The DREAM 6800 Computer

This computer first saw the light of day as a series of articles that featured in the May, June, July and August 1979 issues of the now-defunct magazine *Electronics Australia* and it reappeared in a follow-up booklet entitled *Microprocessors and Personal Computers*, published in 1980.

The DREAM 6800 (Domestic Recreational Educational and Adaptive Microcomputer incorporating the Motorola 6800 microprocessor) was designed by Michael J Bauer from Brisbane's Deakin University. It is not particularly sophisticated but at the time offered those handy with a soldering iron the opportunity to build a simple computer that would be a lot of fun to use. It offered programming in two languages: native 6800 machine code and a strange but quite powerful language called CHIP-8 (Comprehensive Hexadecimal Interpretive Programming – 8 bit). In its basic form, the language features thirty-three two-byte instructions and had been developed at the RCA Laboratories in the US by one Joe Weisbecker in the 70s, principally for those interested in writing their own games programs. RCA also developed a computer called the COSMAC VIP, that could run programs written in CHIP-8. At the time it was termed a "high level" language but it would hardly even begin to qualify for that description nowadays. Still, writing CHIP-8 programs was, and still is, fun and of particular interest to MicroBee users is the fact that a CHIP-8 interpreter, cleverly linked to MicroWorld BASIC, was written by a Melbourne solicitor, Lindsay Ford and enhanced CHIP-8 still further by providing additional instructions. It is to be hoped that it will eventually be able to be run on MicroBee emulators such as PicoMozzy but it does require direct disc access to function, as much of it is menu driven.

The CHIP-8 interpreter and monitor program in the DREAM 6800 is housed in ROM and is termed CHIPOS. It begins at C000H and ends at C3FFH so in all it occupies 1K of memory. Imagine any version of Windows being capable of that? (An unfair comparison you might well say – and you would be correct, of course). Saving and loading programs is taped-based, and utilises the "Kansas City Standard" with frequency shift keying (FSK) frequencies of 1200Hz and 2400Hz at a rate of 300 bits/second. Screen resolution is 64 pixels wide by 32 pixels high (no, I'm not kidding!) and direct memory access (DMA) for the screen begins at 0100H and extends for 256 bytes (i.e. to 01FFH). There is no colour. Programs are run in memory beginning at 0200H and, for very long ones, it is possible to use an additional area from 0080H to 00FFH.

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.

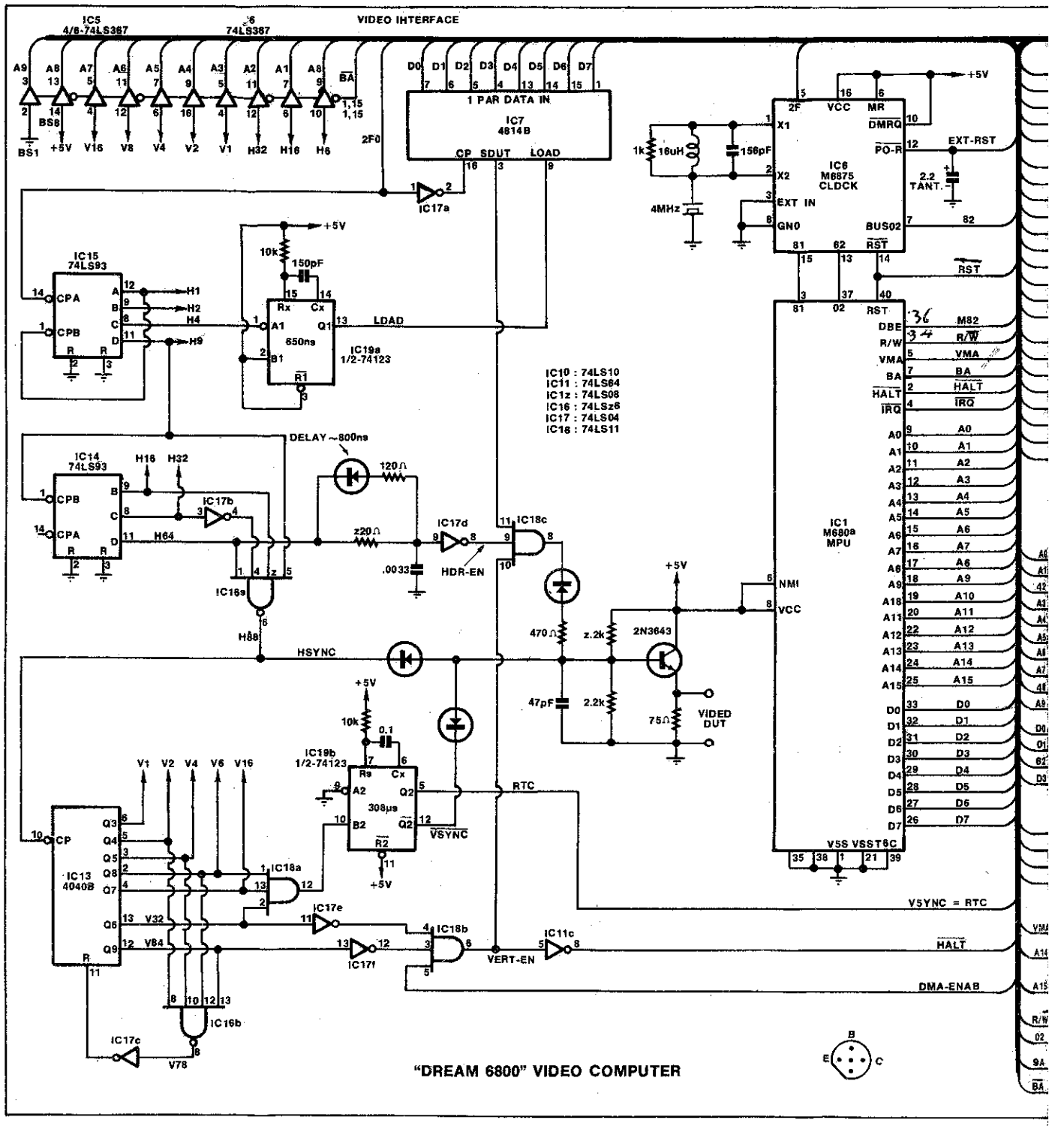
For advanced users, CHIPOS has been written with flexibility in mind. Calls to over 17 useful CHIPOS subroutines can be made from a machine-code program; eg, erase the screen, fill the screen, get random byte, display hex digit, convert byte to decimal, show a user-defined symbol up to 8x16 dots, set 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.

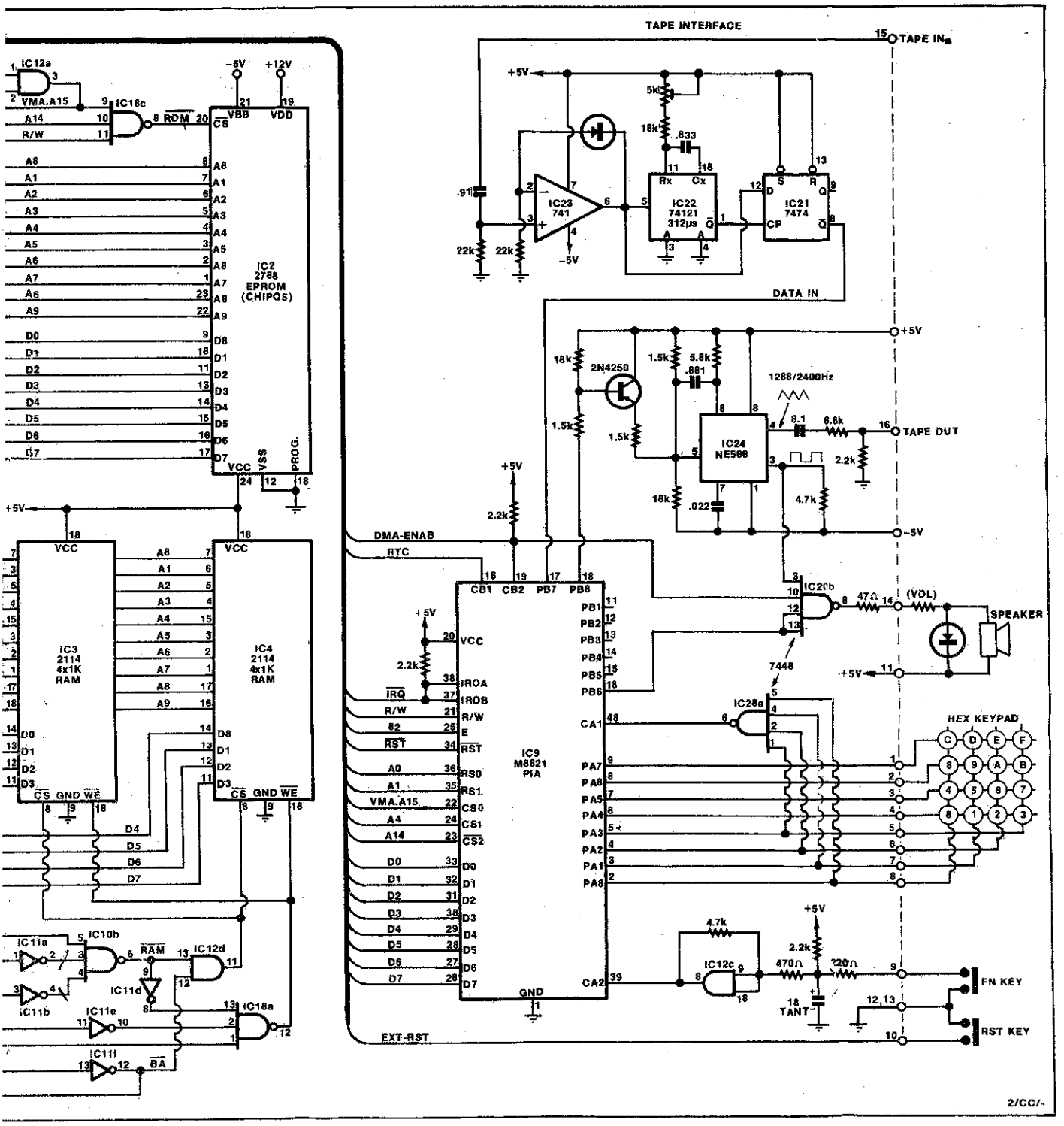
which produces a matrix of dots (64x32) on a TV screen, each dot being a bit in the system's RAM. Nor is a full typewriter-style keyboard required, since all instructions are coded in hexadecimal, and entered via a hex keypad.

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, VO 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.

3. 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 (PBO) 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.

DREAM 6800 GLOSSARY

ACIA: Asynchronous Communications Interface Adaptor; Motorola's answer to the UART, or Universal Asynchronous Receiver Transmitter; it provides the data format and control to interface serial asynchronous data to bus organised systems.
ADDRESS: The label, name or binary number specifying a particular location in memory.

ALPHANUMERIC: Refers to numbers and letters of the alphabet; ie, an alphanumeric code represents numbers and letters.
AND-GATE: A digital logic element with output logic value related to the AND logical function; ie, with all inputs 1, the output is 1 but all other input combinations result in 0 output.
ARRAY: A named group of related variables or constants. Items in the array may be located in consecutive memory locations or they may binked.
ASYNCHRONOUS: Refers to a system or circuit whose elements are not arranged to change state in synchromism.
BASIC: One of the ligh-level programming languages which is user-readable.
BAUD: Used as a measure of serial data flow. 10 baud normally equals 10 bits/second.
BINARY: Refers to the number system with base 2 and expressing all quantities by the numerals 0 and 1.
BIT: Binary digit, either 0 or 1. The minimum amount of information.
BRANCH: An instruction in a program which causes the processor to execute a step not in the usual sequence. A branch can be unconditional or conditional, based on the magnitude or state of some value. Branch is synonymous with Jump.
BUFFER: refers to an amplifier which is interposed between two circuits to avoid undue loading effects. "Buffer" can also refer to an area in computer memory which is used as a work area or to store data for an input/output operation.
BYTE: A group of consecutive binary bits, usually eight, which are operated upon as a unit. A byte can also be a subset of a computer word. Additionslly, byte is a unit of memory size: the Dream/6800 has 1K (1024 bytes) of memory in RAM.
BUS: A circuit or group of circuits which provide a communication path between two or more devices, such as between processor, memory and peripherals. The "S100" bus is based on that originally used in the MITS/Altair 8080 computer and which subsequently became a USA industry standard.
CAI: Computer-aided instruction.
CLOCK: A pulse generator which provided timing signals to which all system operations are synchronized.
CONDITIONAL: See Branch.
DMA: Direct Memory Access; a method of transferring data directly between an external device and system memory without the need for processor intervention. This method significantly increases the data transfer rate and hence system efficiency.
D-TYPE: a particular type of flipflop which, when a clock pulse arrives, stores or latches the logic level at its D-input.
DEBUG: A diagnostic program which helps locate hardware malfunctions in a system or to identify coding errors in newly developed programs.
DISABLE: Use a control voltage to halt system operation.
ENABLE: Use a control voltage to start a system or circuit, or to allow it to function.
ENCODER: A digital circuit which accepts information in uncoded form and generates corresponding coded data.
FLAG: An indicator, usually a single binary bit, used to indicate a condition for a peripheral device or a later stage in a program.

FLIPFLOP: A family of digital circuits capable of assuming either of two stable states and therefore capable of storing one bit of information.
FIELD: Refers here to one vertical scan of a television picture, which occurs 50 times a second.
FSK: Frequency-shift keyed; refers to a digital mode of data transmission wherein the two logic levels are transferred as two distinct frequencies. In the case of the "Kansas City Standard" which resulted from a symposium organised by USA magazine BYTE, logical 1 is a 2400Hz tone while logical 0 is 1200Hz.
GRAPHICS: A system of producing pictorial or graphical information on a video monitor, television screen or chart recorder.
HARDWARE: In this context means the electronic circuitry of the computer and its peripherals.
HEX: Nasty spell placed on person trying to debug a program; also abbreviation for Hexadecimal.
HEXADECIMAL: A number system with base 16 using numbers 0 to 9 and A,B,C,D,E,F to represent its 16 digits. Two hexadecimal digits can be used conveniently to represent 8 bits, or a byte.
I/O: Input/Output. An I/O Port is connection to a processor to provide a data path to or from external devices suchas keyboard, display or cassette recorder. An I/O port of a microprocessor may be input or output or it may be bidirectional.
INSTRUCTION: A set of bits which defines a computer operation and is a basic command understood by the processor.
INTERFACE: A device which transfers data from one system to another.
INTERPRETER: A program that fetches, translates and controls execution of instructions written in a higher-level language.
INTERRUPT: The suspension of a normal program routine of a computer or microprocessor in order to handle a sudden request for service, usually by a peripheral device.
JUMP: See Branch.
LANGUAGE: A set of symbols and expressions used to express a computer instruction or program. More commonly the term is used to describe higher-level languages, or programs expressed in symbolic form for convenience of

DREAM 6800 GLOSSARY . . .

human beings. Programs written in this form must be translated into machine code form before they can be executed by a computer.

LOAD: To store binary information, generally into a processor register.

MACHINE CODE (or language): The binary numeric form of instructions actually understood by a processor.

LOOP: A sequence of instructions so arranged that upon execution the processor is forced to execute the sequence repetitively a certain number of times.

MODEM: Modulator-demodulator; used for transferring digital information over a communications path.

MONITOR: Can refer to a "video monitor" which displays video signals without the need for an RF demodulator; more commonly, in this context, refers to a specific program allied to the processor which looks after communications between the software and system hardware.

