REGISTERED BY AUSTRALIA POST -
Publication No.NBG4250.

DREAMER Nö 17
JAN, '82.

N.S.W. 6800 USERS GROUP,
[redacted]

```
0200    86 37       LDA A #37
0202    BD C2 FE    JSR    C2FE
0205    CE 80 20    LDX    #8020
0208    6F 01       CLR     01,X
020A    6F 03       CLR     03,X
020C    6F 21       CLR     21,X
020E    86 FF       LDA A #FF
0210    A7 00       STA A   00,X
0212    A7 02       STA A   02,X
0214    6F 20       CLR     20,X
0216    86 04       LDA A #04
0218    A7 01       STA A   01,X
021A    A7 03       STA A   03,X
021C    A7 21       STA A   21,X
021E    CE 10 00    LDX    #1000
0221    7F 80 20    CLR     8020
0224    7F 80 22    CLR     8022
0227    A6 00       LDA A   00,X
0229    B1 80 40    CMP A   8040
022C    27 03       BEQ TO 0231
022E    7E CB E3    JMP     CBE3
0231    7E 02 50    JMP     0250
0234    F6 80 22    LDA B   8022
0237    CA 80       ORA B #80
0239    F7 80 22    STA B   8022
023C    86 32       LDA A #32
023E    C6 64       LDA B #64
0240    01          NOP
0241    01          NOP
0242    00          ---
0243    02          ---
```

DREAMER

No. 17 JANUARY, 1982.

Here it is at last, and boy, is it big! Sorry it is a bit late, but people kept sending in bits and pieces which we thought you would enjoy, so we kept adding them in. Reminded us of our early days producing the Dreamer, when we had so much material laying around that we could never decide just what to put in and what to leave for the next issue, and even when we did, could never fit it all in the space we had available. We must admit, when we decided to start a newsletter about a 'cheap little computer' that E.A. had run as a project, then forgotten about, we never envisaged that it would grow to the size that it did, or prove so popular. An enormous volume of information about the DREAM has been distributed to a lot of people this way, and we must thank all those who contributed in any way, as you are the ones who kept the ball rolling, by sharing your ideas and expertise with others, so that they could learn from them. To those of you who have become friends over the past 18 months or so, thank you again, we are sorry that it had to end, but at least you should find plenty in this issue to keep you all occupied for quite a while!

Where can you go from here to get information on your DREAM? Well, the AUSTRALIAN V.I.P. USERS GROUP have offered associate membership of their group to all DREAM owners and users, as the Cosmac V.I.P. also uses the Chip-8 language, and they say that they will be able to provide the necessary information to enable you to convert their programs to run on the DREAM. They have an advertisement elsewhere in this issue giving full details of their offer.

Alternatively, Sid Moorby, of Wyalla Norrie in South Australia, is thinking of starting a Technical newsletter, concentrating more on machine code and assembly language programming. He also has an ad in this issue, and there is also an article from him on the assembler in the DREAM, so if this is more to your liking, drop him a line.

Just before this issue went to the printers, we received a letter from Ashley Emery, of Beverley Hills in N.S.W., telling us that he and a friend were prepared to take on the editing and printing of the Dreamer. We have not had time to get in touch with him yet, but have included the details he gave us in his letter elsewhere in this issue, so if you write to him enclosing a S.A.E. he will send you an introductory letter telling you about himself and what he intends to do.

The Hardware Encoded ASCII keyboard project unfortunately is not in this issue, as we did not receive it from the DREAMSOFT people. They also have a letter in this issue to let you know what their plans are for the future.

To satisfy all those of you who asked in your letters, No, we are NOT bored with, or 'getting out', of computing as a hobby, we are simply moving on to other things. Graeme now has an Apple, and Garry has a new Commodore Vic-20 on order, (if it ever arrives), and we need the time to concentrate on new fields, and learn more about computers, NOT about editing, and typing, and spelling, and deadlines, and all the other things involved in producing a newsletter such as the Dreamer, which we have learnt a great deal about in the past 18 months. (Much to our horror and dismay at times, and with a lot of amusement and enjoyment at others.)

And now, on with the great Assembly Language debate. Lindsay Ford's articles certainly stirred up some comment, and although we do not have room to print all the letters in full, we thought you might like to read the comments from some, and Lindsay's answer to them.

