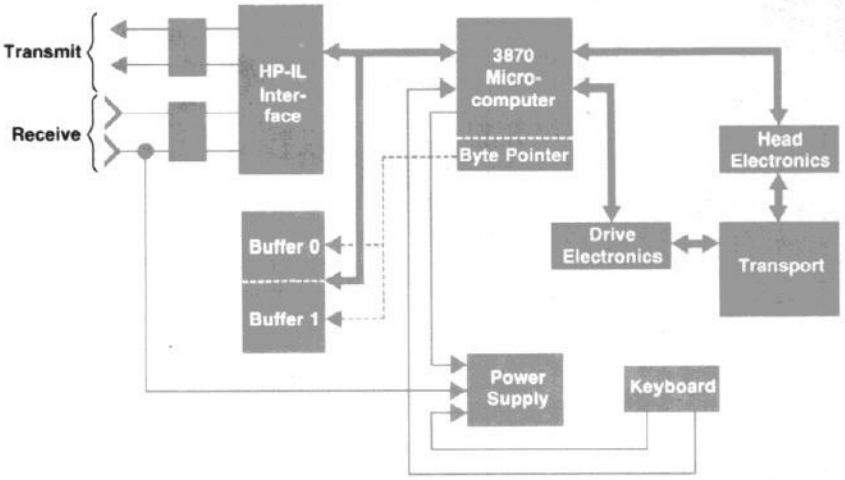


Fig. 2. *Block diagram of electronic system of the 82161A Digital Cassette Drive.*

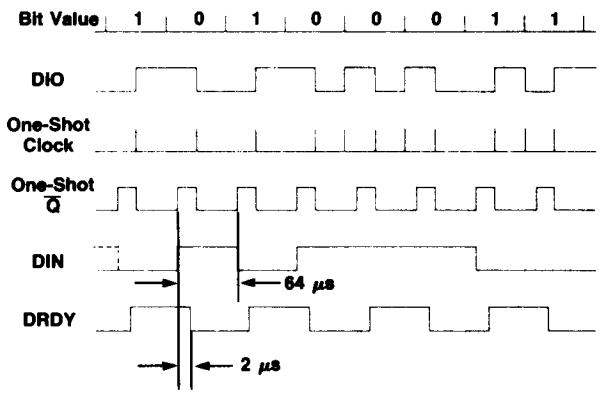


Fig. 3. *Timing diagram for signal lines and one-shot multivibrator states used to decode bit values stored on the 82161A tape cassette.*

record is written. This serves two purposes. The first is to verify that the proper record is being written and the second is to fix the record position on the tape so that it does not move along the tape when it is overwritten.

The major criterion in selecting an encoding method for the 82161A was reliability. The tape drive system requires that the method have a large speed-variation tolerance and use a microcomputer to generate the encoded bit stream during a write operation. The tape lengths required to record a one and a zero should be the same so that the length of a record does not depend on the ratio of ones to zeros within the record. Also, the code should be self-clocking for easy decoding.

The method best qualified is the biphase-level or Manchester code. The rules of this code are 1) there is always a transition in a bit cell center, and its direction specifies the value of the bit, and 2) there is a transition on a bit cell edge only when the two bits on either side have the same value (see Fig. 3).

In a write operation, the time between transitions, bit cell midpoint to bit cell edge, is 64 μs. During this time the transport status (stall, cassette present, and end of tape) is checked, the next nibble is read from the buffer and added to the checksum, and the next transition is calculated. The bit stream generated is sent to the sense amplifier on the DIO line.

There are two signals used in a read operation, DIN (data in) and DRDY (data ready). DRDY is the extracted clock and DIN is the latched data derived from the signal read from the tape. The microcomputer reads DIN and DRDY simultaneously and checks for DRDY to change state. When it does, the value of DIN is shifted into the register building the data. While it is in the read loop, the microcomputer also checks the transport status, stores the complete nibbles in the buffer, adds them to the checksum, and maintains a counter to detect when signal dropouts occur.

When the read/write routine is entered, the motors are turned on, the record number is read and verified, and the data portion of the record is then read or written. If a record number error and/or (in the case of a read) a checksum error is detected, the drive attempts the read or write operation a second time. If the microcomputer still detects an error, it stops the drive and reports the error to the HP-IL controller.

Seek operations are always attempted in a relative manner first. When the new record number is received, it is checked to see if it is in range (i.e., <512), and the difference between the present position and the desired position is calculated. The transport is turned on to move in the appropriate direction, and by watching the DRDY line, the microcomputer counts interrecord gaps until the transport reaches the record immediately before the desired record. This record is read, and the record number is checked. If it is correct, the transport is stopped with the desired record next. If an error is detected, the tape is rewound, and the seek is attempted again, but this time from the beginning of the tape. This gives four chances of reading the record correctly and ensures accurate seeks.

Data Storage and Retrieval

The microcomputer handles digital information to and from the read/write electronics on a bit-by-bit basis using three data-related lines (DIN, DIO, and DRDY, see Fig. 3) and two control lines (REC and TRK). DIO is a bidirectional data line whose transfer direction is controlled by the state of the REC line. In the read mode (REC low), DIO is driven by the sense amplifier, while in the write mode (REC high), the sense amplifier goes into a high-impedance state and DIO is driven directly by the microcomputer. Both DIN and DRDY are generated by the decoder circuitry and are derived solely from DIO level changes. The TRK line is driven by the microcomputer to select which tape track (0 or 1) is read from or written to.

The sense amplifier is a custom bipolar integrated circuit. It contains the signal conditioning and logic circuits to drive the magnetic head during a write operation and to translate the low-level analog signals at the head to time-related digital signals at DIO during a read operation.

Writing to the tape is accomplished by controlling the current flowing through the windings of the magnetic head. These currents produce a magnetic field across the gap at the front of the head. Three wires (two ends and a center tap) are attached to each winding (track) on the head. During a write operation, the center tap is connected to a constant-current sink, and each end of the winding is alternately driven high to control the direction of the current and thus the polarity of the magnetic field at the gap. The use of a current sink allows maximum rate of change of current, yet limits the peak direct current to 150% of that required to completely magnetize the tape.

During a read operation, the voltage at the terminals of the head is proportional to the rate of change of the magnetic flux across its gap and reaches a peak value when the gap is directly opposite a flux reversal on the tape. To decode recorded information properly, a digital signal with level changes corresponding in time to these voltage peaks must be generated. The sense amplifier generates this signal by amplifying and then differentiating the analog signal from the head. A zero crossing at the output of the differentiator corresponds to a peak of the amplified signal and is used to clock DIO level changes. The DIO level (high or low) is related to the polarity of the amplified signal at clock time and indicates the direction of the flux transition. Hysteresis is included to provide protection from unwanted transitions caused by electrical noise.

Data content is encoded by the direction of the DIO level transition at the midpoint of a bit cell. Transitions at bit cell edges are used only as required to set up DIO for the proper change at the next midpoint (see Fig. 3). The decoder hardware ignores these edge transitions and provides the microcomputer with two signals—DRDY and DIN. A change at DRDY notifies the microcomputer that the signal at DIN represents valid data.

For every DIO transition a 100-ns pulse is generated and appears at the trigger input of a nonretriggerable one-shot multivibrator. The timing period of the one-shot multivibrator is set so that, if triggered by a midcell transition, the next edge transition, when it exists, will occur during the cycle and thus be ignored. When the timing cycle expires, the level of DIO, which corresponds to the encoded bit value, is latched into the DIN flip-flop. Approximately 2 μs later, the output of the DRDY flip-flop changes, notifying the microcomputer that data is valid (see Fig. 3).

To ensure that the one-shot multivibrator is triggered only by midcell transitions and that it ignores any cell edge transitions, a sync byte is included at the beginning of each record. This special bit pattern maps to a stream of level changes that includes only midcell transitions. Thus, the one-shot multivibrator is set up to be triggered only at midcell, and, if speed stays within allowable limits, synchronization is maintained throughout the entire record.

The encoder hardware can tolerate timing variations in DIO of ±30%. Electronic jitter, aliasing, phase shift in the amplifiers, external electromagnetic noise, and true speed variations all contribute to the total timing variance at DIO. Fortunately, the wide acceptability range of the decoder easily overcomes these factors and ensures good unit-to-unit compatibility.

Transport Mechanism

The 82161A's mechanical design uses an 8 × 34 × 56 mm minicassette designed especially for digital applications. The cassette contains nominally 24 meters of usable tape 3.81 mm wide, allowing two tracks of data 1.45 mm wide. This tape is hub-driven as opposed to capstan-driven, meaning that the only way the tape can be moved is by turning the appropriate stack of tape directly.

The selection of a hub-driven cassette was the key step in a "simplicity" approach to the design of the 82161A mechanism. It allows the use of a two-motor drive (one per hub) and eliminates additional motors or controlled actuator devices that would be required by capstan mechanisms or single-motor drives. Another key to simplicity is the use of a two-track magnetic head. This eliminates having to move the cassette, either by the user or by a mechanism, to access both tracks. Other factors contributing to a straightforward design are the extensive use of injection-molded thermoplastics, cost-effective fasteners such as adhesives and press fits, and a low part count.

Aside from simplicity, another goal of this mechanism design was modularity. This produced a drive system module that can be removed from the 82161A and designed into other products with a minimum of change, electrical or mechanical.

The primary part of this device is the headframe assembly (Fig. 4). This assembly consists of a molded plastic frame into which a magnetic head is aligned and glued, and an optoelectronic device which forms half of the beginning-of-tape/end-of-tape (BOT/EOT) sensor. The single-gap, two-track head's coil winding parameters were chosen for both read and write functions. The headframe is molded from glass-filled polycarbonate, a very stable compound that allows some key dimensions to be held within a tolerance of 0.025 mm. Two posts on the headframe position the cassette housing relative to the head and two tape guides on the frame guide the tape relative to the head. These features are molded into the headframe.

The headframe assembly is joined with a door, window, and cassette pressure springs to form a door assembly very similar to that used in many conventional cassette tape recorders. The cassette is loaded into a slot in the door and the action of closing the door positions the cassette in the mechanism. When the door is released by pressing the **OPEN** key, a spring pulls it open so that the cassette can be easily removed.

When the door containing a cassette is shut, it pushes a pin, closing a switch spring located directly on the drive printed circuit assembly and informing the electronics that a cassette is present. A 45-degree mirror located within each cassette completes the optical BOT/EOT detection path. The optoelectronic device and the head in the headframe are connected to the drive's printed circuit assembly by a flexible ribbon cable.

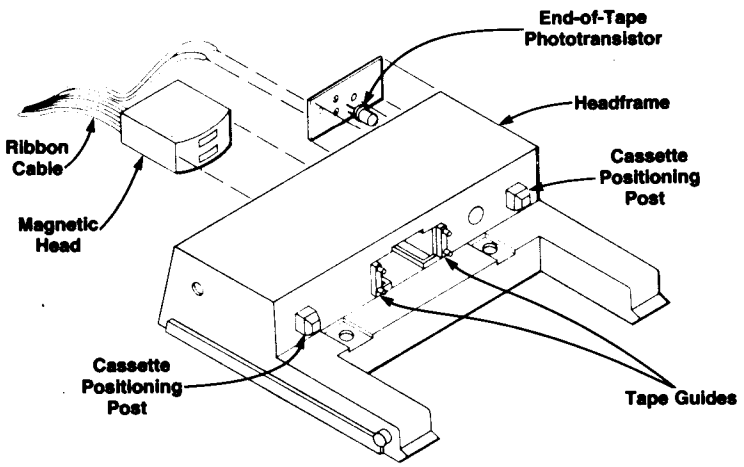


Fig. 4. *Exploded view of the 82161A headframe assembly.*

The backbone of the transport is the mainframe. It is a precision part made of glass-filled polycarbonate formed by injection molding. All of the key subassemblies, including the door/headframe assembly, the motors and gear systems, the door latch, and the printed circuit assembly are fastened to the mainframe to form a complete modular transport mechanism.

