Designed especially for beginners

DREAM 6800

Talks directly to your TV and is programmed in a high-level language!

Are you one of the many people who have been turned off microprocessors and computers by all the complexity and never-ending jargon? Well, here is your chance to really start learning about the subject. This simple and easy to build computer costs around the $100 mark, yet talks directly to your TV without the need for a costly video terminal.

One of the other big features is the built-in cassette interface which means you can store your programs on any cassette recorder. And there is a whole raft of sample programs to get you started. All you have to do is punch them in via the hexadecimal keyboard. In no time you'll have a whole library of your own programs, easily accessible on cassettes.

So start reading now. We've even provided a comprehensive glossary to help you wade through all the jargon which is inevitable in this new and exciting field. The title of the computer is itself a bit of jargon: DREAM 6800, which stands for "Domestic Recreational and Educational Adaptive Microcomputer . . ."

Now we'll let the designer, Michael Bauer, of the Division of Computing and Mathematics at Deakin University, tell his story . . .

Surprising as it may seem, there are very few so-called "hobby computers" which inexpensively satisfy the needs of recreational home computing. The choice is between an "evaluation kit" (eg, 6800-D2, KIM-1, Mini-scamp etc.) or a BASIC system with CRT terminal, 8k memory, etc. The latter will set you back a few hundred dollars, while the evaluation kit doesn't give you enough capabilities. And besides, a hobby is supposed to be pleasurable, not give you headaches. There are much easier, less expensive ways to produce a headache, other than sitting up all night for days on end, hand assembling a ridiculous machine-code program to play "Lunar Lander" (with a 7-segment LED readout), or trying to write an animated video game in a high-level language like BASIC which wasn't invented for that purpose in the first place for a terminal that only displays alphanumerics.

Here's what the "DREAM 6800" home video computer has to offer:—

1. Lower cost: the parts should come to about $100.

2. A more useful display: Chunky graphics output to your colour or B&W TV giving a 64 x 32 dot matrix display.

3. Better software: As well as the usual operating-system or monitor (used for memory examine and deposit, tape load and dump, go to user program, etc), CHIPOS incorporates a high-level language interpreter, CHIP-8, which was specifically invented for video games, graphic displays, simulations, etc. Further, CHIPOS supports machine-language programs as well, for those applications where CHIP-8 is inadequate.

4. Wider appeal: People not into electronics or computing will also find the DREAM 6800 fascinating. Lots of TV games and other programs have already been written in CHIP-8, so you'll be able to impress your "non-believer" friends right away. And you won't hear the old: "Oh yeah, but what does it do?" and similar phrases. This is a fun computer!

5. There are hundreds of applications: TV games; advertising displays; teaching young children elementary arithmetic; practising morse code; timing events in the kitchen; hex/binary (variable base) calculator; metric conversions; bar charts; simulations (like LIFE); data communications experiments; etc. Educational institutions will find it highly motivational for introductory machine-level programming courses. It's also a serious computer!

HARDWARE SPECIFICATION

- ☆ *Processor: Motorola M6800.*
- ☆ *Clock: M6875 with 4.00MHz crystal.*
- ☆ *RAM (On-card): 1K x 8 (2 x 2114) Off-card expansion to 32K.*
- ☆ *ROM (CHIPOS) 1K x 8 (2708).*
- ☆ *Display: 64 x 32 dot matrix; each dot is 4 TV lines square. Uses 256 bytes of RAM at loc. 0100 for refresh by DMA.*
  Video output: 1Vp-p @ 75 ohm.
- ☆ *Input/Output: One M6821 PIA controls:*
  — Hex keypad (16 keys in 4 x 4 matrix) plus 2 extra keys, Function & Reset.
  — Tape I/O: 300 Baud; 2400/1200Hz FSK; Out: 0.5Vpp; In: 300mV — 3Vpp.
  — RTC timer interrupt: 50Hz (frame sync.).
  — Audio bleeper: 2400/1200Hz (8 ohm spkr).
  — Display/DMA enable-disable line.
- ☆ *Add extra PIAs, ACIAs, etc, without any additional logic.*
- ☆ *Power requirements (worst case): +5V (1A), —5V (100mA), +12V (100mA).*

NOTE

Power supply, keypad and TV RF modulator are off-card extras.