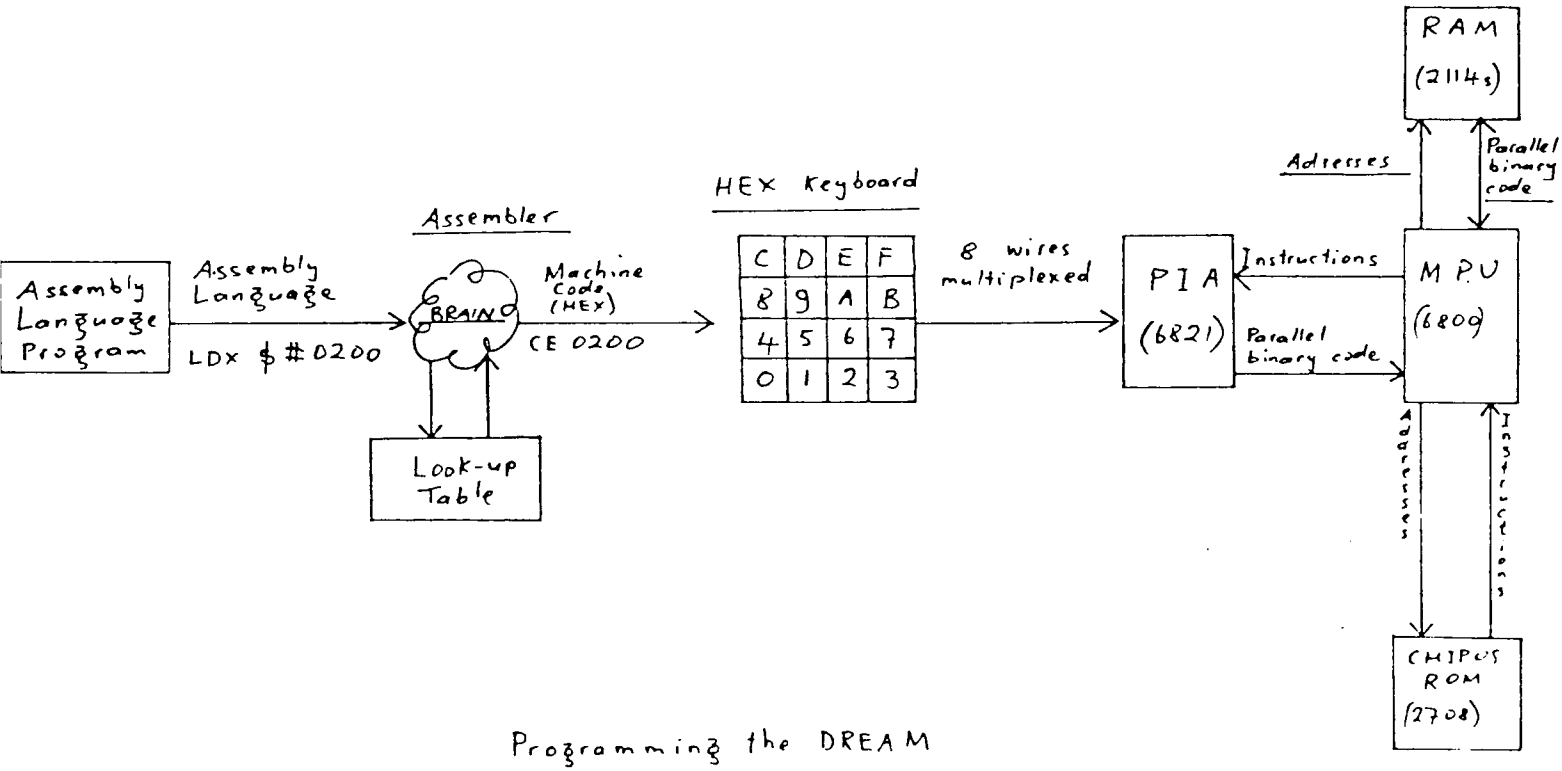
Firstly, we received a telegram from Lawrie Norton, of Upper Hutt, in New Zealand, saying that there were serious errors in Lindsay's first article, and that a letter was following. This duly arrived, (all 7 pages of it) and this is what it said;

"Probably by now you have received my telegram sent Friday 16th., regarding Lindsay Ford's articles about Assembly Language. I enjoyed his style of writing, especially the opening, but I cannot agree with some of the things he has written. The sad thing is that if his article is read, as I guess it is meant to be, by people who do not yet know about assembly language programming, it will start them off on the wrong foot.

Perhaps I could quote parts of the article, with corrections as I believe to be necessary. All right then, here we go:

"The sub circuit that does this is called the Assembler and as you're communicating with it rather than directly with the computer memory the instruction set you use is called Assembly Language."
(Completely wrong! The DREAM does not contain an assembler, and it cannot accept assembly language. The data which is keyed into the DREAM on the Hex keypad is the hexadecimal representation of the binary data which will end up in memory. If you are going to put into the DREAM a program which is written in assembly language, then you must first hand assemble it (using that 2 Gigabytes of dynamic memory you'ré carrying around between your ears) into the equivalent machine code instructions, written in the hexadecimal representation of binary, which is the only code the DREAM can accept.)

The illustration which follows in the original article, showing a keyboard, an assembler and an M.P.U. should be re-drawn to look something like this:



Programming the DREAM from an assembly language program.

To Continue:

"Now assembly language routines can be 'commented' to allow the user to understand what each particular instruction in the program does, but these comments are of little value unless the user can figure out what they mean. Take, for instance:

0080 CE 0200 LDX $#0200

Our Chip-8 experience tells us that the '0080' refers to a location in memory....."
(I gather that Lindsay Ford is telling us that '0080' is the memory location, that the 'CE 0200' is the assembly language instruction, and that the 'LDX $#0200' is the comment, although he does not make this absolutely clear. IF this is what he is saying, then it is incorrect. Certainly the '0080' is the memory location, but the 'CE 0200' is the machine code instruction (in Hex.), and the 'LDX $#0200' is the equivalent instruction in assembly language. This instruction causes the (hex) figures 0200 to be loaded into the index register. This number, (0200 hex) will be used in a following instruction to indicate the start of another section of program. A suitable comment might be, 'Point to start of program', since the index register is pointing to the location 0200, which, in the DREAM, is the location from which CHIPOS runs its programs.)

