

# What's Inside Radio Shack's Color Computer?

Tim Ahrens, Jack Browne, Hunter Scales  
3501 Ed Bluestein Blvd  
Austin TX 78721

The only similarity between Tandy Corporation's new Color Computer and its older brother—the original TRS-80—is the name. Even the microprocessor has been changed. In an apparent breakaway from the Z80, the Color Computer uses the Motorola MC6809E microprocessor as the workhorse of the new silver box. In fact, when we opened the enclosure, we didn't see any semiconductors that weren't made by Motorola.

The Color Computer is totally self-contained—no bulky separate power transformers—and the only cord, the one to the wall socket, has a standard three-prong connector. It can work with any color or black-and-white television set and has provisions for joysticks, a 1500 bps (bits per second) cassette interface, and an

expansion connector for preprogrammed game cartridges.

Our aim in this article is to expose the insides of the computer and show what makes it run. Using this information, you should be able to expand the Color Computer in a number of ways, with a minimum of expertise. We will also describe the graphics interface so that do-it-yourself graphics routines should be a piece of cake.

## System Hardware

Taking the cover off is simply a matter of removing seven screws and lifting the lid. Be warned, however, that Tandy takes a dim view of owners fooling around with their hardware. Opening the case voids the warranty on the machine (one of the screws lies under a paper label that gives this warning).

The first surprise is that the entire computer is built on a single printed-circuit board—including the power supply. Most of the digital circuitry lies inside an RFI (radio-frequency interference) shield—this was probably necessary to get FCC (Federal Communications Commission) Type Approval, but it also helps to give a clean display. To get a look at the parts, simply pry off the top of the shield.

There are only twenty-four DIPs (dual in-line packages) in the system and they are all made by Motorola. (The parts list is shown in table 1.) The machine comes stuffed with 4 K-byte memory circuits; but there is a simple way to change these to 16 K-byte devices and a tricky way to get 32 K bytes of on-board memory—more on this later.

While we do not yet have a schematic diagram, the block diagram in figure 1 should be sufficiently detailed to allow a thorough understanding of the system. There are four basic sections:

- the microprocessor
- the video-display circuitry
- the memory
- the other I/O (input/output) devices (keyboard, cassette, serial port, and joysticks)

The microprocessor is Motorola's advanced 8-bit machine, the MC6809E. It was designed to support today's high-level languages, including the Extended

*Text continued on page 96*

Part	Number of Pins	Quantity	Device Number	Description
MC6809E	40	1	1	Microprocessor
MC6821	40	2	2, 3	Parallel Interface Adapter
MC6883	40	1	4	Synchronous Address Multiplexer
MC6847	40	1	5	Video Display Generator
MCM68A364	24	2	6, 7	8 K-byte Read-Only Memory
MCM4027	16	8	8 thru 15	4 K-bit Programmable Memory
MC74LS138	16	1	16	3-bit Decoder
MC74LS02	14	1	17	Quad 2-Input NOR Gate
MC74LS244	20	1	18	Octal Buffer/Line Driver
MC74LS273	20	1	19	8-bit Latch
MC14050B	14	1	20	Hex Noninverting CMOS Buffer
MC14529B	16	1	21	Dual 4-Channel Analog Color-Subcarrier Modulator
MC1372	14	1	22	Quad Voltage Comparator
MLM339	14	1	23	Voltage Regulator
MC723C	14	1	24	Voltage Regulator
MC78M12	3	1	25	Voltage Regulator
MC79M12	3	1	26	Voltage Regulator
MC79M05	3	1	27	Voltage Regulator
UM1285-8	NA	1	28	ASTEC Video Modulator

**Table 1:** List of integrated circuits used in the TRS-80 Color Computer. Large-scale integration reduces the number of devices necessary to build in sophisticated capabilities, and improves reliability. All circuits used are manufactured by Motorola.

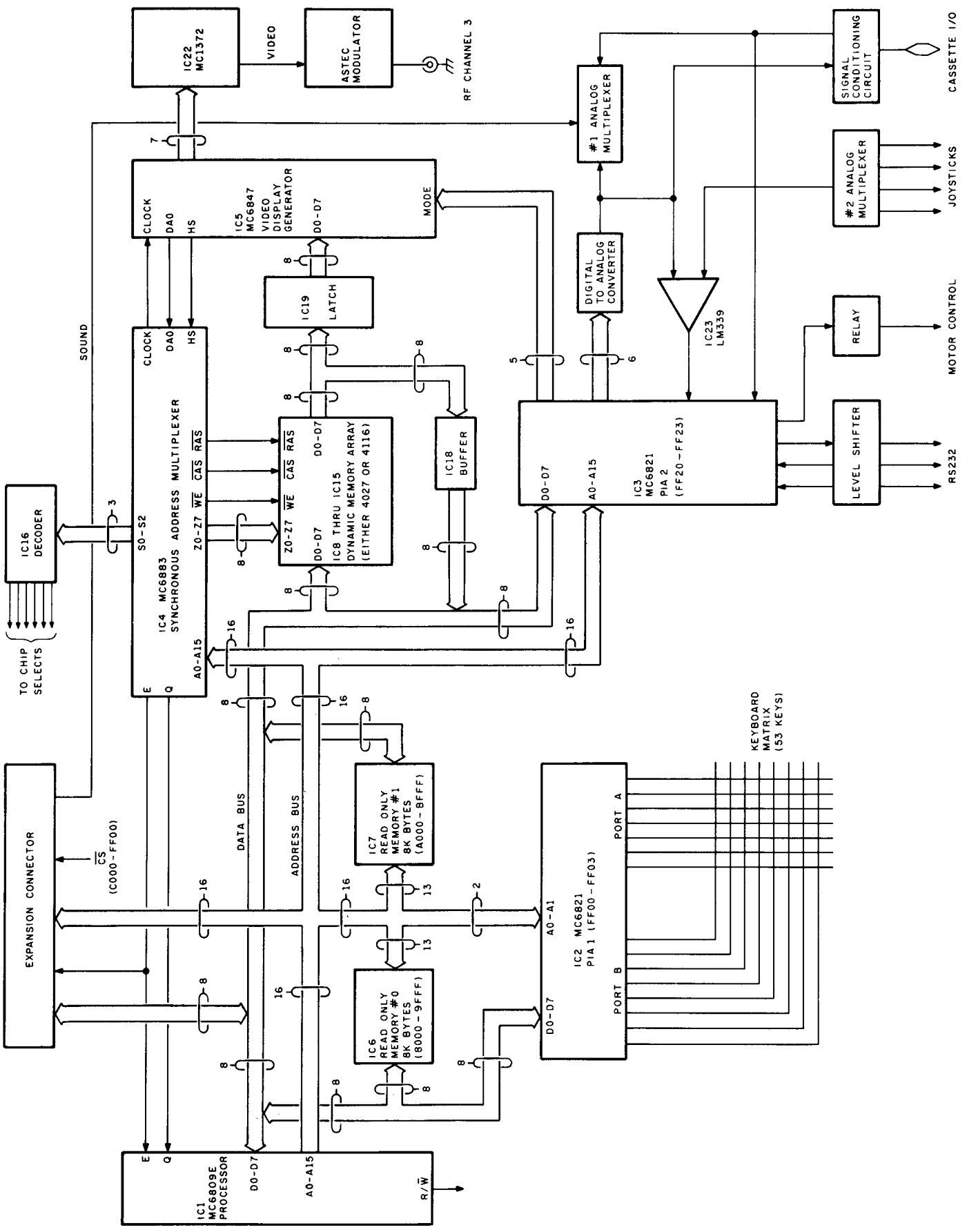
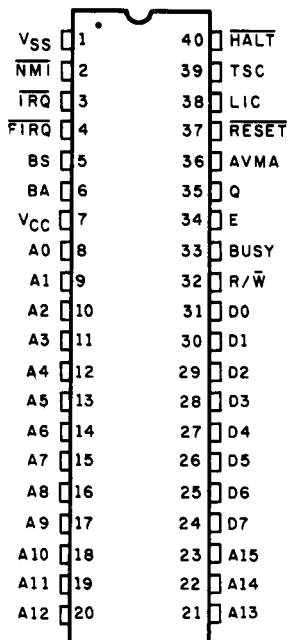
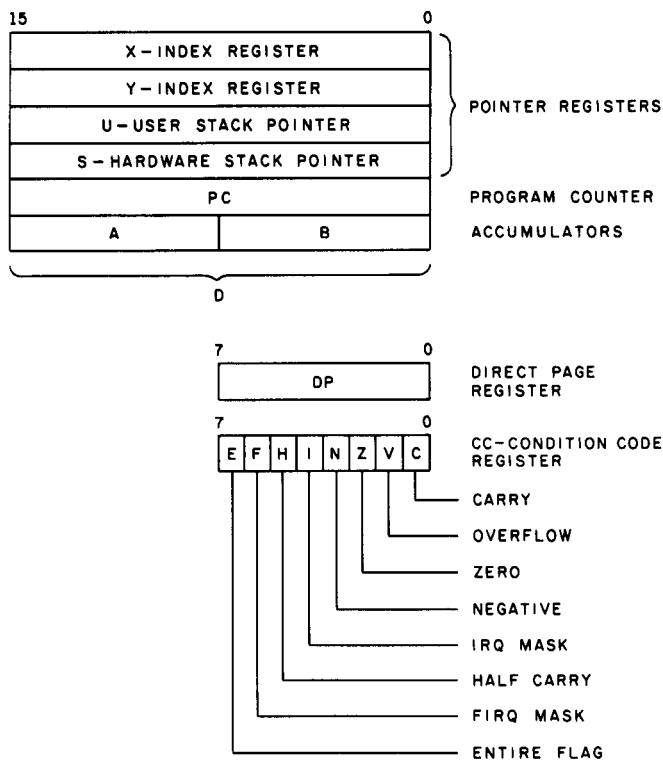


Figure 1: Block diagram of the Radio Shack Color Computer. Although a detailed schematic diagram is not available, the connection of the main components can be readily determined. Note that the use of large-scale integrated circuits (the microprocessor, SAM dynamic-memory handler, video-display generator, and parallel port interfaces) means that a minimum number of components is necessary to build this flexible computer.



**Figure 2:** Pin description of Motorola's MC6809E microprocessor. The device has several 16-bit instructions that, coupled with ease of programming and speed, make for a very powerful 8-bit processor.



**Figure 3:** Registers available in the 6809. Similar in architecture to the 6800, the 6809 has three extra registers to facilitate memory accesses: a direct page register, a user stack register, and a second index register. The instruction set is also more robust, with the addition of 16-bit add, subtract, and multiply operations.