NESTING: The technique of cascading program loops or subroutines. In the case of loops, nesting involves loops-within-loops-within-loops, and so on. In the case of subroutines, nesting involves a subroutine calling another subroutine, which in turn may call another subroutine, and so on (see Subroutine).

PIA: Peripheral Interface Adaptor (See text).

PARALLEL: A method of simultaneously transferring each of a contiguous set of bits over separate wires, one wire for each bit in the set; an eight-bit system requires eight wires.

PROGRAM: A set of computer instructions arranged to control the processor in executing a certain complex task. A program may consist of a linear sequence of instructions, one or more loops, one or more subroutines, or more usually a combination of all of these.

ONE-SHOT: A monostable multivibrator; a logic circuit which delivers just one pulse on reception of a control signal.

OPERAND: Can be the result of a computation, a constant, a parameter, the address of any these quantities or of the next instruction to be executed.

PULL-UPS: Internal or external resistors designed to "pull" a logic output (or input) towards the more positive supply rail when the output is turned off.

POINTER: Registers in the processor or memory that contain memory addresses; i.e., they are used to "point" to memory locations.

PORT: See I/O.

PROCESSOR: Also CPU (Central Processing Unit) or Microprocessor; the major portion of a computer incorporating an arithmetic-logic unit (ALU) to gether wijth various registers and timing circuitry.

RAM: Random-access memory.

ROM: Read-only memory. A memory device in which the stored data is effectively permanent in normal operation and can only be read out, not overwritten.

REGISTER: A device used to store or manipulate numbers of other data; usually a group of flipflops.

ROUTINE: A program or program segment designed to accomplish a single function.

SCHMITT TRIGGER: A controlled switching circuit whose threshold for rising inputs differs from that for falling inputs; usually used for "squaring up" pulses or pulse waveforms. The difference between the two input thresholds is known as hysteresis.

SCRATCH-PAD: Designates an area of memory used for many quick data transfers. It is the most frequently used memory segment. Some microprocessors have simplified instructions which can only be used in a certain part of the memory (say, the first 256 bytes) where the most significant byte of the address is zero. The scratchpad is usually placed in such a location.

SERIAL: A method of sequentially moving a contiguous set of bits (data) over a single wire.

SKIP: An instruction which causes the processor to omit the next instruction in the sequence.

SOFTWARE: Generally refers to all the programs and routines available for a particular computer or microprocessor. More specifically, it refers to those programs which are alterable by the user. Those programs and routines that are not user-changeable, ie, burned into ROMs, are often referred to as Firmware.

STACK: A sequence of registers or memory locations used in Last In First Out (LIFO) fashion. A "stack pointer" specifiers the last-in entry.

STORAGE: Any device in which data can be stored.

STROBE: System of rapidly reading registers or memory in sequence.

SUBROUTINE: A short program segment which performs a specific function and is available for use any number of times by other programs and routines.

SYNC: Describes the pulse trains which synchronise the line scan (15,625Hz) and field rate (vertical sync, 50Hz) for television and video monitors. Sync also refers to the trace synchronising pulses used in oscilloscopes.

TTL: Stands for Transistor-transistor logic. One of several logic systems.

TRI-STATE: Registered trade mark of National Semiconductor Corporation, USA. Describes gates and other circuits which can not only have their outputs at logical 1 or logical 0 levels but can also assume a high impedance state. The latter state allows other gates connected to the same bus system to "talk" alternatively.

UFO: Unidentified flying object. Nothing to do with computers or microprocessors.

VDG: Video Display Generator.

VIP: The RCA microcomputer system bases on their Cosmac microprocessor.

VARIABLE: A named memory (RAM) location which is given some consistent meaning in the program and which may contain different data values during the execution of the program.

You could build this computer in just a few nights!

DREAM 6800

Second article on this innovative design

This month, author Michael Bauer gives the construction, testing and trouble-shooting procedures for the DREAM 6800. If you get started on yours now, youcould have it ready to run the programs we will be featuring next month. Included inthis article is the full hex listing of the DREAM's high-level interpreter/monitor program, CHIPOS.

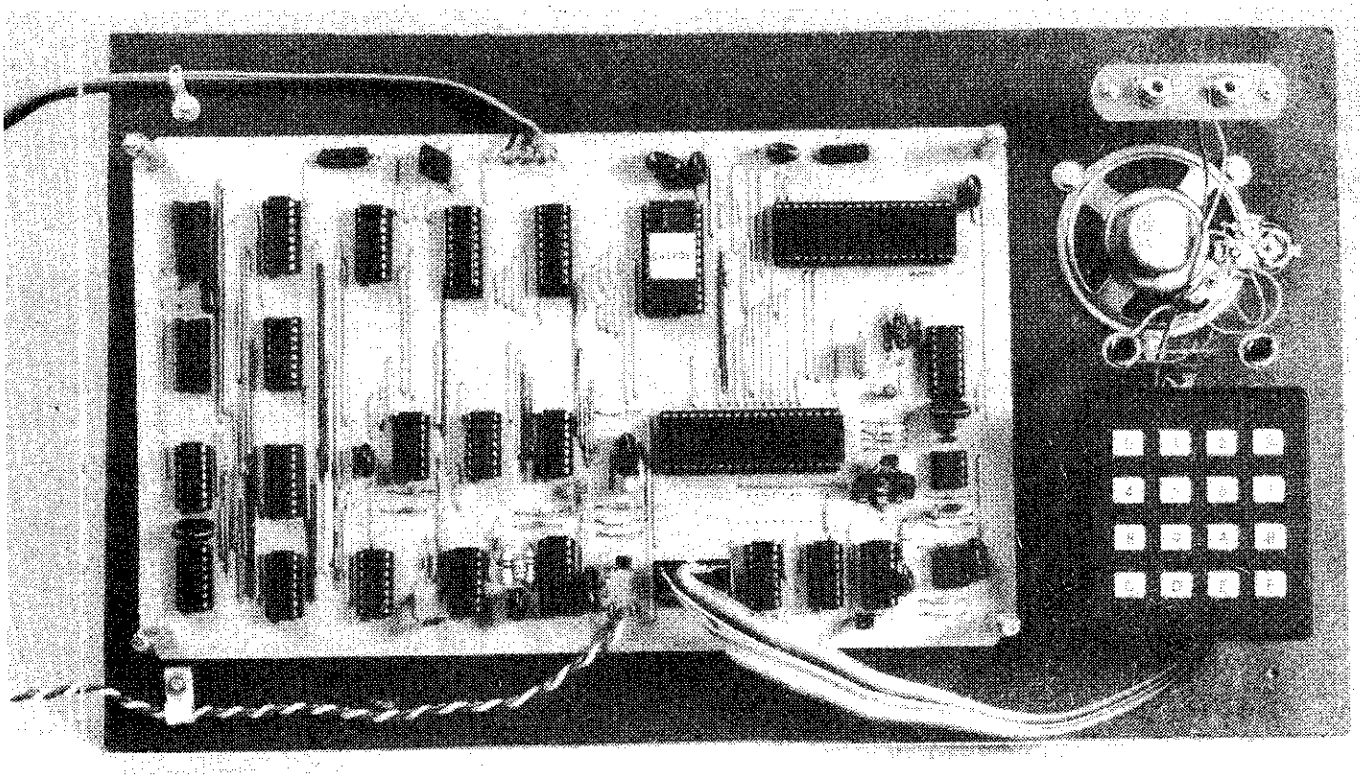
Before soldering, inspect the PCB for flaws. Make sure the PCB is clean. If it is not tin-plated, scrub the copper pattern of the PCB thoroughly with soapy water and steel wool. Solder does not take very well to tarnished or lacquered copper.

You will need a low wattage (20 to 40W) or temperature controlled soldering iron with a fine tip. Some tracks on the PCB come very close together, which means great care is needed to avoid solder bridging. Use 22 gauge, 60/40 resin-cored solder. Do not attempt the job with an old carbon-element iron like a "Mini-Scope" or "Birko". It's about time you bought a precision soldering instrument, anyway.

(Editor's note: The author's recommendation against carbon-element irons refers to the possibility of damage from these irons to some ICs particularly CMOS types. This is because the heavy current flowing in the soldering iron tip prevents it from being effectively earthed. This allows the possibility of damaging voltages being applied to an IC while it is being soldered.)

Bend the leads of components (and links) with pliers before insertion into the board to avoid stressing the casings. Splay the leads outward on the solder side of the board to hold the part in place during soldering. IC's can be held with masking tape, or a finger if you've got three hands. Note that all resistors (except one) have standard half-inch lead spacing. Use the minimum amount of solder practicable for each joint; don't make blobs!

Our prototype was built on a hardboard base with a perspex cover to protect and show off the PC board.



Begin construction with the links, which are considerable in number but think of the $10 or more saved over the price of a double-sided plated-through PCB. Links may be made from tinned copper wire, except for long ones (where there is danger of shorting) which should be insulated. Next, solder in the IC sockets, the keypad I/O socket, and if desired, the expansion sockets. It is recommended to only use sockets for IC's 1,2,3,4,7 and 9. If any others prove faulty, they can be removed with the aid of solder-wick or a solder-vac. The passive components (R,C & L) can then be mounted, followed by the ICs, diodes, pot, transistors, crystal and terminal pins for power and video. Observe orientation of ICs, diodes, tantalum capacitors and transistors. Do not yet plug in ICs 1,2,3 ,4,7 and 9.

Before powering up, inspect your workmanship closely for solder bridges, missing links or components and disoriented parts. By the way, resin residue may be removed from the PCB with a toothbrush dipped in acetone or methylated spirit.

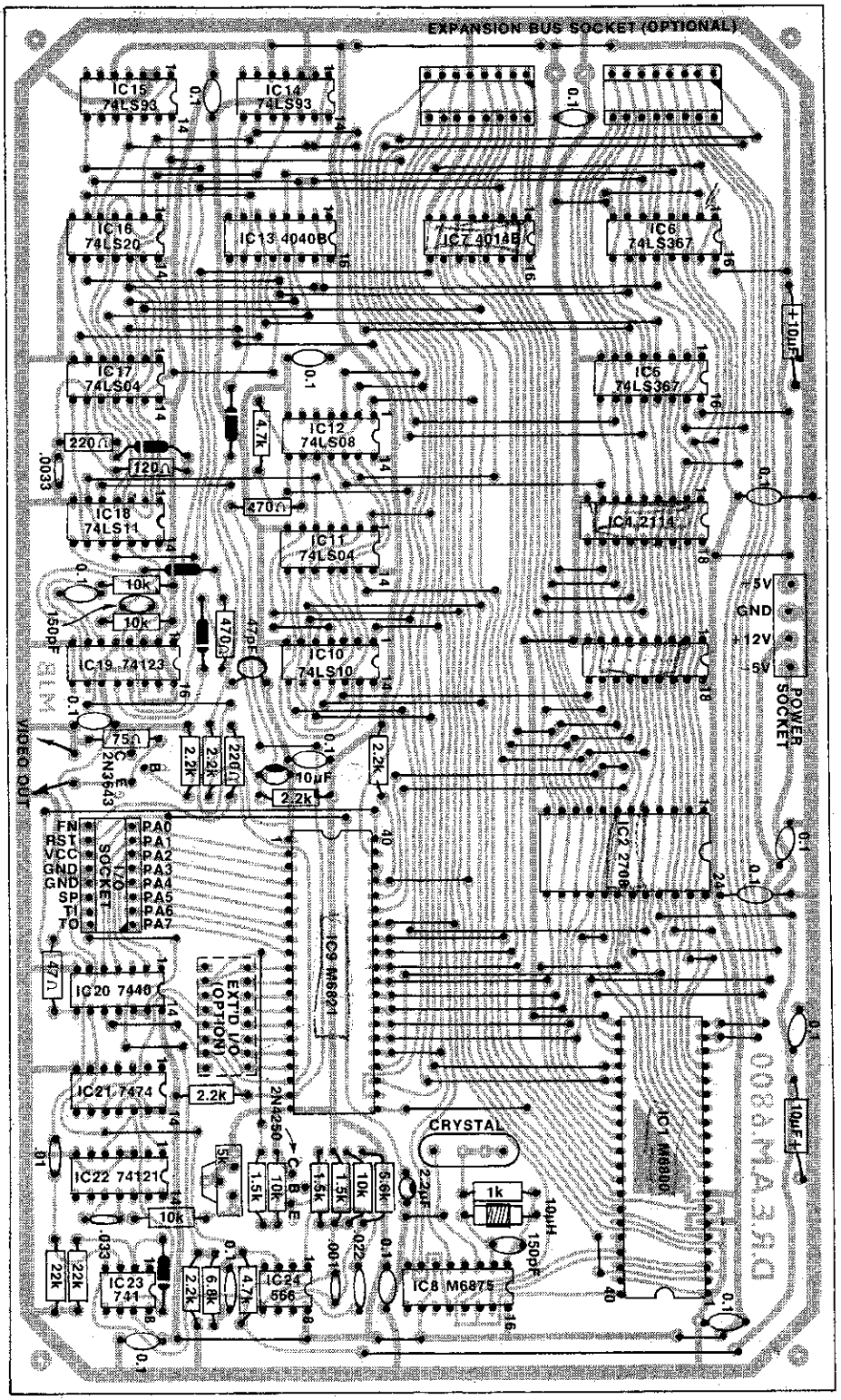
First connect the power supply. Details of a suitable power supply are given elsewhere. You should check the output voltages and regulation under load before applying power to the computer. A 47 ohm, 5 watt resistor will do for a load.

Next, a video monitor needs to be installed. If you can convert your TV to accept a video signal, the results will be worth the effort. Otherwise an RF modulator may be employed to feed the signal into the antenna input. This is the easiest way to get video, and you have the advantage of being able to use your computer on any TV set, anywhere.

Several possibilities exist for choice of keypad. The most economical is to wreck an old calculator. Some obsolete desk-top models used magnetic reed switches which are ideal. The complete computer might be able to be housed in the cabinet of such a calculator. Alternatively, a variety of keyboards, current models and obsolete industry surplus models are available, mainly through the international mail-order houses, at prices from 99 cents to over $10. Whatever you decide on, make sure it is (or can be) wired in a 4 x 4 matrix as shown in the wiring diagram.

Also, it will be an advantage if there is room on or around the keys for labelling (e.g. for special functions, or alphabetic symbols). A further two keyswitches are needed for [FN] and [RST]. As these will be used frequently, they should be of high quality, not cheap and nasty pushbuttons.

Wire the keypad (incl. FN and RST) to a 16-pin DIL plug via a short length (up to 0.5m) of 12-conductor ribbon cable. The GND lines (2) should separate the PA lines from CA2 and RST. Connect a small 8-ohm speaker and diode to pins 11 and 14. The system is now ready for the smoke test.



Install the wire links before mounting any other components on the PCB.

With ICs 1, 2, 3, 4, 7 and 9 removed, power and TV connected, switch on and immediately check voltages at the power socket. If there are no loud noises nor signs of molten silicon, adjust your TV (and RF modulator) to get a large white rectangle centred on the screen. If it is not forthcoming, chances are that there's something wrong with your RF modulator, or a certain 2N3643 is in backwards. Having eliminated these as the source of error, but still getting no picture, go to the section on trouble-shooting.

One you've got a picture (large white rectangle), remove power and proceed to plug in ICs 1,2, 3, 4, 7 and 9, and the keypad and speaker. It is very important when handling MOS devices that everything is at the same potential and preferably earthed, in particular your

printed board (GND line), your work surface, and yourself. Don't wear nylon pantyhose or rub ebonite rods in your hair while handling MOS!