Carrying on:
"As you will commonly encounter threeof these symbols, let's sort them out;
[$] - means that the numbers following it relate to a memory location."
(WRONG! The '$' means that the number following it is in Hexadecimal. It may be a memory location, but it may be data. There are several of these indicators of the base of numbers, here they are.

$	means	the	number	following	is	Hexadecimal. (Base 16.)
@	"	"	"	"	"	Octal. (Base 8.)
%	"	"	"	"	"	Binary. (Base 2.)
(nothing)	"	"	"	"	"	Decimal. (Base 10.)

(This is the default case.)

" [#] - means that the numbers following it are data."
(In a rather round about way this is correct. According to the text books, the hash mark (#) means the immediate mode of addressing:- see further down the article.)

" [*] - means that a number.....is irrelevant."
(I believe this could be correct in some instances, although according to the text books, the star (*) is used to indicate that the following number is an offset from the program counter.)

All these are very clearly shown in the book, "6800 Assembly Programming" by Lance A. Leventhal, published by Osborne & Associates.
(Hopefully, by now you will be able to understand that the 'LDX $#0200' means, 'Load, (using the immediate mode) in to the index register, the hex value 0200.')

Somewhat further down the original article, we arrive at.
"MACHINE LANGUAGE CONCEPTS:" (There is NO machine language, there is machine code.)

"When we were programming in Chip-8 we could happily sit back and talk to the interpreter (that's the device that translates Chip-8 instructions into assembly language....." (etc)
(WRONG! The interpreter is the program which translates Chip-8 instructions into machine code. Note however, that Chip-8 is a high level language, and therefore each Chip-8 instruction will be translated into a group of machine code instructions which will perform the required function. Note also that because Chip-8 instructions are written as a sequence of hex digits, it is often difficult for newcomers to DREAM programming to distinguish between Chip-8 instructions and machine code instructions.)

Not far down page 5 of Dreamer No.14, there is a chart showing (diagramatically) the index register, containing the binary data
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
Below that is the same number in hexadecimal representation, labled, 'Assembly Language Operand', whereas it is simply the hex.representation of the binary number above it.

Further again down Page 5, under the heading, "6800/6802 ADDRESSING CONVENTIONS" there is a very good description of the various addressing modes used in machine code programming. I found these descriptions easy to follow, and quite clear to understand. The table shown would be clearer if it were re-drawn as follows:

Function	Assembly Language	Machine Code					Comment
	Mnemonic	Immediate	Direct	Indexed	Extended	Inherent	
Load Index Register	LDX	CE	DE	EE	FF		Point to (whatever)
Decrement Index Register	DEX					09	Point to one position previous.

It is a table of this sort that you would use if you were hand assembling a program from assembly language into machine code, to key it into the DREAM via its hex. keypad.

Incidentally, for anyone wishing to 'get into' assembly language programming, I would strongly recommend the book mentioned earlier, "6800 Assembly Language Programming" by Lance A. Leventhal, published by Osborne & Associates.

You may notice that I have always referred to 'Hand assembly', rather than just 'assembly'. This is because I am writing specifically about the DREAM. With a large computer, and I am thinking of one having around 16K of RAM, there will be an assembler program and the computer will have a full 'typewriter' keyboard with a complete alphabet as well as the dollar sign and hash mark. Therefore, you can type in assembly language (to use our previous examples; LDX $#0200) and have the assembler program in the computer assemble it into machine code for you. (CE 0200 in this example.) Note that an assembler is a program , not a hard wired logic device as seems to be indicated by Lindsay Ford's diagram on page 3 of Dreamer No. 14. The DREAM does not have enough RAM to contain an assembler program, let alone have room to put the program after it has been assembled. The DREAM also does not have a full alphabetic keyboard (it has only a hexadecimal keyboard). It is not possible to type assembly language into the DREAM, because many of the required keys are not on the keyboard.

To combine assembly language and the DREAM, you must program your brain with an assembler program, and use it (together with a hard copy look-up table) to assemble the program into machine code, and key this, (in hexadecimal representation) into the DREAM.

Have I added sufficiently to the confusion?

Kind regards,
Lawrie Norton. (ZL2AUT)

AND THEN,

Michael Bauer, in one of his regular letters, (yes, he still talks to us), had this to say;

"At the risk of becoming something of a critic, I wonder if you noticed some of the glaring errors in Mr. Ford's article on assemblers. He has got the meaning of the symbols ($, # and *) all wrong! A note in next month's editorialwould be appropriate, don't you think? (Otherwise, everyone will be confused all the more.) If people want to get into assembler programming, (which is the same thing as machine code programming if you are hand-assembling, isn't it?), then I would recommend Ron Bishop's "Basic Microprocessors and the 6800" (Hayden, 1979) or one of the many other excellent texts."