Text continued from page 90:

BASICs now available. It has two 16-bit index registers and two 16-bit stack pointers, as well as two 8-bit accumulators that can be used as a double-precision 16-bit accumulator. It supports both position-independent code (code that can be executed anywhere in memory without reassembly) and reentrant (interruptible) code.

The video display is generated by the Motorola MC6847 VDG (video display generator). This is a 40-pin LSI (large-scale integration) part that reads from ½ K bytes to 6 K bytes of memory, depending on mode, to produce an analog video signal. This signal is fed to the MC1372 color-subcarrier modulator to get composite video, which is then modulated by the ASTEC video modulator to channel 3 or 4.

The Color BASIC interpreter is stored in an 8 K by 8 bit ROM (read-only memory). Its companion, Extended BASIC, comes in another ROM of the same type. The basic machine comes with only the first ROM; the extended ROM costs \$99 plus installation.

As mentioned, the computer comes with eight MCM4027 4 K-bit dynamic memory circuits. Tandy will upgrade your system to 16 K bytes by replacing these with MCM4116s (16 K-bit devices) for \$119. Or you can buy the system with 16 K bytes and the Extended BASIC ROM for \$599.

These memory circuits are controlled and refreshed by a special part, the MC6883 SAM (synchronous address multiplexer). It provides all the signals for the memory and the VDG and also provides the timing signals for the microprocessor.

The other I/O functions are all handled by parallel ports in the form of MC6821 PIAs (peripheral interface adapters). The keyboard is connected to these and is scanned and decoded in software. The serial port and cassette port are both derived from a single parallel line and are selected by software. The optional joysticks are encoded with an A/D (analog-to-digital) converter composed of a resistive-summing network hooked to a 6-bit parallel port and an LM339 comparator.

### The MC6809E Microprocessor

The third-generation MC6809E 8-bit microprocessor features several 16-bit operations. This puts it functionally between the 8- and the 16-bit processors. A description of the MC6809E signals appears in figure 2.

The programming model of the MC6809E is shown in figure 3. Three registers were added to the register set of the original MC6800:

- a direct page register
- a user stack pointer
- a second index register

There are two 8-bit accumulator registers, the A register and the B register, that are used for data manipulation and serve as holding registers for arithmetic calculations. The MC6809E has many 16-bit arithmetic operations, including additions, subtractions, loads, stores, and an 8 bit by 8 bit multiplication. The 16-bit arithmetic operations use both accumulators—with the A register treated

---

## Efficient position-independent code can be written using the capabilities of the MC6809E.

---

as the most significant byte. When the A and B registers are concatenated, they are referred to as the D register.

The DP (direct page) register is one of the new registers. Its contents form the high-order byte of the address bus during instructions utilizing the direct addressing mode. This register may be changed to allow direct addressing anywhere in the 64 K-byte memory map, as compared to the MC6800, which allowed direct addressing only in the first 256 bytes of the memory map. Direct addressing uses the immediate byte of the instruction as a 1-byte pointer into a single 256-byte "page" of memory. This shortens instruction execution time because the high-order byte is furnished by the direct page register. MC6800 source code compatibility is ensured because actuation of the RESET line clears the direct page register.

The MC6809E has four 16-bit pointer registers available to the user. The U and S registers support stack-oriented instructions such as PSH and PUL. The S register is used as the hardware stack pointer to support interrupts and subroutine calls. The U register gives the designer the capability of maintaining an independent stack.

The other two registers, X and Y, are intended primarily for use as index registers, although special indexing

modes allow them to be used to maintain additional stack areas. All four pointer registers can be used as index registers, allowing indexed addressing, indirect addressing, or indexed indirect addressing. These pointer register capabilities permit the MC6809E to function efficiently as a stack processor, allowing the microprocessor to support graphics, high-level languages, and modular programming techniques.

The microprocessor's program counter, while primarily used by the processor to address the next instruction, may be referenced as an index register, thus allowing addressing relative to the program counter.

The condition code register defines the state of the microprocessor such that conditional branch instructions may be used. The condition code register also allows masking of some of the interrupts.

The register set is manipulated with the 59 instructions shown in table 2. Over 1460 different op codes are available to the programmer if all modes of the instructions are considered. However, only the 59 mnemonics must be remembered when using an assembler.

Efficient PIC (position-independent code) can be written using the capabilities of the MC6809E. The program counter can be used as a pointer to provide offsets within the program. For example, when a portion of PIC is executed, the stack addresses, peripheral addresses, and other addresses may be specified as offsets from the current program counter address.

Other key factors in effective position-independent code writing are the use of long and short relative-branch

## 8-BIT OPERATIONS

Mnemonic	Description
ABX	Add B register to X register unsigned.
ADCA, ADCB	Add memory to accumulator with carry.
ADDA, ADDB	Add memory to accumulator.
ANDA, ANDB	AND memory with accumulator.
ANDCC	AND immediate with condition code register.
ASLA, ASLB, ASL	Arithmetic shift left accumulator or memory.
ASRA, ASRB, ARS	Arithmetic shift right accumulator or memory.
BITA, BITB	Bit test memory with accumulator.
CLRA, CLRB, CLR	Clear accumulator or memory.
CMPA, CMPB	Compare memory with accumulator.
COMA, COMB, COM	Complement accumulator or memory.
DAA	Decimal Adjust A accumulator.
DECA, DECB, DEC	Decrement accumulator or memory.
EORA, EORB	Exclusive OR memory with accumulator.
EXG R1, R2	Exchange R1 and R2.
INCA, INCB, INC	Increment accumulator or memory.
LDA, LDB	Load accumulator from memory.
LSLA, LSLB, LSL	Logical shift left accumulator or memory.
LSRA, LSRB, LSR	Logical shift right accumulator or memory.
MUL	Unsigned multiply (8 bit by 8 bit = 16 bit).
NEGA, NEGB, NEG	Negate accumulator or memory.
ORA, ORB	OR memory with accumulator.
ORCC	OR immediate with condition code register.
PSHS (register list)	Push register(s) on hardware stack.
PSHU (register list)	Push register(s) on user stack.
PULS (register list)	Pull register(s) from hardware stack.
PULU (register list)	Pull register(s) from user stack.
ROLA, ROLB, ROL	Rotate accumulator or memory left.
RORA, RORB, ROR	Rotate accumulator or memory right.
SBCA, SBCB	Subtract memory from accumulator with borrow.
STA, STB	Store accumulator to memory.
SUBA, SUBB	Subtract memory from accumulator.
TSTA, TSTB, TST	Test accumulator or memory.
TFR R1, R2	Transfer register R1 to register R2.

## 16-BIT OPERATIONS

Mnemonic	Description
ADDD	Add to D accumulator.
SUBD	Subtract from D accumulator.
LDD	Load D accumulator.
STD	Store D accumulator.
CMPD	Compare D accumulator.
LDX, LDY, LDX, LDU	Load pointer register.
STX, STY, STS, STU	Store pointer register.
CMPX, CMPY, CMPU, CMPS	Compare pointer register.
LEAX, LEAY, LEAS, LEAU	Load effective address into pointer register.
SEX	Sign extend
TFR register, register	Transfer register to register.
EXG register, register	Exchange register to register.
PSHS (register list)	Push register(s) onto hardware stack.
PSHU (register list)	Push register(s) onto user stack.
PULS (register list)	Pull register(s) from hardware stack.
PULU (register list)	Pull register(s) from user stack.

**Table 2:** *The 6809 instruction set.*

## INDEXED ADDRESSING MODES

Mnemonic	Description
0, R	Indexed with zero offset.
[0, R]	Indexed with zero offset indirect.
,R +	Autoincrement by 1.
,R + +	Autoincrement by 2.
[,R + +]	Autoincrement by 2 indirect.
, - R	Autodecrement by 1.
, - - R	Autodecrement by 2.
[, - - R]	Autodecrement by 2 indirect.
n, P	Indexed with signed n as offset (n = 5, 8, or 16 bits).
[n, P]	Indexed with signed n as offset indirect.
A, R	Indexed with accumulator A as offset.
[A, R]	Indexed with accumulator A as offset indirect.
B, R	Indexed with accumulator B as offset.
[B, R]	Indexed with accumulator B as offset indirect.
D, R	Indexed with accumulator D as offset.
[D, R]	Indexed with accumulator D as offset indirect.

**NOTE:** R = X, Y, U, or S; P = PC, X, Y, U, or S. Brackets indicate indirection. D means use AB accumulator pair.

## 6809 RELATIVE SHORT AND LONG BRANCHES

Mnemonic	Description
BCC, LBCC	Branch if carry clear.
BCS, LBCS	Branch if carry set.
BEQ, LBEQ	Branch if equal.
BGE, LBGE	Branch if greater than or equal (signed).
BGT, LBGT	Branch if greater (signed).
BHI, LBHI	Branch if higher (unsigned).
BHS, LBHS	Branch if higher or same (unsigned).
BLE, LBLE	Branch if less than or equal (signed).
BLO, LBLO	Branch if lower (unsigned).
BLS, LBLs	Branch if lower or same (unsigned).
BLT, LBLT	Branch if less than (signed).
BMI, LBMI	Branch if minus.
BNE, LBNE	Branch if not equal.
BPL, LBPL	Branch if plus.
BRA, LBRA	Branch always.
BRN, LBRN	Branch never.
BSR, LBSR	Branch to subroutine.
BVC, LBVC	Branch if overflow clear.
BVS, LBVS	Branch if overflow set.

## 6809 MISCELLANEOUS INSTRUCTIONS

Mnemonic	Description
CWAI	Clear condition code register bits and wait for interrupt.
NOP	No operation.
JMP	Jump.
JSR	Jump to subroutine.
RTI	Return from interrupt.
RTS	Return from subroutine.
SEX	Sign extend B register into A register.
SWI, SWI2, SWI3	Software interrupts.
SYNC	Synchronize with interrupt line.

Text continued from page 98:

instructions and LEA (load effective address) instructions. The relative-branch instructions allow PCR (program counter relative) branching. When an 8-bit offset is used, control may be transferred anywhere within a 256-byte area. A 16-bit offset allows transfer of control anywhere in the entire 64 K-byte address space. The following are examples of the relative-branch instructions:

DECA	Decrement A Accumulator
BEQ CAT	If A = 0 then go to CAT (CAT is within $\pm 128$ bytes)
INCA	Increment A Accumulator
LBEQ DOG	If A = 0 then go to DOG (DOG is within $\pm 32,768$ bytes)

The LEA instructions work by calculating the effective address of an indexed instruction and storing it in the specified pointer register. This allows the programmer to use all the internal addressing hardware of the microprocessor. Below are some examples of the LEA instructions.