Apply power once again, and you should notice a short bleep in the speaker and something resembling fig. 2 on the screen. The actual pattern and 4-digit number seen are just random garbage in the RAM at switch-on. Try keying in any 4 hex digits. This number should then appear on the screen, and if so, your computer is, in all probability, fully operational. See how many 4-letter words you can make from the hex digits: A,B,C,D,E and F. If your system fails to display the above information (after resetting), see how many 4-letter words you can mutter to yourself and then proceed calmly to the section on trouble-shooting.

Once the video generator and processor appear to be working, you can try using the memory-modify command. Hit [RST], then enter 0, 1, 0, 0, the starting address of the display refresh buffer. Now key [FN] followed by [0] to get into "memod". The display window will show a 2-digit number beside the address. This is the contents of location 0100, which can also be seen in binary at the upper LHS of the screen. (A white dot = 1, no dot = 0.) Step through the memory by pressing [FN] repeatedly. Go back to 0100 (hit [RST], 0, 1, 0, 0, [FN], 0). This time, write into the buffer by keying in a pair of hex digits, and another, and another . . . noting the binary pattern formed by each byte.

Notice that as data bytes are deposited into memory, the address flips to the next address, before you see the byte just keyed in. This is a design compromise (not an oversight), but should be of no concern because you're not likely to be looking at the screen anyway, when keying in a program. One eye will be on a listing, the other on the keypad. The data, having been entered, can be verified later by stepping through with the [FN] key.

Getting the feel for it, and want to try a CHIP-8 program? Try the simplest possible! Use memod to enter this data at 0200:

Address	Data	Mnemonic
0200	F0	go to monitor
0201	00	

The instruction F000 does not exist, and will result in a jump back to the monitor (CHIPOS), but first the interpreter clears the screen, as it does at the start of each new program, (unless you start from C002). To run this "program", hit RST, C000, FN, 3 (GO from C000).

Here's something to watch CHIPOS's random number generator at work:

0200	CA3F	VA is random x-coord (00-63)
0202	CB1F	VB is random y-coord (00-31)
0204	A20A	Point to pattern byte (I=20A)
0206	DAB1	Display 1 byte at coords (VA, VB)
0208	1200	Go to loc'n 0200 for next instr'n
020A	8000	DATA: 80hex = 10000000 binary = dot.

Before the programs get too much bigger, you'll want to save them on cassette. If your recorder has line (auxiliary) input and output, you're fortunate because these voltage levels are optimum for use with the DREAM-6800's tape modem. Also it is highly desirable (but not essential) that the recorder's internal speaker not be muted, so that the leader tone can be located by sound. Hence, recorders with only an EXT-SPKR (or earphone) jack should be modified such that insertion of the plug does not result in disconnection of the speaker. If this is awkward, another speaker can always be connected externally.

Do not operate the recorder at high volume when connected to the computer. Voltage levels exceeding 5V peak-peak could damage IC23, but this is improbable at normal listening levels. Further, recorders without an AUX input may require a much lower signal level from the computer, for use with the "MIC" input. This problem is easily solved by inserting a 220k resistor in series with the "TAPE-OUT" line (pin-16). Shielded cable should be used for the tape connections, with the shield wired to pin-13 of the I/O plug.

To test the cassette functions, proceed as follows. Use the "memod" function to create a pattern on the screen, as described earlier. Then define the beginning and ending addresses of the block you want to save, in this case the display buffer page from 0100 to 01FF. For convenience, the ending location PLUS ONE is specified. Hence to dump the display, deposit the following data at 0002:

0002	01	Beginning address MSB
0003	00	Beginning address LSB
0004	02	Ending address (+1) MSB
0005	00	Ending address LSB

HOW WE BUILT OURS:

Since the author did not present a proto-type with his article, we decided to build our own, both to confirm the design and to aid presentation of this attractive system in the magazine. With the latter idea paramount, we decided to mount the PCB on a hardboard base with a perspex cover to protect and show off the unit.

The perspex cover is also used to mount the keyboard, two pushbuttons and the tape interface sockets. This method of construction is easy to build, is very economical and produces an attractive unit.

Readers will note that we have used IC sockets for the ICs on the PCB. We did this as a precaution — if bugs had shown up, we wanted to be able to change ICs with a minimum of work. Nevertheless we are inclined to agree with the author's comments on IC sockets.

The miniature speaker is mounted face down on the hardboard base. Even so we found the loud bleeps it emitted quite annoying, so we muted it with a preset pot, as suggested by the author.

Rather than salvage a keyboard from a calculator or other source, we took the easy but expensive approach of buying a new one. We used a Digitran KL0043 keyboard, which has the buttons connected in a 4 x 4 matrix as required but with a slightly different numbering to that shown on the circuit published last month. However, this presents no problem.

The keyboard can be purchased from Radio Despatch Service, 869 George Street, Sydney, NSW 2000. Radio Despatch Service have notified us that since the wholesale price of the keyboard is high, they have reduced their own margin to a minimum. Even so, the KL0043 will set you back by $20.43, including sales tax. Radio Despatch Service also have a ready source of suitable perspex in the form of surplus smoke-tinted record deck covers, at $2 each.

For the RST and FN pushbuttons we used two good quality momentary contact switches. We didn't bother to label these as they are used so often that it soon becomes second-nature. The FN button is mounted on the right and the RST on the left, immediately above the keyboard.

The major ICs for our unit, with the exception of the CHIPOS EPROM, were supplied by Total Electronics, 155 Willoughby Road, Crows Nest, NSW 2065. Silicon Valley Stores and Applied Technology Pty Ltd, 1a Pattison Avenue, Waitara, NSW 2077 will be able to supply all the IC's, including the CHIPOS (2708) EPROM.

Two other firms have EPROM programming services: A.J.F. Systems & Components Pty Ltd, 29 Devlin Sreet, Ryde, NSW 2112 and Warburton Franki (Sydney) Pty Ltd, 199 Parramatta Road, Auburn, NSW 2144.

DREAM 6800

[N.B.: 01FF+1=0200; MSB = Most significant (high-order) byte; LSB = Least significant (low-order) byte.]

Thus, a 256-byte block is defined, from 0100 to 0200, not including the last byte (at 0200). The same block applies to a load or dump. This simple tape format lets you load a file (or part thereof) into any place in RAM, regardless of where it was dumped from, thereby allowing relocation of data or programs.

Having got that, reset the system, start the cassette in RECORD mode and adjust the recording level, and let it run for several seconds to write a "leader" tone (steady 2400 Hz). Then key [FN][2] (dump/save). The screen will be disabled until the dump is complete, because the serial I/O software cannot tolerate the display refresh delays.

To verify the dump, and to test your demodulator, power down the system to destroy RAM contents. Once again, enter the begin and end locations, as above. Set the DEMOD trimpot to mid-position. Rewind and play the tape until the leader tone is heard, then press [FN][1] (load). The display will again black out and should return at the instant the last byte is accepted, hopefully revealing your saved pattern.

If anything goes wrong, first retry the above steps. Then try various recording and playback levels, or try adjusting the DEMOD trimpot (although this should be non-critical in the majority of cases). As a last resort, you might have to check the modem with a CRO, but be suspicious of external troubles first. Also note that it pays to use good quality cassettes.

That concludes the testing procedure. Now you can look forward to entering and saving much larger programs. Be sure to write down the block loading addresses on the cassette index. It's a good idea to always use "standard" size blocks; e.g. 0200-0300 for a small program; 0200-0400 for a medium; and 0080-0400 to dump all usable RAM. Refrain from dumping/loading 0000-0080, because this area is reserved for CHIPOS's scratchpad and stacks.

Just a final note for perfectionists. The width of the first and last dot (on every row) is controlled by the delay network on H64, (120 ohms, 220R, 220 ohms, .0033uF). If the RHS dots are too narrow, first try increasing C to .0047uF. Also, the frequency of the cassette modulator (2400Hz, marking) can be adjusted by the 5.6k resistor. Speaker volume can be reduced with a series resistor or 500 ohm trimpot.

This is the complete listing for the CHIPOS interpreter/monitor program.

```
C000  8D 77 CE 02 00 DF 22 CE 00 5F DF 24 DE 22 EE 00
C010  DF 28 DF 14 BD C0 D0 96 14 84 0F 97 14 8D 21 97
C020  2E DF 2A 96 29 44 44 44 44 8D 15 97 2F CE C0 48
C030  96 28 84 F0 08 08 80 10 24 FA EE 00 AD 00 20 CC
C040  CE 00 2F 08 4A 2A FC A6 00 39 C0 6A C0 A2 C0 AC
C050  C0 BA C0 C1 C0 C8 C0 EE C0 F2 C0 FE C0 CC C0 A7
C060  C0 97 C0 F8 C2 1F C0 D7 C1 5F D6 28 26 25 96 29
C070  81 E0 27 05 81 EE 27 0E 39 4F CE 01 00 A7 00 08
C080  8C 02 00 26 F8 39 30 9E 24 32 97 22 32 97 23 9F
C090  24 35 39 DE 14 6E 00 96 30 5F 9B 15 97 15 D9 14
C0A0  D7 14 DE 14 DF 22 39 DE 14 DF 26 39 30 9E 24 96
C0B0  23 36 96 22 36 9F 24 35 20 E8 96 29 91 2E 27 10
C0C0  39 96 29 91 2E 26 09 39 96 2F 20 F0 96 2F 20 F3
C0D0  DE 22 08 08 DF 22 39 BD C2 97 7D 00 18 27 07 C6
C0E0  A1 D1 29 27 EB 39 C6 9E D1 29 27 D0 20 D5 96 29
C0F0  20 3B 96 29 9B 2E 20 35 8D 38 94 29 20 2F 96 2E

C100  D6 29 C4 0F 26 02 96 2F 5A 26 02 9A 2F 5A 26 02
C110  94 2F 5A 5A 26 0A 7F 00 3F 9B 2F 24 03 7C 00 3F
C120  5A 26 0A 7F 00 3F 90 2F 25 03 7C 00 3F DE 2A A7
C130  00 39 86 C0 97 2C 7C 00 2D DE 2C 96 0D AB 00 A8
C140  FF 97 0D 39 07 C1 79 0A C1 7D 15 C1 82 18 C1 85
C150  1E C1 89 29 C1 93 33 C1 DE 55 C1 FA 65 C2 04 CE
C160  C1 44 C6 09 A6 00 91 29 27 09 08 08 08 5A 26 F4
C170  7E C3 60 EE 01 96 2E 6E 00 96 20 20 B0 BD C2 C4
C180  20 AB 97 20 39 16 7E C2 E1 5F 9B 27 97 27 D9 26
C190  D7 26 39 CE C1 BC 84 0F 08 08 4A 2A FB EE 00 DF
C1A0  1E CE 00 08 DF 26 C6 05 96 1E 84 E0 A7 04 09 86
C1B0  03 79 00 1F 79 00 1E 4A 26 F7 5A 26 EB 39 F6 DF
C1C0  49 25 F3 9F E7 9F 3E D9 E7 CF F7 CF 24 9F F7 DF
C1D0  E7 DF B7 DF D7 DD F2 4F D6 DD F3 CF 93 4F DE 26
C1E0  C6 64 8D 06 C6 0A 8D 02 C6 01 D7 0E 5F 91 0E 25
C1F0  05 5C 90 0E 20 F7 E7 00 08 39 0F 9F 12 8E 00 2F

C200  DE 26 20 09 0F 9F 12 9E 26 34 CE 00 30 D6 2B C4
C210  0F 32 A7 00 08 7C 00 27 5A 2A F6 9E 12 0E 39 D6
C220  29 7F 00 3F DE 26 86 01 97 1C C4 0F 26 02 C6 10
C230  37 DF 14 A6 00 97 1E 7F 00 1F D6 2E C4 07 27 09
C240  74 00 1E 76 00 1F 5A 26 F5 D6 2E 8D 28 96 1E 8D
C250  15 D6 2E CB 08 8D 1E 96 1F 8D 0B 7C 00 2F DE 14
C260  08 33 5A 26 CB 39 16 E8 00 AA 00 E7 00 11 27 04
C270  86 01 97 3F 39 96 2F 84 1F 48 48 48 C4 3F 54 54
C280  54 1B 97 1D DE 1C 39 C6 F0 CE 80 10 6F 01 E7 00
C290  C6 06 E7 01 6F 00 39 8D EE 7F 00 18 8D 55 E6 00
C2A0  8D 15 97 17 C6 0F 8D E1 E6 00 54 54 54 54 8D 07
C2B0  48 48 9B 17 97 17 39 C1 0F 26 02 D7 18 86 FF 4C
C2C0  54 25 FC 39 DF 12 8D BF A6 01 2B 07 48 2A F9 6D
C2D0  00 20 07 8D C2 7D 00 18 26 EC 8D 03 DE 12 39 C6
C2E0  04 D7 21 C6 41 F7 80 12 7D 00 21 26 FB C6 01 F7
C2F0  80 12 39 8D 00 37 C6 C8 5A 01 26 FC 33 39 CE 80

C300  12 C6 3B E7 01 C6 7F E7 00 A7 01 C6 01 E7 00 39
C310  8D 13 A6 00 2B FC 8D DD C6 09 0D 69 00 46 8D D3
C320  5A 26 F7 20 17 DF 12 CE 80 12 39 8D F8 36 6A 00
C330  C6 0A 8D BF A7 00 0D 46 5A 26 F7 32 DE 12 39 20
C340  83 86 37 8D B9 DE 02 39 8D F7 A6 00 8D DD 08 9C
C350  04 26 F7 20 0B 8D EA 8D B7 A7 00 08 9C 04 26 F7
C360  8E 00 7F CE C3 E9 DF 00 86 3F 8D 92 8D 43 0E 8D
C370  CE 4D 2A 10 8D C9 84 03 27 23 4A 27 D8 4A 27 C8
C380  DE 06 6E 00 8D 0C 97 06 8D 06 97 07 8D 23 20 DF
C390  8D AD 48 48 48 48 97 0F 8D A5 9B 0F 39 8D 12 DE
C3A0  06 8D 25 8D 9A 4D 2B 04 8D E8 A7 00 08 DF 06 20
C3B0  EC 86 10 8D 2B CE 01 C8 86 FF BD C0 7D CE 00 06
C3C0  8D 06 08 8D 03 8D 15 39 A6 00 36 44 44 44 44 8D
C3D0  01 32 DF 12 BD C1 93 C6 05 BD C2 24 86 04 9B 2E
C3E0  97 2E 86 1A 97 2F DE 12 39 7A 00 20 7A 00 21 7D
C3F0  80 12 3B DE 00 6E 00 00 C3 F3 00 80 00 83 C3 60
```

TROUBLE-SHOOTING

In the unlikely event that your computer malfunctions, the cause must be either a constructional error or a faulty component. Therefore, proceed to double check the board. Inspect the solder side with a magnifying glass and if any tracks appear to be touching, scrape between them with a sharp pointed instrument. Remember to ground yourself and the board. Look for disoriented components, and incorrect values. Check that all links are present. From here on, it is assumed that the wiring is correct and that your power supply and video monitor are working properly.

The first step is to get the video display generator up. ICs 1, 2, 3, 4, 7 and 9 should be removed at this stage. First check the clock (IC8). There should be 1MHz square waves at pins 7, 13, and 15, and 2MHz at pin 5 (to VDG). Also check RST (pin 14) is high. If trouble, check that the crystal is oscillating (1MHz sine-wave at pin 2), using a x10 probe on your CRO. If not, try it without the L-C tank circuit (150pF/10uH). If no success, you have a bad crystal, or 6875.