Right about now the sceptics will be saying: "But there's only 1K of RAM and the video refresh buffer's got to be in there somewhere, and a scratchpad, and a stack or two . . . good grief! there won't be enough left for a program! In

Sungravure staffer Adriane Hill puts our prototype DREAM 6800 through its paces with a random number display.

fact, there are 640 bytes free. That's either a damned long machine-code program to hand-assemble, or a 320 statement CHIP-8 program. Most users will find this more than adequate. CHIP-8 is a lot more memory-efficient than BASIC, assuming the application is graphics oriented and does not require any heavy number-crunching, or text manipulations.

For experimenters, there are a few spare I/O lines on the PIA and the system bus is terminated on two 16-pin sockets allowing memory and I/O expansion.

Most hobby computer designers take advantage of the increase in sophistication and lower cost of hardware to produce a more powerful system for the same price as earlier designs. The DREAM-6800 philosophy is to retain the meek processor power and small memory size of past generations, but at a much reduced cost, and to more effectively utilise the available memory. This is not to imply that the '6800 lacks power; it is a superlative 8-bit MPU in every respect.

SOFTWARE DESCRIPTION — CHIPOS

Q: "When is a computer not a computer?"

A: "When there's no software to go with it."

CHIPOS packs about as much into 1024 bytes as is possible. The monitor and interpreter share many subroutines, such as the keypad encoder and display routines. There are four commands, selected by the function key (FN) followed by a hex digit, viz:—

[FN] [0] for memory modify ("memod"): allows RAM contents to be examined or changed. First, a 4-digit hex address is keyed in and appears in the display readout. The [FN] key is used to step through memory, each byte being displayed one by one. To write data into RAM, a 2-digit number (one byte) is entered, and the address is incremented automatically.

[FN] [1] for tape load. First, "memod" (above) is used to enter the beginning and ending locations of the block to be loaded (or dumped), at 0002 and so on.

[FN] [2] for tape dump.

[FN] [3] for "GO". First, a 4-digit starting address is entered; eg C000 to run a CHIP-8 program. (This address is remembered for subsequent runs, unless memod is used in the meantime.) At any time, the reset key [RST] may be used to regain monitor control

That's it; you've learned just about all there is to know in order to enter, verify, save, load and run any program. Details of how to write and debug your own CHIP-8 programs will be given later.