Regards,
M.J.B.

Enclosed with the above letter was a copy of page 4 of Dreamer No. 14, showing Michaels interpretation of the symbols to be;

$ - means that the numbers following it are hexadecimal.

- means that the numbers following it are data and are used in immediate mode.

SO,

We then sent copies of these letters to Lindsay Ford, telling him that we intended to publish them, (or parts thereof), and asking him did he wish to reply to them.

HE DID, AND THIS IS WHAT HE SAID.

"I was rather surprised to receive the copys of letters from Michael Bauer and Lawrie Norton concluding that some of the terms I used in the first part of the article were 'wrong', as this assumes that they must be 'right', so let's look a little closer;

1) Michael Bauer takes me up on my use of the symbols '$' and '#'. Where I use the '$' sign to refer to a "memory address" operand, he says that it means'hexadecimal' and where I use '#' for 'data' he says it denotes an instruction in the immediate addressing mode. What a pity Michael didn't give this a little more thought before putting pen to paper, as not only are both definitions used in the literature, they are entirely reconcilable!

A computer programmer working with the ordinary instruction set of an MPU (i.e. instructions outside the 'relative' mode) is only concerned with hexadecimal operands where they relate to memory addresses. In the immediate mode the programmer is concerned with the binary data in the operand as this is what gets stored in the register or accumulator.

So Michael is 'right' in that,

'#' refers to the 'immediate mode' (his definition) and immediate mode operands deal with 'data' (my definition).

'$' deals with 'hex numbers' (his definition) and as they are outside the immediate mode they must, by definition, relate to 'memory addresses' (my definition).

The aim of my article was to reduce the amount of confusion surrounding assembly language and in this my definitions were both correct and effective. I am sure Michael's definitions are equally correct, but I am not so sure about them being effective in explaining abstract and difficult concepts to novices.

2) Lawrie Norton defines 'Assembly Language' as the mnemonics that are used to 'comment' hex instructions and takes me to task on this basis for use of terms such as 'assembler' and 'assembly language'. In this he is supported by Lance Leventhal in his excellent book on 6800 Assembly Language (to which I referred in my article). Unfortunately for Lawrie's thesis, however, MOTOROLA, (who really ought to know) do not use the same terms with any degree of consistency in their publications. In some cases they refer to the mnemonics as 'the operator', in others as 'the executive instruction', and even, in some publications, they call the whole listing (i.e. Operator/label op. field and operand) "Assembly Language".

My readings on the subject led me to the conclusion that there was NO 'right' answer and so I opted for the choice that resulted in the greatest clarity of explanation. Had I adopted Lawrie's approach I would have needed to introduce concepts such as 'hand assembly' (most Dreamer's would not have known what that was), and to go into so much detail that few people would have penetrated beyond the first page. Lawrie's drawing is an excellent example of just this point.

In conclusion, may I say that I entirely agree that inaccuracies should be corrected. I am just a bit reserved about introducing huge volumes of technical detail in a "beginners' article" where this serves no practical purpose other than the introduction of complexity."

Regards,
Lindsay.

P.S. What Lawrie calls an 'assembler' I have seen called a 'cross assembler', 'mnemonic assembler', and even an 'assembly mnemonic compiler'. Similarly, my definition is often replaced with 'hex loader' and I have even seen 'assembly/binary translator'. How can you possibly win with jargon like this????

Well, there you have it. We will leave it to you to make up your own minds what you want to call them. Personally, we think that Lindsay did a great job with the three articles, and are sure that a lot more people now know considerably more about assembly language and/or machine code programming than they did before they started, particularly if they took his advice in the last article, and bought a couple of texts to do some follow up work with.

That is it from us, thank you all again for your support,
May all your DREAMs be happy ones,

GRAEME and GARRY

****

DREAM EPROM PROGRAMMER (0200 - 04D0)

RUSS VERDON,
[redacted]

I began to design this eprom programmer because I am building a new 6802 system and realised that I would need to write a new operating system and save it in rom. Mike Bauer's article on HIRES in DREAMER No 14 hastened completion of the project so I can program a HIRES eprom for my DREAM.

So here is an eprom programmer which will allow you to program 2708, 2716 or 2732 type eproms. A complete unit including box costs $21, or if you have a spare PIA which can be pressed into service(eg. J R expansion board) then less than $15 (by omitting 6821 and socket). Similarly if you only want to program 2716's the cost will reduce to $12 (omit switch 1, 26volt pulse circuit).

CIRCUIT

The circuit is quite straightforward. The data bus is connected to the eprom via the A side of a PIA. PB0-3 are used to control the programming via software timing loops , and are used to reset and increment the address on the eprom socket by means of a 12 bit counter (cmos 4040). The other circuitry consists of two switches to select the type of eprom and whether the program or read mode is required, and a 25/26 volt power supply and pulse amplifier to provide programming voltages.

EPROMS

2708 (1K) eproms were early technology and require +12v (pin19), -5v (pin21) and pin20 needs 0v to select the device and +12 when programming. To program it , a high pulse is applied by PB1 and amplified by DS549 and T1 to pin 18; when this occurs,the byte on the data bus is stored internally in the eprom at the location on its address pins. Each pulse is 100ms long and 100 are needed for each location.