Instruction	Operation
LEAX 10,X	$X + 10 \rightarrow X$
LEAY A,Y	$Y + A \rightarrow Y$
LEAX D,Y	$Y + D \rightarrow X$
LEAU -10,U	$U - 10 \rightarrow U$
LEAX TABLE,PCR	(see text below)

Note how the registers may be incremented or decremented using the LEA instructions. In addition, registers may be used as offsets, as explained above. The program counter may be used as a pointer register with 8- or 16-bit signed offsets. As in relative addressing, the offset is added to the current contents of the program counter register to create the effective address.

The last example calculates the offset of TABLE and adds it to the current value of the program counter register. This value is then placed in the X register. Tables related to a particular routine will maintain the same relationship after the routine is moved, since addresses are calculated when the code is executed.

Position-independent code is not without disadvantages, the major being that it generally takes 5 to 10 percent more space than nonrelocatable code. In addition, PIC usually takes 5 to 10 percent more time to execute. Typically, PIC would be used for utility programs where the run-time addresses are dynamically determined. This eliminates the need for a linking loader to perform a relocation operation. Common examples of this type of code would be machine-language utilities such as graphic routines and subroutines called by BASIC programs.

The MC6809E has several very interesting hardware features also. Referring to the signal descriptions of figure 2, note that not only does the microprocessor have 16 address lines, 8 data lines, and an  $R/\bar{W}$  (read/write) line, but there are several other control lines. The MC6809E is synchronized to the video-display circuit by the two clock inputs, E and Q. These two clocks control internal operation of the microprocessor. Figure 4 shows typical timing diagrams for bus operations.

Three interrupt control lines,  $\overline{\text{NMI}}$ ,  $\overline{\text{FIRQ}}$ , and  $\overline{\text{IRQ}}$ , allow peripherals to request (demand!) support. Each interrupt causes the microprocessor to retrieve a vector from a specific address and use it to begin executing instructions.

The Color Computer uses  $\overline{\text{IRQ}}$  (interrupt request) and  $\overline{\text{FIRQ}}$  (fast interrupt request) to support real-time clock input (driven by the horizontal and vertical sync signal from the VDG) and to auto-start read-only memory cartridges. The  $\overline{\text{NMI}}$  (nonmaskable interrupt) input is reserved for use by the expansion port.

These interrupts function in different manners. The  $\overline{\text{NMI}}$  cannot be disabled or postponed under software control and is useful in real-time interrupt-servicing disk transfers. The other two interrupts are maskable under software control. One is "faster" than the other in that a response to an  $\overline{\text{FIRQ}}$  saves only the condition code register and the program counter on the stack. The other,  $\overline{\text{IRQ}}$ , "stacks" all the registers, as does  $\overline{\text{NMI}}$ . Separate interrupts were used for the PIAs (parallel interface adapters) to provide independent vector addresses for the service routines, thereby minimizing the software overhead.

The interrupt vectors in the Color Computer are mapped to the top of the BASIC ROM by the SAM chip. These vectors point to locations in programmable memory starting at address hexadecimal 100. On reset, the BASIC program stores jump instructions in these locations which point to the interrupt-service routines. Each jump call consists of 3 bytes: the jump extended op code (hexadecimal 7E) and the address of the routine. If a particular interrupt is not being used, all 3 bytes of its jump call would contain 00. See table 3 for a map of the interrupt-service addresses.

To define a jump call, program the 3 bytes with the required jump instruction. For example, if the SWI (software interrupt) service routine is located at hexadecimal 8000, the SWI jump call should be loaded with 7E 80 00. The following BASIC program would load the SWI jump call with this vector:

```
POKE 264,0
POKE 263,128
POKE 262,126
```

This example program defines the last byte of the jump call first, then the middle byte, then the first byte. This approach is required to prevent interrupt service until the jump call is completely defined. If the jump call was defined by starting with the first byte, an interrupt could be vectored to the wrong address. All interrupt-service routines should end with a hexadecimal 3B (Return from Interrupt op code) to restore the Color Computer to the proper state.

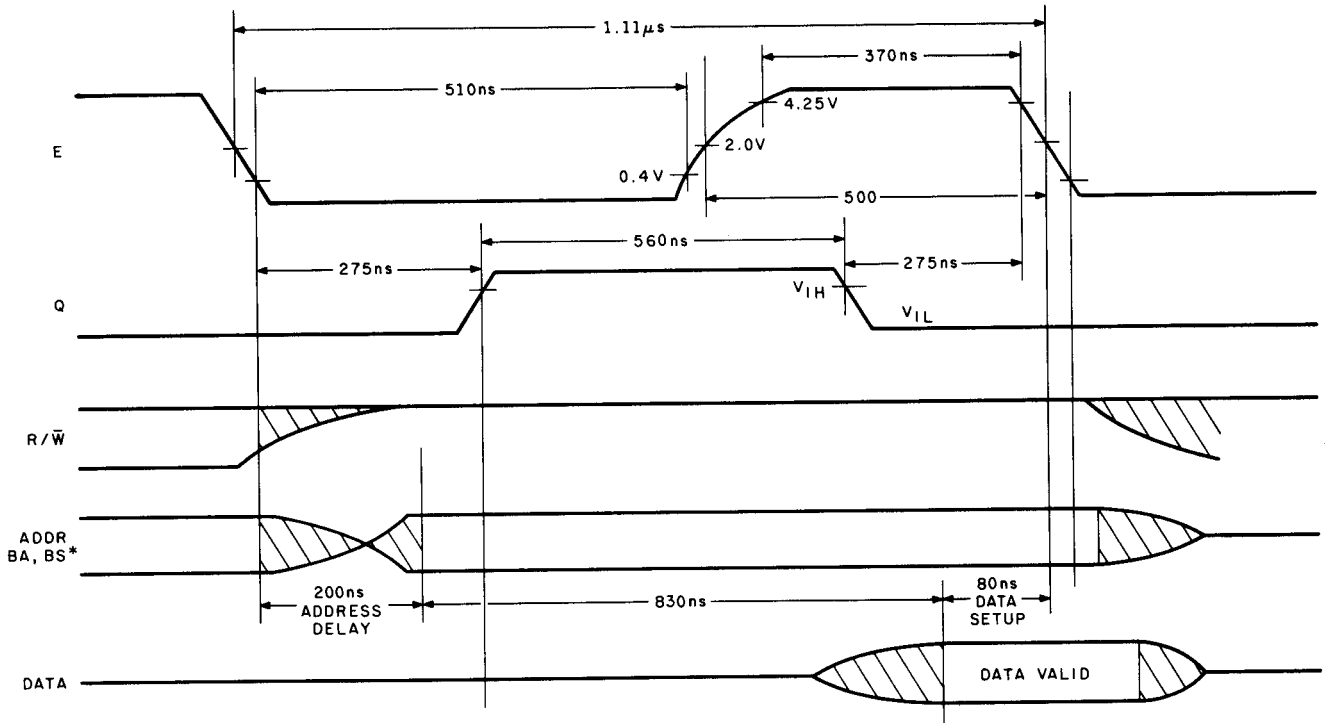
Two other MC6809E input-control signals used by the Color Computer are HALT and RESET. RESET is controlled by the pushbutton switch on the rear right-hand portion of the Color Computer. When the switch is pressed, RESET goes low to initiate a restart routine. The HALT input is connected to the expansion port. When HALT goes low, the MC6809E completes the current instruction, then releases the address, data, and R/W lines to the high-impedance state. This allows another device,

*Text continued on page 110*



(4a)

READ DATA FROM MEMORY OR PERIPHERALS



(4b)

WRITE DATA TO MEMORY OR PERIPHERALS

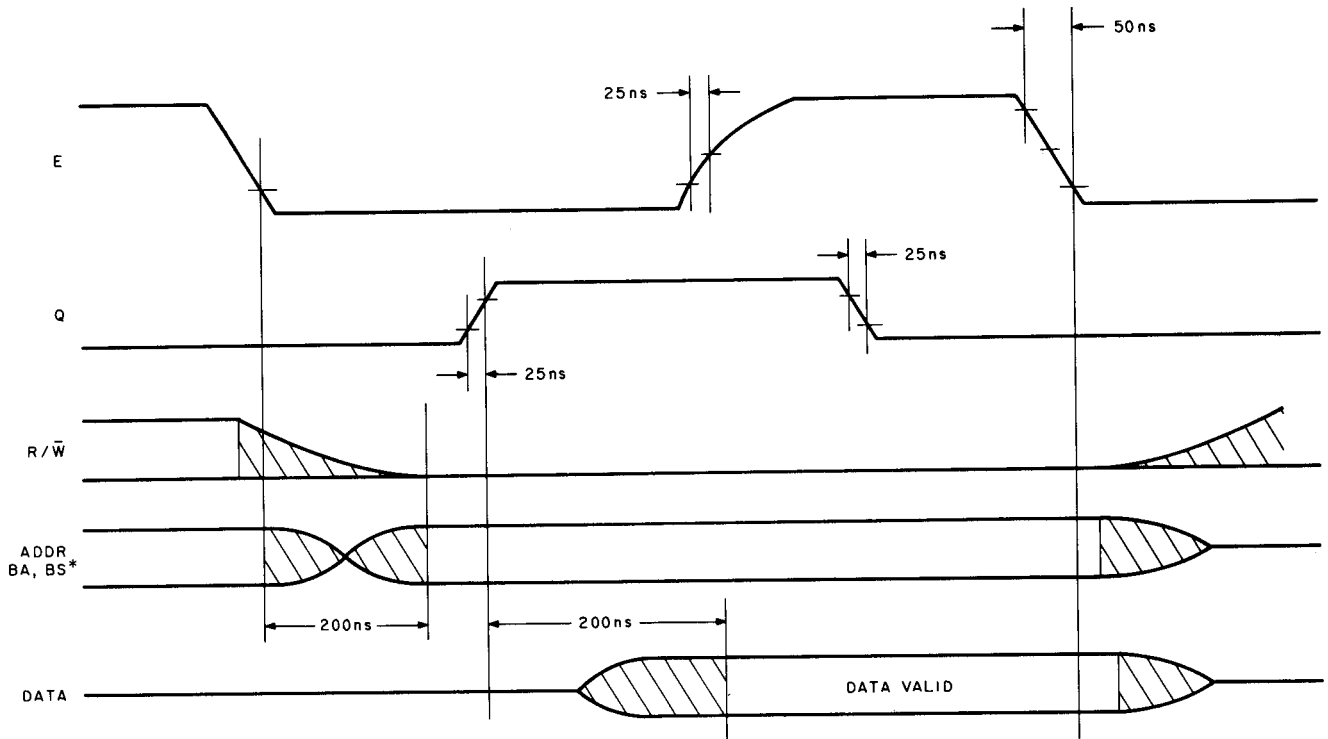


Figure 4: Timing diagrams for 6809 bus operations. As with the 6800, both memory and peripherals are accessed in the same way and share the same address space. The complete instruction cycle for reads (figure 4a) and writes (figure 4b) is the same: approximately 1.1 μs.