If there appears to be some video output, but you can't get the picture to lock, the trouble is probably in your RF modulator. Try reducing the level of the video input signal to the modulator. Also, beware of harmonics; perhaps you have been trying to tune in to a spurious signal eminating from the thing.

Assuming the presence of a 2MHz clock signal, check for horizontal and vertical sync pulses (4us every 64us, and approx. 300us every 20ms, resp.). If no sync, check counter outputs (ICs 15, 14, 13, in that order). Vertical problems could also be caused by IC 13 not resetting or by a faulty one-shot (IC19b). There's not much else that can go wrong with the VDG itself, except when interacting with the MPU.

Having obtained a rock solid white rectangle display, the next step is to check operation with the processor. With all ICs installed, switch on (and reset) the system again. Press a few hex keys. Are the keystrokes being acknowledged with a bleep, but something incoherent is being displayed? If so, do the following, in order given:—

1. Check the LOAD pulse (IC7, pin 9); should be 500 to 800 nanosec, every 8 dot-clock cycles (4us).
2. Remove ICs 1, 2 and 9 (MPU, ROM, PIA); connect BA (IC11, pin 13) to +5V; proceed to check the DMA address bus (outputs of buffers, IC5, 6). The signals should be the same as the respective inputs. Now remove the +5V connection to BA. The outputs should no longer follow the inputs, but "float". If any of the Tristate buffers appear to be faulty, replace it.
3. Re-insert all ICs previously removed. The screen should show RAM contents, usually some kind of vaguely ordered pattern, or random dots. Try grounding WE (IC3, pin 10) momentarily with a jumper lead, a few times, while the system is running. The display should change each time. If not, suspect the 4014 (IC7).
4. Finally, the least likely cause of the above symtoms is a bad EPROM.

At this stage, we are assuming that the video is behaving itself, but a processor malfunction is suspected. With all chips on board, press the [RST] key. The speaker should bleep when the key is held down (even if the PIA is at fault), and the RST line (IC8, pin 14) should go LOW momentarily. If not, check the 2.2uF tantalum capacitor and RST wiring. Note that the Reset function is performed by the 6875. If the system does not appear to be resetting, you could have a faulty EPROM, RAM, MPU or PIA!

Check that the MPU address lines are all HIGH (except A0) when RST is LOW (hold down [RST]). Before trying a new MPU chip, note that any faulty device on the address bus might be holding a bus line LOW (incl. ICs 5, 6).

Assuming the actual reset circuit is operating, but CHIPOS refuses to spring to life (i.e. no 4-digit readout on screen, or no keypad response), the fault is almost certainly in EPROM or RAM, or the associated select logic. Less likely is a bad PIA, but this can be checked. If you have a good display, but no I/O response, check the PIA initialization. After resetting, PB0 is HIGH, PB1-PB7 are all LOW, PA0-PA3 HIGH, and PA4-PA7 LOW. When a hex key is pressed, the PAx lines will reverse momentarily, if CHIPOS and the PIA are both operating.

If you have an acquaintance who is also constructing a DREAM-6800, see if you can arrange to borrow the MPU, RAM, EPROM, and PIA chips. One by one, substitute a chip for one of your own.

In conclusion, it must be said that, provided due care is taken in construction, the probability of success at switch-on is very high. Readers who are contemplating the project, should not be put off by the trouble-shooting section, which was included to help isolate rare, hard-to-find bugs. Problems of a minor nature should be able to be handled by enthusiasts with a moderate amount of experience, with the help of the theory-of-operation section.

PARTS LIST

HARDWARE
- *1 PC board, 244 x 142mm*
- *1 4.000MHz crystal*
- *1 hexadecimal keypad (4 x 4 matrix)*
- *2 momentary-contact pushbuttons*
- *2 RCA phono sockets*
- *1 small loudspeaker*
- *1 10uH inductor*

SEMICONDUCTORS
- *1 6800 microprocessor*
- *1 6821 peripheral interface adaptor*
- *1 6875 clock generator*
- *1 2708 EPROM (programmed with CHIPOS)*
- *2 2114 static RAMs*
- *1 4040B CMOS counter/divider*
- *1 4014B CMOS static shift register*
- *2 74LS04 hex inverter*
- *1 74LS08 quad 2-input gate*
- *1 74LS10 triple 3-input gate*
- *1 74LS11 triple 3-input gate*
- *1 74LS20 dual 4-input gate*
- *1 7440 dual 4-input buffer*
- *1 7474, 74LS74 dual D-flipflop*
- *2 7493, 74LS93 binary counter*
- *1 74121 one-shot multivibrator*
- *1 74123 dual one-shot*
- *2 74LS367 Tristate buffer*
- *1 566 function generator*
- *1 741 operational amplifier*
- *1 2N3643 NPN transistor*
- *1 2N4250 PNP transistor*
- *6 1N4148 silicon diodes*

IC Sockets
- *2 40 pin*
- *1 24 pin*
- *2 18 pin*
- *3 16 pin*
- *1 16 pin DIL plug*

CAPACITORS
- *2 10uF/16VW aluminium electrolytic*
- *1 10uF/16VW tantalum electrolytic*
- *1 2.2uF tantalum electrolytic*
- *14 0.1uF polyester or ceramic*
- *1 .033uF polyester*
- *1 .022uF polyester*
- *1 .01uF polyester*
- *1 .0033uF polyester*
- *1 .001uF polyester*
- *2 150pF ceramic*
- *1 47pF ceramic or polystyrene*

RESISTORS
(¼W, 10% tolerance)
2 x 22k, 5 x 10k, 1 x 6.8k, 1 x 5.6k, 2 x 4.7k, 6 x 2.2k, 3 x 1.5k, 1 x 1k, 2 x 470 ohms, 2 x 220 ohms, 1 x 120 ohms, 1 x 75 ohms (or 2 x 150 ohms), 1 x 47 ohms.
1 5k trimpot (vertical mounting)

MISCELLANEOUS
Ribbon cable, tipped copper wire, spaghetti sleeving, shielded cable, PC pins, 22g solder, 3 extra DIL plugs and sockets (if required for expansions).

Designed especially for beginners . . .

DREAM 6800

3rd article has interesting programs

In this third article in the series on Michael Bauer's innovative design for the DREAM 6800 computer, we described how to connect it to a TV set and provide listings and instruction for sample programs. Next article in the series will give the lowdown on CHIP-8 programming.

STOP PRESS

PCB suppliers such as RCS Radio Pty Ltd have indicated that their flux-coated PCB should not be scrubbed with steel wool and soap. The author's remarks apply only to non-flux-coated PCB's. For the many who have asked, the PCB pattern will be published in the August issue. We also hope to give a solution to the looming shortage of 6875 clock chips.

Last month's article which gave complete construction details for the DREAM 6800, made only a brief mention of the video connection from the DREAM. The easiest way to connect the system to your TV set is to use an RF modulator (strictly speaking, this should be called a video-modulated RF oscillator), which enables a simple connection to the TV set antenna terminals.

You may be able to wreck a defunct video game and use its modulator. While the modulators used in most video games do not provide very sharp resolution, they are quite suitable for the chunky graphics display of the DREAM. You can also purchase a suitable modulator complete with instructions, from Dick Smith Stores. Catalog price is $3.00.

Our approach was to make a direct video connection to the TV set. With the DREAM 6800, the method of video connection is less critical than for the usual "glass terminal" which uses the full screen and has small alpha-numeric characters. The fact that the DREAM uses a rectangle in the centre of the screen means that its relatively simple sync pulse "trains" will not cause "flag-waving" (horizontal jitter) at the top of the screen. The centrally located rectangular display also takes advantage of the superior linearity available in this area of the screen. So even old valve sets with quite poor linearity will give a reasonable display of the DREAM graphics.

The other reason why the DREAM is relatively non-critical of the method of video connection is that the chunky graphic display does not require as wide a picture bandwidth as a normal computer's alpha-numeric display. This means there is no need to improve the picture bandwidth by removing sound traps or other modifications.

Even so, the use of a direct video connection gives a quite worthwhile improvement in picture quality compared with that available via a RF modulator. And there is also less chance of interference to other TV sets in the near vicinity.

Our approach is to connect the video output from the DREAM to the input of the video amplifier in the TV set; ie, immediately after the video detector. If you have access to the circuit diagram of the set you should be able to find the appropriate spot in the circuit without any trouble. Ideally, the circuit will also show the shape and amplitude of the composite sync/video waveform which is normally present at the input to the video amplifier stage.

For example, in a small valve portable TV set we modified for this purpose, the composite sync/video waveform is normally 2 volts peak-to-peak with positive video and negative sync. This is in the right ball-park for the DREAM, which has a composite sync/video amplitude of 1 volt peak-to-peak. All that we did was to connect the video from the DREAM via a 100uF/16VW electrolytic capacitor to the grid of the video amplifier valve.

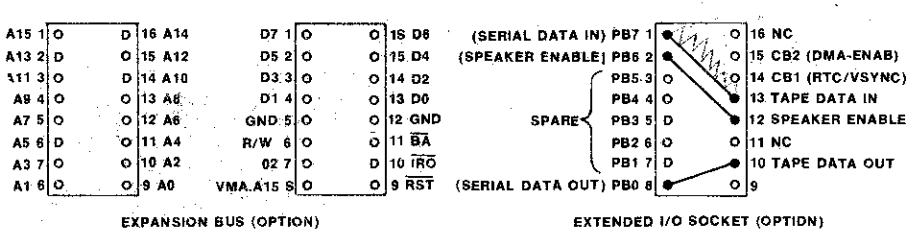
Much the same approach applies to solid state sets. Find the video detector and check the video waveform. Provided its polarity is correct and the amplitude is in the ball-park, you can feed the DREAM video signal into the base of the following video amplifier stage via a 100uF capacitor, as before.

The TV set tuner is set to an unused channel. This means that no video modulation is present from within the set. The DREAM video signal will swamp the noise to produce a sharp display.

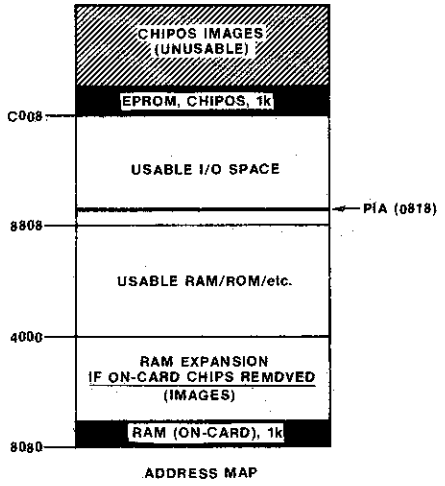
By suitably adjusting the brightness and contrast controls, a bright and steady display is obtained. The polarity of the electrolytic coupling capacitor must be correct and it must have low leakage to avoid upsetting the bias of the following stage.

All the foregoing assumes that you have a set with earthed chassis and transformer isolation from the mains supply. If not, you will just have to use an RF modulator.

Some other sets which have a separate sync detector will not be suitable for the above method of video



Held over from last month, this diagram shows the leadouts for the output sockets on the PCB.



ADDRESS MAP

DEVICE	BASE ADDRESS	ADDRESS LINES															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RAM (ON-CARD)	0808	0	0	–	–	–	–	*	*	*	*	*	*	*	*	*	*
RAM (EXT)†	0000	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*
ANYTHING	4080	0	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*
PIA (ON-CARD)	8818	1	0										1	–	–	*	*
EXT. I/O DEVICES 1	8020	1	0									1		*	*	*	*
EXT. I/O DEVICES 2	8840	1	0								1			*	*	*	*
EXT. I/O DEVICES 3	8880	1	0							1				*	*	*	*
EXT. I/O DEVICES 4	8100	1	0						1					*	*	*	*
EXT. I/O DEVICES 5	8208	1	0					1						*	*	*	*
EXT. I/O DEVICES 6	8408	1	0				1							*	*	*	*
CHIPOS EPROM	C808	1	1	–	–	–	–	*	*	*	*	*	*	*	*	*	*

DEVICE-SELECT TABLE

KEY

0	LINE MUST BE LOW TO SELECT DEVICE
1	LINE MUST BE HIGH TO SELECT DEVICE
*	LINE IS DECODED BY DEVICE TO SELECT REQUIRED BYTE; NOTE: I/O DEVICES MAY HAVE UP TO 16 ADDRESSABLE 1-BYTE REGISTERS
–	IRRELEVANT (DONT CARE)
(BLANK)	LINE IS NOT DECODED, BUT MUST BE LOW TO AVOID BUS CONTENTION
†	IF EXTERNAL RAM IS PUT AT 8888, THEN ON CARD RAM MUST BE REMOVED OR RELOCATED (E.G. TO 4808);

Memory map for the Dream 6800.

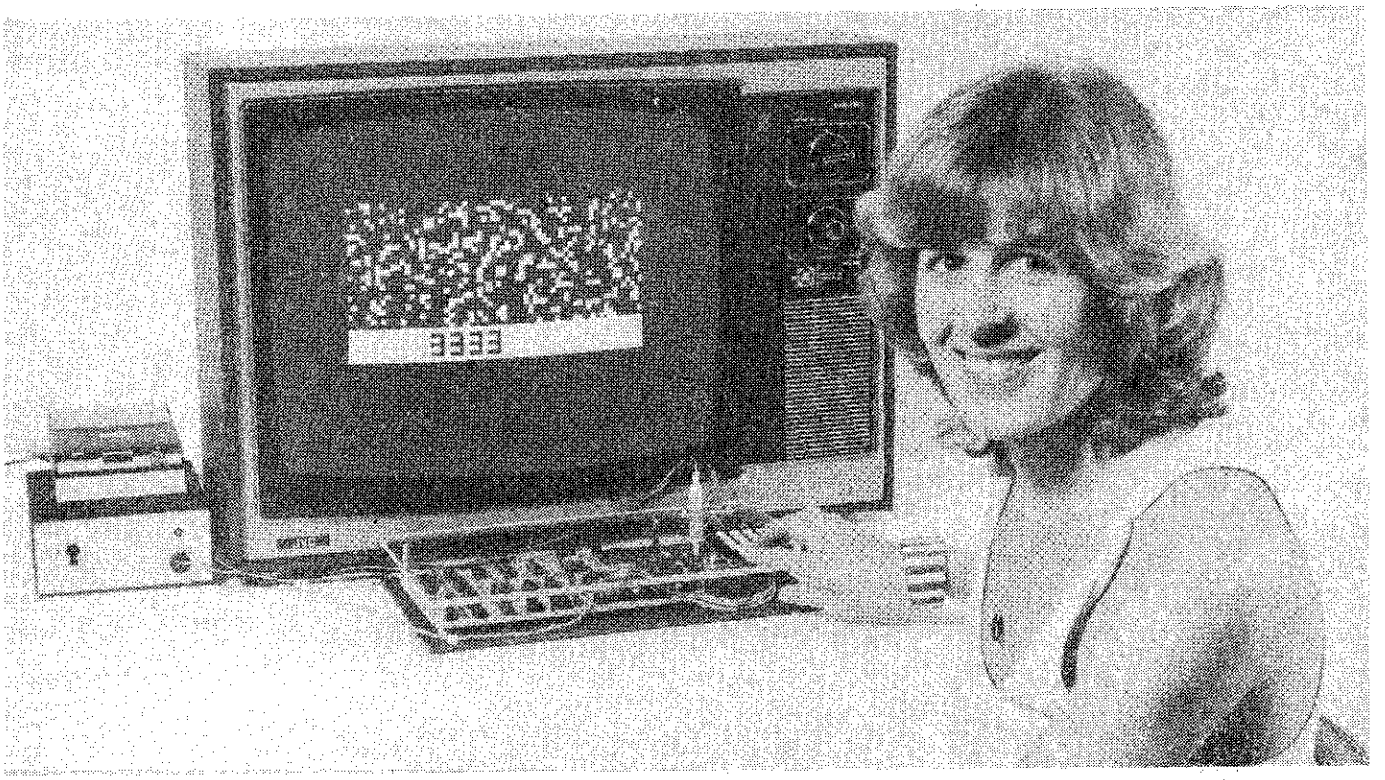
connection. In these cases it may be possible to connect the sync and video from the DREAM separately, rather than use the composite waveform.