Two identical drive motors are used. One drives the left stack of tape and is called the forward motor. The other drives the right stack and is called the reverse motor. The use of two motors coupled directly to the tape stacks in this fashion makes possible a simple speed control scheme for reading and writing data. The motor selection involved a tradeoff between motor performance (hence cost) and product capability (primarily data capacity). Ironless-rotor motors with their low inertia/torque ratio were selected. The software definition requires interrecord gaps long enough to allow the tape velocity to change between zero and read/write speed between records. Selecting low-inertia motors allows record length to interrecord gap ratios of around 3:1. If higher-inertia motors had been used, this ratio, as well as data capacity, would have been lower. Another important characteristic of ironless-rotor motors in this application is their linear tachogenerator feature. The construction of the motors is such that their EMF, with the commutation ripple filtered out, can be used to detect motor speed changes typically within 2 percent. Perhaps the most important result of the selection of low-inertia motors is the reduction of dynamic tape tension. With the two-motor, hub-drive technique, the driving motor must pull the tape against the other motor's inertia. When accelerating, the trailing motor's inertia is multiplied by the square of the gear reduction as it is reflected into the tape. Hence, peak tape tensions are very sensitive to motor inertia. Tape life was found to be dominated by dynamic tape tensions and so the long tape life achieved in the 82161A is strongly related to the selection of low-inertia motors.

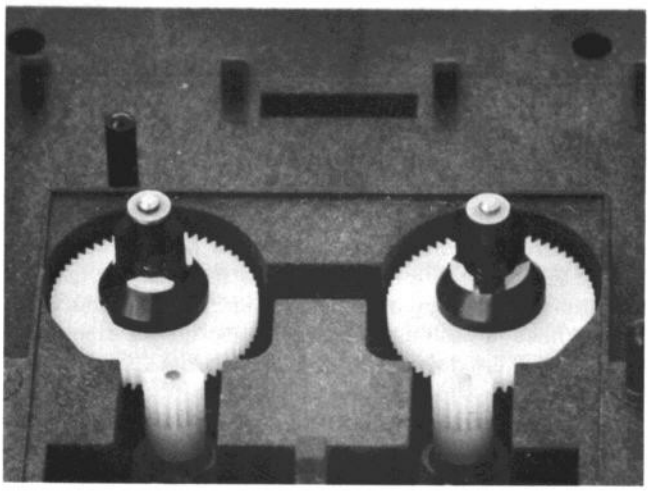
Motors could not be found that would go slow enough while maintaining the torque required to drive the tape hubs directly. Therefore, a speed reducer is required. Many methods were considered, beginning with the logical choice of motor/gearbox combinations, but these proved to be too expensive. O-ring and toothed-belt drive designs were tried, but both exhibited a common problem of requiring increased belt tension to avoid slipping. The higher belt tension produced higher shaft friction, which in turn, led to increased tape tension and reduced tape and bearing life.

The 82161A uses a custom gear drive (Fig. 5) consisting of a pinion and drive gear for each motor with a ratio of 1:4 (15 teeth to 60 teeth). Both gears are injection-molded at a custom gear-molding house and exceed AGMA* quality No. 7. The diametral pitch is 96 and the gear material is lubricated acetal resin. The pinions are pressed on the motor shafts and the drive gears run free on ground stainless-steel shafts pressed into the mainframe (Fig. 5a). The motors are positioned by concentric collars fastened also to the mainframe. Precision molding of the mainframe allows the motor-to-drive-gear-shaft dimension to be held to within 0.025 mm. Also running on the drive gear shafts, and axially coupled to the drive gears, are the drive splines (see Fig. 5b). These splines fit into the cassette hubs and transmit torque to them from the driven gears. In the event that a cassette hub does not align with the spline when the door is closed, the spline is spring-loaded and can be pushed down, allowing the door to shut. When that particular side of the drive moves, spline relative to hub, the spring pushes the spline up to engage it with the hub. These spline springs also produce a braking action or a drag friction. This drag was optimized for system speed performance by adjusting the spring constant and preload.

*American Gear Manufacturers' Association

Head Alignment

Accurate alignment of the magnetic head in the headframe is crucial to give unit-to-unit read/write compatibility for systems using multiple transports. This alignment is done electrically, with the head actually reading signals



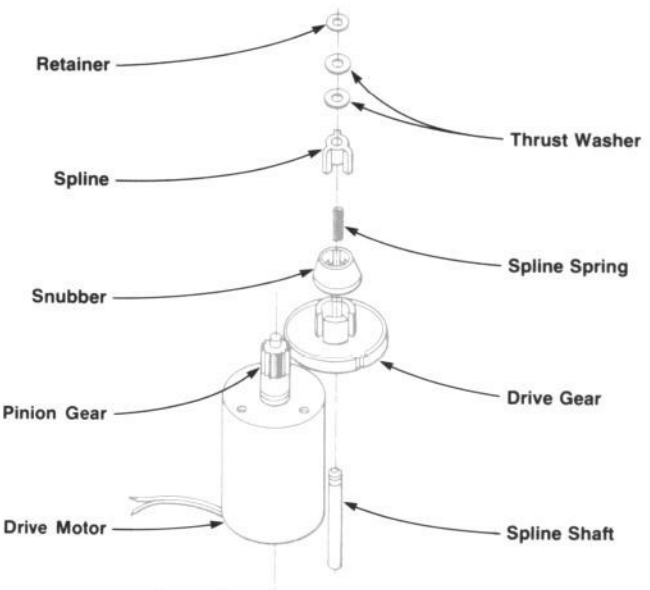


Fig. 5. *The tape drive system in the 82161A uses two identical motors, each driving a spring-loaded spline. (a) Close-up photograph of drive gear and motor assembly. (b) Exploded drawing of motor and drive gear system.*

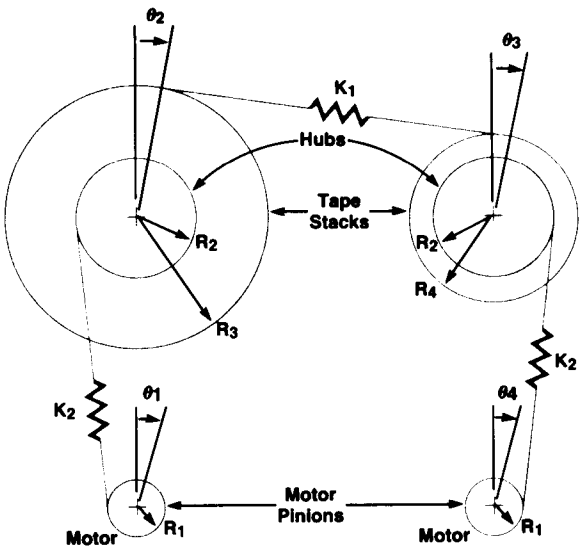
from two tracks simultaneously, rather than aligning optically as has been done by HP in the past.[3,4] The head aligner tool consists of an endless-loop tape deck, an oscilloscope, and an ac voltmeter. A master head on the tape deck is used to write perfectly phased signals on both tracks of a tape loop. This loop is then read by a head requiring alignment. The head is held inside a headframe by a tooling fixture that sets penetration and varies azimuth (perpendicularity of the head gap relative to tape motion) and tracking (vertical position of the head poles relative to tape position). The two signals read are observed on the oscilloscope and the azimuth is adjusted until both tracks are in phase and the combination of signals with maximum amplitudes has been found. This ensures that the azimuth has been "perfectly" aligned in the fixture. Next, the tracking is adjusted by moving the head up and down until the sum of the signals from both tracks is a maximum as measured by the voltmeter. This ensures that the pole spacing on the head optimally matches the track positions on the alignment tape. The two adjustments are practically decoupled on the alignment fixtures, so convergence is not necessary. After the head is aligned, the assembly and tooling fixture is removed, and the head is glued in place with a fast-curing acrylic adhesive. The headframe assembly can then be removed from its fixture and the alignment rechecked to observe any movement caused by glue cure. This procedure produces azimuth alignment better than ±5 arc-minutes, and tracking alignment, relative to the headframe, better than ±0.05 mm. The master head is periodically used to monitor the accuracy of the alignment tape. To check the master head, a Möbius tape is installed on the tape deck. This tape alternately presents front and back sides to the master head. A signal is written on the front side and read from the back side. The amplitude is reduced, but only the track-to-track phase is important. If the master head is "perfectly" aligned in azimuth, no phase difference will occur. An iterative process is used to align the master head if necessary.

System Modeling

The electromechanical system of the 82161A tape transport was modeled to allow studies of tape velocities and the effects thereon of motor parameters and mechanism inertias and frictions. The basic model is shown in Fig. 6 and the basic equations of motion are

$$
\begin{aligned}
I_1\Theta_1 &= R_1\left[K_2(R_2\theta_2 - R_1\theta_1)\right] \\
I_2\Theta_2 &= -R_2\left[K_2(R_2\theta_2 - R_1\theta_1)\right] + R_3\left[K_1(R_4\theta_3 - R_3\theta_2)\right] \\
I_3\Theta_3 &= -R_4\left[K_1(R_4\theta_3 - R_3\theta_2)\right] + R_2\left[K_2(R_1\theta_4 - R_2\theta_3)\right] \\
I_1\Theta_4 &= -R_1\left[K_2(R_1\theta_4 - R_2\theta_3)\right]
\end{aligned}
$$

This multi-degree-of-freedom system was studied by identifying different motor-to-tape-stack and motor-to-motor modes whose frequencies depend on the reduction method (belts or gears) and tape stack ratios. Many of the natural frequencies found by this model can be satisfactorily filtered out by altering the servo design, but one mode consistently showed up in the analysis that cannot. This was identified as a "tape mode" or the opposing oscillation of both hub stacks with the spring being the length of tape between the two stacks. During read/write operation, when the servo is controlling tape speed, this tape mode, if excited, is superimposed on the steady-state tape velocity and becomes what was found to be the major cause of tape speed jitter. In all of the configurations studied, this jitter frequency is near 500 Hz. This tape mode cannot be effectively corrected by the servo because it is totally isolated from the motors and hence is "invisible" to the servo. This tape mode creates oscillations in tape tension, but the magnitude of the oscillations was found to be less than the steady-state tape tension. Thus, the treatment of the tape as a linear spring was not negated by the fact that the tape cannot be put into compression. This phenomenon will be discussed further in reference to the slow-start circuitry in the motor drive.



θ_1 = Left motor position
θ_2 = Left stack position
θ_3 = Right stack position
θ_4 = Right motor position
I_1 = Motor inertia
I_2 = Left-hub-and-tape-stack inertia
I_3 = Right-hub-and-tape-stack inertia
K_1 = Connecting tape stiffness
K_2 = Gear mesh stiffness
R_1 = Pinion radius
R_2 = Hub radius
R_3 = Left stack radius
R_4 = Right stack radius

Fig. 6. *Diagram defining tape motor drive parameters used to model 82161A transport system behavior.*

The tape mode oscillations would not be a major difficulty if there was no 500-Hz excitation in the mechanism to get them going. Unfortunately, because of the size of the transport, the gear pitch selection was limited, and 500-Hz perturbations caused by gear backlash are unavoidable. Hence, two methods are used to damp the tape mode. First, friction is added by using the spline spring as described previously, and second, viscous rubber snubbers are added between the drive gear and the spline (see Fig. 5). In this position, the snubbers serially provide a viscoelastic element between the perturbation (gear backlash) and the elements (the cassette stacks) producing the tape mode oscillations. The combination of these two damping schemes reduces speed jitter by approximately 50% to a range of 10% of average tape speed.

Motor Drive

The 82161A mechanism combines software, electronics, and mechanics to control both the position and the velocity of the tape. TTL-compatible inputs to the motor drive circuitry allow the microcomputer to select any of five possible modes of operation.

The fast forward and rewind modes move the tape at 76 to 152 cm/s, during which time the microcomputer counts interrecord gaps to determine tape position (record number). Once the desired position has been reached, the slow forward mode is activated for a data read/write operation. Forward and reverse braking is accomplished by using the back EMF of the trailing motor to generate a reverse torque to decelerate the system.

The fast forward, rewind, and slow forward modes use the leading motor as the actuator and the trailing motor is "pulled" by the tape. The no-load friction of the trailing motor and its associated gears provides tape tension to aid speed control and help keep the tape in contact with the magnetic head. The forward and reverse braking modes use the trailing motor as the actuator and the tape as the mechanical link to decelerate the leading motor.

The heart of the motor drive electronics is the velocity control circuitry (Fig. 7). To ensure read/write compatibility, linear tape velocity past the magnetic head must be a controlled, repeatable function of tape position. Although holding the angular velocity of one motor constant would satisfy this objective, tape capacity would be severely limited because the linear tape speed would vary over a wide range as the radii of the takeup and supply reels, respectively, increase and decrease. However, holding the sum of the angular velocities of both motors constant not only satisfies the above requirements, but dramatically increases data capacity by maintaining a more uniform linear tape velocity.