Interrupt Source	Address of Interrupt Vector (hexadecimal)	Indirect Routine Call Address (hexadecimal)	Contents of Indirect Routine Call (hexadecimal)	
Reset	FFFE	A027	none	direct call to restart
NMI	FFFC	0109	undefined	not used
SWI	FFFA	0106	undefined	not used
IRQ	FFF8	010C	A9B3	Extended BASIC uses 894C to update real-time clock.
FIRQ	FFF6	010F	A0F6	not used
SWI2	FFF4	0103	undefined	not used
SWI3	FFF2	0100	undefined	not used

**Table 3: Interrupt vectors for Color Computer BASIC.** At the reception of an interrupt, control is transferred to a service routine via a call to an address stored near the top of the 64 K address space (occupied by the BASIC ROM). The address points to a 3-byte jump instruction (loaded into programmable memory when BASIC is initialized); that, in turn, points to an interrupt-handling routine.

Bus Available Signal	Bus Status Signal	Machine State
low	low	Normal (running)
low	high	Synchronize Acknowledge
high	low	Interrupt Acknowledge
high	high	Halt/Bus-Grant Acknowledge

**Table 4: The four possible machine states.** The Bus Available and Bus Status signals can be decoded to detect when the bus is not being used by the processor.

Text continued from page 104:

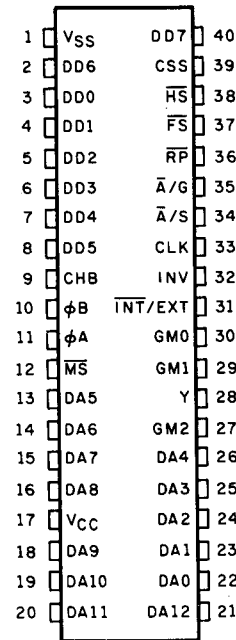
such as a DMA (direct-memory access) controller, to control the bus.

Since the microprocessor is not halted until completion of the current instruction, the external bus controller has to wait 20 bus cycles before driving the bus. This delay is required because the longest execution time for an MC6809E instruction is 20 cycles for a CWAI instruction (see table 2).

This delay could have been minimized if the BA and BS lines were brought out to the expansion port. BA and BS (Bus Available and Bus Status) indicate one of four machine states. These four states and the BA and BS signal combinations are shown in table 4.

Of the four states, the *Halt/Bus-Grant Acknowledge* is the only one pertinent to the design of the Color Computer. The *Normal* state indicates that the microprocessor is executing code. The *Synchronize Acknowledge* state, which allows the processor to be synchronized to an external event, is not required in the Color Computer. Nor is the *Interrupt Acknowledge* state, which indicates that vector fetches are occurring.

Four other MC6809E signals were ignored by the Color Computer's designers: TSC, AVMA, BUSY, and LIC. TSC (Three State Control) is used to put the buses into the high-impedance state for cycle-stealing operations.



**Figure 5: Pin description of Motorola's MC6847 Video Display Generator.** In concert with the Synchronous Address Multiplexer (see figure 6), this device interprets the contents of a block of memory to create a color display (using either an internal character generator or an external one). The output signal is converted to composite video by an MC1372, while a device built of discrete components modulates the signal to radio frequencies for reception on a standard television.

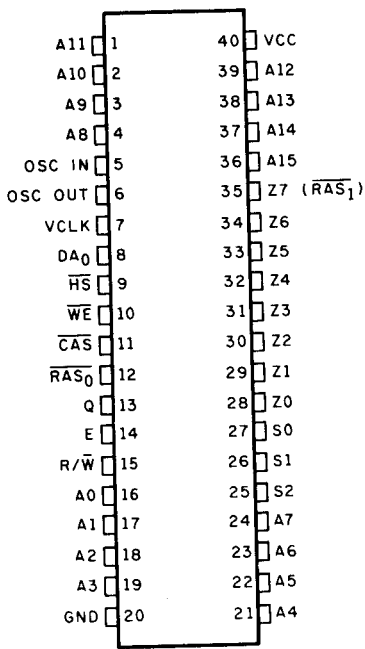
This type of operation is typically used for DMA or dynamic-memory refresh and is not needed in the Color Computer.

AVMA, BUSY, and LIC are intended primarily for use in multiprocessor systems (which the Color Computer is not). AVMA (Advanced Valid Memory Access) is the signal indication that the processor will use the bus during the next cycle. The BUSY output provides the "indivisible" memory indication required for a "test and set" operation (operations of this type are required for efficient multiprocessor support on a common bus). LIC (Last Instruction Cycle) indicates that the first byte of an op code will be latched at the end of the present bus cycle.

The MC6809E was the best choice of the microprocessors available for use when the Color Computer was designed. The external clock inputs allow the microprocessor to be synchronized to the video display to allow interleaved memory accesses. In addition, the power of the MC6809E instruction set allows the efficient graphics drivers supported by the Extended BASIC.

### The Video Display and the Memory Controller

The "Color" in Color Computer comes from the MC6847 Video Display Generator. This device can display information stored in memory using a variety of alphanumeric, semigraphic, and graphic modes. To understand how it works, refer to the signal description shown in figure 5. Normally the address lines DA0 thru DA12 would be connected to a block of programmable memory (usually static devices such as MCM2114s) shared with the microprocessor. Depending on the mode selected, the VDG would read the memory and, taking



**Figure 6:** Pin description of Motorola's MC6883 Synchronous Address Multiplexer. This device provides the complex timing signals required by the microprocessor and for refresh of dynamic memories, as well as multiplexing addresses going into the memories. The various programmable modes of the video-display generator are provided for so that the SAM can help to refresh the video display. (This occurs during the portions of instruction cycles that the processor does not access memory.)

the information off its data lines (DD0 thru DD7), it would format and shift out video information to its companion part (the MC1372 Color Television Modulator) to be transmitted to a TV receiver.

This method of using the part is fine, but it has a few drawbacks. First, there needs to be a way to allow the microprocessor to write its output data to memory. This means that there must be three-state buffers between the microprocessor bus and the VDG bus (and logic to control them). A control pin on the VDG, Memory Select (MS), must be used to put the VDG's address lines in the high-impedance state when the processor accesses the memory.

One side effect of this is that the VDG shift registers will be filled with the data from its data bus as usual, except that the address lines are under the control of the microprocessor, and so the data that gets sent out on the video lines is incorrect. This results in "sparkles" of random color on the TV screen and can be annoying when you are trying to move your TIE fighter out of enemy gunshots!

Second, there is only one block of memory for the VDG to "look" at. In trying to implement computer animation, it would be nice to allow the microprocessor to draw one picture while another is being displayed. Then you would simply swap memory pages and, voilà, the horse moves! You can't do this with the system outlined above unless you resort to fancy hardware.

Of course, both of these problems can be overcome. We have seen it done with an entire board full of TTL (transistor-transistor logic) packages but this is expensive and not for the faint of heart. Fortunately, these problems have a solution in the form of another LSI device from—you guessed it—Motorola. The MC6883 SAM (Synchronous Address Multiplexer) is a 40-pin TTL part that marries the MC6809E and the MC6847 to some dynamic programmable memory.


### SAM, the Synchronous Address Multiplexer

The little jewel called the SAM should really interest computer experimenters. In the first place, it provides the clock signals needed by the microprocessor. The E and Q clocks are derived from the 14.31818 MHz crystal—they are normally 895 kHz—but this can be changed, as we will see. Secondly, the SAM also provides RAS (row-address strobe) and CAS (column-address strobe) signals for dynamic-memory refresh. As anyone who has tried to design a dynamic-memory board can tell you, it isn't easy; and one of the hardest things is deriving RAS and CAS and hiding the refresh cycle from the processor. The SAM does it all and could do it even without a VDG. A complete memory board could be designed around this device even if you didn't want a video display. A signal description of the MC6883 is given in figure 6.

To conserve the number of pins on a dynamic-memory circuit the address is multiplexed in 6-bit pieces (7 bits for 16 K-bit devices). The SAM takes all the microprocessor address lines, multiplexes them to the memory, and controls RAS, CAS, and WE (Write Enable). A typical read cycle is shown in figure 7.

The microprocessor puts out an address to read a location in the dynamic memory. The SAM splits this address into the row address and the column address. First the row address is presented to the memory on the output

**INTRODUCING**



**MULTIPLE APPLICATION PROCESSING SYSTEMS**

## MP/M™ USERS GROUP


Digiac Corporation, a major manufacturer and supplier of automated Educational Training Systems, is proud to announce the formation of **MAPS**, a **National MP/M Users Group** which will provide all MP/M users with a vehicle to exploit MP/M's benefits.