The 2716 and 2732 eproms remove the need for the +12 and -5v supply on pins 19 and 21; they become A10 and +25 (2716), or A10 and A11 (2732). Pin 20 becomes (read/program) 0,+5v for the 2716 and 0,+25 for the 2732. Pin 18 is given a single 50ms pulse to program both devices(via PB1).

POWER SUPPLY

The power supply can be derived as shown from your existing supply to save the cost of a new transformer. Simple current limiting is provided by the 22ohm resistor and IN914's across base and emitter of BD139. When constructed, check that +26 is within ± 1volt; adjust zener value(or add IN914's between zener and 0v to increase voltage).

NOTES ON CONSTRUCTION

The whole unit can be built in a small size 'zippy' box of the 130x42x68 size. A circuit board is shown which has been designed to fit within such a box, and also has the option of being cut along the dotted line if a PIA is already available. The eprom socket will need to be mounted on a second, smaller pc board which is screwed to the front panel. Pads are shown for connection of this board- direct soldering to the copper side is necessary because the component side will be flush with the front panel. Suggested front panel layout and switch wiring are also shown which may assist your construction. The READ/PROGRAM switch cut-out,is for a Dick Smith 4pole 2pos toggle switch (s1301), others may need a different size hole.

Don't forget the 6821 and 4040 are static sensitive - ground everything. A slot or hole will be needed in the side of the box to take out the interface cable/s(rainbow cable makes a neat job).

OPERATION AND SOFTWARE ROUTINES

First connect the unit to your DREAM/expansion board. Select READ mode and set Sw1 to the type of eprom you want to plug in, and plug it in.

NOTE WELL : Don't switch Sw1 to different eprom types while power is applied(I learnt the hard way!)- if your switch is make-before-break, you will short the +25 and -5v supplies (my 7905 regulator punched through, putting -20v on the -5v rail!).

It is also worth mentioning that in the idle state, T1's collector load(1k 1w) dissipates 2/3watts, and gets quite warm. It is therefore preferable to disconnect the 26v supply or unplug the unit when not in use.

Turn on and load the software routine listed below (0200-04D0). Load, key in or block shift your data to be stored, at:

0500-08FF .....for 2708
0500-0FFF 2716
0500-14FF 2732

Type in 0200 Fn3. The program will now initialise the PIA ports and prompt:

1 2708
2 2716
3 2732

To type in a1,2 or 3. You then select one of the following routines:

1 VERIFY
2 PROGRAM
3 CHECK

Again select the routine you require.

VERIFY checks that the eprom in the socket is fully erased(all 1's,ie.FF's). This routine should run OK if there is no eprom plugged in as the data lines should "pull-up" high.

CHECK compares the data stored in eprom with the data in ram(0500....) to see if programming has been successful.

PROGRAM transfers the data in RAM (0500...) to the eprom. When PROGRAM is selected, you are asked to switch from READ to PROGRAM mode when PIA ports have been configured. When you have thrown this switch, pushing any key will initiate programming, which takes up to 4mins depending on eprom type. While in the program mode, the LED will light as an indicator. When programming is complete, you are asked to switch back to the READ mode and can then use VERIFY to confirm a successful "burn".(READ mode ie. LED off, should always be selected unless instructed otherwise by the program.)

NOTES ON THE PROGRAM

1. The routines assume aPIA located at 8040(ie. 6821 has $\overline{CS2}$ connected to A14, CS0 to VMA.A15 and CS1 to A6). To use a PIA at a different address, change locations:
038E,0391,0395,0398,039D,03A0,03B9,03BC,03C0,03C3,03C8,03CE,03D8,03DB,
03FB,0401,0406,042E,0430,0438,0444,044A,0450,0455,0458,0493,0496,049F
04A0,04A3,04A9,04B4,04B7, from 4X to 2X,6X or wherever your PIA is at.
2. At 03E9 two NOP's ($01) exist. This location is just before a fail message is displayed when running VERIFY or CHECK. If you stick a SWI ($3F) at 03E9 and store DREAMBUG at 0080 , then you will see where the unerased section or mismatch of data was detected (ACCA=byte in eprom, X= location, ACCB= byte in RAM if using VERIFY routine).

REFERENCES

1. CHIPOS Manual - M.J. Bauer
2. Son of Cheap Video - D. Lancaster
3. Electronic Design 8 12/4/80
4. National Semiconductor CMOS Data Book