The input to the servo is a controllable reference voltage. The servo acts to hold the sum of the back EMFs of the two motors equal to this reference. As shown in Fig. 7, the forward transfer path consists of an error amplifier, a power stage, and the mechanical system. The back-EMF summer forms the feedback path. All necessary frequency compensation is implemented in the error amplifier and includes a pole at the origin to integrate out dc errors, a low-frequency zero at 4 Hz to compensate a pole of the mechanical system, and a second pole at ≈40 Hz to filter out unwanted motor commutation noise that appears in the feedback signal. At frequencies within the range of interest (40 Hz), the open-loop transfer function of the system, including compensation, consists of a single pole at the origin. Local feedback for the error amplifier is derived from the output of the power stage to minimize crossover distortion. As discussed earlier, the transfer function of the mechanical system is quite complex and includes several oscillatory modes. Fortunately, these modes are either at frequencies well outside the bandwidth of the servo or are invisible to the servo so that no serious electronic stability problems arise.

A novel feature of this servo is the speed sensor which sums the back EMF from each motor. Since no current flows through the trailing motor, the back EMF is simply its terminal voltage and is readily available to determine motor speed. However, the current required to produce drive torque generates a voltage across the rotor resistance of the leading motor which is superimposed on its back EMF. In the past, this speed measurement problem has been avoided by using either pulse-width modulators, which sample back EMF by momentarily removing power, or transducers which do not rely on back EMF. For this application, low-frequency pulse-width modulators would dissipate additional power in the motor and generate electrical and mechanical noise caused by their switching transients. Transducers are too expensive, too large, and require too much additional hardware.

The chosen scheme dynamically sums the terminal voltage of each motor and subtracts the voltage caused by the drive currents in the leading motor. Referring to Fig. 7,

$$V_0 = EMF_L + I_M R_M - \left[(EMF_L + I_M R_M + I_M R_S) - (EMF_L + I_M R_M)\right](R3/R2) + \left[(EMF_L + I_M R_M) - (EMF_L + I_M R_M - EMF_T)\right](R3/R1)$$

If R1=R3, and canceling terms,

$$V_0 = EMF_L + I_M R_M - I_M R_S (R3/R2) + EMF_T$$

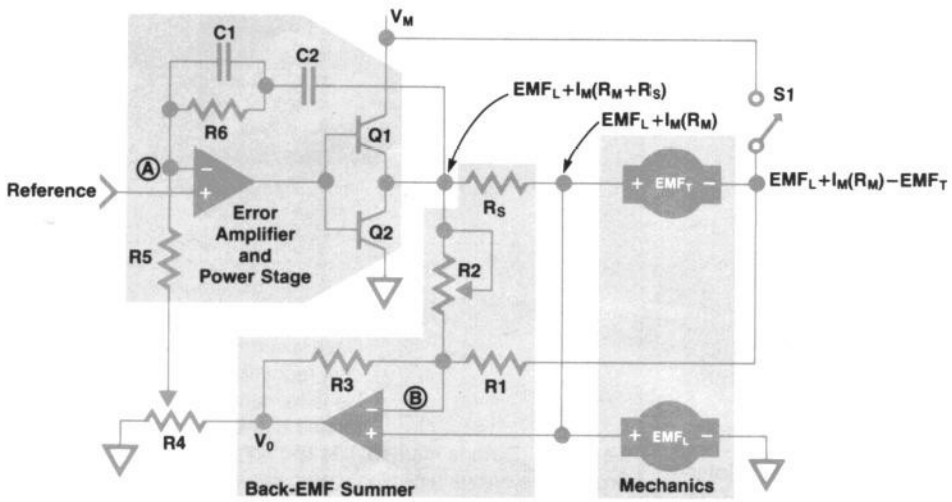


Fig. 7. *Simplified schematic of velocity control servo.*

Then, if R2 is adjusted such that $R_M/R_S=R3/R2$,

$$V_0=EMF_L+EMF_T$$

As can be seen from the derivation above, if resistance matching is done (using potentiometer R_2), the output of the feedback amplifier is the true sum of the back EMF of the motors. R_S is specified as a copper wirewound resistor so that its temperature coefficient of resistivity will match that of the motor's ironless rotor, thus holding the ratio of R_M to R_S constant and ensuring consistent speed control over the full operating temperature range.

The fast forward and rewind modes are implemented by "fooling" the servo. Grounding point A in Fig. 7 eliminates the feedback and causes the output of the error amplifier to go high and drive the motor at high forward speed. Forcing point B low causes V_0 to be high, thus forcing the output of the error amplifier low. This, in combination with closing switch S1 and lifting the ground on the leading motor (using transistor switches), results in the rewind mode.

To use the feedback from both motors to control speed, it is essential that the motors be mechanically linked in a predictable, linear fashion. In the 82161A, this link is the tape. Because the tape cannot support compressive forces, slack in the tape can occur and totally uncouple the leading and trailing motors. The typical result is a "bang-bang" servo action. The leading motor is driven until its back EMF equals the reference value. Suddenly the tape slack is taken up and the trailing motor begins to move and injects a step function into the feedback signal. The error amplifier responds by slowing the leading motor, which allows the trailing motor to spool up and form another loop. This starts the process all over again.

Once the 82161A has attained stable slow forward operation, this problem is prevented by the tape tension generated by system friction. However, when the slow-forward mode is initiated, there is always some amount of slack in the tape, and this slack must be eliminated first before accelerating to full speed. In addition, since overshoot can cause the same problem, the rate of change of the speed reference voltage must be slowed to the point where the servo can keep up.

The slow-start circuit performs these functions by controlling the reference voltage to the servo. When the slow forward mode is selected, the reference is held to a low value for approximately 130 ms, during which time the slack is removed from the tape. Then, the reference voltage rises exponentially towards an asymptotic value, allowing a smooth acceleration to read/write speed without overshoot.

Acknowledgments

The authors would like to thank Roger Quick and Tom Braun for their direction and leadership in the development of the 82161A, and George Custer for the product package design, head sourcing, and innumerable other design details. Also greatly appreciated is the work of Mark Matsler in developing the head alignment process, and the hard work of everyone else who helped bring the 82161A Digital Cassette Drive to production.

References

1. R.D. Quick and S. L. Harper, "HP-IL: A Low-Cost Digital Interface for Portable Applications," Hewlett-Packard Journal, Vol. 34, no. 1, January 1983.
2. R.D. Quick and D.L. Morris, "Evolutionary Printer Provides Significantly Better Performance," Hewlett-Packard Journal, Vol. 31, no. 3, March 1980.
3. D.J. Collins and B.G. Spreadbury, "A Compact Tape Transport Subassembly Design for Reliability and Low Cost," Hewlett-Packard Journal, Vol. 31, no. 7, July 1980.
4. R.B. Taggart, "Designing a Tiny Magnetic Card Reader," Hewlett-Packard Journal, Vol. 25, no. 5, May 1974.

William A. Buskirk

Bill 'Buzzy' Buskirk joined HP in 1977 after receiving a BSEE degree from the University of Colorado. He worked in production engineering on various calculators before moving to R&D in 1978. Bill worked on the card reader for the HP-41 Handheld Computer and the 82161A Cassette Drive before assuming his current responsibility as a project manager. He was born in Bloomington, Indiana and during his undergraduate studies, worked for the National Oceanic and Atmospheric Administration. Bill is married, has a son, and lives in Albany, Oregon. Outside of work, he is kept busy remodeling his home, reading, camping, working with home computers, and helping his wife run their gift shop.

Charles W. Gilson

Charlie Gilson graduated from California Polytechnic State University with a BSME degree in 1973. He worked three years on computer modeling of missile shock isolation and air-launch systems before joining HP in 1976. He worked on the mechanical design of various calculators and the 82161A Cassette Drive's transport mechanism before moving to production engineering to do cost-reduction design. Born in San Francisco, California, he now lives in King's Valley, Oregon. He is married and has two children, a girl and a boy. His interests include raising sheep and slowly remodeling an old farmhouse.

David J. Shelley

Dave Shelley was born in Seattle, Washington and attended the nearby University of Washington, earning a BSEE degree in 1973. He then joined HP's San Diego Division and contributed to the electrical design for the 7245A Plotter/Printer and the 9872A Graphics Plotter. In 1977, he transferred to HP's Corvallis Division and worked on the electrical design for the 82143A Printer and the 82161A Cassette Drive. Dave is currently a project manager for cost reduction of portable computers. He is named coinventor for a patent related to the thermal head design for the 7245A. Dave is married, has two sons, and lives in Corvallis, Oregon. He enjoys bicycling, camping, and playing racquetball.

Scientific Pocket Calculator Extends Range of Built-In Functions

Matrix operations, complex number functions, integration, and equation solving are only some of the numerous preprogrammed capabilities of HP's latest scientific calculator, the HP-15C.

by Eric A. Evett, Paul J. McClellan, and Joseph P. Tanzini

THE NEW HP-15C Scientific Programmable Calculator (Fig. 1) has the largest number of preprogrammed mathematical functions of any handheld calculator designed by Hewlett-Packard. For the first time in an HP calculator, all arithmetic, logarithmic, exponential, trigonometric, and hyperbolic functions operate on complex numbers as well as real numbers. Also, built-in matrix operations are provided, including addition, subtraction, multiplication, system solution, inversion, transposition, and norms.

The HP-15C also performs the **SOLVE** and $\int_y^x$ functions, which are very useful tools in many applications. The **SOLVE** operator numerically locates the zeros of a function programmed into the calculator by the user.[1] The $\int_y^x$ operator numerically approximates the definite integral of a user-programmed function.[2]

Design Objectives

The HP-15C was designed with the following goals in mind:

- Provide all functions of the HP-11C and HP-34C Calculators in the same slim-line package used for the HP-11C
- Provide additional convenient, built-in advanced mathematical functions which are widely used in many technical disciplines.

Achieving the first objective posed a keyboard layout problem. The nomenclature for the HP-11C functions filled every position on the keyboard. Since the display is limited to seven-segment characters, functions could not be removed from the keyboard and accessed by typing the function name as is done on the HP-41 Handheld Computers. Therefore, to free some space on the keyboard, only the two most common conditional tests are placed on the keyboard, **x=0** and **x≤y**. A **TEST** prefix is added to access the other ten

Fig. 1. *The HP-15C is an advanced programmable calculator with special functions that enable the user to solve problems involving matrices, integrals, complex arithmetic, and roots of equations. Its slim-line design fits easily in a shirt pocket.*

tests by executing **TEST 0**, **TEST 1**, ..., **TEST 9**. A table on the back of the calculator indicates the correspondence between the digits and tests. This frees enough positions on the keyboard to add the **SOLVE** and $\int_y^x$ functions, plus a few more.

In striving to attain the second objective, we noted that nearly every text covering advanced mathematics for science and engineering includes chapters on complex analytic functions and matrix algebra. They are fundamental tools used in many disciplines. Furthermore, the complex functions and many of the matrix operations can be viewed as extensions of the functions already on the keyboard. This is an important consideration because of the limited number of key positions available. Thus, our goal was to extend the domain of some of the built-in functions to complex numbers and matrices in a natural way without altering how those functions operate on real numbers.

Complex Mode

A complex mode was introduced in which another register stack for imaginary numbers is allocated parallel to the traditional register stack for real numbers (Fig. 2). Together they form what is referred to as the complex RPN* stack.

The real X register is always displayed. A complex number $a+ib$ is placed in the X register by executing **a**, **ENTER**, **b**, **I**. The user may display the contents of the imaginary X register by executing **Re≶Im** to exchange the contents of the real and imaginary X registers. Or the user may hold down the **(i)** key to view the imaginary part without performing an exchange.

ENTER, **R↓**, **R↑**, **x≶y**, and **LST x** all operate on the complex stack, but **CLx** and **CHS** operate only on the real X register so that one part of a complex number can be altered or complemented without affecting the other. For example, the complex conjugate is performed by executing **Re≶Im**, **CHS**, **Re≶Im**.