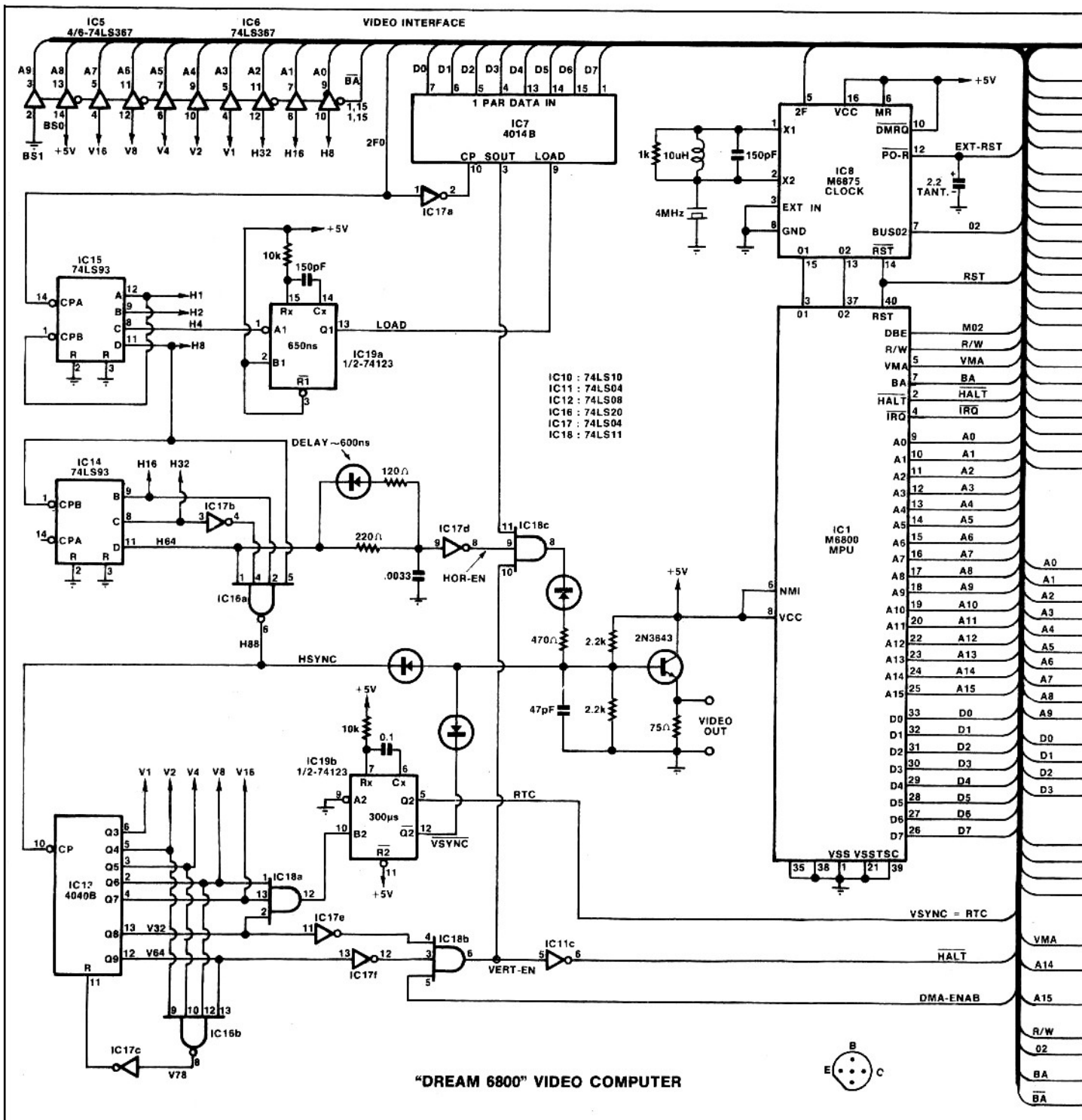
For advanced users, CHIPOS has been written with flexibility in mind. Calls to over 17 useful CHIPOS subroutines can be made from a machine-code program; eg, erase the screen, fill the screen, get random byte, display hex digit, convert byte to decimal, show a user-defined symbol up to 8x16 dots, set display coordinates, turn screen on/off, input a key-code, make a bleep, delay 3.33 milliseconds, test keypad status, input byte from serial I/O line (tape port), output same, wait for frame sync interrupt, return to monitor, etc. A user can load his own machine-language debugging utility (DREAMBUG!) in RAM at 0080. A software interrupt causes a jump to this address. A typical debug routine would then display all register contents. Further, the IRQ vector is in RAM, so that a user can supply his own interrupt service routine. This feature is handy if you want to program your DREAM-6800 as an "intelligent alarm clock", or if you want the keypad to be interrupt driven.

Don't worry if the last paragraph made little sense to you, because most of those subroutines, and many additional ones are far easier to utilise via the CHIP-8 interpreter, which uses a two-byte "macro" instruction to perform any given task. Add a few more instructions to do arithmetic and logic, perform conditional branching, and do loads/stores on the variables, and presto — you've got a high-level computer language.

CHIP-8 LANGUAGE SUMMARY

CHIP-8 was originally developed by Joe Weisbecker at RCA Labs (USA), primarily to allow users of low-cost microcomputers to write their own video game programs without the tedium of hand assembling machine language programs.

CHIP-8 does not use an expensive video terminal like BASIC, but rather a low-cost interface to the processor



Here is the complete circuit of the DREAM 6800 minus the power supply.