ADDRESS	LABEL	CODE	MNEMONIC	
0200	GO	7E02F8	JMP START	
0203	C1			LOOP COUNTER
0205	XTEM			TEMP XSTORE
0207	TIME			
0209	ROM			ROM FLAG
020B	TABN	56DA	V	NEW 5x3 ALPHA CHARACTERS
D		B75E	R	
F		E92E	I	
0211		4BDA	Y	
3		93DE	P	
5		F6CE	G	
7		B6FE	M	
9		B7DA	H	
B		B76A	K	
D		B6DE	N	
F		0000	SPACE	
0221		F34E	E	
3		934E	F	
5		F6DE	O	
7		B7DE	A	
9		F24E	C	
B		F248	L	
D		492E	T	
F		C546	S	
0231		FFDA	W	
3		D6DC	I	
0235	LIST1	0108		READ
		0E14		
		FF		
023A	LIST2	0A00		VERIFY
		0B01		
		020C		
		03FF		
0242	LIST3	0A04		PROGRAM
		010D		
		0501		
		0E06		
		FF		
024B	LIST4	0A0F		CHECK
		0709		
		0F08		
		FF		
0252	LIST5	1213		SWITCH TO
		0111		
		0F07		
		0A11		
		0D0A		
		FF		
025D	LIST6	0207		=708
		0008		
		FF		

ADDRESS	LABEL	CODE	MNEMONIC	
0262	LIST7	0D08		OK
		FF		
0265	LIST8	0C0E		FAIL
		0210		
		FF		
026A	LIST9	0207		2716
		0106		
		FF		
026F	LISTA	0207		2732
		0302		
		FF		
0274	OUTSH	A600	LDA 0,X	OUTPUT A HEX STRING
6		81FF	CMPA #$FF	STARTING AT X
8		2601	BNE OUTSH1	IF LAST CHAR = FF THEN
A		39	RTS	RETURN
027B	OUTSH1	8D0F	BSR OUTP	ELSE DISPLAY IT
D		08	INX	MOVE ALONG STRING
E		20F4	BRA OUTSH	AND CONTINUE
0280	OUTSN	A600	LDA 0,X	OUTPUT STRING FROM NEW TABLE
2		81FF	CMPA #$FF	CHECK IF LAST CHAR = FF
4		2601	BNE OUTSN1	
6		39	RTS	YES SO RETURN
0287	OUTSN1	8D1E	BSR OUTN	NO SO DISPLAY IT
9		08	INX	MOVE ALONG STRING AND
A		20F4	BRA OUTSN	CONTINUE
028C	OUTP	FF0205	STX XTEM	SAVE X
F		BDC193	JSR LDSP	FORM HEX PATTERN OF
0292	OUTP1	C605	LDB# $05	5BYTES AND
4		BDC224	JSR SHOWI	DISPLAY IT
7		D62F	LDB VY	RESET VY (WE GET AN AUTO
9		C005	SUBB# $05	LINEFEED DURING SHOWI)
B		D72F	STB VY	
029D	SPACE	C604	LDB# $04	MOVE VX ONE SPACE
F		DB2E	ADB VX	
1		D72E	STB VX	
3		FE0205	LDX XTEM	RECOVER X
6		39	RTS	
02A7	OUTN	FF0205	STX XTEM	SAVE STRING POSITION X
A		CE0209	LDX TABN-2	X = START OF NEW TABLE
D		BDC198	JSR LDSP1	FORM PATTERN
0		20E0	BRA OUTP1	AND DISPLAY IT
02B2	OK	8D1E	BSR DMAON	TURN ON DISP
4		BDC079	JSR ERASE	CLEAR SCREEN
7		4F	CLRA	
8		972E	STA VX	VX, VY = 0
A		972F	STA VY	
C		CE0262	LDX#LIST7	
F		8DBF	BSR OUTSN	PRINT 'OK'
02C1		39		