It is possible that the polarity of the video waveform within your set is reversed to that from the DREAM. This will result in poor or incorrect picture sync and a negative (ie, reversed) picture. The solution in this case is to build a single-stage common-emitter amplifier which will provide the necessary waveform polarity reversal.

Finally, if you propose to use an old set for which no circuit diagram is available, it is usually possible to identify the video amplifier relatively quickly. Just take note of the single wire from the picture tube socket which is the video output. Trace this back to the appropriate valve. From there it should be easy to identify the grid. This can be done by measuring voltages — the grid will usually be a few volts negative with respect to chassis.

The same approach would apply to solid state black and white TV sets. The video output transistor can be found by tracing the video output lead to the picture tube, back to its source. From there it's a matter of identifying the base of the transistor and then feeding the signal in via a 100uF capacitor, as before.

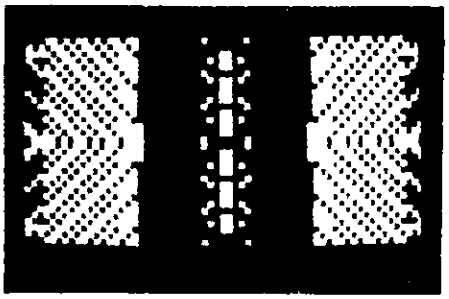
Well now you should be champing at the bit to get some programs entered and running. Enter each program in the following sequence and, as soon as you have it running, dump it on cassette. There is nothing so boring as having to enter the same hex listing twice! So make sure you dump all your programs onto tape. Note: Of the following programs, "Block Puzzle" and "TV Typewriter" were written by M. J. Bauer while the others were adapted from the "RCA Cosmac VIP" Instruction Manual.



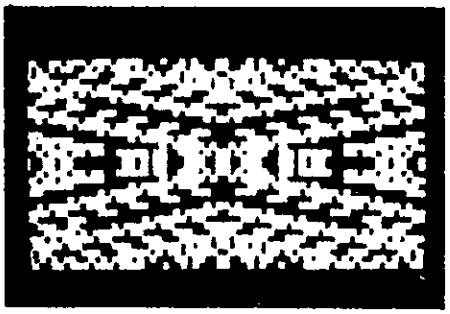
Repeated from the first article in May 1979, this photograph shows the TV displaying the random number generator after the program has been stopped. The "3333" address does not normally occur but was typed in to give good digit display (when this photograph was taken, our prototype was blurring some digits becase of a low spec 4014B IC.)

KALEIDOSCOPE

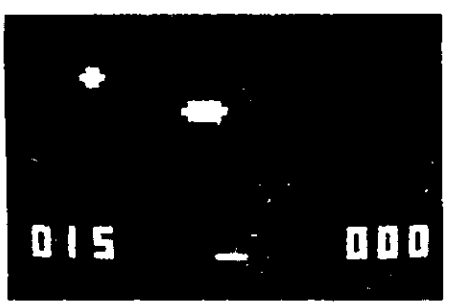
Use keys 2,4,6 and 8 to enter a short sequence of movements. then press key 0 and watch the computer repeat the sequence to create a moving, symmetrical pattern. Try 44444442220, then experiment with other nice patterns.



Just two of the many Kaleidoscope patterns.



```
0200  6000 6380 611F 620F
0208  2232 A200 F31E F00A
0210  F055 4000 121C 7301
0218  3300 1208 6380 A200
0220  F31E F065 4000 121C
0228  7301 4300 121C 2232
0230  121E 4002 72FF 4004
0238  71FF 4006 7101 4008
0240  7201 A277 6AE0 8A12
0248  6B1F 81B2 3A00 7201
0250  6AF0 8A22 6B0F 82B2
0258  3A00 7101 6B1F 81B2
0260  D121 8A10 6B1F 8B25
0268  DAB1 6A3F 8A15 DAB1
0270  8B20 DAB1 00EE 0180
0278  0000
```

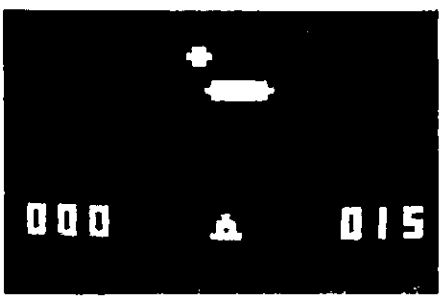


Start of the UFO Intercept game.

UFO INTERCEPT

Launch a missile with key 4, 5 or 6 (left, up, right). Hit the small UFO to score 15, the big one to score 5. You have 15 shots.

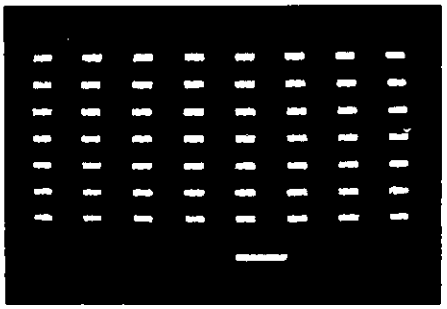
Two more views of the UFO Intercept game.



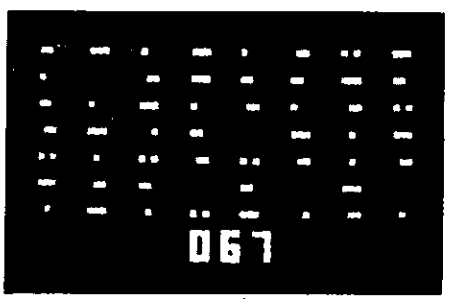

```
0200  A2CD 6938 6A08 D9A3
0208  A2D0 6B00 6C03 DBC3
0210  A2D6 641D 651F D451
0218  6700 680F 22A2 22AC
0220  4800 1222 641E 651C
0228  A2D3 D453 6E00 6680
0230  6D04 EDA1 66FF 6D05
0238  EDA1 6600 6D06 EDA1
0240  6601 3680 22D8 A2D0
0248  DBC3 CD01 8BD4 DBC3
0250  3F00 1292 A2CD D9A3
0258  CD01 3D00 6DFF 79FE
0260  D9A3 3F00 128C 4E00
0268  122E A2D3 D453 4500
0270  1286 75FF 8464 D453
0278  3F01 1246 6D08 8D52
0280  4D08 128C 1292 22AC
0288  78FF 121E 22A2 7705
0290  1296 22A2 770F 22A2
0298  6D03 FD18 A2D3 D453
02A0  1286 A2E8 F733 6300
02A8  22B6 00EE A2F8 F833
02B0  6332 22B6 00EE 6D1B
02B8  F265 F029 D3D5 7305
02C0  F129 D3D5 7305 F229
02C8  D3D5 00EE 017C FE7C
02D0  60F0 6040 E0A0 F8D4
02D8  6E01 6D10 FD18 00EE
```

WIPE OFF

Use keys 4 and 6 to serve ball and move bat. Score is shown at end of game, after 20 balls. For smaller bat, change data at 1oc. 02CD to FO. For a bat with a hole, use E7!



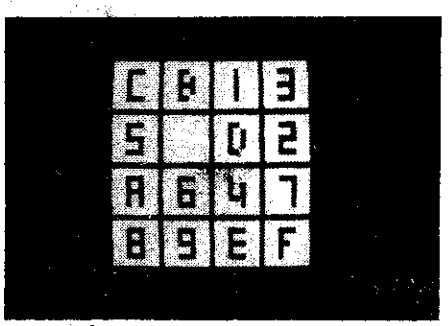
Start and finish of the game of Wipe-Off.



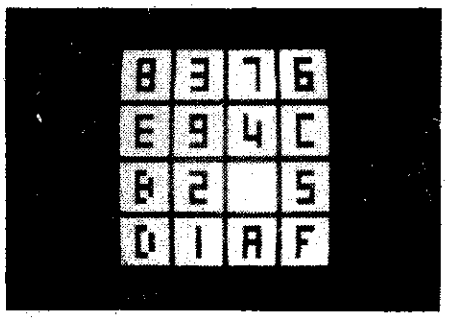

```
0200  A2CC 6A07 6100 6B08
0208  6000 D011 7008 7BFF
0210  3B00 120A 7104 7AFF
0218  3A00 1206 6600 6714
0220  A2CD 6020 611E D011
0228  631D 623F 8202 77FF
0230  4700 12AA FF0A A2CB
0238  D231 65FF C401 3401
0240  64FF A2CD 6C00 6E04
0248  EEA1 6CFF 6E06 EEA1
0250  6C01 D011 80C4 D011
0258  4F01 1298 4200 6401
0260  423F 64FF 4300 6501
0268  431F 12A4 A2CB D231
0270  8244 8354 D231 3F01
0278  1242 431E 1298 6A02
0280  FA18 7601 4670 12AA
0288  D231 C401 3401 64FF
0290  C501 3501 65FF 1242
0298  6A03 FA18 A2CB D231
02A0  73FF 1236 A2CB D231
02A8  1228 A2CD D011 A2E0
02B0  F633 F265 6318 641B
02B8  F029 D345 7305 F129
02C0  D345 7305 F229 D345
02C8  12C8 0180 44FF
```

BLOCK PUZZLE

The screen shows a 4 x 4 board with symbols 0 to 9, A to F arranged in order, with a blank square upper left. Watch the computer jumble the blocks, then you try to re-order them using keys 2 (down), 4 (left), 6 (right) and 8 (up) to move a block into the blank space.



Each start for the Block Puzzle is different.



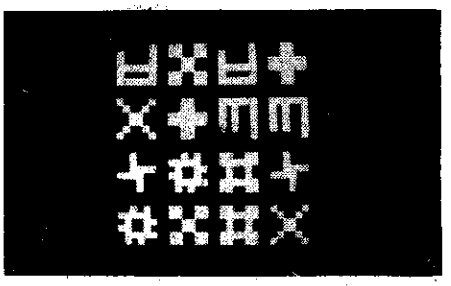
```
0200  6A12 6B01 6110 6200
0208  6000 A2B0 D127 F029
0210  3000 DAB5 7108 7A08
0218  3130 1224 6110 7208
0220  6A12 7B08 A300 F01E
0228  F055 7001 3010 120A
0230  6A12 6B01 6C00 62FF
0238  C006 7002 2252 72FF
0240  3200 1238 6E00 0000
0248  F00A 2252 7E01 0000
0250  1248 84A0 85B0 86C0
0258  3002 1264 4501 1264
0260  75F8 76FC 3008 1270
0268  4519 1270 7508 7604
0270  3006 127C 4412 127C
0278  74F8 76FF 3004 1288
0280  442A 1288 7408 7601
0288  A300 F61E F065 8100
0290  6000 A300 F61E F055
0298  A300 FC1E 8010 F055
02A0  F129 D455 DAB5 8A40
02A8  8B50 8C60 00EE EE5E
02B0  FEFE FEFE FEFE FEFE
```

CONCENTRATION

Two players, A and B, take turns to find matching pairs of symbols arranged in a 4 x 4 matrix. The hex keys correspond to board positions, so just press a key to see the symbol there. When player A gets a match, the computer replaces his two symbols with A's, and same for B, so you can see who won. The computer also shows whose turn it is to play.

```
0200  A385 6002 6102 6202
0208  6302 6402 6502 6602
0210  6702 F755 6300 A385
0218  C107 F11E F065 4000
0220  1216 70FF A385 F11E
0228  F055 A38E F31E 8010
0230  F055 7301 3310 1216
0238  2314 C501 22C4 6B00
0240  6D10 F00A A375 F01E
0248  F065 90D0 1242 8D00
0250  22D8 3B00 125E 6B0F
0258  8CD0 89A0 1242 6020
0260  F015 F007 3000 1262
0268  99A0 1278 22C4 7501
0270  6001 8502 22A0 123C
0278  6020 F018 7E01 22A0
0280  A385 FA1E 60DD F055
0288  4500 1296 A367 D346
0290  A367 D126 12B8 A33F
0298  D346 A33F D126 12B8
02A0  22D8 8130 8240 8DC0
02A8  22D8 00EE A36D FA1E
02B0  F065 A334 F01E 00EE
02B8  0000 123E 22C4 6060
02C0  F018 12C2 6300 6408
02C8  A33F 4500 12D2 633A
02D0  A367 D346 00EE 5555
02D8  A38E FD1E F065 8A00
02E0  A385 F01E F065 40DD
02E8  1242 22AC 6310 6400
02F0  600C 80D2 4004 6408
02F8  4008 6410 400C 6418

0300  6003 80D2 4001 6318
0308  4002 6320 4003 6328
0310  D346 00EE 2324 6040
0318  F015 F007 3000 131A
0320  2324 00EE 6D00 22D8
0328  7D01 4D10 1330 1326
0330  00EE 0101 1010 1E78
0338  0808 1818 7E7E 1818
0340  2424 3C24 2466 6618
0348  1866 667E 2424 7E66
0350  4224 1818 2442 7E52
0358  5252 527E 4242 7E42
0360  7E14 7C26 643E 287C
0368  243C 2424 7C00 0611
0370  161C 2227 2D0C 0D0E
0378  0F08 090A 0B04 0506
0380  0700 0102 0300
```



This is the start of the Concentration game.