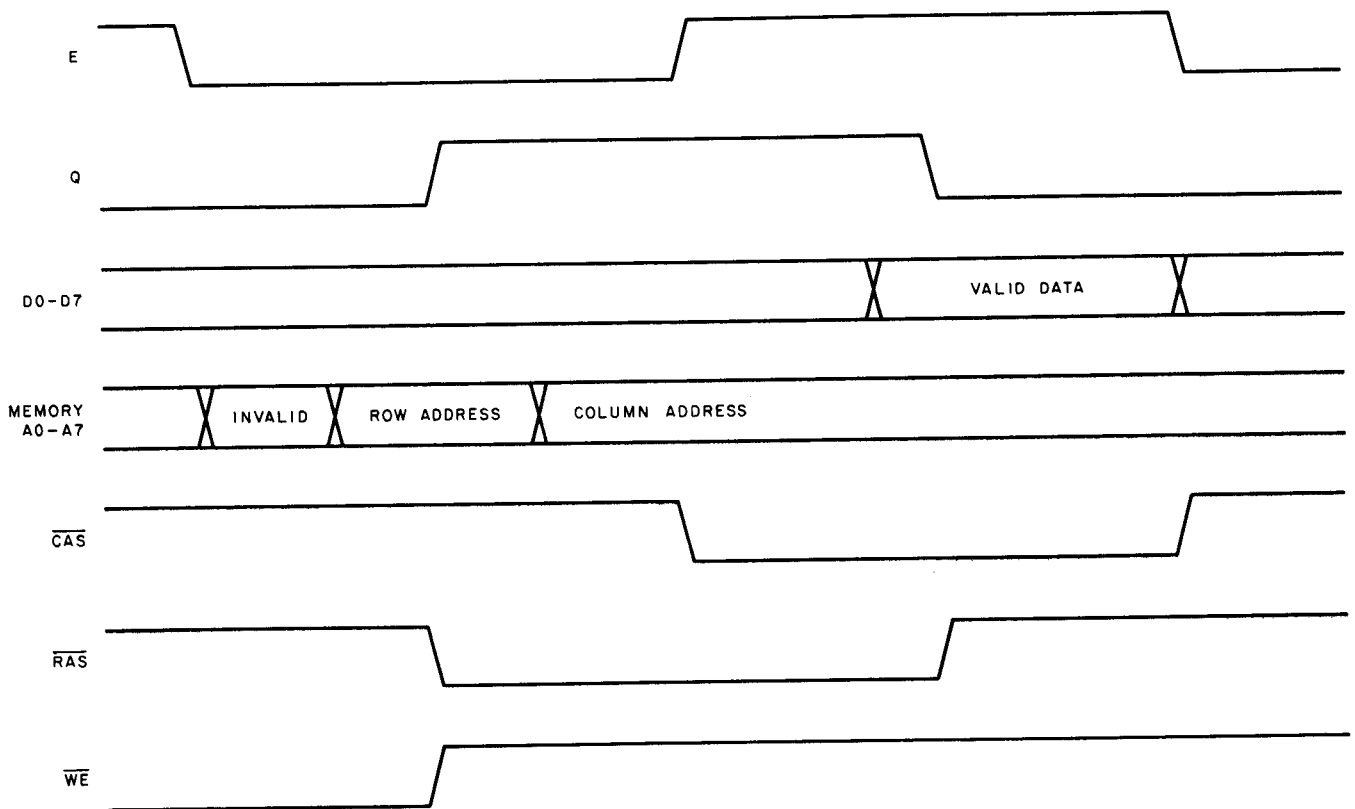
### MP/M SUPPORT PRODUCTS

Digiac is supporting MP/M with a series of exclusive S100 products:

- **Universal MP/M Support Module**
- **MP/M XIOS Configurations for popular Disk Systems**
- **MP/M Multibank Memory Module**
- **CT-80 Multi-Workstation System**

For Additional Information:  
Contact Lorraine Keckeisen  
**MAPS**  
Commercial Products Div.  
**DIGIAC CORPORATION**  
175 Engineers Road  
Smithtown, New York 11787  
Phone (516) 273-8600  
MP/M is a trademark of the Digital Research Corporation





**Figure 7:** Typical read cycle of 4116-type dynamic memory circuits. To reduce the number of pins required, the memory device interprets the address being accessed as two sets of 7 bits that come at different times over the same set of pins. The memory cells of each device are arranged in an array, and the two sets of bits define a row address and a column address. When a set of address bits is valid, either the CAS (column-address strobe) or the RAS (row-address strobe) signal is sent to latch in the respective portion of the address.

lines Z0 thru Z5, and the falling edge of RAS causes the memory to latch this part of the address into internal decoders. The SAM then puts out the column address and drops CAS. This causes the memory device to latch the column address and decodes the location in the internal memory array. The memory's stored data is then put on the data-output lines and through a buffer to the microprocessor.

Now, what about refreshing? Dynamic-memory circuits are made of small capacitor cells and, unless they are refreshed, the charge that represents the stored information will bleed off in a very short time. The memories are constructed such that merely accessing all the row addresses every 2 ms will keep the data alive. Usually this is done with counters that need only count from 0 to 63 (0 to 127 for 16 K-bit devices). The trick is to hide this from the microprocessor.

In the MC6809E, this is possible because the microprocessor needs to access memory only during the time that the E clock is high, so all that must be done is to refresh the memories when E is low. The SAM also does this little chore.

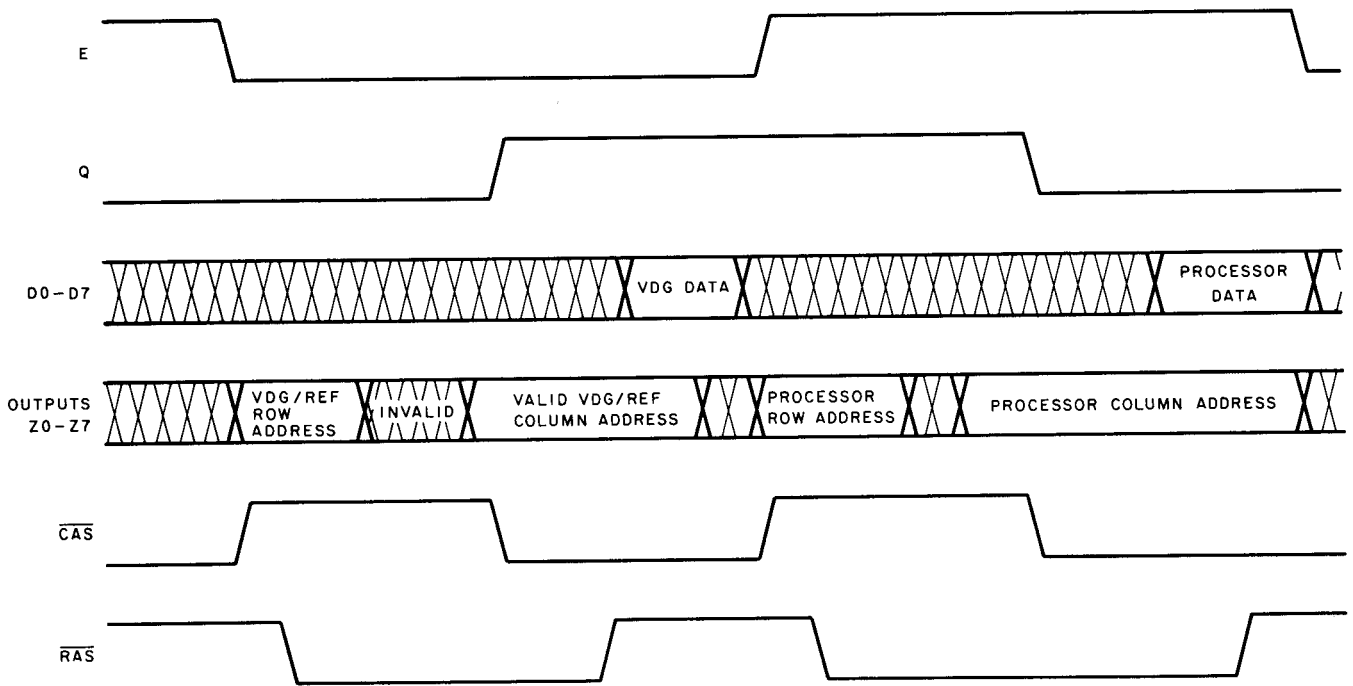
There are two differences between a system that uses 4 K-bit circuits and one that uses 16 K-bit devices. First, the MCM4116 integrated circuits have an extra address line which must be connected to the Z6 output of the SAM. Second, the refresh counters in the SAM must be programmed to put out 128 refresh addresses for the MCM4116s instead of the 64 needed for the MCM4027s.

The SAM has to be programmed to do this. How this is done will be detailed later.

In the Color Computer, the change is simple. There are only two jumpers that need to be switched to select either 4 K-bit or 16 K-bit memory devices. One of these connects the seventh address line, and one is connected to a PIA input line. Upon reset, the BASIC interpreter reads this bit and sets up the SAM for the type of memory indicated. That's all there is to it.

So what does all this have to do with the VDG? Since the VDG needs to be able to read memory to refresh the video screen, the SAM takes care of this, also. The address lines of the VDG are not connected at all in this system. Rather, the SAM is programmed into the same mode as the VDG and duplicates the timing of the VDG's address bus, except that it accesses memory to refresh the VDG during the E low time (so that the VDG accesses are transparent to the microprocessor). Since there is no possibility of a bus fight between the processor and the VDG, there is no need to deny the VDG access to the memory and the screen remains glitchless.

The full timing is shown in figure 8. The SAM usually provides memories with the address needed to access the data for the VDG to output as video. During the active display time (one frame of video) these addresses automatically refresh the memory devices. During the vertical retrace time, the SAM puts out refresh addresses. The microprocessor can access the memory at any time E is high and is therefore not affected.



**Figure 8:** Diagram of a typical dynamic-memory refresh cycle. The SAM provides every dynamic memory with a signal on each row address, as required, to refresh the data contained within.

## The VDG supports one alphanumeric mode, two semigraphic modes, and eight full graphic modes.

### Programming the VDG

The VDG has 5 mode-control pins that determine how the address lines behave and how the data that is obtained from the memory is to be interpreted. In this system, these lines are connected to lines PB3 thru PB7 of PIA2. The data-output register for this device is located at address hexadecimal FF22. The microprocessor can write directly to this port to select the VDG mode. In fact, Extended Color BASIC has a statement, PMODE, to do just this.

The VDG has one alphanumeric mode (using its internal character generator or an external one), two semigraphic modes, and eight full-graphic modes. The modes and the way the mode-control pins must be programmed are shown in table 5.

The alphanumeric mode is the one used by BASIC to print on the screen. The VDG sequentially reads 512 bytes from memory for each TV frame. The data is interpreted as character codes, with the first byte corresponding to the top left corner ("home" position). There are 16 rows of 32 characters for a total of 512 characters on the screen. The character code is given in table 6.

Lowercase characters are displayed as inverted (light characters on a dark background). This is done by tying bit 6 (DD6) of the VDG to the INVERT pin. Because this bit is set in all lowercase numbers, they are inverted.

To support the SET and RESET commands in Radio Shack's Level I BASIC, data line DD7 on the VDG is connected to the alpha/semigraphic pin (A/S). Whenever

this bit is set, the VDG will interpret the data in the manner shown in table 5, under the semigraphic-4 mode. Instead of displaying a character, a colored block that is divided into four smaller blocks is displayed. The code in the byte read from memory determines which pattern of blocks is shown and what color it is. Using the smaller element within the block as a pixel, this gives a grid of 64 by 32 blocks, which are the dimensions of the SET and RESET commands. The other semigraphic mode is similar to this, but each large block is divided into six blocks (instead of four) and has a choice of two sets of four colors, controlled by the CSS (Color Set Select) pin. (Refer to the semigraphic-6 mode in table 5.)