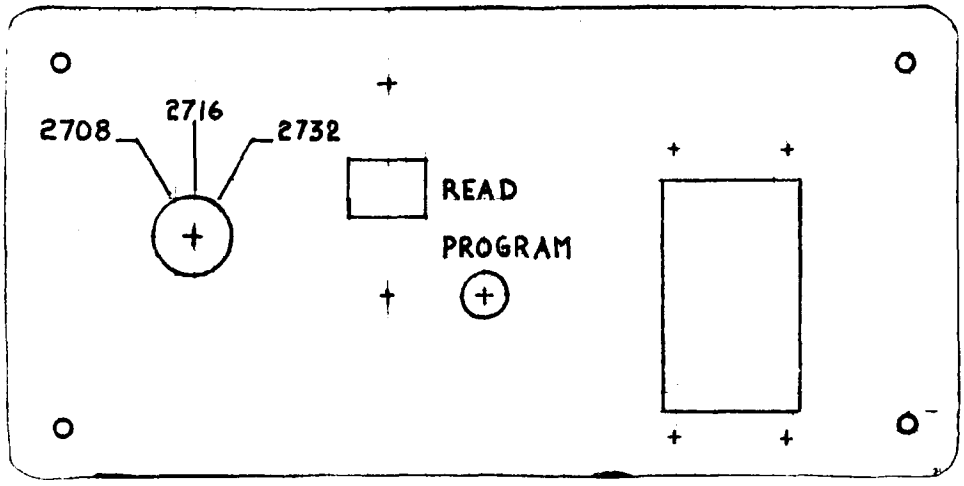
ADDRESS	LABEL	CODE	MNEMONIC	
02C2	FAIL	BDC079	JSR ERASE	CLEAR SCREEN
02C5		8D0B	BSR DMAON	TURN ON SCREEN
02C7		4F	CLRA	
02C8		972E	STA VX	
02CA		972F	STA VY	
02CC		CE0265	LDX# LIST8	
02CF		8DAF	BSR OUTSN	DISPLAY 'FAIL'
02D1		39	RTS	
02D2	DMAON	863F	LDA# $3F	TURN ON DISPLAY
02D4		BDC2FE	JSR PBINZ	
02D7		39	RTS	
02D8	DMA OFF	8636	LDA# $36	TURN OFF DISPLAY
02DA		20F8	BRA DMAON+2	
02DC	1MS	7F0207	CLR TIME	DELAY FOR 1MS
02DF	1MS1	7C0207	INC TIME	
02E2		B60207	LDA TIME	
02E5		8131	CMPA # $31	
02E7		01	NOP	
02E8		01	NOP	
02E9		26F4	BNE 1MS1	
02EB		39	RTS	
02EC	49MS	8627	LDA # 127	DELAY FOR 49MS
02EE	49MS1	C69C	LDB # $9C	
02F0	49MS2	5A	DECB	
02F1		01	NOP	
02F2		26FC	BNE 49MS2	
02F4		4A	DECA	
02F5		26F7	BNE 49MS1	
02F7		39	RTS	
02F8	START	4F	CLRA	BEGIN MAIN PROGRAM
02F9		B7020A	STA ROM+1	
02FC		8609	LDA # $09	INITIALISE ROM
02FE		B70209	STA ROM	
0301		BDC079	JSR ERASE	CLEAR SCREEN
0304		4F	CLRA	
0305		972E	STA VX	
0307		972F	STA VY	
0309		4C	INCA	A = 1
030A		BD028C	JSR OUTP	DISPLAY IT
030D		8D8E	BSR SPACE	
030F		CE025D	LDX# LIST6	'2708'
0312		BD0274	JSR OUTSN	
0315		7F002E	CLR VX	CARRIAGE RETURN
0318		8606	LDA # $06	
031A		972F	STA VY	LINE FEED
031C		8602	LDA # $02	
031E		BD028C	JSR OUTP	'2'
0321		BD029D	JSR SPACE	
0324		CE026A	LDX# LIST9	'2716'
0327		BD0274	JSR OUTSN	
032A		7F002E	CLR VX	CR

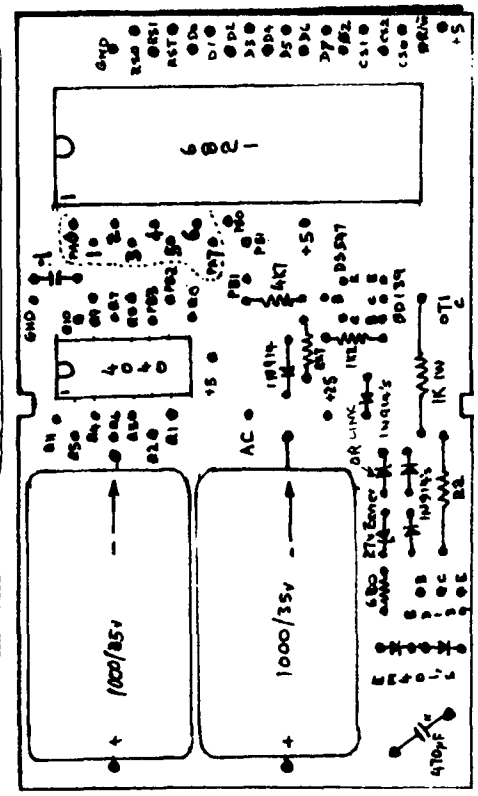
ADDRESS	LABEL	CODE	MNEMONIC	
032D		860C	LDA $0C	NEXT LINE
F		972F	STA VY	
0331		8603	LDA $03	
3		BD028C	JSR OUTP	'3'
6		BD029D	JSR SPACE	
9		CE026F	LDX# LIST A	'2732'
C		BD0274	JSR OUTSH	
F		BDC2C4	JSR GETKEY	WAIT FOR KEY INPUT
0342		4A	DECA	
3		2710	BEQ START1	IF 2708 LEAVE ROM=$09
5		C60D	LDB #$0D	IF 2716 THEN ROM=$0D
7		F70209	STB ROM	
A		4A	DECA	
B		2708	BEQ START1	
D		C615	LDB #$15	IF 2732 ROM=$15
F		F70209	STB ROM	
0352		4A	DECA	
3		26A3	BNE START	ILLEGAL INPUT - ASK AGAIN
0355	START1	BDC079	JSR ERASE	
8		4F	CLRA	CLEAR SCREEN
9		972E	STA VX	VX, Y = 0
B		972F	STA VY	
D		4C	INCA	'1'
E		BD028C	JSR OUTP	
0361		CE023A	LDX# LIST2	'VERIFY'
4		BD0280	JSR OUTSN	
7		8606	LDA #$06	LINE FEED
9		972F	STA VY	
B		7F002E	CLR VX	CR
E		8602	LDA #$02	'2'
0370		BD028C	JSR OUTP	
3		CE0242	LDX#LIST3	'PROGRAM'
6		BD0280	JSR OUTSN	
9		7F002E	CLR VX	CR
C		860C	LDA #$0C	
E		972F	STA VY	NEWLINE
0380		8603	LDA #$03	'3'
2		BD028C	JSR OUTP	
5		CE024B	LDX #LIST4	'CHECK'
8		BD0280	JSR OUTSN	
B		4F	CLRA	
C		B78041	STA CRA	SELECT DDREGS A, B BY
039F		B78043	STA CRB	CLEARING CONTROL REGS
2		43	COMA	
3		B78040	STA DDRA	MAKE PIA, B ALL OUTPUTS
6		B78042	STA DDRB	
9		8604	LDA #$04	MAKE BIT2 = 1 TO
B		B78041	STA CRA	SELECT OUTPUT REGS
E		B78043	STA CRB	
03A1	KEY	BDC2C4	JSR GETKEY	WAIT FOR KEY INPUT