The following functions include complex numbers in their domain: **+**, **−**, **×**, **÷**, **1/x**, $\sqrt{x}$, **x^2**, **ABS** (magnitude), **LN**, **e^x**, **LOG**, **10^x**, **y^x**, **SIN**, **COS**, **TAN**, **SIN^{-1}**, **COS^{-1}**, **TAN^{-1}**, **SINH**, **COSH**, **TANH**, **$SINH^{-1}$**, **$COSH^{-1}$**, and **$TANH^{-1}$**. These functions assume the complex inputs are in the rectangular form, $a+ib$.

Often complex numbers are expressed in polar form: $re^{i\theta} = r(\cos\theta + i\sin\theta)$. In complex mode, the polar-to-rectangular conversion functions **→P** and **→R** provide a convenient means for converting between the polar and rectangular forms of a complex number.

*Reverse Polish notation, a logic system that eliminates the need for parentheses and "equals" keystrokes in calculator operations.

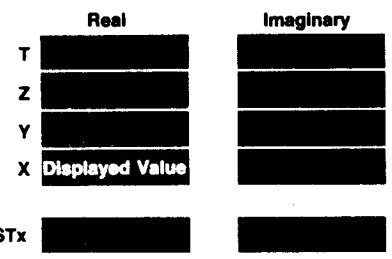
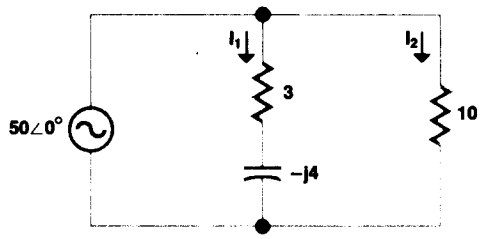


Fig. 2. *To handle complex numbers, the HP-15C uses another register stack in parallel with the traditional RPN stack. Only the contents of the X register in the real stack are displayed.*



$Z_1 = 3 - j4$
$Z_2 = 10$
$Z_{eq} = 1/(1/Z_1 + 1/Z_2)$

Fig. 3. *The complex arithmetic capabilities of the HP-15C make it easy to solve for the equivalent impedance of this parallel circuit (see text).*

Complex numbers are used extensively in electrical engineering. For example, to find the equivalent impedance in the parallel circuit shown in Fig. 3, perform the following steps on the HP-15C:

Keystrokes	Calculation
3 **ENTER** 4 **CHS** **I**	Z_1
1/x	$1/Z_1$
1 0	Z_2
1/x	$1/Z_2$
+	$1/Z_1 + 1/Z_2$
1/x	$Z_{eq} = 1/(1/Z_1 + 1/Z_2)$. Hold down **(i)** key to view imaginary part. $Z_{eq} = 2.9730 - 2.1622i$
→P	Convert to phasor form. $Z_{eq} = 3.6761 \angle -36.0274°$

This example is a very elementary application of the built-in complex function capability. Since complex operations can be used in conjunction with the **SOLVE** and $\int_y^x$ functions, the HP-15C can be programmed to carry out some sophisticated calculations such as computing complex line integrals and solving complex potentials to determine equipotential lines and streamlines.[3]

Matrix Descriptors

No set of matrix operations is complete without addition, subtraction, multiplication, system solution, and inversion. To provide these operations on the HP-15C, it seemed natural to extend the domains of the **+**, **−**, **×**, **÷**, and **1/x** functions to include matrix arguments. Since these functions operate on the stack contents, a means of placing a matrix name (descriptor) on the stack is essential. The set of alpha characters that can be represented in a seven-segment font is limited, but the letters A, B, C, D, and E have reason-

Fig. 4. *The seven-segment font used in the HP-15C's liquid-crystal display allows representations of the alphabetic characters A, B, C, D, and E as shown above for use in labeling matrices.*

able representations (Fig. 4).

Thus the decision was made to allow up to five matrices to reside in memory simultaneously, named **A**, **B**, **C**, **D**, and **E**. Their descriptors are recalled into the X register by the sequence **RCL MATRIX** followed by the appropriate letter. When the X register contains a matrix descriptor, the matrix name and dimensions are displayed. Matrix descriptors may be manipulated by stack operations and stored in registers just like real numbers, and certain functions accept matrix descriptors as valid inputs. For example, suppose **C** and **D** are 2-by-3 and 3-by-4 matrices, respectively, which are already stored in memory. To compute the matrix product **CD** and place the result in matrix **A**, the user parallels the steps required for real number multiplication, except that the result destination must be specified:

Keystrokes	HP-15C Display
RCL MATRIX C	C 2 3
RCL MATRIX D	d 3 4
RESULT A	d 3 4

At this point, the HP-15C's RPN stack contains the information shown in Fig. 5a. The matrix operands are in the stack, and the result matrix is specified. The user now executes × to compute the matrix product. When × is executed, the presence of the matrix descriptions in the Y and X registers is detected, the matrices are checked for compatible dimensions, the result matrix **A** is automatically dimensioned to a 2-by-4 matrix, the product is computed, and the matrix descriptor of the result is placed in the X register and displayed (Fig. 5b).

The operators + and − work similarly, and ÷ performs the matrix operation $\mathbf{X}^{-1}\mathbf{Y}$ if the X and Y registers contain matrix descriptors. This is useful for linear system solution, since the solution to the matrix equation $\mathbf{XR}=\mathbf{Y}$ is $\mathbf{R}=\mathbf{X}^{-1}\mathbf{Y}$. The **1/x** function key performs matrix inversion.

Other important matrix operations that are not natural extensions of functions on the keyboard are accessed by the prefix **MATRIX** followed by a digit. These include transpose, determinant, and matrix norms. A table on the back of the calculator indicates the correspondence between the digits and matrix operations.

Internal Format of Descriptors

Normal floating-point numbers are internally represented in the HP-15C by using a 14-digit (56-bit) binary-coded-decimal (BCD) format (Fig. 6).

The exponent e is given by XX if XS=0, and by −(100−XX) if XS=9. The value of the number is interpreted as $(-1)^{S}(\text{M.MMMMMMMMM})\times 10^{e}$. For example,

01234000000002 represents 1.234×10^{2}

and

91234000000994 represents -1.234×10^{-6}.

Matrix descriptors, on the other hand, are distinguished by a 1 in the mantissa sign digit and a hexadecimal digit corresponding to the matrix name in the most significant digit of the mantissa field. For example, the matrix descriptor C is represented internally as 1C000000000000.

When a matrix descriptor is detected in the X register, the matrix name is displayed, and the current dimensions of that matrix are fetched from a system memory location and also displayed.

Creating Matrices and Accessing Individual Elements

A matrix is dimensioned by entering the row and column dimensions in the Y and X registers of the stack, respectively, and then executing the **DIM** prefix followed by the matrix name. Individual matrix elements are accessed by executing the **STO** or **RCL** prefixes followed by the matrix name. The element accessed is determined by the row and column indexes stored in registers R0 and R1, respectively.

Matrix data is usually entered or reviewed from left to

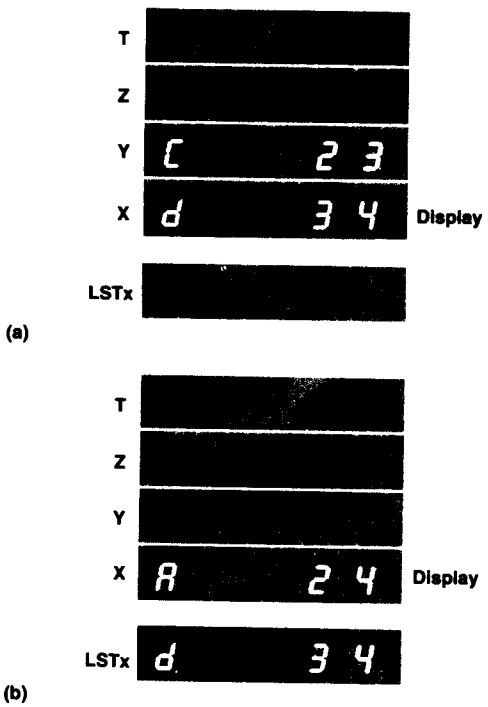


Fig. 5. *Before multiplying matrices* **C** *and* **D**, *the information in the RPN stack is located as shown in (a). After multiplication, the result, matrix* **A**, *is located as shown in (b) and the* **LSTx** *register contains the information for matrix* **D**.

right along each row and from the first row to the last. To facilitate this process, a user mode is provided in which the indexes are automatically advanced along rows after each **STO** or **RCL** matrix access operation. After the last element of the matrix has been accessed, the indexes wrap around to 1,1. As an added convenience, executing **MATRIX 1** initializes the indexes to 1,1.

The following example illustrates some of these features by solving the following matrix equation for C:

$$\underset{A}{\begin{bmatrix} 5 & -2 \\ 4 & 6 \end{bmatrix}} \underset{C}{\begin{bmatrix} c(1,1) & c(1,2) \\ c(2,1) & c(2,2) \end{bmatrix}} = \underset{B}{\begin{bmatrix} 8 & 3 \\ 2 & -6 \end{bmatrix}}$$

Keystrokes	Display	Comments
USER		Select **USER** mode.
MATRIX 1		Initialize indexes in registers R0 and R1 to 1.
2 ENTER DIM A	2.0000	Dimension matrix **A** to 2 by 2.
DIM B	2.0000	Dimension matrix **B** to 2 by 2.
5 STO A	5.0000	a(1,1)
2 CHS STO A	−2.0000	a(1,2)
4 STO A	4.0000	a(2,1)
6 STO A	6.0000	a(2,2). Indexes wrap around to 1,1.
8 STO B	8.0000	b(1,1)
3 STO B	3.0000	b(1,2)
2 STO B	2.0000	b(2,1)
6 CHS STO B	−6.0000	b(2,2). Indexes wrap around to 1,1.
RCL MATRIX B	b 2 2	Recall right-hand side
RCL MATRIX A	A 2 2	Recall coefficient matrix
RESULT C	A 2 2	Specify matrix **C** as result.
+	C 2 2	Compute $\mathbf{C}=\mathbf{A}^{-1}\mathbf{B}$.
RCL C	1.3684	c(1,1)
RCL C	0.1579	c(1,2)
RCL C	−0.5789	c(2,1)
RCL C	−1.1053	c(2,2)

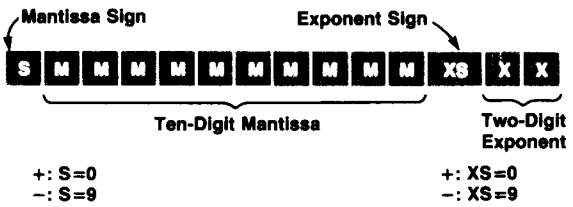


Fig. 6. *The internal representation for floating-point numbers in the HP-15C uses a 14-digit (56-bit), binary-coded-decimal format.*

Available Matrix Memory, Speed

A maximum of 64 matrix elements can be distributed among the five matrices. Since the HP-15C can invert matrices in place, up to an 8-by-8 matrix can be inverted. There is also enough memory to solve a 7-by-7 linear system of equations. Table I specifies the approximate time required to perform certain matrix operations.

Table I
Time in Seconds for Selected Matrix Operations

Order of Matrix	Determinant	Solving a System	Matrix Inversion
1	0.5	0.5	0.5
2	1.3	2.0	1.8
3	2.8	4.2	5.3
4	5.3	7.6	12
5	9.1	12	22
6	14	19	36
7	21	28	55
8	30	—	80

Designing the Complex Function Algorithms

After deciding to extend the real-valued functions and the RPN stack to the complex domain, our next step was to design the algorithms for complex arithmetic. Although their defining formulas are very simple, some disturbing examples made us question what accuracy should be achieved to parallel the high quality of the real-valued functions.

The real functions are generally computed with a small relative error (less than 6×10^{-10}) except at particular points of certain functions, where it is too costly in execution time or ROM space for the result to be computed that accurately.[3]

The relative difference R(x,y) between two numbers x and y is given by

$$R(x,y)=\frac{|x-y|}{|y|}$$

When X is an approximation of x, then we say R(X,x) is the relative error of the approximation X. Notice that the size of the relative error is related to the number of digits that are accurate. More precisely, $R(X,x)<0.5\times10^{-n}$ implies that X

is an approximation to x that is accurate to n significant digits.

If we always wish to obtain small relative errors in each component of a complex result, then the outcome of the following example is very disappointing. For simplicity we will use four-digit arithmetic, instead of the 13 digits used internally to calculate the 10-digit results delivered to the X register of the calculator.