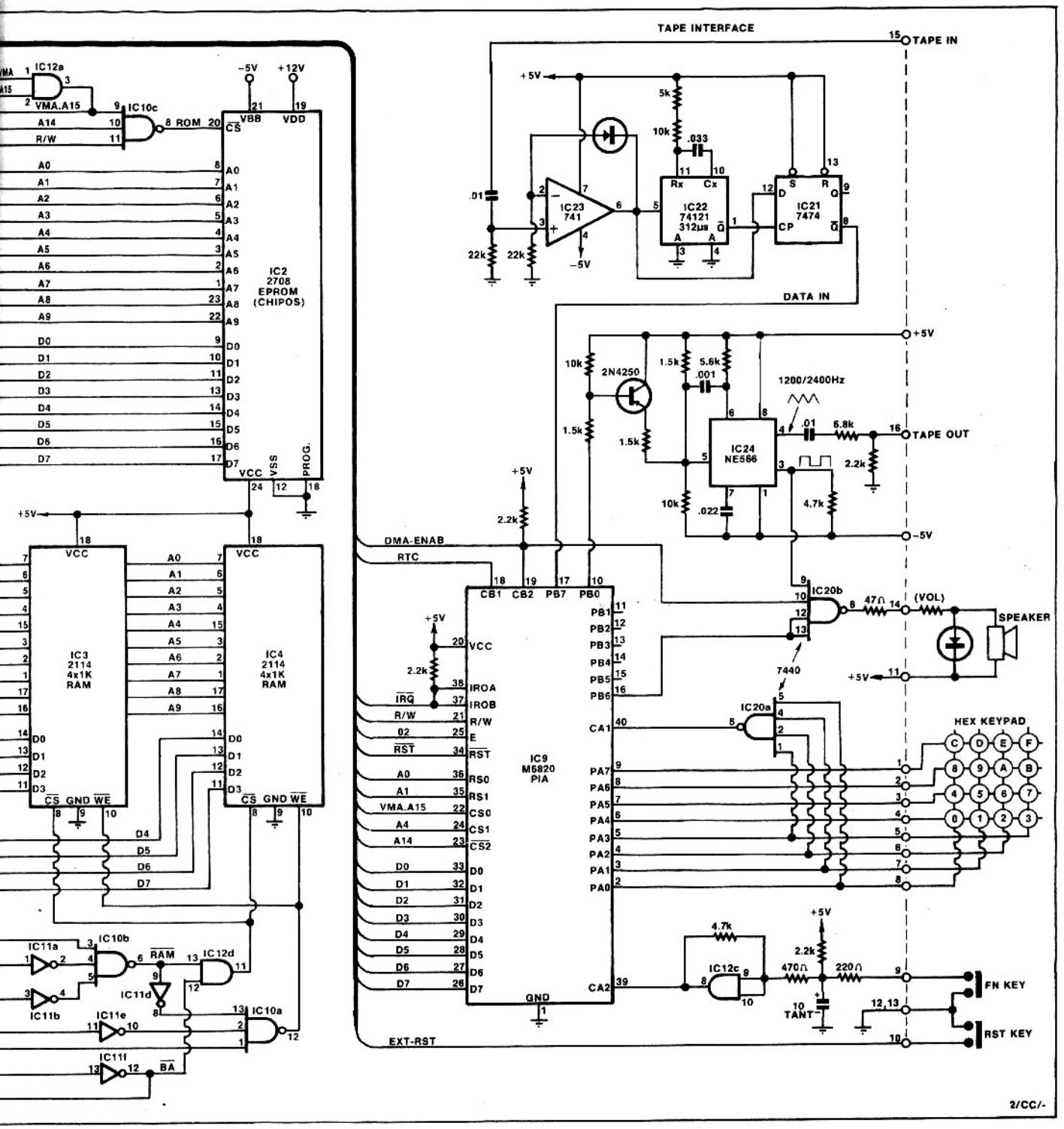
which produces a matrix of dots (64x32) on a TV screen, each dot being a bit in the system's RAM. Nor is a full typewriter-style keyboard required, since all instructions are coded in hexadecimal, and entered via a hex keypad.

CHIP-8 produces quite compact programs for video games, graphics, advertising displays, animations, simulations, educational tests, drills, and so on. A fascinating video kaleidoscope program requires only 60 instructions. An animated UFO/missile intercept program, with on-screen decimal scoring, requires only 104. A tank battle game easily fits into the minimum (1K) system.

CHIP-8 is an interpreter, ROM-resident with the operating system. The user's program instructions reside in RAM, starting at 0200. Each statement is stored as 4 hex digits (2 bytes) which are designed to be easily encoded by hand. There are 33 instructions in the set, including No Operation (0000) and STOP (F000).

The language provides 16 one-byte variables, V0 thru VF, which can be manipulated with a variety of arithmetic/logic and conditional branching instructions. A 12-bit pointer (I) indexes memory locations for load/store and display instructions. This allows multiple sets of variables or array processing. CHIP-8 is limited to the first 4K of memory, because of the 3-digit (12-bit) memory operands.