The remaining eight modes are of the bit-mapped graphic type. They require 1, 1.5, 2, 3, or 6 K bytes of memory, depending on the mode. Basically, the data in memory is interpreted as pixels. In the four-color modes (1-C, 2-C, 3-C, and 6-C), each pixel is represented by 2 bits, selecting one of four colors. The set of colors is selectable by the CSS pin. In the two-color modes (1-R, 2-R, 3-R, and 6-R), each bit is mapped one-to-one on the screen. If the bit is set, the pixel is colored, and if it is not set, the pixel is black. The color set can be changed so the pixel can be either buff or green; color sets are controlled by the CSS pin. The resolution of these modes varies from 64 by 64 to 256 by 192 pixels horizontal and vertical respectively.

To use these graphic modes, you simply program the VDG by writing the mode code into the PIA output register, and write to the "screen memory" addresses. The only problem is that the VDG's address lines are not connected to any memory. As mentioned before, the SAM provides the addresses and the VDG interprets the data from the memory, so the SAM must be programmed to be in the same mode as the VDG in order to get a mean-

*Text continued on page 120*

MODES	VDG PINS								COLOR		TV SCREEN	VDG DATA BUS		
	MS	A/G	A/S	INT/EXT	GM2	GM1	GM0	CSS	INV	Character Color	Background	Border	Display Mode	E <sub>1</sub> E <sub>0</sub> A <sub>5</sub> A <sub>4</sub> A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> extra ASCII code
The alphanumeric internal mode uses an internal character generator which contains the following 5 dot by 8 character sets: @ABCDEFGHIJKLMNORSTUVWXYZ _    — SP # \$ % & ' ( ) * _ - / 0 1 2 3 4 5 6 7 8 9 ; < = > ? The 6 bit ASCII code leaves 2 bits free and these may be externally connected to the mode pins (A/G, A/S, INT/EXT, GM2, GM1, GM0, CSS or INV).	1	0	0	0	X	X	X	0	0	Green Black Orange Black	Black Green Black Orange	Black Black	32 characters across 16 characters down	E <sub>1</sub> E <sub>0</sub> A <sub>5</sub> A <sub>4</sub> A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> extra ASCII code
The alphanumeric external mode uses an external character generator as well as a row counter. Thus, custom character fonts are graphic symbol sets with up to 256 different 8 dot by 12 dot "characters" that may be displayed.	1	0	0	0	X	X	X	0	0	Green Black Orange Black	Black Green Black Orange	Black Black	32 characters across 16 characters down	one row of custom characters
The semigraphic-4 mode uses an internal "coarse graphics" generator in which a rectangle (8 dots by 12 dots) is divided into four equal parts. The luminance of each part is determined by a corresponding bit on the VDG data bus. The color of illuminated parts is determined by 3 bits.	1	0	1	0	X	X	X	X	X	Lx C2 C1 C0 X X X 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0	Color Black Green Yellow Blue Red Black Buff Cyan Magenta Orange	Black	64 display elements across 32 display elements down	X C <sub>2</sub> C <sub>1</sub> C <sub>0</sub> L <sub>3</sub> L <sub>2</sub> L <sub>1</sub> L <sub>0</sub>
The semigraphic-6 mode is similar to the semigraphic-4 mode with the following differences: the 8 dot by 12 color rectangle is divided into six equal parts. Color is determined by the 2 remaining bits.	1	0	1	1	X	X	X	1	X	Lx C1 C0 X X 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	Color Black Green Yellow Blue Red Black Buff Cyan Magenta Orange	Black	64 display elements across 48 display elements down	C <sub>1</sub> C <sub>0</sub> L <sub>5</sub> L <sub>4</sub> L <sub>3</sub> L <sub>2</sub> L <sub>1</sub> L <sub>0</sub>
The graphic 1-C mode uses a maximum of 1024 bytes of display memory in which one pair of bits specifies one picture element.	1	1	X	X	0	0	0	0	X	C1 C0 0 0 0 0 1	Color Green Yellow Blue Red Buff Cyan Magenta Orange	Green or Buff	64 display elements across 64 display elements down	C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub>
The graphic 1-R mode uses a maximum of 1024 bytes of display memory in which one bit specifies one picture element.	1	1	X	X	0	0	1	1	0	Lx 0 1 0 1 0 1 0 1 0 1 0 1 0 1	Color Black Green Black Buff	Green or Buff	128 display elements across 64 display elements down	L <sub>7</sub> L <sub>6</sub> L <sub>5</sub> L <sub>4</sub> L <sub>3</sub> L <sub>2</sub> L <sub>1</sub> L <sub>0</sub>
The graphic 2-C mode uses a maximum of 2048 bytes of display memory in which one pair of bits specifies one picture element.	1	1	X	X	0	1	0	0	X	Same color as graphic 1-C mode	Green or Buff	Green or Buff	128 display elements across 64 display elements down	C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub>
The graphic 2-R mode uses a maximum of 1536 bytes of display memory in which one bit specifies one picture element.	1	1	X	X	0	1	1	1	X	Same color as graphic 1-R mode	Green or Buff	Green or Buff	128 display elements across 96 display elements down	L <sub>7</sub> L <sub>6</sub> L <sub>5</sub> L <sub>4</sub> L <sub>3</sub> L <sub>2</sub> L <sub>1</sub> L <sub>0</sub>
The graphic 3-C mode uses a maximum of 3072 bytes of display memory in which one pair of bytes specifies one picture element.	1	1	X	X	1	0	0	0	X	Same color as graphic 1-C mode	Green or Buff	Green or Buff	128 display elements across 96 display elements down	C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub>
The graphic 3-R mode uses a maximum of 3072 bytes of display memory in which one bit specifies one picture element.	1	1	X	X	1	0	1	1	X	Same color as graphic 1-R mode	Green or Buff	Green or Buff	128 display elements across 192 display elements down	L <sub>7</sub> L <sub>6</sub> L <sub>5</sub> L <sub>4</sub> L <sub>3</sub> L <sub>2</sub> L <sub>1</sub> L <sub>0</sub>
The graphic 6-C mode uses a maximum of 6144 bytes of display memory in which one pair of bits specifies one picture element.	1	1	X	X	1	1	0	0	X	Same color as graphic 1-C mode	Green or Buff	Green or Buff	128 display elements across 192 display elements down	C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub> C <sub>1</sub> C <sub>0</sub>
The graphic 6-R mode uses a maximum of 6144 bytes of display memory in which one bit specifies one picture element.	1	1	X	X	1	1	1	1	X	Same color as graphic 1-R mode	Green or Buff	Green or Buff	256 display elements across 192 display elements down	L <sub>7</sub> L <sub>6</sub> L <sub>5</sub> L <sub>4</sub> L <sub>3</sub> L <sub>2</sub> L <sub>1</sub> L <sub>0</sub>

Table 5: Video Display Generator modes.

		Low-Order Hexadecimal Digit															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
High-Order Hexadecimal Digit	0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	1	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	:
	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	6	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	:
7	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	

**Table 6:** Codes for the characters stored in the VDG's internal character generator. The lower section contains inverse-video characters, dark characters on a light background.

Text continued from page 116:  
ingful display.

### Programming the SAM

With a SAM in the system, the memory map is pretty much fixed. The SAM directly decodes the addresses from the processor to access memory, and provides device selects for the rest of the system on the S0 thru S2 pins. These pins are decoded by a 3-to-8 decoder (74LS138) to get the active-low select signals for the rest of the system. Refer to the memory map shown in figure 9.

The reset vector and interrupt vectors at the top of the map are mapped from hexadecimal FFF2 thru FFFF to BFF2 thru BFFF. This allows these vectors to be stored in the 8 K-byte BASIC ROM beginning at address hexa-

decimal A000. The addresses of the two PIAs, the second ROM, and the off-board ROM cartridges are also shown in figure 9.

The block of addresses from hexadecimal FFC0 to FFDF are the locations of the SAM registers. The SAM is programmed and its various options selected by writing to these locations. The data is immaterial since the data bus is not connected to the SAM. Each register bit has two unique locations, an even location and an odd one. Writing to the even location will clear the register bit. Writing to the odd location will set the bit. By encoding the bits, and accessing the appropriate locations, the SAM can be programmed.

The memory map in figure 9 shows the modes and the locations associated with each. S stands for set and C for clear in the diagram. The programmable attributes include:

- VDG mode – mode of address lines during VDG refresh time.
- Display offset – the base address of the memory used by the VDG is specified here. This is the address of the pixel in the upper left-hand corner of the screen in graphic mode. Programmable in 1/2 K pages.
- Memory size – 4 K-bit, 16 K-bit or 64 K-bit dynamic memories or a full map of static memory and I/O.
- Microprocessor clock rate – can be set for 0.8, 1.8 MHz or address-dependent rate.
- Page – allows two 32 K-byte memory pages between hexadecimal 0000 and 7FFF.

The VDG mode bits in the SAM must be programmed to match the mode selected for the VDG on its mode pins. Table 7 shows the correspondence between the SAM and the VDG modes. If the two modes do not agree, interesting results can be obtained. Some of these "mixed" modes include graphics mixed with alphanumeric.

The VDG address offset specifies where the SAM should start the address counters. Figure 10 shows the address sent by the SAM as a function of this offset. This allows the VDG display to be "paged" through memory in 512-byte pages, allowing fast page swapping for animation, etc. On reset, BASIC will set the offset to hexadecimal 400 so all the screen output of the BASIC interpreter is at locations hexadecimal 400 thru 5FF. Try POKEing to these locations to use the alphanumeric and semigraphic modes.