Example 1: Using the definition for complex multiplication,

$$(a + ib)(c + id) = (ac - bd) + (ad + bc)i,$$

consider the four-digit calculation of $Z \times W$, where $Z = 37.1 + 37.3i$ and $W = 37.5 + 37.3i$. We get,

$$\begin{aligned} Z \times W &= (1391 - 1391) + (1384 + 1399)i \\ &= 0 + 2783i \end{aligned}$$

Since the exact answer is $-0.04 + 2782.58i$, it is clear that accurate components are not always achieved by a simple application of this formula. The difference $a \times c - b \times d$ has been rounded off to result in a loss of all significant digits of the real part. The loss can be eliminated, but the calculation time would increase roughly by a factor of 4. Is it really worth this higher cost in execution time? For comparison we will consider an alternative definition of accurate complex results.

Complex Relative Error

As with real approximations we often want our errors small relative to the magnitude of the true answer. That is to say, we want $|$ (approximate value)$-$(true value)$|/|$(true value)$|$ to be small enough for our purposes. So relative error may be extended to the complex plane by $R(Z,z) = |Z-z|/|z|$. This extension may be applied to vectors in any normed space. A simple geometric interpretation is illustrated in Fig. 7. Approximations Z of z will satisfy $R(Z,z) < \delta$ if and only if the points Z lie inside the circle of radius $\delta|z|$ centered at z. This condition for complex relative accuracy is weaker than that for component accuracy. If the errors in each component are small, then the complex error is small. To show this, assume that $R(X,x) < \delta$ and $R(Y,y) < \delta$ where $z = x + iy$. Then,

$$\begin{aligned} R(Z,z) &= |(X - x) + i(Y - y)|/|z| \\ &\leq |X - x|/|z| + |Y - y|/|z| \\ &\leq R(X,x) + R(Y,y) \\ &< 2\delta \end{aligned}$$

Actually, $R(Z,z)$ is less than δ, but this is slightly more difficult to show. On the other hand, however, a small complex error does not imply small component errors. Referring back to Example 1, we see that $R(ZW, zw) = 0.0002$, which is respectably small for four-digit precision, even though the real component has no correct digits.

It is not unusual for only one component to be inaccurate when the result is computed accurately in the sense of complex relative error. In fact, because the error is relative to the size $|z|$, and because this is never greatly different from the size of the larger component, only the smaller component can be inaccurate.

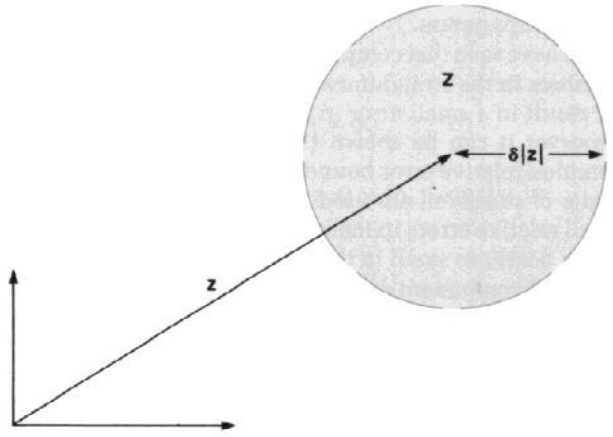


Fig. 7. *A simple geometric representation of complex relative error $R(Z,z) < \delta$.*

To show this we shall assume, without loss of generality, that $|x|$ is the larger component. Then

$$|z|/|x| = \sqrt{1 + |y/x|^2},$$

which implies that $1 \leq |z/x| \leq \sqrt{2}$, since $|y| \leq |x|$ by assumption. Thus $|x|$ and $|z|$ do not differ greatly. The important part is that $|x| \geq |z|/\sqrt{2}$. This gives

$$|X-x|/|x| \leq |Z-z|/|x| \leq \sqrt{2}|Z-z|/|z| < \sqrt{2}R(Z,z)$$

So the relative error of the larger component (assumed to be x here) is very nearly as small as the complex relative error bound $R(Z,z)$. It also follows that the smaller component is accurate relative to the larger component's size (i.e., $|Y-y|/|x| \leq |Z-z|/|x| \leq \sqrt{2}R(Z,z)$.

This provides a quick way to determine which digits of a calculated value can possibly be incorrect when it is known that the calculated value has a certain complex error. By representing the smaller component with the exponent of the larger component, the complex error indicates the number of correct digits in each component.

For instance, in Example 1 we obtained the approximation $Z = 0 + 2783i$ of the true answer $-0.04 + 2782.58i$. Since the larger component is 2.783×10^3 we will represent the first component with the same exponent (0.000×10^3) to obtain $Z = 0000.0 + 2783i$. These components must be accurate to nearly four digits since $R(Z,z) = 0.0002$.

Perhaps the zero component of Z confuses the issue here, so another example may be appropriate. First, let

$$Z = 1.234567890 \times 10^{-10} + 2.222222222 \times 10^{-3}i$$

Then think of Z as

$$Z = 0.0000001234567890 \times 10^{-3} + 2.222222222 \times 10^{-3}i$$

If the complex relative error indicates 10-digit accuracy, i.e., $R(Z,z) < 0.5 \times 10^{-10}$, then this implies that the first 10 digits are correct, that is,

$$Z = 0.000000123 \times 10^{-3} + 2.222222222 \times 10^{-3}i$$

Error Propagation

We have seen that computing the product of two complex numbers in the straightforward manner does not necessarily result in a small error in each component (Example 1). However it can be shown that the product does have a complex relative error bound of roughly 10^{-n} whenever n digits of precision are used in the calculation. Moreover, small relative errors in the input values give rise to relative errors nearly as small in the output values. This is not true for small component errors. One acceptable rounding error in an input value may produce an inaccurate component, even when the multiplication is exact. This is illustrated by the following example.

Example 2: Let $z = (1 + 1/300) + i$ and $w = 1 + i$, then using four-digit precision we have

$$Z = 1.003 + 1.000i$$
$$\text{and } W = 1.000 + 1.000i$$

Therefore,

$$ZW = 0.003 + 2.003i$$
$$= 3.000 \times 10^{-3} + 2.003i$$

exactly, yet

$$zw = 3.333 \times 10^{-3} + 2.003i$$

to four digits. The single rounding error of $1+1/300 \rightarrow 1.003$ in the component of the input Z was magnified from a relative error of 0.0003 to 0.1.

So, in general, computing accurate components will not improve the result of a chain calculation because intermediate input values are often inexact (this is the idea of backward error analysis and is explained more fully in reference 3). It is important to realize that this is not, in itself, a good reason to forsake accurate results based on the assumption that the input values are not exact. For example, if we assume that X has an error in its eleventh digit and thus decide that sin(X) for $X > 10^5$ degrees, say, need not be computed accurately, then we would have failed to provide a useful result for those special cases where we know that the input value is exact.

As a simple illustration consider accurately calculating the value sin(1,234,567,899.1234567890) where the argument is in degrees. Using

$$\sin(1{,}234{,}567{,}899) = 0.9876883406$$

is grossly inaccurate. Instead, let $x = 1.234567899 \times 10^9$ and $y = 0.123456789$, then evaluate

$$\sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y).$$

Here we know x is exact, and since sin(x) and cos(x) are computed accurately by the HP-15C, the final result $\sin(x+y) = 0.9873489744$ is very accurate.

The point here is that clean results (in particular accurate components) are desirable, but in our estimation the cost of adding ROM and increasing execution time was too high on this machine to provide complex arithmetic that is accurate in each component. However, accurate components are delivered in those functions where it is more practical. This is discussed further in the following section.

In general, the HP-15C delivers complex results that satisfy $R(Z,z) < 6 \times 10^{-10}$, except where functions involving trigonometric calculations (in radians) are evaluated at very large arguments or near transcendental zeros such as multiples of π. This inaccuracy is embedded in the real-valued functions and is an example of an error that is too costly to correct completely.[3,4]

Some Specific Complex Functions

For complex arithmetic we obtained accurate results (i.e., small complex relative errors) from the standard formulas used to define each operation. But, in general, defining formulas are usually not accurate for computers. In this section we will single out two particular functions, sin(z) and $\sqrt{z}$, and very briefly focus on some difficulties that arise.

Sin(z). A typical defining formula for the complex sine function is given by

$$\sin(z) = \frac{e^{iz} - e^{-iz}}{2i} \qquad (1)$$

If this is used to compute sin(z) for small $|z|$, the two exponential terms will be nearly equal and thus cause a loss of accuracy. This will result in a large complex relative error even though each step of the calculation is very accurate. If equation (1) is replaced by

$$\sin z = \sin(x)\cosh(y) + i\cos(x)\sinh(y) \qquad (2)$$

where $z = x+iy$, then the relative error problem for small $|z|$ will be solved, and furthermore the components will become accurate (except for the trigonometric difficulty with large angles mentioned earlier). To observe the striking difference in results, we calculate

$$w = \sin(1.234567 \times 10^{-5} + 9.876543 \times 10^{-5} i)$$

for each formula. The outcome is represented below.

Eqn.	W (10-digit calculation of w)	R(W,w)
(1)	$1.234567006 \times 10^{-5} + 9.876530000 \times 10^{-5} i$	10^{-6}
(2)	$1.234567006 \times 10^{-5} + 9.876543015 \times 10^{-5} i$	10^{-10}

The HP-15C's internal calculation is based on equation (2), with minor modifications that exploit the relationships between the real functions to eliminate redundant computation.

$\sqrt{z}$. The most common definition of the principal square root is

$$\sqrt{z} = \sqrt{|z|}\, e^{i\theta/2} \qquad (3)$$

where θ is the Arg (z), satisfying $-\pi < \theta \leq \pi$.

This formula is accurate with respect to complex relative error, but not accurate in each component. This can be seen by working through the calculation of $\sqrt{a}$, where $a = -1 + (-1 \times 10^{-15} i)$, with 10-digit precision. Here $\theta/2$ rounds to precisely 90 degrees, thus causing $\sqrt{a} \rightarrow 0 - i$, while the true

value rounds to $5\times10^{-16}-i$. The complex error is small but certain information in the real component is lost. The fact that $\sqrt{a}$ lies on the right side of the imaginary axis can be critical when computing near discontinuities called branch cuts. For example, $\ln(-i\sqrt{a}) \approx -i\pi/2$, but the inaccurate component of $\sqrt{a}$ will cause it to evaluate to $i\pi/2$ since $-i\sqrt{a}$ is near the branch cut of ln(z). More will be said about branch cuts in the next section.

It turns out that $\sqrt{z}$ can be computed with accurate components and without loss in execution time. This function, along with the inverse trigonometric and hyperbolic functions, is computed on the HP-15C with accurate components. Their algorithms are not described by a simple formula as with sin(z) in equation (2), but rather are described in terms of their components. These accurate components are achieved by recognizing and eliminating errors such as those described above.

Principal Branches

The function $\sqrt{z}$ is an inverse function of $f(z) = z^2$. As is often the case with defining inverses, we must select from more than one solution to define the principal branch of the inverse. This is done for the real function by selecting the non-negative solution of $x^2 = a$ and denoting it by $\sqrt{a}$. Because of the branch point at 0, any branch for $\sqrt{a}$ must have a discontinuity along some slit (branch cut). In equation (3) above, it is along the negative real axis. Notice in Fig. 8 that values below the negative real axis map to values near the negative imaginary axis, while above the slit, values map near the positive imaginary axis. Since it is traditional to have $i = \sqrt{-1}$ we must attach the slit (negative real axis) to the upper half plane, making it continuous from above and not from below, that is, $-\pi<\theta\leq\pi$. One will occasionally see $\sqrt{z}$ defined for $0\leq\theta<2\pi$, which places the discontinuity along the positive real axis. We have avoided doing something like this in the branches of all of the complex inverse functions so that each will be analytic in a region about its real domain. This is important since complex computation is often performed in a region about the real domain in which the function's values are defined by the analytic continuation from the real axis.

The placement of the branch cuts and the function values along the slit are fairly standard for $\sqrt{z}$ and ln(z), but the inverse trigonometric and hyperbolic functions have not, as yet, become standardized. However, by following a few reasonable rules there is not much room for variation.

The first rule, analyticity about the real domain, has already been mentioned. Secondly, we have tried to preserve fundamental relationships such as the oddness or evenness of functions (e.g., $\sin(-z) = -\sin(z)$) and the computational formulas relating functions to the standard principal branches of ln(z) and $\sqrt{z}$ (e.g., $\pi/2-\sin(z) = g(z)\sqrt{1-z}$ where g(z) is analytic at 1, that is, a power series in $z-1$).