The SHOW instruction takes N bytes from the location at I, and writes them vertically into the display refresh buffer, using the values of any two



variables as the x,y coordinates of the symbol. If an attempt is made to write dots on top of existing dots, then the overlapping area is erased and variable VF is set to 1.

A hex numeric digit (contents of any variable, LSD) can be displayed by preceding a SHOW instruction (DXY5) with an I=DSP, VZ (FZ29). The MI=DEQ,VZ instruction stores the 3-digit (unsigned) decimal equivalent of variable VZ in memory at I.

Additional instructions let you make a bleep of variable duration in the speaker, preset or test a timer (a variable which is decremented every 20 milliseconds), input a digit from the keypad, or simply test to see if a given key is being pressed.

Subroutine nesting to 12 levels, and calls to machine-language programs are catered for. CHIP-8 subroutines may call machine-code subroutines.

THEORY OF OPERATION

The design is centred on the video interface, which shares both RAM and time with the processor. The picture is active for 128 out of a total of 312 TV lines, which is about 40% of the frame. During this time, the MPU is halted and the video display generator (VDG) has sole access to the memory, which is held in read mode. Therefore, no intermediate buffer storage is required, such as a recirculating register, for display refresh.

The compromise is between picture size and processor throughput. The chosen format of 64 x 32 dots has several advantages apart from VIP compatibility. Some of the "coincidences" are incredible; eg:—

1. The clock frequency worked out to be 1.998 MHz (for 50.00Hz field freq.), and the M6875 has an auxiliary 2.00MHz output.
2. 64x32 = 4096 bits = 256 bytes = 1

page; ie one of 256 bytes can be selected with an 8-bit address, thereby grossly simplifying the display driver software.

In order to produce square dots, the width of each dot had to be 0.5 usec (hence 2MHz clock), but a standard video line is 64 usec making the total line 128 dots wide — a binary multiple, thus simplifying the horizontal counter circuitry.

Carving 40% off the M6800's effective speed makes negligible difference in this application. In fact, the 6800 CHIP-8 interpreter runs faster than the Cosmac VIP. To keep everyone happy, the VDG can be turned on and off under program control to allow maximum, uninterrupted MPU speed when requiried.

Video is produced by loading a parallel-in/serial-out shift register (IC7) with a byte of RAM, via the data bus. The bits are then clocked out and combined with sync signals to produce composite video. A set of counters (IC's 15, 14, 13) are responsible for producing sync pulses, and for supplying the RAM address for each byte. This address is applied to the system bus via a set of tristate buffers (IC5,6), which are enabled by the bus-available (BA) control line. BA is put HIGH by the MPU after a HALT request, telling that the address and data busses from the MPU are floating (high impedance state), and can be used by other devices (eg the VDG).

Horizontal timing is produced by counting dots (2MHz pulses). When the counters are reset, the first byte is being addressed. This byte is the upper LHS of the active picture, where a sync pulse is NOT required. So a gate (IC16a) looks for a set of conditions which will tell when a horizontal sync pulse is needed. By a welcome coincidence, the pulse out of this gate is 4 usec long, which is near enough to the desired 4.7 usec for HSYNC. The pulse occurs on the 88th dot after the start of the active portion of the picture. Now we need a way to blot out the unwanted part of the picture. Only 64 of the 128 dot positions are valid picture dots.

The counter output called H64 changes state every 64 dot positions, and thus can be used for a gating signal to enable the picture during the first 64 dots.

Here's where complications set in. We have to allow 450 ns RAM access time, and allow for the propagation delay in the shift register, between the instants of supplying a valid address and receiving valid data. This problem was solved by delaying the H64 signal with an RC delay network.

Due to the low threshold of TTL different rates of charge and discharge were necessary, hence the extra resistor and diode. Having delayed and inverted H64, a suitable "horizontal-enable" signal (HOR-EN) has been derived. (Refer timing diagram.)