The Extended BASIC supports the higher-resolution

Text continued on page 124

**K.25**  
**BISTYNG**  
**SDIOWSMA**  
Software for your Microcomputer

IBM 2770  
IBM 2780  
IBM 3270  
IBM 3741  
IBM 3780  
IBM 3776

8080 Z80 8086 Z8000

WINTERHALTER & ASSOCIATES, INC.  
SPECIALISTS IN DATA COMMUNICATIONS  
3825 ZEEB ROAD  
DEXTER, MICHIGAN 48130  
313-426-3029 or  
313-665-5582

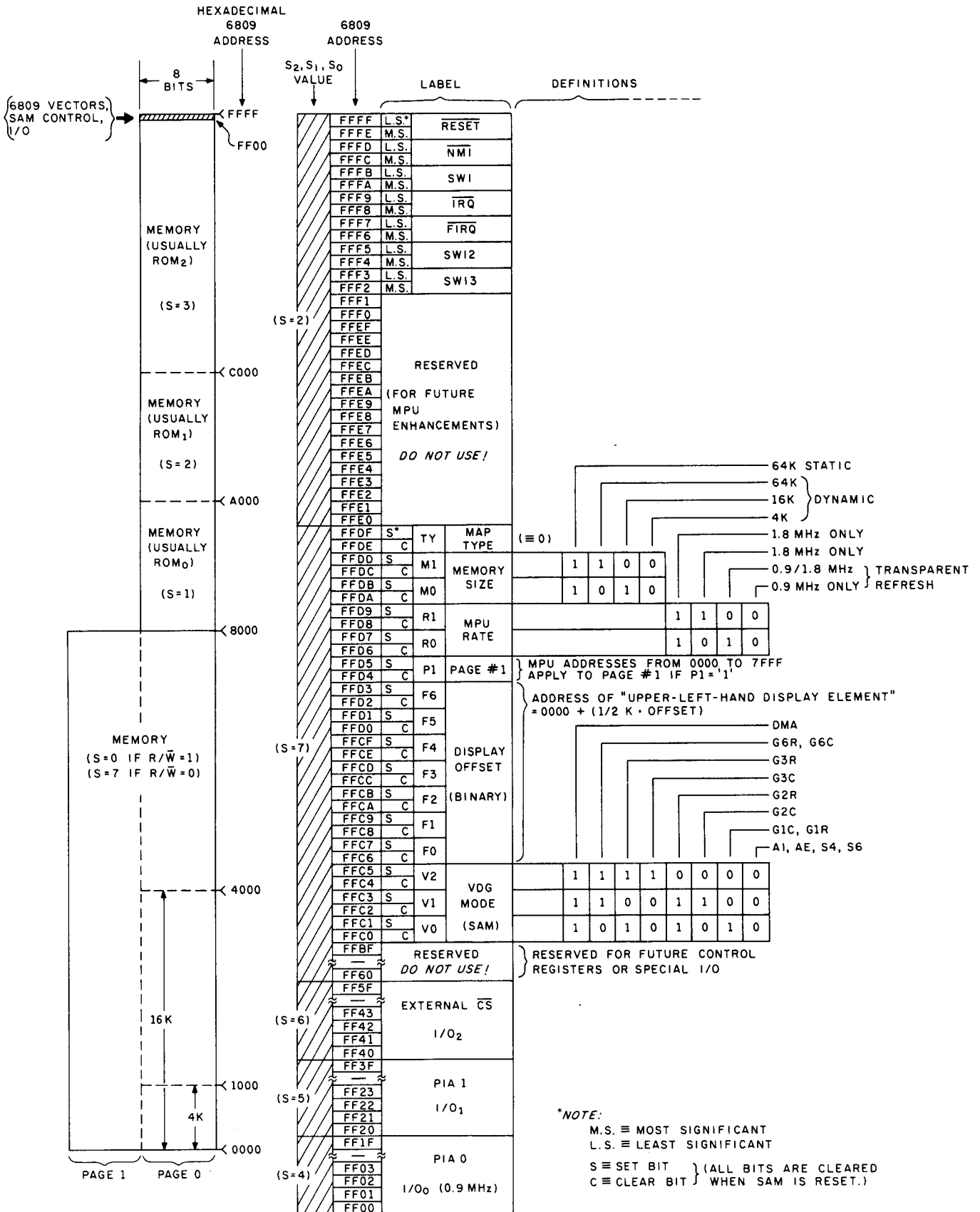


Figure 9: Memory map of the Color Computer address space. The general division of addresses is provided at the left, while the SAM programming registers and the processor-interrupt vectors are expanded at the right.



Mode	Video Display Generator Signals					Synchronous Address Multiplexer Signals		
	G/ $\bar{A}$	GM2	GM1	GM0 EXT/I	CSS	V <sub>2</sub>	V <sub>1</sub>	V <sub>0</sub>
Internal alphanumeric	0	X	X	0	x	0	0	0
External alphanumeric	0	X	X	1	X	0	0	0
Semigraphic-4	0	X	X	0	X	0	0	0
Semigraphic-6	0	X	X	1	X	0	0	0
Full graphic 1-C	1	0	0	0	X	0	0	1
Full graphic 1-R	1	0	0	1	X	0	0	1
Full graphic 2-C	1	0	1	0	X	0	1	0
Full graphic 2-R	1	0	1	1	X	0	1	1
Full graphic 3-C	1	1	0	0	X	1	0	0
Full graphic 3-R	1	1	0	1	X	1	0	1
Full graphic 6-C	1	1	1	0	X	1	1	0
Full graphic 6-R	1	1	1	1	X	1	1	0
Direct memory access	X	X	X	X	X	1	1	1

Table 7: Mode correspondence between the SAM and the VDG.

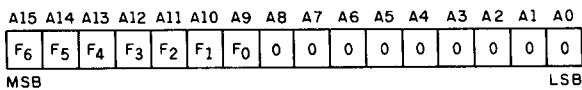


Figure 10: Mapping of the video-display refresh address. The SAM uses a 7-bit offset to determine the start of video-display memory. This allows the use of 512-byte "pages" for display refreshing, making it possible to page through memory to create fast animation effects, etc.

Text continued from page 120:

graphics and can also allocate more memory for multiple pages, up to eight. It provides graphic operations, such as LINE, DRAW, and CIRCLE, that are fast enough to allow the programming of real-time games using the joysticks as controllers.

Memory type is self-explanatory. The SAM must be programmed for the type of memory devices used in the system to produce the correct timing signals. If 16 K-bit circuits (MCM4116 or the equivalent) are used, pin 35 can be used for RAS1. This is needed to select a second bank of devices to provide 32 K bytes of memory. One way to do this on the Color Computer is to piggyback a second set of eight MCM4116s on top of the existing integrated circuits, paralleling all the pins except for the RAS pin. When this is jumpered to pin 35 on the SAM, the system then has 32 K bytes of user-programmable memory.

The microprocessor clock rate is also programmable. There are three modes, as shown in figure 9. In mode 0, the clock rate is fixed at one-sixteenth the crystal frequency. In this case, that is 895 kHz. Mode 2 gives a fixed rate of one-eighth the crystal frequency, or 1.8 MHz. This can be used with an MC68B09E, a 2 MHz version of the microprocessor. However, there are no memory or VDG addresses output in this mode, so don't use it.

Mode 1 is the most interesting. It gives a dual-rate clock of 895 kHz or 1.8 MHz depending on the address used in the bus cycle. When the processor accesses addresses from hexadecimal 0000 to 7FFF and FF00 to FF1F, the lower rate is used, allowing for slower memory and peripherals. When all other addresses are accessed, the processor runs at 1.8 MHz. Using fast ROMs will almost double the speed of the system because a majority of the microprocessor's memory references are to fetch op

codes. If you want to try this, execute the following BASIC statement:

```
POKE 65495,0
```

This will set bit R0 of the microprocessor rate register at location hexadecimal FFD7 and put the SAM into the dual-rate mode. If your microprocessor can run at the higher speed (a pretty good bet), you will see the changing-color cursor flashing about twice as fast as normal. Your BASIC programs will now run about twice as fast, too. There is one problem, though — don't try to use the SOUND, CLOAD, or CSAVE statements in this mode. The PIA used by these statements is at location hexadecimal FF20 and it will probably not run at the higher speed.

The other two registers do not apply to the Color Computer. The Map Type bit chooses a mixed programmable/read-only type of system such as the Color Computer or a fully programmable system such as a disk-based one. The Page bit allows two 32 K-byte pages of memory to be accessed between locations hexadecimal 0000 and 7FFF. This can't be done on this system.

### Keyboard Scanning

The keyboard is configured as an 8 by 7 matrix of keys. The Color Computer uses a software routine to encode the keyboard in a manner similar to that of the TRS-80 Model I. This is done by shifting a 0 through the B port of PIA IC8. The B port drives the 8 rows of the keyboard; the 7 columns are connected to the A port of IC8. The A port has internal pull-up resistors that provide a logic 1 level unless a key is depressed. When the shifted 0 occurs on the row of the closed key contact, the low level is passed to port A. By repeating the scanning procedure several times, debounced inputs are recognized.

If you need to monitor the keyboard during a program, a function (INKEY\$) is provided. The BASIC statement

```
A$ = INKEY$
```

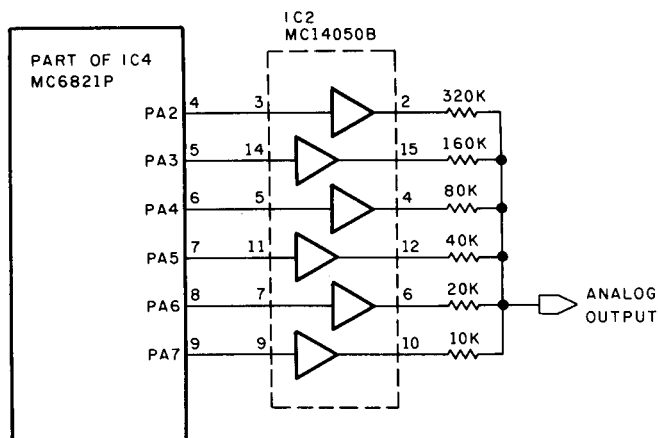
will return a character if a key is closed when the function is called. An example use of this function would be to monitor the keyboard during a "Tank" game for direction

keys and a "Fire" key. This would allow you to play a "Tank" game without having a set of joysticks.

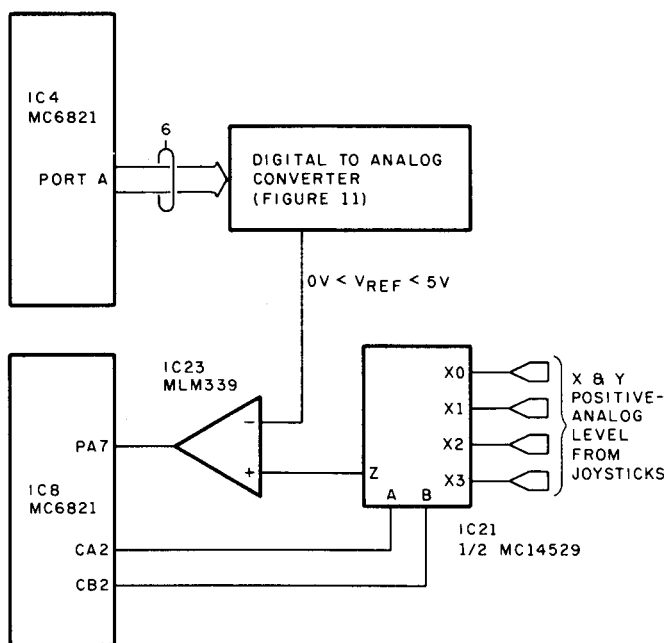
## Digital-to-Analog Converter

The D/A (digital-to-analog) converter allows the Color Computer to send analog waveforms. These signals are used for the cassette output, sound to the video modulator, and as a reference signal for A/D (analog-to-digital) conversion.