The determination of formulas involving a choice of branches is often quite complicated. W.M. Kahan has presented a very enlightening discussion[5] of branch cuts and has pointed out to us that the HP-15C branch cuts should satisfy certain simple formulas relating them to the principal branch of ln(z). These formulas are satisfied and are reproduced below.

$$\ln(z) = \ln(|z|) + i\,\mathrm{Arg}(z)$$

and

$$\sqrt{z} = \exp(\ln(z)/2)$$

with $-\pi < \mathrm{Arg}(z) \leq \pi$ and $\sqrt{0} = 0$

$$\begin{aligned}
\operatorname{arctanh}(z) &= [\ln(1+z) - \ln(1-z)]/2 \\
&= -\operatorname{arctanh}(-z) \\
\arctan(z) &= -i\,\operatorname{arctanh}(iz) \\
&= -\arctan(-z) \\
\operatorname{arcsinh}(z) &= \ln(z + \sqrt{1+z^2}) \\
&= -\operatorname{arcsinh}(-z) \\
\arcsin(z) &= -i\,\operatorname{arcsinh}(iz) \\
&= -\arcsin(-z) \\
\arccos(z) &= \pi/2 - \arcsin(z) \\
\operatorname{arccosh}(z) &= 2\ln[\sqrt{(z+1)/2} + \sqrt{(z-1)/2}\,]
\end{aligned}$$

These are not intended as algorithms for computation, but as relations defining precisely the principal branch of each function.

Matrix Calculations

As mentioned earlier, the HP-15C can perform matrix addition, subtraction, and multiplication. It can also calculate determinants, invert square matrices, and solve systems of linear equations. In performing these last three operations, the HP-15C transforms a square matrix into a computationally convenient and mathematically equivalent form called the LU decomposition of that matrix.

LU Decomposition

The LU decomposition procedure factors a square matrix, say **A**, into a matrix product **LU**. **L** is a lower-triangular square matrix with 1s on its diagonal and with subdiagonal elements having values between −1 and 1, inclusive. **U** is an upper-triangular square matrix. The rows of matrix **A** may be permuted in the decomposition procedure. The possibly row-permuted matrix can be represented as the matrix

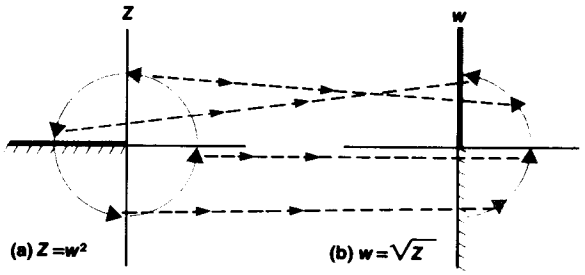


Fig. 8. *The complex function $Z = w^2$, shown in (a) has an inverse function $w = \sqrt{Z}$, shown in (b), which maps the Z plane onto the right half plane of w with a branch cut along the negative real axis of the Z plane.*

product **PA** for some invertible matrix **P**. The LU decomposition can then be represented by the matrix equation $\mathbf{PA} = \mathbf{LU}$ or $\mathbf{A} = \mathbf{P}^{-1}\mathbf{LU}$.

The HP-15C uses the Doolittle method with partial pivoting to construct the LU decomposition. It constructs the decomposition entirely within the result matrix. The upper-triangular part of **U** and the subdiagonal part of **L** are stored in the corresponding parts of the result matrix. It is not necessary to save the diagonal elements of **L** since they are always equal to 1.

Partial pivoting is a strategy of row interchanging to reduce rounding errors in the decomposition. The row interchanges are recorded in the otherwise underused XS format fields of the result matrix's diagonal elements. The recorded row interchanges identify the result matrix as containing an LU decomposition and the result matrix's descriptor includes two dashes when displayed.

The determinant of the decomposed matrix **A** is just $(-1)^r$ times the product of the diagonal elements of **U**, where r is the number of row interchanges represented by **P**. The HP-15C computes the signed product after decomposing the argument matrix **A** into the result matrix.

The HP-15C calculates the inverse of the decomposed matrix using the relationship

$$\mathbf{A}^{-1} = \left[\mathbf{P}^{-1}\mathbf{LU}\right]^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{P}$$

It does this by inverting both **U** and **L**, computing the product of their inverses, and then interchanging the columns of the product in the reverse order of the row interchanges of **A**. This is all done within the result matrix.

Solving a system $\mathbf{AX}=\mathbf{B}$ for **X** is equivalent to solving $\mathbf{LUX}=\mathbf{PB}$ for **X**, where $\mathbf{PA}=\mathbf{LU}$ denotes the LU decomposition of **A**. To solve this system, the HP-15C first decomposes the matrix **A** in place. The calculator then solves the matrix equation $\mathbf{LY}=\mathbf{PB}$ for matrix **Y** (forward substitution) and finally $\mathbf{UX}=\mathbf{Y}$ for matrix **X** (backward substitution), placing the solution **X** into the result matrix.

The LU decomposition is returned by a determinant or system solution calculation and can be used instead of the original matrix as the input to subsequent determinant, matrix inverse, or system solution calculations.

Norms and the Condition Number

A norm of a matrix **A**, denoted by $\|\mathbf{A}\|$, is a matrix generalization of the absolute value of a real number or the magnitude of a complex number. Any norm satisfies the following properties:

- $\|\mathbf{A}\| \geq 0$ for any matrix and $\|\mathbf{A}\| = 0$ if and only if $\mathbf{A} = 0$
- $\|a\mathbf{A}\| = |a| \times \|\mathbf{A}\|$ for any number a and matrix **A**
- $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$ for any matrices **A** and **B**
- $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \times \|\mathbf{B}\|$ for any matrices **A** and **B**.

One measure of the distance between two matrices **A** and **B** is the norm of their difference, $\|\mathbf{A}-\mathbf{B}\|$. A norm can also be used to define a condition number of a square matrix, which measures the sensitivity of matrix calculations to perturbations in the elements of that matrix.

The HP-15C provides three norms. The Frobenius norm of a matrix **A**, denoted $\|\mathbf{A}\|_F$, is the square root of the sum of the squares of the matrix elements. This is a matrix generalization of the Euclidean length of a vector.

The HP-15C also provides the row (or row-sum) norm. The row norm of an m-by-n matrix **A** is the largest row sum of absolute values of its elements and is denoted by $\|\mathbf{A}\|_R$:

$$\|\mathbf{A}\|_R = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|$$

The column (or column-sum) norm of a matrix **A** is denoted by $\|\mathbf{A}\|_C$ and is the largest column sum of absolute values of its elements. It can be computed as the row norm of the transpose of the matrix **A**.

For any choice of norm, a condition number K(**A**) of a square matrix **A** can be defined by

$$K(\mathbf{A}) = \|\mathbf{A}\| \times \|\mathbf{A}^{-1}\|$$

Then $K(\mathbf{A}) = \|\mathbf{A}\| \times \|\mathbf{A}^{-1}\| \geq \|\mathbf{AA}^{-1}\| = \|\mathbf{I}\| \geq 1$ for any norm. The following discussion assumes the condition number defined by the row norm. Similar statements can be made for the other norms.

If rounding or other errors are present in matrix elements, these errors will propagate through subsequent matrix calculations. They can be magnified significantly. Consider, for example, the matrix product **AB** where **A** is a square matrix. Suppose that **A** is perturbed by the matrix $\Delta\mathbf{A}$. The relative size of this perturbation can be measured as $\|\Delta\mathbf{A}\|/\|\mathbf{A}\|$. The relative size of the resulting perturbation in the product is then

$$\begin{aligned}\|\Delta\mathbf{A}\,\mathbf{B}\|/\|\mathbf{A}\,\mathbf{B}\| &= \|\Delta\mathbf{A}\,\mathbf{A}^{-1}\mathbf{A}\,\mathbf{B}\|/\|\mathbf{A}\,\mathbf{B}\| \\ &\leq \|\Delta\mathbf{A}\,\mathbf{A}^{-1}\| \\ &\leq \|\Delta\mathbf{A}\| \times \|\mathbf{A}^{-1}\| \\ &= K(\mathbf{A})\ \|\Delta\mathbf{A}\|/\|\mathbf{A}\|\end{aligned}$$

with equality for some choices of **A**, **B**, and $\Delta\mathbf{A}$. Hence K(**A**) measures how much the relative uncertainty of a matrix can be magnified when propagated into a matrix product.

Uncertainties in the square system matrix **A** or the matrix **B** of the system of equations $\mathbf{AX}=\mathbf{B}$ will also propagate into the solution **X**. For small relative uncertainties $\Delta\mathbf{A}$ in **A**, say $\|\Delta\mathbf{A}\|/\|\mathbf{A}\| << 1/K(\mathbf{A})$, the condition number is a close approximation to how much the relative uncertainty in **A** or **B** can be magnified in the solution **X**.[6]

A matrix is said to be ill-conditioned if its condition number is very large. We have seen that errors in the data—sometimes very small relative errors—can cause the solution of an ill-conditioned system to be quite different from the solution of the original system. In the same way, the inverse of a perturbed ill-conditioned matrix can be quite different from the inverse of the unperturbed matrix. But both differences are bounded by the condition number; they can be relatively large only if the condition number is large.

Singular and Nearly Singular Matrices

A large condition number also indicates that a matrix is relatively close to a singular matrix (determinant = 0). Suppose that **A** is a nonsingular matrix.

$$1/K(\mathbf{A}) = \min (\| \mathbf{A}-\mathbf{S} \| / \| \mathbf{A} \|)$$

$$\text{and } 1/\| \mathbf{A}^{-1} \| = \min (\| \mathbf{A}-\mathbf{S} \|),$$

where each minimum is taken over all singular matrices **S**.[6] $1/\| \mathbf{A}^{-1} \|$ is the distance from **A** to the nearest singular matrix. 1/K(**A**) is this distance divided by the norm of **A**.

For example, if

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0.9999999999 \end{bmatrix}$$

then

$$\mathbf{A}^{-1} = \begin{bmatrix} -9{,}999{,}999{,}999 & 10^{10} \\ 10^{10} & -10^{10} \end{bmatrix}$$

and $\| \mathbf{A}^{-1} \| = 2\times10^{10}$. Therefore, there should exist a perturbation matrix $\Delta\mathbf{A}$ with $\| \Delta\mathbf{A} \| = 5\times10^{-11}$ that makes $\mathbf{A} + \Delta\mathbf{A}$ singular. Indeed,

$$\Delta\mathbf{A} = \begin{bmatrix} 0 & -5\times10^{-11} \\ 0 & 5\times10^{-11} \end{bmatrix}$$

has $\| \Delta\mathbf{A} \| = 5\times10^{-11}$, and

$$\mathbf{A} + \Delta\mathbf{A} = \begin{bmatrix} 1 & 0.99999999995 \\ 1 & 0.99999999995 \end{bmatrix}$$

is singular.

In principle, because the HP-15C's matrices are bounded in size, exact arithmetic and exact internal storage could be used to ensure 10-digit accuracy in matrix calculations. This was considered prohibitively expensive, however. Instead, the HP-15C is designed to perform arithmetic and store intermediate calculated values using a fixed number of digits.

Numerical determinant, matrix inversion, and system solution calculations using a fixed number of digits introduce rounding errors in their results. These rounding errors can be conceptually passed back to the input data and the calculated results interpreted as exact results for perturbed input data $\mathbf{A}+\Delta\mathbf{A}$. If the norm of the conceptual perturbation $\Delta\mathbf{A}$ is comparable to $1/\| \mathbf{A}^{-1} \|$, the original nonsingular input matrix **A** may be numerically indistinguishable from a singular matrix.

For example, a square matrix is singular if and only if at least one of the diagonal elements of **U**, the upper triangular matrix in the LU decomposition of **A**, is zero. But because the HP-15C performs calculations with only a finite number of digits, some singular and nearly singular matrices cannot be distinguished in this way.