Vertical timing is derived by counting HSYNC pulses. A field is composed of 312 lines, which is 78 rows of dots. Only the first 32 rows are valid picture, so a vertical enable signal (VERT-EN) can be formed by gating V32 and V64. Field sync is derived by looking for the 56th row (IC18a) and using this transition to fire a one-shot (IC19b) of about 300 usec. The vertical counter (IC13) is reset on the 78th row by another gate (IC16b).

The vertical enable signal serves two tasks. Firstly it masks out the unwanted part of the picture in the field, and secondly it serves to HALT the MPU during the time that display refresh is required. Note that VERT-EN can be prevented from going HIGH altogether, by the DMA-ENAB signal, allowing software enable/disable of the screen.

Let us now focus our attention on the Input/Output interface, controlled by an M6821 Peripheral Interface Adaptor. Whoever designed this device is a wizard! For starters, every individual bit can be set up to be either an input or an output, under program control, including two control lines. Next, the B side port is Tristate (the lines float when programmed as inputs), and the A side has internal 5k pullups for versatility. The control lines can be programmed to be flags, senses, interrupts or strobes. In this particular application, the data and control lines are all functioning independently.

The tape interface is a simple frequency-shift keyed (FSK) modem, which uses the "Kansas City Standard" frequencies and data rate (2400/1200Hz @ 300bits/sec). The modulator is a 566 function generator (IC24) which free runs at 2400Hz, unless the serial data out line (PB0) is LOW. Then, the control voltage on pin 5 is changed to produce 1200Hz, by turning on the transistor. This transistor is wired in the "inverse switching mode" (upside-down!) to get a lower Vce(sat). The triangle waveform changes gracefully from one frequency to the other, thereby producing an ideal FSK output for taping.

Using supply voltages +5V and —5V on the 566 gives TTL compatability for both voltage control and square wave output. This output drives a speaker when its enable line (PB6) is HIGH. The speaker is inhibited during tape I/O, by using the same line that inhibits the display (CB2). The main use for the speaker is as a "bleeper" to acknowledge valid keystrokes.

The tape demodulator consists of a comparator (IC23) to square up the incoming signal, followed by a pulse-width discriminator (IC22,21). The input signal from the tape deck should be in

TABLE OF CHIP-8 INSTRUCTIONS

Stored Code	Mnemonic	Description
1MMM	GOTO MMM	Jump to instruction at location MMM.
BMMM	GOTO MMM + VO	Computed GOTO; Jump to MMM + VO.
2MMM	DO MMM	Do CHIP-8 subroutine at MMM.
00EE	RETURN	Return from CHIP-8 subroutine.
3XKK	SKF VX = KK	Skip next instruction if VX = KK (hex).
4XKK	SKF VX ≠ KK	Skip if VX NOT = KK.
5XY0	SKF VX = VY	Skip if VX = VY.
9XY0	SKF VX ≠ VY	Skip if VX NOT = VY.
EX9E	SKF VX = KEY	Skip if key down = VX; no wait.
EXA1	SKF VX ≠ KEY	Skip if key NOT = VX; no wait.
6XKK	VX = KK	Assign hex constant KK to variable.
CXKK	VX = RND.KK	Get random byte; AND with KK.
7XKK	VX = VX+KK	Add (2's comp.) KK to VX.
8XY0	VX = VY	Copy VY to VX.
8XY1	VX = VX!VY	Logical OR VX with VY.
8XY2	VX = VX,VY	Logical AND VX with VY.
8XY4	VX = VX+VY	Add VY to VX; If result >FF, VF=1.
8XY5	VX = VX-VY	Subtract VY; If VX <VY, VF=0, else 1.
FX07	VX = TIME	Get current timer value.
FX0A	VX = KEY	Input hex keycode (wait for keydown).
FX15	TIME = VX	Initialize timer; 01 = 20 millisec.
FX18	TONE = VX	Bleep for 20 x VX milliseconds.
AMMM	I = MMM	Set memory index pointer to MMM.
FX1E	I = I+VX	Add VX to memory pointer.
FX29	I = DSP, VX	Set pointer to show VX (LS digit).
FX33	MI = DEQ, VX	Store 3-digit decimal equiv. of VX.
FX55	MI = VO:VX	Store VO thru VX at I; (I = I + X + 1).
FX65	VO: VX = MI	Load VO thru VX at I; (I = I + X + 1).
00E0	ERASE	Clear the screen.
0MMM	CALL MMM	Call machine-code subr. (MMM>200).
DXYN	SHOW N@VX, VY	Display N-byte pattern at (VX, VY).
0000	NOP	No Operation.
F000	STOP	Jump to monitor (CHIPOS).