SECRET NUMBER

The computer is thinking of a secret (random), 3-digit, decimal number. You try to guess what it is, with the help of "clues" from the computer. Simply enter your 3-digit guess, shown upper left. The computer's clue, shown bottom left (momentarily), is a number calculated as follows: starts with 0; adds 2 for each correctly guessed digit in the correct position; then adds 1 for each guess digit which is present in the secret number but in the wrong place. The number of tries you took is shown bottom right.

```
0200  6E00 A3F0 22A0 22A0
0208  22A0 6500 6000 6100
0210  6200 F255 22AE 6534
0218  22D0 A3F6 22E2 22E2
0220  22E2 6500 22AE A3F6
0228  F265 A3F3 F255 6500
0230  22AE 6402 6D00 A3F3
0238  22F4 A3F3 F255 8500
0240  A3F0 22F4 A3F0 F255
0248  9500 1300 9510 1252
0250  9520 7D01 4400 125C
0258  74FF 1236 6508 22D0
0260  6534 22D0 7E01 6534
0268  22D0 4D06 1288 4E63
0270  1282 6130 F115 F107
0278  3100 1276 6508 22D0
0280  121A A3F0 652C 22AE
0288  6108 6002 F018 6F10
0290  71FF FF15 FF07 3F00
0298  1294 3100 128A 0C00
02A0  6409 C00F 8405 4F00
02A8  12A0 F055 00EE 6600
02B0  3500 12C6 A3F3 F265
02B8  F029 22CA F129 22CA
02C0  F229 22CA 00EE A3F0
02C8  12B6 D565 7508 00EE
02D0  6618 3508 12DA FD29
02D8  12CA A3F6 FE33 F265
02E0  12BC F00A 400F 1282
02E8  6109 8105 4F00 12E2
02F0  F055 00EE F265 8300
02F8  B010 8120 8230 00EE
0300  7D02 1254
```

T.V. TYPEWRITER

Starts with cleared screen and a cursor in the upper LHS corner. Enter a 2-digit number (character code) from 00 to 2F (total of 48 codes), noting the characters produced by each. The cursor can be moved by entering a 2-digit control code. The first digit specifies the direction: C (left), D (down), E (up), F (right). The second digit specifies how many dot positions to move. A mistake can be erased by positioning the cursor on top of the offending character and re-keying its code.

```
0200  6A00 6B00 602F 0266
0208  DAB5 F00A 0277 8100
0210  F00A 8101 602F 0266
0218  DAB5 8010 62C0 8122
0220  41C0 1234 4016 1250
0228  4020 1254 0266 DAB5
0230  7A04 1204 8200 64F0
0238  8242 640F 8042 42C0
0240  8A05 42D0 8B04 42E0
0248  8B05 42F0 8A04 1204
0250  A25C 1256 A25D DAB5
0258  7A06 1204 F8A8 A8A8
0260  A850 0000 0000 9630
0268  810F 2203 7EC1 9380
0270  10CE 027E 7EC1 9896
0278  3048 4848 4897 3039
0280  F6CE B7DA E92E F492
0288  B75A F248 B7FA B6DE
0290  F6DE 93DE 5EDE BBDE
0298  C546 492E F6DA 56DA
02A0  BFDA B55A 4BDA F11E
02A8  0024 2A22 88A8 8000
02B0  0BA0 0380 1550 1110
02B8  0820 1C70 419E FFFE
```

TANK BATTLE

Use keys 4 (left), 9 (up), 6 (right) and 1 (down) to move your tank about. Fire a shot with key F. Hit the randomly moving hostile enemy vehicle and you score 10 points. If you allow the target to collide with your tank, 5 shots will be forfeited. After each round, the score (left) and remaining number of shots are shown.

```
0080  76FB 6020 8065 4F00
0088  6600 1354

0200  6E00 6DA0 6A01 6906
0208  6804 6709 6619 6410
0210  630C 6200 6106 A092
0218  FA55 23A4 6040 F015
0220  F007 3000 1220 23A4
0228  22DA 2332 A092 F565
0230  227E 2296 22BC 3F01
0238  22E4 3F01 22BC 3F01
0240  22BC 3F01 224C 4F01
0248  1336 1232 A092 F565
0250  4600 3500 1258 135C
0258  E7A1 6209 E8A1 6204
0260  E9A1 6206 EAA1 6201
0268  4200 00EE 227E 8120
0270  236A 237C 6C01 6200
0278  6F00 A092 F555 A3CF
0280  4109 6000 4104 6013
0288  4106 600D 4101 6006
0290  F01E D347 00EE 600F
0298  E09E 00EE 450F 00EE
02A0  650F 76FF A092 F555
02A8  7403 7303 236A 236A
02B0  236A A0A3 F555 A3E9
02B8  D341 00EE A0A3 F565
02C0  4500 00EE A3E9 D341
02C8  236A 6C02 238E 4BBB
02D0  12DA D341 A0A3 F555
02D8  00EE 6500 6000 A097
02E0  F055 12D4 A09D F565
02E8  350F 1314 A3EA D345
02F0  3200 1302 C103 A099
02F8  F11E F065 8100 C20F
0300  7201 236A A3EA 6C03
0308  72FF 6F00 D345 A09D
0310  F555 00EE C407 A3EF
0318  F41E F065 8300 A3F7
0320  F41E F065 8400 A3EA
0328  D345 6020 F018 650F
0330  130E 6500 130E 4C01
0338  1080 4C02 1352 A0A3
0340  F565 4500 1080 A3E9
034B  D341 6F00 D341 3F01
0350  1080 7E0A 6040 F018
0358  00E0 121A 00E0 23A4
0360  6060 F018 1364 6E00
0368  1354 4109 74FF 4104
0370  73FF 4106 7301 4101
0378  7401 00EE 4400 7401
0380  4300 7301 4338 73FF
03B8  4418 74FF 00EE 6B00
0390  4400 139E 4300 139E
0398  433F 139E 441F 6BBB
03A0  6F00 00EE 6308 6408
03A8  A0A9 FE33 F265 23BC
03B0  6328 A0A9 F633 F265
03B8  23C2 00EE F029 D345
03C0  7306 F129 D345 7306
03C8  F229 D345 00EE 0110
03D0  547C 6C7C 7C44 7C7C
03D8  6C7C 5410 00FC 786E
03E0  78FC 003F 1E76 1E3F
03E8  0080 A870 F870 A80B
03F0  1B28 3830 2010 0000
03F8  0000 081B 1B1B 13D4
```

4th article gives the lowdown on

Chip-8 programming for the DREAM 6800 computer

In this fourth article on the DREAM 6800 the author gives hints on CHIP-8 programming. Also featured is a substitute circuit for the 6875 clock chip using low cost TTL devices and the full size artwork for the PCB. A future article will give details of RAM expansion.

by MICHAEL J. BAUER

After a while, when the provided video games become a bit of a yawn, you will want to write your own programs. There is no language as powerful as CHIP-8 which can be learned with such ease. The function of most of the instructions can be understood from the table, but some need further explanation. First, it might be an idea to re-read the CHIP-8 summary given in the May article.

The display instruction (DXYN) is the most important. It treats the screen as a coordinate grid of dots, numbered from 0 to 63 (00-3F hex) from left to right across the screen, and from 0 to 31 (00-1F hex) from top to bottom. Two variables of your choice are used to specify the coordinates of a symbol to be displayed. The symbol may be any size up to 8 dots across by 16 dots down. Larger symbols may be shown by using more than one DXYN instruction, possibly in a loop. Various symbols are defined by making up a pattern of bytes and storing this data along with the program. As an example, let us say we want to show an "X", 7 x 7 dots in size. Thus, N is 7. The screen coordinates we will choose to be variables VA and VB, i.e. X = A and Y = B. Thus the instruction will be DAB7. But how does the interpreter know where to find our symbol pattern? A special index variable, called "I", can be set to point to anywhere in the bottom 4k of memory, using an AMMM instruction. Let us put our pattern at location 0210 onwards, thus:-

Address	Binary Data	Hex Data
0210	1000 0010	82
0211	0100 0100	44
0212	0010 1000	28
0213	0001 0000	10
0214	0010 1000	28
0215	0100 0100	44
0216	1000 0010	82

To display this pattern in the upper left hand corner of the screen, we would initialize variables VA and VB to zero, and set I = 210. Note, if N = 0, a 16 byte pattern will result. The program, with comments, is shown below:-

0200	6A00	VA = 00	Set coordinates
0202	6B00	VB = 00	
0204	A210	I = 210	Set pointer
0206	DAB7	SHOW 7@VA,VB	Show 7-byte pattern
0208	F000	STOP	Jump to monitor
020A			
020C			
020E			
0210	8244	DATA	Pattern for "X"
0212	2810		
0214	2844		
0216	8200		

Note that the first CHIP-8 instruction must be at 0200. The program is executed by a GO from C000, which is the interpreter's starting address. Try setting VA and VB to different starting values, then re-run the program. Note that these values specify the position of the upper LH corner of the symbol.

An important feature of the SHOW instruction is that if a symbol is displayed and it overlaps another symbol already there, then the overlapping spots are erased and variable VF (the "flag" variable) is set to 01. If no overlap, VF = 00. This feature can be used to erase a symbol, by showing it again at the same coordinates, without erasing the whole screen (which can be done with a 00E0). Of course, you have to keep track of the positions of each different symbol used in this way. Variable VF can be used to see if two objects collided, in an animated game. An object can be made to move about on the screen by erasing it and re-showing it in a new position each time.

This program illustrates:-

```
0200  6A00  VA=00
0202  6B00  VB=00
0204  A210  I=210
0206  DAB7  SHOW 7@VA,VB
0208  DAB7  SHOW 7@VA,VB
020A  7A01  VA=VA+01
020C  7B02  VB=VB+02
020E  1206  GOTO 206
0210  8244  DATA
0212  2810
0214  2844
0216  8200
```

The speed and direction of motion can be manipulated with the instructions at 20A and 20C by changing the values which are added. Note that adding FF is the same as subtracting 01; (refer to a text on "two's complement" arithmetic). The motion can be slowed down by putting a time delay inside the loop.

The random byte generator in CHIPOS is unique, and achieves longer sequences and higher randomisation than conventional software pseudo-random sequencers by utilising the fact the program bytes are "kind-of" random. In a VX=RND.KK instruction (CXKK), a variable is set to a random value which has been masked by (i.e. ANDed with) a constant (KK). Thus, random numbers covering a specified range, and falling into precise intervals, can be selected. For example, a C01E instruction will give only even numbers in the range 0 to 30 (00-1E hex).

CHIPOS has built-in patterns for the symbols 0 to 9 and A to F, and CHIP-8 provides an instruction to allow you to display the contents of any variable as a hex digit. Only the least significant 4

bits are relevant. For example, to show the hex value of V6, we would use an F629 followed by a DXY5, where X and Y are again arbitrary. The F629 sets up I to point to the symbol corresponding to the value of V6 (LSD).

Another useful instruction, FX33, lets you find the 3-digit decimal equivalent of any variable; e.g: F433 would store 3 bytes in memory, at the location specified in I. To display this 3-digit number, you will need to be familiar with another pair of instructions: FX55 and FX65.

FX55 takes the values of variables V0 up to VX (incl.) and stores them in successive memory locations, indexed by I. FX65 does the reverse, i.e. re-loads the variables from memory. These powerful instructions not only extend the number of available variables, but also let you perform array processing. Note that the pointer (I) auto-increments with these two instructions, i.e. I is advanced by the number of variables stored or loaded (X+1), provided that a page boundary is not crossed. (A "page" is 256 bytes.) Also note that if X=0, only one variable (V0) is affected.

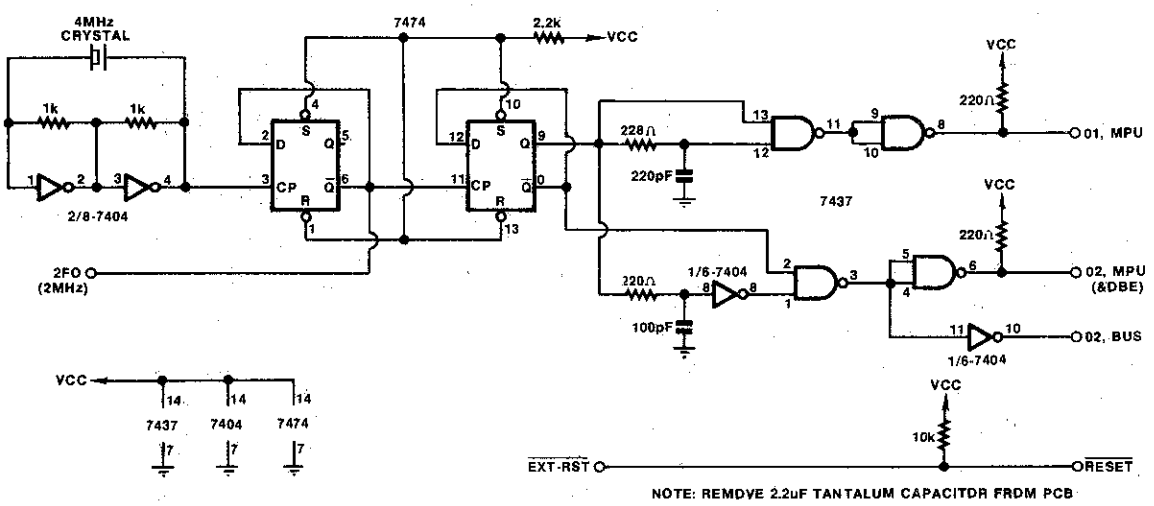
We can therefore use F256 to load variables V0, V1 and V2 from memory at I. If we had previously used an F433 to store the 3-digit decimal equivalent of V4 in memory at I, then V0, V1 and V2 would now contain the "hundreds", "tens", and "units" (resp.) of the value of V4. These can be displayed with the FX29 and DXYN instructions, as explained. While this may be confusing at first, it makes for a very versatile language, as you will come to appreciate.

A small part of a program, called a subroutine, can be accessed several times from different parts of a larger program. Further, a subroutine can "call" other subroutines (known as subroutine nesting). Each subroutine must end with a RETURN statement (00EE) so that, upon completion, control will return to the instruction following the particular "DO" statement (2MMM) that called it.

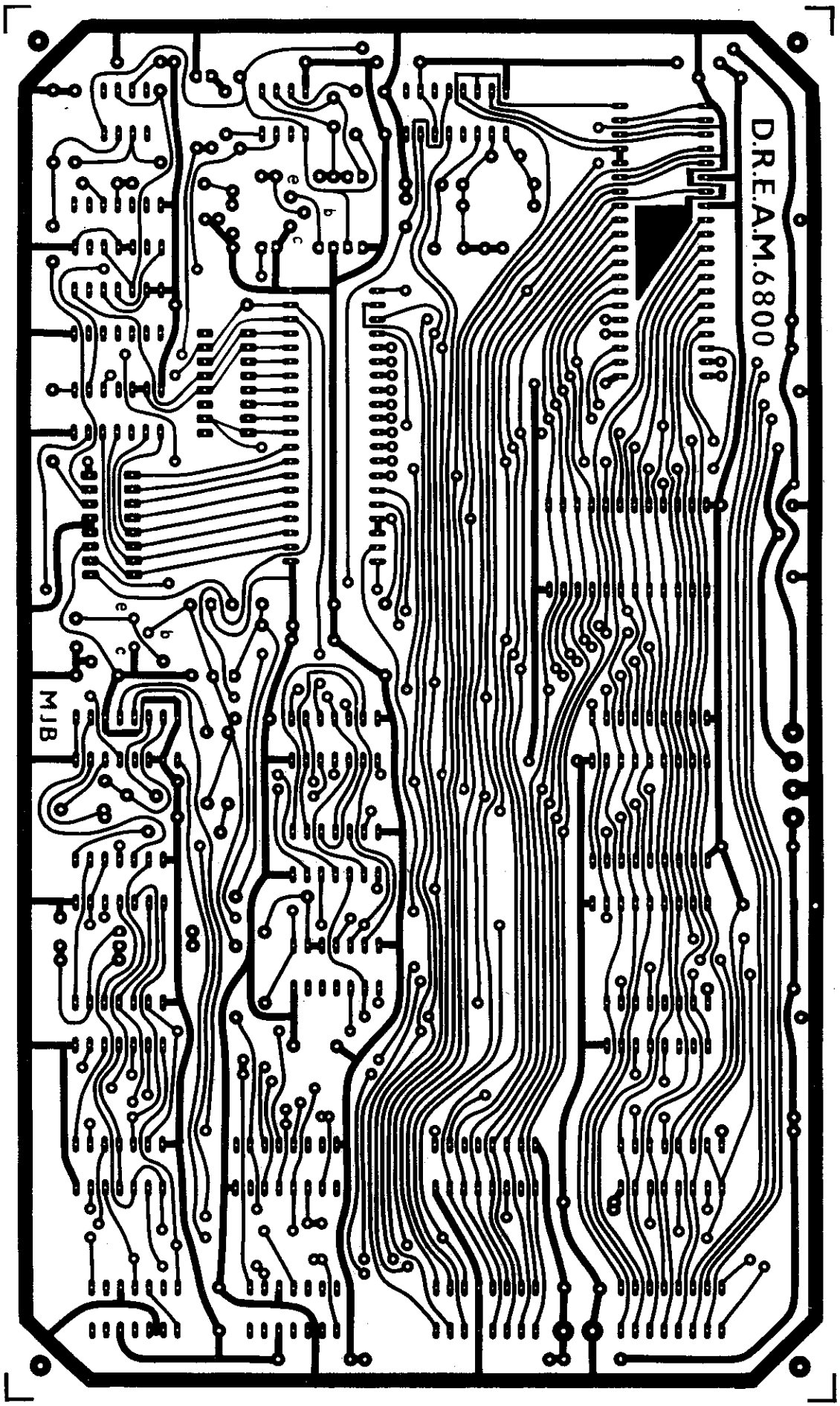
Everything discussed in the last few paragraphs is illustrated in the following program, plus the use of the timer and tone instructions, so you can see how easy it is. Have a close look, and try to understand its workings. The program counts down V3, converting it to decimal, showing it, and bleeping, every second.

```
0200  63FA  V3=FA          start counter at 250
0202  A240  I=240          point to workspace
0204  F333  MI=DEQ,V3      store dec. eq. of V3
0206  F265  VO:V2=MI       Load same into V0:V2
0208  6418  V4=18          set display position
021A  6510  V5=10
021C  F029  I=DSP,V0       display V2
021E  222C  DO 22C
0210  F129  I=DSP,V1       display V0
0212  222C  DO 22C
0214  F229  I=DSP,V2       display V1
0216  222C  DO 22C
0218  6602  V6=02          bleep for 2x20 msec
021A  F618  TONE=V6
021C  6630  V6=30          wait for 48x20 msec
021E  F615  TIME=V6        (total 1 sec)
0220  F607  V6=TIME        check timer
0222  3600  SKF V6=00
0224  1220  GO TO 220
0226  73FF  V3=V3+FF       decrement counter
0228  00E0  ERASE          clear screen
022A  1202  GO TO 202      repeat . . .
022C  D455  SHOW 5@V4,V5   subr. to show digit
022E  7404  V4=V4+04       move "cursor" right
0230  00EE  RETURN
```

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.