The matrix

$$\mathbf{B} = \begin{bmatrix} 3 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/3 & 1 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 0 & 0 \end{bmatrix} = \mathbf{L}\,\mathbf{U}$$

is singular. Using 10-digit accuracy, the calculated LU decomposition is

$$\mathbf{L}\,\mathbf{U} = \begin{bmatrix} 1 & 0 \\ 0.3333333333 & 1 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 0 & 10^{-10} \end{bmatrix}$$

which is the decomposition of the nonsingular matrix

$$\mathbf{D} = \begin{bmatrix} 3 & 3 \\ 0.9999999999 & 1 \end{bmatrix}$$

Hence the calculated determinants of **B** and **D** are identical. On the other hand, the matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 3 \\ 1 & 0.9999999999 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/3 & 1 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 0 & -10^{-10} \end{bmatrix} = \mathbf{L}\,\mathbf{U}$$

is nonsingular. Using 10-digit accuracy, the calculated LU decomposition is

$$\mathbf{L}\,\mathbf{U} = \begin{bmatrix} 1 & 0 \\ 0.3333333333 & 1 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 0 & 0 \end{bmatrix}$$

which is the LU decomposition of the singular matrix

$$\mathbf{C} = \begin{bmatrix} 3 & 3 \\ 0.9999999999 & 0.9999999999 \end{bmatrix}$$

The calculated determinants of **A** and **C** are also identical.

Because the calculated LU decompositions of some singular and nonsingular matrices are identical, any test for singularity based upon a calculated decomposition would be unreliable. Some singular matrices would fail the test and some nonsingular ones would pass it. Therefore, no such test is built into the HP-15C.

Instead, if a calculated diagonal element of **U**, which we call a pivot, is found to be zero during the LU decomposi-

tion, rather than aborting the matrix calculation and reporting the input matrix to be singular, the HP-15C replaces the zero pivot by a small positive number and continues with the calculation. This number is usually small compared to the rounding errors in the calculations. Specifically, it will be about 10^{-10} times the largest absolute value of any element in that column of the original matrix. If every element in that column of the original matrix has an absolute value less than 10^{-89}, the value 10^{-99} is used instead.

An advantage of replacing zero pivots by nonzero pivots is that matrix inversion and system solution calculations will not be interrupted by zero pivots. This is especially useful in applications such as calculating eigenvectors using the method of inverse iteration. Example programs calculating eigenvalues and eigenvectors can be found in reference 3.

The effect of rounding errors and possible intentional perturbations causes the calculated decomposition to have all nonzero pivots and to correspond to a nonsingular matrix usually identical to or negligibly different from the original matrix.

Complex Matrix Calculations

The HP-15C only operates on real matrices, that is, matrices with real elements. However, it is possible to represent complex matrices as real matrices and to perform matrix addition, subtraction, multiplication, and inversion of complex matrices and to solve complex systems of equations using these real representations.

Let $\mathbf{Z} = \mathbf{X} + i\mathbf{Y}$ denote a complex matrix with real part $\mathbf{X}$ and imaginary part $\mathbf{Y}$, both real matrices. One way to represent $\mathbf{Z}$ as a real matrix is as the partitioned matrix

$$\mathbf{Z}^{\mathrm{P}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix}$$

having twice the number of rows but the same number of columns as $\mathbf{Z}$. Complex matrices can be added or subtracted by adding and subtracting such real representations.

Another computationally useful real representation for $\mathbf{Z}$ is

$$\tilde{\mathbf{Z}} = \begin{bmatrix} \mathbf{X} & -\mathbf{Y} \\ \mathbf{Y} & \mathbf{X} \end{bmatrix}$$

having twice the number of rows, and columns as $\mathbf{Z}$. The HP-15C's built-in matrix operation **MATRIX 2** performs the transformation

$$\mathbf{Z}^{\mathrm{P}} \rightarrow \tilde{\mathbf{Z}}$$

The operation **MATRIX 3** performs the inverse transformation

$$\tilde{\mathbf{Z}} \rightarrow \mathbf{Z}^{\mathrm{P}}$$

Suppose $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are complex matrices and $\mathbf{A}$ is invertible. Then complex matrix multiplication, inversion, and system solution can be performed with real matrices and built-in HP-15C operations using the relationships:

$$(\mathbf{AB})^{\mathrm{P}} = \tilde{\mathbf{A}}\mathbf{B}^{\mathrm{P}},$$

$$(\widetilde{\mathbf{A}^{-1}}) = (\tilde{\mathbf{A}})^{-1},$$

$$\mathbf{A}\,\mathbf{C} = \mathbf{B} \rightarrow \mathbf{C}^{\mathrm{P}} = (\tilde{\mathbf{A}})^{-1}\,\mathbf{B}^{\mathrm{P}}.$$

These procedures are illustrated in the *HP-15C Owner's Handbook*.

Matrix Transpose

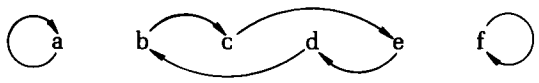
The operations **MATRIX 2** and **MATRIX 3** perform their transformations using a matrix transpose routine. The rows and columns of a matrix are interchanged to form the transpose of that matrix. The transformation is performed in place, replacing the original matrix by its transpose. This routine is available to the user as **MATRIX 4**. Consider the following example:

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \rightarrow \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$$

Here the elements of the matrices have been displayed in a two-dimensional format. However, they are stored in a one-dimensional sequence within the calculator's memory. For this example, the transpose operation changes the ordering of the elements within the calculator memory as

a b c d e f → a d b e c f.

The **MATRIX 4** operation moves the elements according to



These movements form disjoint loops. The first value in the sequence is the first candidate for moving. As a value is copied into its destination, that destination is tagged in its XS field. The previous value at that location is the next candidate for moving. Movement along a loop continues until a destination is encountered that is already tagged. The content of the tagged destination is not changed and the current loop is terminated. The value in the location immediately following that tagged destination is the next candidate for moving.

This operation continues moving values along loops until the sequence is exhausted, at which point all destination tags are removed. Finally, the recorded dimensions of the matrix are switched.

Accuracy of Matrix Calculations

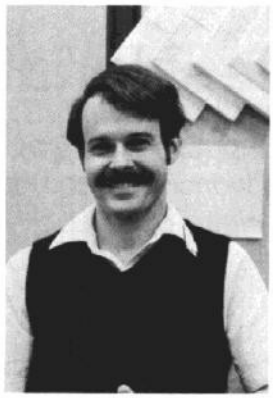
Accuracy specifications for all matrix operations are given in reference 3. These specifications are stated in terms of both backward and forward error analysis. Reference 3 includes a general rule of thumb for the number of significant digits in a calculated matrix inverse or system solution. It also includes descriptions of techniques to improve upon the accuracy of calculated system solutions and to reduce the ill-conditioning of systems of equations.

Acknowledgments

Numerous individuals made valuable contributions to the HP-15C software effort. As the software project manager, Rich Carone helped formulate some of the original design concepts and kept the software effort on track. Diana Roy, Robert Barkan, and Hank Schroeder wrote the *HP-15C Owner's Handbook*. We would like to give special thanks to Professor William Kahan, who contributed many design ideas, provided strong guidance in developing the mathematical algorithms, and wrote a portion of the *HP-15C Advanced Functions Handbook*. His unbounded enthusiasm for the product helped keep us going, especially when we still had features to implement and no ROM space left.

References

1. W.M. Kahan, "Personal Calculator Has Key to Solve Any Equation f(x)=0," Hewlett-Packard Journal, Vol. 30, no. 12, December 1979.
2. W.M. Kahan, "Handheld Calculator Evaluates Integrals," Hewlett-Packard Journal, Vol. 31, no. 8, August 1980.
3. *HP-15C Advanced Functions Handbook.*
4. D.W. Harms, "The New Accuracy: Making $2^3=8$," Hewlett-Packard Journal, Vol. 28, no. 3, November 1976, p. 16.
5. W.M. Kahan, "Branch Cuts for Complex Elementary Functions," working document for IEEE Floating-Point Standards Subcommittee, September 9, 1982.
6. E. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons, 1978, pp. 461-463.



Paul J. McClellan

Paul McClellan started working part-time at HP in 1979 while he was a graduate student at Oregon State University. He has a BS degree in physics and mathematics awarded by the University of Oregon in 1974 and an MS degree in statistics from Oregon State. He plans to complete the requirements for the PhD degree in statistics this year. Paul began full-time work at HP in late 1982 and worked on the **SOLVE** and matrix computation routines for the HP-15C. He is a member of the American Statistical Association and coauthor of the *HP-15C Advanced Functions Handbook*. When not busy with work or his studies, Paul enjoys rock climbing, mountaineering, cross-country skiing, and visiting Portland, Oregon. He is single and lives in Corvallis, Oregon.

Joseph P. Tanzini

Joe Tanzini was born in Trenton, New Jersey and attended Trenton State College, receiving a BA degree in mathematics in 1973. He continued his mathematics studies at Lehigh University and earned the MS and PhD degrees in 1975 and 1982. Joe started at HP as a software engineer in 1980 and wrote firmware for calculators, including coding the integrate algorithm and complex functions for the HP-15C. He also coauthored the *HP-15C Advanced Functions Handbook*. Joe enjoys bicycling and walking during his leisure time. He is married, has two daughters, and lives in Corvallis, Oregon.

A Pocket Calculator for Computer Science Professionals

This compact, yet powerful pocket calculator is designed for technical professionals working in computer science and digital electronics. Boolean operations and bit manipulation are some of its capabilities.

by Eric A. Evett

LOGIC DESIGN and computer programming require mathematical operations not ordinarily provided by small calculators. A large amount of tedious paperwork is often required to convert among number bases, perform logic operations, shift and rotate bits in a word, or check processor instruction flow. To simplify such work, Hewlett-Packard recently introduced a programmable pocket calculator especially designed for people who deal with bits. The HP-16C (Fig. 1), like other HP calculators, uses a reverse-Polish-notation (RPN) system and provides standard floating-point decimal arithmetic (including square root). Its novel capabilities become apparent, however, when the HP-16C is switched into the integer mode. Only integers are allowed in this mode, and they can be keyed in and displayed in either hexadecimal, octal, binary, or decimal format. In this mode, number base conversion, integer arithmetic, logical operations, and bit manipulations can be done.

Integer Mode

In the integer mode, all numbers are represented internally in binary form. The word size is selected by the user and can range from 1 to 64 bits. The user also can select whether the numbers are to be interpreted as one's complement, two's complement, or unsigned integers. In the unsigned integer mode with a 64-bit word size, numbers up to 2^{64-1} (18,446,744,073,709,551,615) can be represented. Although the HP-16C normally displays the eight least-significant digits of a number, a scrolling capability is provided to display higher-order digits.

Programming

In addition to the four-register RPN stack, 203 bytes of user memory are available for storing program steps and use as storage registers. When the program memory is cleared, all 203 bytes are allocated to storage registers. The number of storage registers available depends on the selected word

Fig. 1. *The HP-16C Programmable Calculator is designed for computer science and digital electronics applications. Besides the normal four-function calculator features, it has a number of capabilities for setting number bases and word sizes, performing Boolean operations, and manipulating bits.*

Real (Floating-Point) Format

Real numbers are represented in the HP 3000 memory by 32 bits (two consecutive 16-bit words) separated into three fields. These fields are the sign, the exponent, and the mantissa. The format is known as excess 256. Thus, a real number consists of (see Fig. 1):

- Sign (S), bit 0 of the first 16-bit word. Positive=0, negative=1. A value X and its negative −X differ only in the value of the sign bit.
- Exponent (E), bits 1 through 9 of the first 16-bit word. The exponent ranges from 0 to 777 octal (511 decimal). This number represents a binary exponent biased by 400 octal (256 decimal). The true exponent, therefore is E−256; it ranges from −256 to +255.
- Fraction (F), a binary number of the form 1.xxx, where xxx represents 22 bits stored in bits 10 through 15 of the first 16-bit word and all bits of the second 16-bit word. Note that the 1 is not actually stored, there is an assumed 1 to the left of the binary point. Floating-point zero is the only exception. It is represented by all 32 bits being zero.

The range of nonzero real values for this format is from 0.863617×10^{-77} to 0.1157920×10^{78}. The formula for computing the decimal value of a floating-point representation is: Decimal value = $(-1)^S \times 2^{E-256} \times F$.