(X,Y,N and M are arbitrary hex digits, O to F.

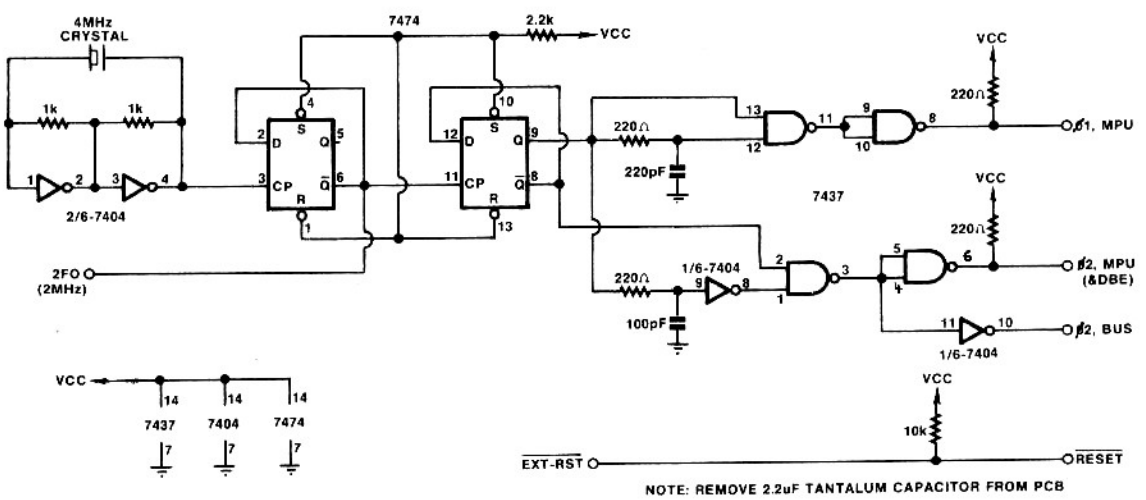
the range 300mV to 3V peak-to-peak (100mV to 1V rms). The square wave output from the 741 triggers a one shot whose period is 3/4 cycle at 2400Hz, ie 312 usec. The same square wave is sampled by a D-type flip-flop when the one-shot times out. If the signal was low at this instant, then it must be 2400Hz, but if it was still HIGH, it must be 1200Hz (see diagram). Hence the demodulated signal is the inverse output from the flip-flop. Note that the Schmitt-trigger facility of the 74121 is exploited, for added noise immunity.

The task of converting 8-bit parallel data to an asynchronous serial bit stream at 300 Baud, and vice-versa, is done by software in CHIPOS. Thus the tape modem may be disconnected, and PB0,PB7 used as a serial data communications port, if such an application is envisaged (eg "smart" terminal).

Operation of the hex keypad is very simple from a hardware point of view. Normally, the rows (PA4 to PA7) are outputs, held LOW, while the columns (PA0 to PA3) are inputs, held HIGH by internal pullups. If any key is pressed, one column must go LOW causing a rising edge at CA1. The software encoding routine then can determine which column went LOW, then reverse the roles (ie data directions) of the rows/columns to determine which row is active. This program incorporates debounce and error-checking sequences, ensuring ultra-reliable functioning.

The [FN] key posed a special problem: how to detect closure of an SPST contact without getting bounce or noise, and without introducing an extra chip. The final solution was a Schmitt-trigger made from a spare AND-gate with feedback.

Here is a substitute circuit for the 6875 clock chip:



Just before the July issue was due to be run on the presses, the shortage of 6875 clock chips became apparent. It seems that it could be several months before Motorola, Inc, USA is able to restore supplies. In the meantime, designer M.J. Bauer has produced a substitute circuit for the 6875 using cheap and readily available TTL ICs. This circuit may be built up on a small section of Veroboard and linked to the DREAM PCB via a ribbon cable fitted with DIL plug (16-pin) and IC socket, in the 6875 position. When the time comes, the TTL circuit can be discarded and the 6875 plugged in, instead. Note that the reset circuitry must also be changed slightly, as noted on the circuit above.