Here is the full size artwork for the printed circuit board.

While the above program serves to illustrate some of the trickier CHIP-8 statements, it is not a good example of the power and efficiency of the language. To see that, one has to analyse a more complex, graphics oriented program, such as an animated video game. It is good experience to "dis-assemble" one or more of the games provided, to see how the programmer tackled the problem. You should therefore deduce: which numbers are instructions and which are data; what each variable is used for; and what is stored in various memory workspaces; etc. (Kaleidoscope and TV-Typewriter not recommended for starters.) Flowcharting is also a handy programming tool that will increase your expertise.

I have presented only a very sketchy description of how to write programs. A lot of practical experience is the only way to learn and become proficient. Test the operation of each of the instructions in a short routine, so that its operation becomes clear. Before attempting any complex video games, try some of these simpler exercises:-

1. A program that waits for a key depression, then displays the corresponding hex digit on the screen. (Looping indefinitely.)
2. Same as (1), but rejects keys A to F by returning to monitor.
3. Show an 8 x 8 symbol of your choice on the screen and make it move left when key 4 is held down and right when key 6 is held (using EX9E or EXA1).
4. Make the above 8 x 8 symbol move randomly about the screen.
5. Program the game of NIM. Show 21 objects on the screen. Two players take turns to remove 1, 2 or 3 objects. Player to take last object(s) wins.
6. Imagine a 4 x 4 square game board. The keypad is also a 4 x 4 matrix. Program accepts a hex key, then places a symbol in corresponding position on screen.
7. As above, but alternating between two different symbols.
8. Invent a two-player game based on the above principle, and program your computer to win against a human opponent.

Once you can do the above, you're ready for Lunar Lander,LIFE, Blackjack, and other favourites. Add a 2k RAM board and you can try for CHESS or STAR-TREK.

APPENDIX: HEXADECIMAL

There's nothing complicated about it, but it might help if you had 8 fingers on each hand. Then you could count from 0 to 15 (instead of 0 to 9) before having to use carry. HEX is convenient because each digit can be represented by exactly 4 binary digits (bits), without having any missing codes or extraneous codes:-

Decimal	Binary	HEX
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

The symbols A to F are used to denote the numerals 10 to 15. Furthermore, 4 divides into 8 exactly; so you can represent an 8-bit binary number with 2 hex digits, without having any bits left over; unlike the OCTAL (base 8) system, which has had many programmers pulling out their hair! Thus we can easily convert between binary and hex, simply by grouping bits into fours: e.g.:-

What is 26F0 in binary?

2 6 F 0

Answer = 0010 0110 1111 0000

(from above table)

What is 01111100 in hex?

0111 1100

Answer = 7 E

As well, 16=4x4, and 4+4=8, and PIAs have 8-bit ports, which makes 16-key keypads ideally suited. So HEX is very convenient all round, and easy to master once you memorize the above table!

Editor's Note: 6802 chip has on-board clock

Shortly after the DREAM 6800 was first described, in the May, June, July and August 1979 issues of Electronics Australia, there was a serious shortage of 6875 clock chips manufactured by Motorola Inc., USA.

A short-term solution was provided by an "out-rigger" clock unit described in the August 1979 issue (page 104 of this handbook) and this is still a valid approach for those who may prefer it.

A little later a more elegant solution appeared: the 6802 microprocessor which carries its own on-board clock chip and replaces the 6800/6875 combination. This is now the preferred approach.

The 6802 requires a slightly modified board and we know of at least one firm, J. R. Components, who are supplying a complete kit based on the 6802 and including a modified board. Readers contemplating construction should keep the above history in mind when buying a kit and ensure that the board supplied is suitable for whichever processor is supplied.

The address of J. R. Components is PO Box 128, Eastwood, NSW, 2122. Phone (02) 85 3385.

4K RAM Expansion for the DREAM 6800

The long-awaited DREAM 6800 expansion project is here! This uncomplicated circuit allows DREAM users to expand their computer's memory to a total of 4K and fits inside the cabinet of the original DREAM. It's just the thing for those who have gone beyond the initial stages of programming and now wish to write longer programs.

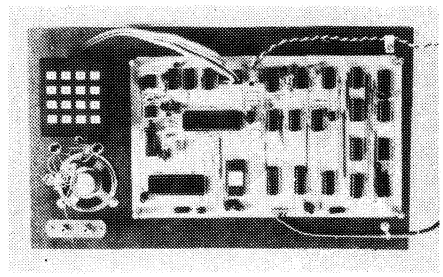
by K. ZALKALNS

Although CHIP-8 is a very memory efficient language, I and no doubt other owner-drivers of Dream computers have had times when the available memory storage is just not large enough. A quick answer is to make simpler therefore shorter programs, but that seems to be a retrograde step.

With more RAM, various possibilities suggest themselves. Often used subroutines could be pre-stored at the top of the stack, and called from a number of programs, a block of programs could be entered at once and a key pressed to select a chosen program, or even "number crunching", etc, etc. But enough of suggestions. After building the board you can dream up your own ideas on how to use it (sorry about the pun).

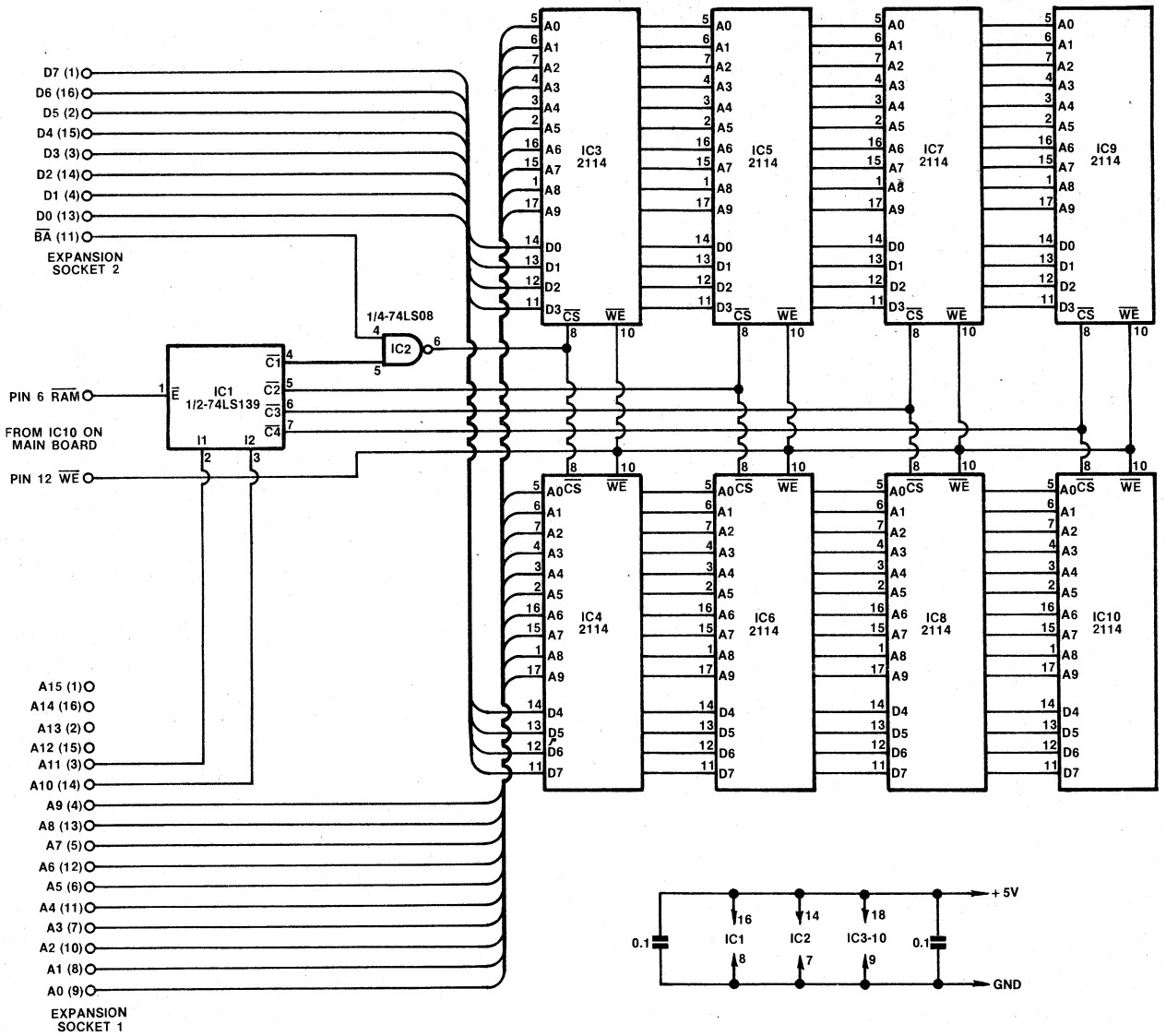
I decided to limit the expansion of memory to a total of 4K. This is the maximum extension which is possible without buffering the data and address lines, and in any case the 12-bit address operand of the CHIP-8 language will only allow addressing up to location OFFF (4095 decimal). Because of the efficiency of the CHIP-8 language, 4K of memory should be ample.

To keep the circuit simple several signals are taken from the main board. The conditions required to address the RAMs for the CHIP-8 use are that address lines 14 and 15 are low and lines 10 and 11 are decoded to select the correct 1K block (see table 1). The signal for the former function is already available at pin 6 of IC10 (74LS10), labelled RAM and can be used as the enable input for the address decoder (½ 74LS139). The four decoded outputs (active low) are then used to select the correct RAM. The Read/Write function is also available at IC10 (pin 12) and is fed to the WE input of all RAMs.



Published in May, June, July and August 1979, the DREAM 6800 was a highly popular project.

The only further decoding required is to supply RAM1 with the BA signal for DMA (Direct Memory Access) use for the video page. This can be achieved in two ways. The first is to cut the track near pin 13 of IC12 and apply the C1 signal from the extension board to the same pin, leaving the main board RAMs in their current location. The other method, which I opted for, is to use one gate of a 74LS08 on the extension board with the BA signal taken from the expansion bus. This does mean most of the package is unused, but the price is low enough and the main board doesn't have to be mangled in the process. If you use this approach there will be no RAM chips on the main board. The entire 4K memory will be on the expansion board.



EA 4K RAM EXPANSION FOR DREAM 6800

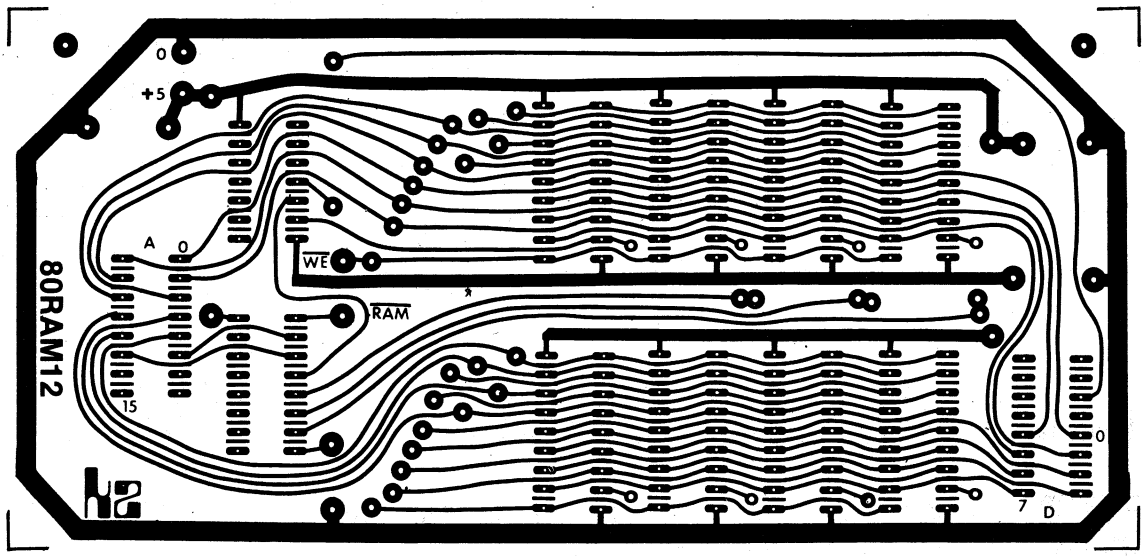
2/CCI-

Just two ICs, in addition to the memory chips, are required for this RAM expansion.

As can be seen from the board pattern, a single sided board has again been used to keep the cost down, so there are a number of wire links to install. Whether you make your own board or buy one, the first job should be to check for shorts or breaks in the tracks, as most of them, of necessity, are closely spaced and quite narrow. Next, solder the 23 wire links, using sleeving where necessary, Install IC sockets for the RAMs and bus extension at least, and finally the two capacitors and TTL ICs.

The extra two signals $\overline{RAM}$ and $\overline{WE}$ must now be obtained from the main board. If you didn't follow the advice given for building the main board, and did install sockets for all the ICs a simple method is available. Take a 14 pin header, solder and an IC socket to the top and like magic, you've now got a high rise socket for IC10. Solder leads from pin 6 ($\overline{RAM}$) and pin 12 ($\overline{WE}$) and you're in business. If you did solder the ICs either PC pins will have to be installed at the correct locations, or alternatively, the two leads could be soldered to the bottom of the board. The only other leads required are for power, which can come from the two pads between the expansion sockets on the main board.