Six of the eight port A lines are configured as outputs and buffered to drive a resistive adder network for analog signal generation, as shown in figure 11. The resultant analog signal ranges from 0 V to +5 V in 78 mV steps.



**Figure 11:** Schematic diagram of the Color Computer's digital-to-analog converter. In a rather simple scheme, the output lines of a parallel port drive a resistive adding network to provide conversion. The resulting analog signals are used for recording on a cassette, providing the video modulator with sound, and also as part of the analog-to-digital converter.



**Figure 12:** Diagram for the analog-to-digital converter circuit. Also used as the joystick interface, this circuit applies the successive-approximation method (see figure 13) to change analog signals to digital form.

This type of converter is accurate to  $\pm \frac{1}{2}$  the least significant bit, or in this case  $\pm 39$  mV.

## Cassette Port

The Color Computer has a cassette port which connects to a low-cost recorder. Motor-control capability is included that allows the cassette recorder to be started or stopped as required. The motor can be turned on and off with the statements MOTOR ON and MOTOR OFF. This allows the user to fast-forward or rewind tapes without having to unplug connections to the Color Computer.

Data is output to the recorder from the D/A converter. If an oscilloscope is connected to the data-output line, pin 5 of the cassette jack, an 800 mV 1500 bps signal will be seen.

When data is loaded from the cassette recorder, the playback signal can be routed to the modulator sound input in a manner that allows you to monitor the cassette signal via the speaker of a television set. This is done with the AUDIO ON and AUDIO OFF statements.

The cassette data-output can be used for an analog output level because the D/A converter can be controlled by a user program. The motor-control relay can be used to control loads up to 6 V DC at 500 mA.

## Joystick Interface

Two joystick ports are provided which allow full  $x,y$  directional control. Each joystick has a pushbutton for use with games (eg: paddle control for the Pinball game). Each joystick consists of two potentiometers, each connected across +5 V and ground. The wiper of each potentiometer is connected to the input of an analog multiplexer controlled by PIA IC8. The voltage level from each of the four potentiometers is routed to the A/D converter to get a digital value for the position. This value will range between 0 and decimal 63. The JOYSTK(j) function returns the digital value of the joystick position.

Analog voltage levels from the joysticks are digitized using a successive-approximation technique. This is one of the more popular methods of A/D conversion. The 6-bit D/A converter is used in a feedback loop to generate a known analog signal to which the unknown analog joystick input is compared. This technique is not as fast as a flash converter, nor is it as slow as a binary counter.

Figure 12 shows the block diagram for the successive-approximation converter circuit. Figure 13 shows a flowchart for this approach. The D/A converter inputs are controlled by the microprocessor to form a successive-approximation register. The analog output is compared to the analog joystick input by the MLM339 comparator whose output is monitored by the MC6809E.

At the start of a conversion the MSB (most significant bit) of the D/A converter is turned on by the microprocessor, producing an output equal to half the full-scale value. This output is compared to the analog input and if it is greater than the joystick voltage, the microprocessor turns the MSB off. However, if the D/A output is less than the joystick voltage, the MSB remains on.

Following the trial of the MSB, the next most significant bit is turned on and again the comparison is made

between the converter's output and the joystick voltage. The same criteria apply and this bit is either kept on or turned off. This procedure of testing each bit continues four more times until the 6 bits of the D/A converter have been set to the proper level.

Once the conversion is complete the microprocessor reads the joystick output by reading port A of PIA IC4. The internal structure of port A allows a read of the port to sample the output logic levels. Now the Color Computer has the digital value for the joystick voltage. The time necessary to do this conversion is constant and does not vary with the analog voltage level.

Note that the Color Computer has an on-board A/D converter that accepts a signal between +5 V and ground and can digitize it with less than a 40 mV error. This means you can use the appropriate joystick inputs to monitor various analog voltages. The switch inputs are connected to the PIA (the left switch to IC8 pin 3, PA1; and the right switch to IC8 pin 2, PA0). You can write a program to monitor these bits for use with external devices. Figure 14 shows the connectors for the joysticks (which are not shown in the TRS-80 Color Computer Operation Manual).

### RS-232 Interface

An RS-232 interface is also provided. This allows you to connect all manner of devices to the Color Computer. The standard RS-232 Transmit Data, Receive Data, and Carrier Detect signals are provided. This is the fundamental signal subset used by most devices. Tandy sells an off-the-shelf line of printers and a modem that are readily

usable.

### Expansion Port

The expansion port provides the capability to interface almost anything to the Color Computer. Table 8 lists the pins and their functions. Note that the entire address bus is brought out. There is also a decode-defeat pin which disables the 74LS138 that decodes ROMs and peripherals. This allows the expansion port to redefine the memory map. For instance, a flip-flop could be toggled to remove the BASIC and Extended BASIC ROMs from the memory map and replace them with programmable memory. A disk-controller board could also contain 48 K bytes of memory to fill the system from address hexadecimal 0000 to FF60.

The Vector Graphic company makes a wire-wrap prototype board (part number 4609) that fits the expansion connector of the Color Computer. This allows you to build your own peripheral boards. We are working on an interface to the General Instrument "Cricket" sound generator. The output from this circuit can be routed to the video modulator through a pin on the expansion connector. If you want, you can also build your own game

## Selectric® Interface System

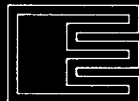
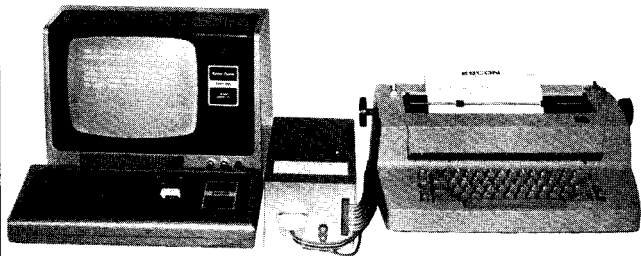
**E**ASILY interfaced to any IBM Selectric I, II, or III.

**S**TOP spinning your wheels. Letter quality at an affordable price.

**C**ONNECTS via Parallel or RS-232, accommodates varied handshaking.

**O**NLY \$575 to \$599. Dealer inquiries invited.

**N**EW design provides added features.



ESCON Products, Inc.  
12919 Alcosta Blvd.  
San Ramon, Ca., 94583  
(415) 820-1256

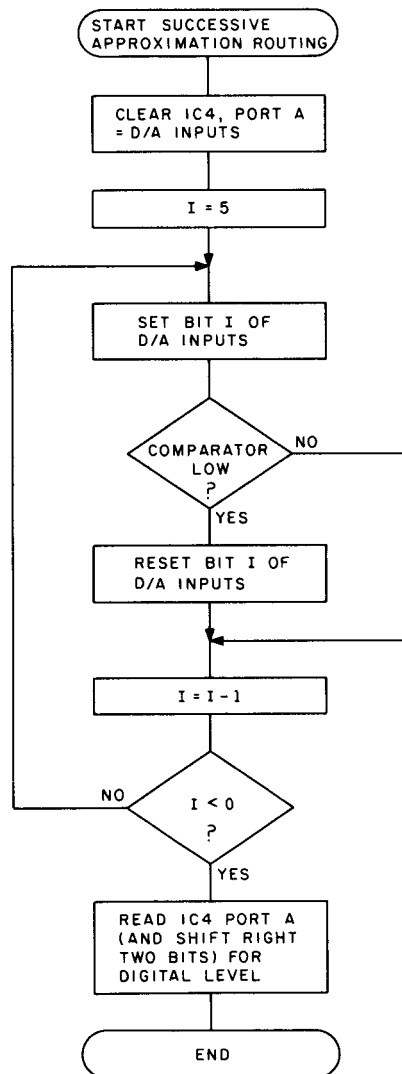
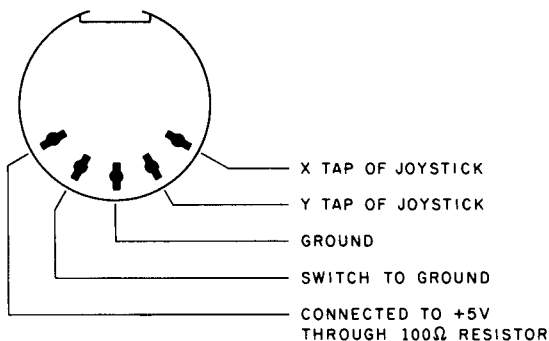


Figure 13: Flowchart of the successive-approximation algorithm used by the Color Computer.



**Figure 14:** Pin designations of the Color Computer joystick connectors. The connectors will mate with a standard 5-pin DIN plug, and any signal within the A/D converter's range may be monitored under program control.

Expansion Port Pin		Description	
pin	function	pin	function
1	- 12 V	2	+ 12 V
3	HALT	4	Nonmaskable Interrupt
5	RESET	6	E
7	Q	8	CB1 of IC4
9	+ 5 V	10	D0
11	D1	12	D2
13	D3	14	D4
15	D5	16	D6
17	D7	18	R/W
19	A0	20	A1
21	A2	22	A3
23	A4	24	A5
25	A6	26	A7
27	A8	28	A9
29	A10	30	A11
31	A12	32	C000 thru FEFF
33	Ground	34	Ground
35	Analog In	36	FF40 thru FF5F, CS
37	A13	38	A14
39	A15	40	Decode Defeat

**Table 8:** Signals available at the expansion port.

cartridges. If you want them to auto-start like the Tandy cartridges, connect pins 7 and 8 together. This runs the Q clock into the CB1 input of PIA IC4, causing an FIRQ interrupt. The FIRQ interrupt-service routine jumps to hexadecimal C000 and starts execution. There is also a device select on pin 32 that is decoded from hexadecimal C000 to FEFF.

## Summary

We have tried to completely describe the architecture of the Color Computer and deduce the reasoning behind the design trade-offs. Tandy certainly is to be complimented on the amount of "bang for the buck"—every part is fully used and several innovative design ideas are evident. We believe that the Color Computer has the capability to surpass the Model I in sales.

In a later article we will take a detailed look at the Extended BASIC and discuss its capabilities. We are currently implementing several popular video games in BASIC. Once the algorithms are proven, we plan to convert them to machine language to increase the speed, although with the power of the Extended BASIC we may not have to. ■