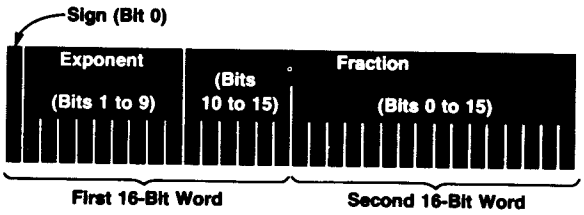


Fig. 1. *Diagram of real (floating-point) format used in the HP 3000.*

```
LBL A
2'S          Set two's complement mode.
OCT          Convert to octal integer mode,
             and return integers y and x
             such that 2^y=original input.
SF 3         Leading zeros will be displayed.
X=Y          Was input 0?
GTO 1        If yes, then branch to Label 1.
4
3
7
+            Bias exponent. 287=256+31
STO I        Store biased exponent in index register.
X≷Y          Swap exponent and mantissa.
SF 0         Set flag 0.
X<0          Mantissa negative?
CF 0         If yes, clear flag 0.
ABS          Absolute value of mantissa.
4
0
WSIZE        Set word size to 32.
4
0
0
+            Round mantissa to 23 bits.
2
7
MASKL        Create mask of 23 bits, left-justified.
AND          Extract upper 23 bits.
F? 4         Did round cause a carry-out
             of most significant bit?
ISZ          If yes, increment exponent.
SL           Shift off implied 1 bit.
RCL I        Recall biased exponent.
OR           Concatenate exponent to fraction part.
F? 0         Is mantissa sign to be positive?
GTO 1        If yes, branch to Label 1.
1
1
SB           Set the sign bit.
LBL 1
1
2
RRn          Rotate sign, exponent, and fraction
             to proper position.
STO 0        Store the 32-bit result in register 0.
2
0
WSIZE        Change word size to 16 bits.
RCL 1        Recall 16-bit word 1.
RCL 0        Recall 16-bit word 2.
RTN
```

Fig. 2. *Outline of HP-16C subroutine to convert numbers given in the HP 3000 Computer's real format to decimal floating-point format.*

Using the HP-16C

Listing the features of a programmable calculator rarely provides a complete picture of its capabilities. Examples of the application of the calculator's features are often required to demonstrate to the user what can be done and why a particular feature is useful. Hence, several examples of the use of the HP-16C are given below.

Add with Carry

The HP-16C can be programmed to simulate instructions commonly found in commercial processors. The following subroutine performs an add with carry (Y+X+C→X). It adds the numbers in registers X and Y along with the carry bit indicated by the state of flag 4 and returns the result in the X register. The carry flag is set (indicated by the C annunciator in the display) if there is a carry-out of the most significant bit of the result.

001	**LBL A**	Labels subroutine
002	**0**	
003	**RLC**	Generates 0 or 1 depending on carry flag
004	**+**	Adds carry to second operand
005	**CF 0**	
006	**F? 4**	Copies carry flag 4 to flag 0
007	**SF 0**	
008	**+**	Adds first operand to the total
009	**F? 0**	
010	**SF 4**	Sets carry flag if first add carried
011	**RTN**	

To use this routine, enter the two operands in registers Y and X, and press **GSB A**.

Example:

2'S	Set two's complement mode
HEX	Set number base to hexadecimal
8	
WSIZE	Set word size to 8
CF 4	Clear carry flag
FE	First operand
ENTER	Enter first operand into Y register
72	Second operand
GSB A	Displays 70 (FE+72+0) with carry set (C annunciator on)

Bit Extraction

The following subroutine extracts a field from a bit pattern. The field is specified by the bit numbers of the pattern corresponding to the lowest-order and highest-order bits of the field. The least-significant bit of the bit pattern is bit number 0. Hence, the result in the X register is the bits of the pattern in the Z register from the bit number in the Y register to the bit number in the X register, inclusive.

001	**LBL B**	Labels subroutine
002	**R↓**	Bring down value in Y register
003	**RRn**	Right-justifies field
004	**R↑**	Raise stack
005	**LST x**	Recall Y value
006	**–**	Subtract Y from X
007	**1**	
008	**+**	Computes number of bits in field
009	**MASKR**	Creates mask same width as field
010	**AND**	Extracts field
011	**HEX**	Exits in the hexadecimal mode
012	**RTN**	

size. When the word size is eight bits, 203 registers are available; a 16-bit word size results in 101 available registers, and so on. Each programmable instruction takes one byte of memory. As program steps are inserted, the number of available storage registers decreases. A program can have up to 203 steps if no storage registers are required.

Editing capabilities to make program development easier include insert, delete, back-step, single-step, and go-to-line-number operations. The user may single-step through program execution to help debug programs. Other programming features include label addressing (sixteen labels), subroutines (up to four levels deep), conditional tests, branching, and six user flags.

These flags can be set, cleared, and tested under program control. Three of the flags are special. Leading zero digits in a word are suppressed in the display unless flag 3 is set. Flag 4 is the carry flag, and flag 5 is the overflow flag. Two annunciators in the display (C for carry and G for > largest representable number) give a visual indication of the state of flags 4 and 5, respectively. The overflow flag is set if the true result of an operation cannot be represented in the selected word size and complement mode. The carry flag is set under various conditions, depending on the operation. For example, addition sets the carry flag if there is a carry-out of the most significant bit; otherwise the carry is cleared (see box above for examples). The shift-left instruction sets the carry if a 1 bit is shifted off the left end of the word; otherwise the carry is cleared.

Logic Operations

The rich selection of bit manipulation and logical operations, along with user-selectable complement mode and word size, make the HP-16C a flexible logic and program design tool. Programs can be written to simulate individual instructions commonly found on commercial processors, to extract a field from a bit pattern, or to convert from one numeric format to another.

A common problem is the conversion between the internal binary floating-point format of a particular machine and decimal floating-point format. The HP-16C provides a feature that can be used to great advantage in programs designed to perform such conversions. This feature provides a mode for performing standard decimal floating-point calculations. Upon switching from the integer mode to decimal floating-point mode (by using the **FLOAT** function), the integers y and x in the Y and X stack registers are converted to the floating-point equivalent of $2^x y$, which is then placed in the X register and displayed. Converting back to integer mode (by pressing the **HEX**, **DEC**, **OCT**, or **BIN** keys), causes the contents of the X register to be converted to a pair of integers y and x such that y is a 32-bit integer ($2^{31} \leq |y| < 2^{32}$ unless $y=0$) and $2^x y$ is equal to the value in the X register before mode conversion. The integers y and x are then placed in the Y and X registers.

To use this routine, the user places the bit pattern in register Z, the number of the lowest-order bit in the field in register Y, and the number of the highest-order bit in the field in register X. The user then presses **GSB B**.

Example: Extract bits 2 through 5 from 39_{16} (00111001).

8	
WSIZE	Set wordsize to 8
HEX	Set hexadecimal mode
39	Bit pattern
ENTER	
2	Lowest-order bit
ENTER	
5	Highest-order bit
GSB B	Displays E (1110) as result.

Conversion Between Binary and Gray Code

Gray code has the property that only one bit changes between the representations of any two adjacent numbers. If the word size is n bits, then binary-to-Gray-code conversion is given by

$$G_0 = B_0 \text{ XOR } B_1$$

$$G_1 = B_1 \text{ XOR } B_2$$

$$\vdots$$

$$G_{n-2} = B_{n-2} \text{ XOR } B_{n-1}$$

$$G_{n-1} = B_{n-1}$$

where G is the Gray code number, B is the binary number, and subscript 0 indicates the least-significant bit of G and B, subscript 1 indicates the next least-significant bit, and so forth.

The Gray-code-to-binary conversion is given by

$$B_0 = G_0 \text{ XOR } G_1 \text{ XOR } \cdots G_{n-1}$$

$$B_1 = G_1 \text{ XOR } G_2 \text{ XOR } \cdots G_{n-1}$$

$$\vdots$$

$$B_{n-2} = G_{n-2} \text{ XOR } G_{n-1}$$

$$B_{n-1} = G_{n-1}$$

Binary-to-Gray-code subroutine:

001	**LBL C**	
002	**ENTER**	Copies binary number to Y register
003	**SR**	Shifts binary number in X register to the right
004	**XOR**	Computes Gray-code equivalent
005	**RTN**	

Gray-to-binary-code subroutine:

001	**LBL D**	
002	**ENTER**	Copies Gray code number to Y register
003	**LBL 2**	
004	**SR**	Shift Gray code number in X register to the right
005	**XOR**	Exclusive OR operation
006	**LST x**	Recall previous number
007	**X≠0**	Loop until Gray code number is 0
008	**GTO 2**	
009	**R↓**	
010	**RTN**	

To use these routines, the user sets the HP-16C to the binary mode by pressing **BIN**, places the number in the X register and presses **GSB C** for binary-to-Gray or **GSB D** for Gray-to-binary-code conversions.

LBL B	
HEX	Set hexadecimal base mode.
2'S	Set two's complement mode.
2	
0	
WSIZE	Set word size to 32 bits.
X≷Y	Swap word 1 and word 2.
1	
0	
RLn	Shift word 1 16 bits to left.
OR	Concatenate word 2 to word 1.
SL	Shift sign bit left into carry flag.
ENTER	
X=0	Is input 0?
GTO 0	If yes, branch to Label 0.
1	
7	
MASKR	Create mask of 23 bits, right-justified.
AND	Extract fraction part.
XOR	Extract biased exponent part.
LST x	Recover fraction part.
1	
7	
SB	Set bit 23 (the implied 1-bit). Bit 0 is the least significant.
F? 3	Test carry flag. Was sign bit negative?
CHS	If yes, complement mantissa.
ASR	Arithmetic shift right mantissa 1 bit.
X≷Y	Swap mantissa and exponent part.
9	
RLn	Rotate exponent part 9 bits left, placing it at right end.
1	
1	
6	
−	Unbias exponent. $E-2^7\bar{8}=E-256-32$
LBL 0	
FLOAT .	Compute $2^{\pm y}$
RTN	

Fig. 3. *Outline of HP-16C subroutine to convert decimal floating-point numbers into the format used by HP 3000 Computers.*

The subroutines listed in Fig. 2 and Fig. 3 convert between the HP 3000 Computer's FORTRAN real (floating-point) format[2] and decimal floating-point format (see box on page 37). Because the HP-16C views bit 0 as the least significant bit of a word and the HP 3000 views it as the most significant bit, some of the steps listed in Fig. 2 and Fig. 3 are used to convert between these two opposing views.

To use these programs after they are entered in the HP-16C, a user performs the following steps.

- HP 3000 to decimal:
  1. Select octal base (**OCT**).
  2. Enter word 1 in the Y register and word 2 in the X register.
  3. Execute **GSB B**. Answer is displayed.
  4. Repeat steps 1, 2, and 3 for each new conversion.
- Decimal to HP 3000:
  1. Select decimal floating-point mode (**FLOAT 4**).
  2. Enter number in the X register.
  3. Execute **GSB A**. Word 1 is placed in the Y register, word 2 in the X register.
  4. Repeat steps 1, 2, and 3 for each new conversion.

Acknowledgments

Several people made significant contributions to the HP-16C software effort. John Van Boxtel developed many of the fundamental design concepts. Rich Carone also contributed to the software design and coded the complex routines that format and build the display. Stan Blascow did a large portion of the software testing and Diana Roy wrote the owner's handbook.

References

1. M.E. Sloan, *Computer Hardware and Organization*, Science Research Associates, Inc., 1976, pp 95-97.
2. *FORTRAN Reference Manual*, HP 3000 Computer Systems, Hewlett-Packard, Santa Clara, 1978, Section II.

Eric A. Evett

Eric Evett received the BS and MS degrees in mathematics from the University of Arizona in 1970 and 1972. He taught mathematics there until 1975 and then spent three years developing software for calculators before joining HP in late 1978. Eric wrote portions of the microcode for the HP-11C, HP-15C, and HP-16C Calculators and now is an R&D software project manager at HP's Corvallis, Oregon facility. He was born in Colfax, Washington, and now lives in Corvallis. He is married and has two sons. His outside interests include jogging, hiking, basketball, reading, movies, and tennis (he played on his college's tennis team which was then ranked 7th in the U.S.A.).

CORRECTION

In the March issue, the Pascal statements at the top of the back page were printed in the wrong order. Here is the correct version.

```
buffer[w]:=getch;
c:=c+a[buffer[w]];
w:=w+1;
```


Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, California 94304

HEWLETT-PACKARD JOURNAL

MAY 1983 Volume 34 • Number 5

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, 3000 Hanover Street
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
Van Heuven Goedhartlaan 121
1181 KK Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

5953-8511