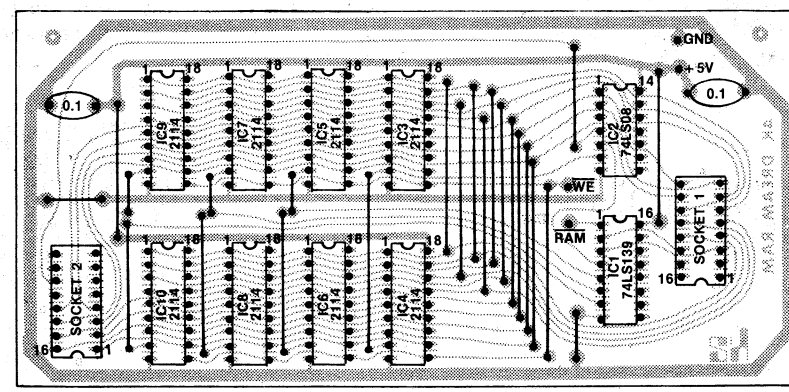
The extra board has been designed to mount above the main board over the expansion sockets by using spacers and longer screws. Prior to fitting the board, thoroughly check it again. It's better to be safe than sorry. If everything checks



Above is the full-size artwork for the PC board while below is the component layout diagram.

out OK, it's time for the big test.

Remove the RAMs from the main board. Don't forget they are MOS devices, so chain yourself to your earthed metal workbench before handling them. You don't want to blow their brains out, now, do you? Plug in the bus extenders (the address lines plug into the socket nearest the corner on the main board) and the other connections and insert the ICs on the RAM board. The next



step is obvious. Take a deep breath to steady yourself and switch on. If all is well you should be confronted with a totally awe uninspiring picture, very similar or identical to the one displayed prior to brain surgery. Think of that – a Dream 6800 that to all intents and purposes appears to be stock-standard, but is in reality waiting for you to fill it's vast memory with useful things to do.

Dreaming of 1979

Now and then it's worth indulging in a little nostalgia. Computer technician Neil Helson gave us the perfect excuse when he got in touch about his surviving Dream 6800 microcomputer. Here is the venerable machine's story.

Residents on a suburban Lower Hutt street are abuzz with talk of a new high-tech device with a 4MHz processor and a grunty 1KB of RAM. The year is 1979, and Neil Helson's new Dream 6800 microcomputer is the talk of the neighbourhood.

Back then, plans for the Dream 6800 had just been published in *Electronics Australia*. According to the computer's designer Michael Bauer, it was inexpensive and easy to build, weld and program from scratch.

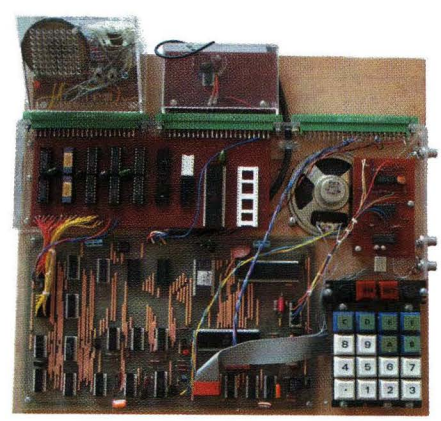
Helson, who was teaching a data processing course at Wellington Polytech at the time, built the microcomputer according to the plans using components purchased from a hand radio store in central Wellington.

Once assembled it sported a Motorola M6800 processor with 4MHz clock-speed, 1KB of RAM, 1KB of ROM and an audio beeper for sound effects. It could be hooked into a black and white TV to provide a 64 x 32 dot matrix display and data was input using a customised hexadecimal keyboard ripped from an old calculator.

"You needed to be able to use a soldering iron and you had to be adept at constructing things like that, but if anything, finding the parts was the biggest issue," Helson said.

"Things like keyboards didn't exist, so I salvaged one quarter of an old calculator and relabelled some of the additional keys with letters A through to F. That gave me the 16 hexadecimal keys I needed and it had two others – which were labelled reset and function."

Helson built the Dream both as a project – he had a love of electronics – and as a way to save money. The components cost him less than $100 at the time, making the computer a bargain compared to the $US600 TRS80 or US$1,298 Apple II.



His children also enjoyed playing console games and buying new games and a constant supply of batteries was expensive. The Dream allowed Helson to program a steady supply of games for his kids in its CHIP-8 programming language.

"The language was very compact... For example an animated UFO mission intercept program with onscreen decimal scoring took only 104 instructions."

Programs and games were keyed into memory and then saved to tape using a standard cassette tape recorder. Numerous CHIP-8 programs were published in books and most took about 10 to 15 minutes to key into memory, Helson said.

"The word soon got out that there was this device that nobody had seen. Neighbourhood children would come around to play the games."

Helson replaced the beeper with a homemade soundcard based on the Texas Instruments SN76488 sound synthesiser.

chip so he could add sounds such as bombs dropping and explosions to his games.

He also built a paddle, similar to a joystick, which his kids and their friends could use to play games he had designed using bouncing balls to knock out bricks. Over time, as games he programmed became more complex, Helson upgraded the Dream with an extra 4K of RAM by building an expansion board.

"Our children still fondly remember the simple games, and the hours spent designing, coding and testing new games and applications."

Back to the future

Fast forward to 2010 and Helson's Dream has long since moved into retirement at his house in Darfield, Canterbury. It still works fine 31 years later – something almost unheard of when it comes to modern PCs – but it's now more of a museum piece than anything else.

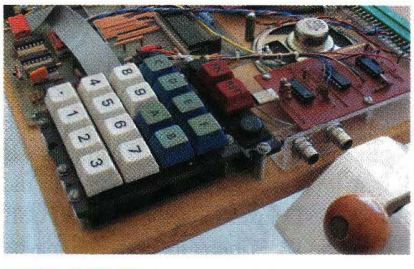
"I've shown it to people more recently and they look at me sideways. I've had it running games and they just have such a low resolution compared with modern computers."

But despite the huge number of changes in PCs during the last 30-odd years, the basics remained fundamentally the same, Helson said. PCs still had a processor, memory, video, sound, a means of storing and retrieving data and generally a keyboard and mouse, but those components had all become infinitely more complex.

That was generally a good thing, though it was a pity that an individual hobbyist would find it almost impossible to build and program a modern PC from scratch these days, Helson said.

"I am glad that we are no longer limited to 256 bytes of video memory, and that colour displays replaced old black and white TVs, and have such wonderfully high resolution. Storing programs on cassette tape was all we had, and it was painful. I could never have imagined that home computers would use laser technology to record vast amounts of data on optical media, or that you could purchase a 1TB external disk for $120. Listening to music in full stereo has to be better than the whizzes, pops and bangs which were all the synthesiser chip could provide."

James Heffield



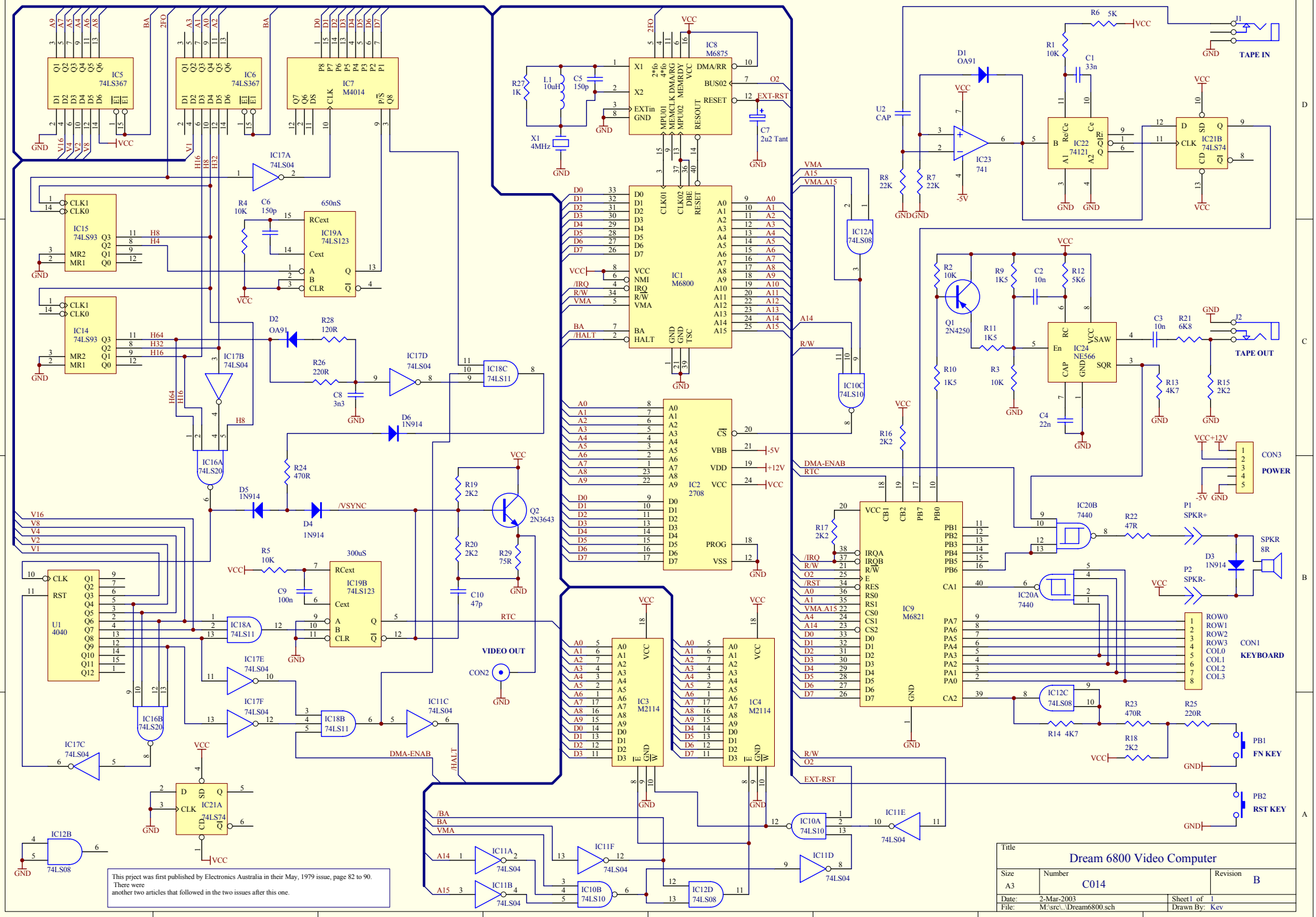
HOBBYIST CENTRAL: Low res, high fun.

Dream 6800

Processor: Motorola M6800
Clock speed: 4MHz
RAM: Onboard 1K, expandable to 32K with off-card expansion
ROM: 1K
Display: 64 x 32 dot matrix
Input/output: Motorola M6821 to control a hex keypad, tape I/O and audio beeper

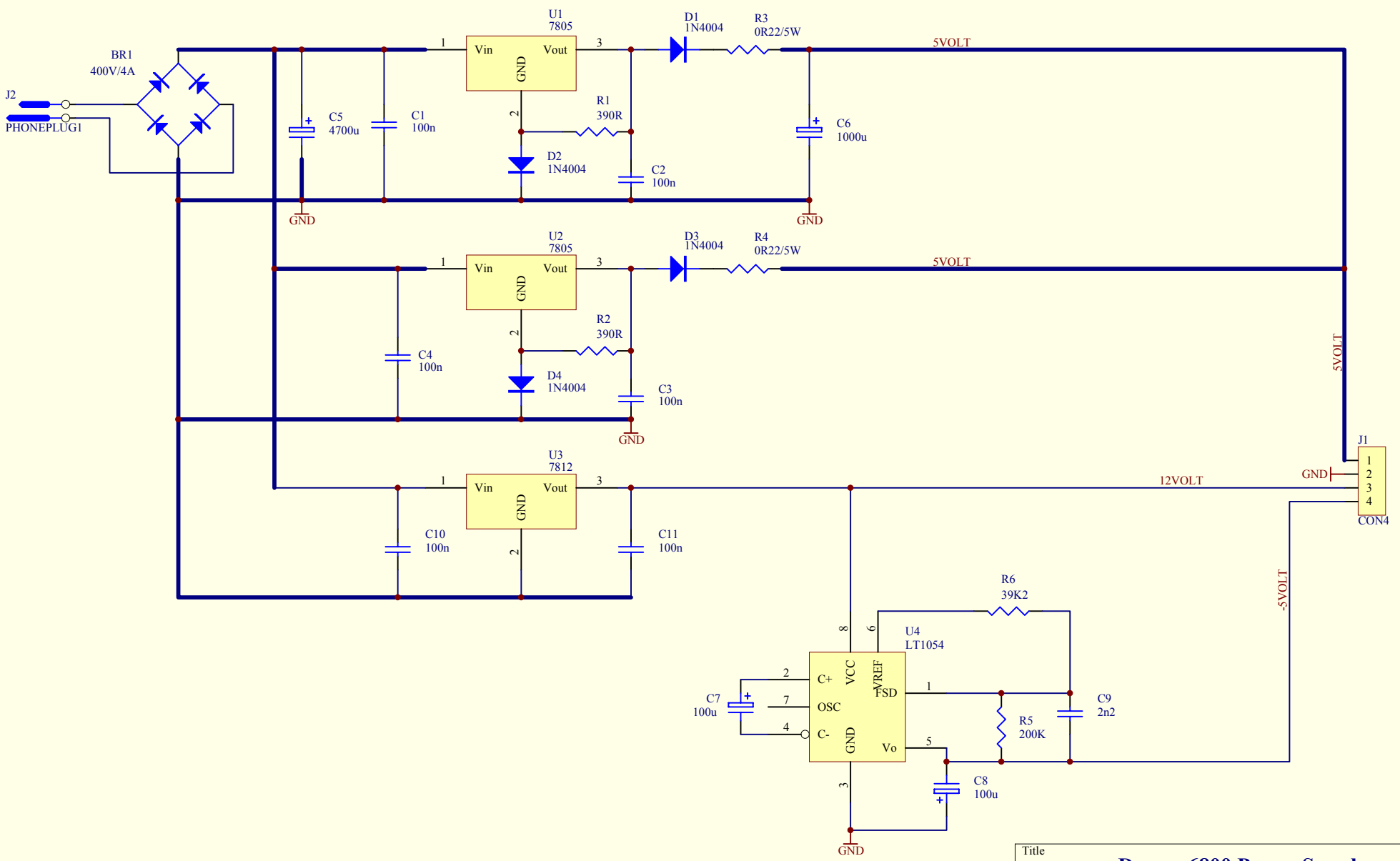
The Dream, full name Domestic Recreational Educational Adaptive Microcomputer, was a do-it-yourself kit PC designed in 1979 by Michael Bauer, a tutor in computing and mathematics at Australia's Deakin University. Bauer described it as a "superlative 8-bit MPU in every respect".

The Dream was compatible with the CHIP-8 language and hobbyists were required to key in the code for its CHIPOS operating system manually. CHIP-8 was developed by RCA Labs researcher Joseph Weisbecker. It was described as a "high level language interpreter" and was invented especially for video games, graphic displays and simulations.



This prject was first published by Electronics Australia in their May, 1979 issue, page 82 to 90.
There were
another two articles that followed in the two issues after this one.

Title	Dream 6800 Video Computer	
Size: A3	Number: C014	Revision: B
Date: 2-Mar-2003	Sheet1 of 1	
File: M:\src\..\Dream6800.sch	Drawn By: Kev	



Title	**Dream 6800 Power Supply**	
Size: A4	Number: **C015**	Revision: **EA**
Date: 16-Jun-2002	Sheet lof 1	
File: M:\src\..\Dream6800_PSU.sch	Drawn By: Kev	