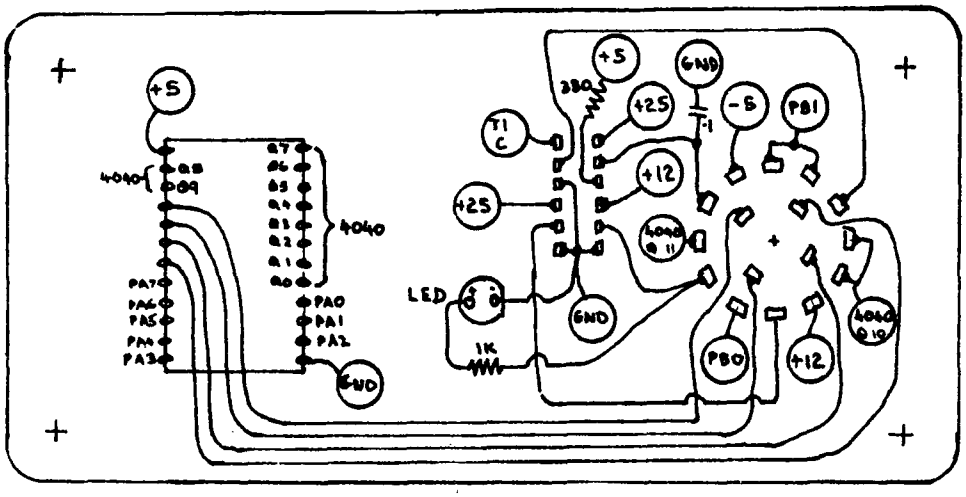
ADDRESS	LABEL	CODE	MNEMONIC	
03A4		8101	CMPA #$01	
6		270A	BEQ NEW	JUMP TO SELECTED
8		8102	CMPA #$02	ROUTINE
A		2748	BEQ PGM	
C		8103	CMPA #$03	
E		2736	BEQ CKLINK	
03B0		20EF	BRA KEY	INPUT AGAIN
03B2	NEW	BD02D8	JSR DMAOFF	ROUTINE TO CONFIRM EPROM
5		8604	LDA #$04	IS BLANK
7		B78042	STA O/R B	RESET 4040
A		7F8042	CLR O/R B	(PB2 HIGH THEN LOW)
D		4F	CLRA	
E		B78041	STA CRA	SELECT DDR A
03C1		B78040	STA DDRA	MAKE PIA INPUTS
4		8604	LDA #$04	
6		B78041	STA CRA	SELECT OUTPUT REG A
9		CE0500	LDX #$0500	X=START OF ROM IN RAM
03CC	NEW1	B68040	LDA A/R A	IN THIS CASE JUST A COUNTER
F		81FF	CMPA #$FF	GET A BYTE FROM ROM, IS IT FF
03D1		2616	BNE NEW2	- NO SO ROM IS NOT ERASED
3		08	INX	YES SO INC COUNTER
4		C608	LDB #$08	AND INC ROM ADDRESS
6		F78042	STB O/R B	(IE PB3 GOES HIGH THEN
9		7F8042	CLR O/R B	LOW TO CLOCK 4040)
C		BC0209	CMPX ROM	END OF ROM
F		26EB	BNE NEW1	NO SO CONTINUE
03E1		BD02B2	JSR OK	YES SO SAY 'OK'
4		2008	BRA NEW3	AND RETURN TO START
03E6	CKLINK	7E0490	JMP CHECK	LINK TO REACH CHECK
03E9	NEW2	0101	NOP NOP	SWI (3F) AT 03E9 PLUS DREMBUG
B		BD02C2	JSR FAIL	CAN BE USED - ELSE PRINT FAIL
03EE	NEW3	BDC2C4	JSR GET KEY	WAIT FOR KEY INPUT
1		7E02F8	JMP START	BEFORE RESTARTING
03F4	PGM	4F	CLRA	BURN ROUTINE
5		B70203	STA $C1_H$	RESET LOOP COUNTER C1
8		B70204	STA $C1_L$	
B		B78041	STA CRA	SELECT DDR A
E		43	COMA	
F		B78040	STA DDRA	MAKE PIA OUTPUTS
0402		8604	LDA #$04	
4		B78041	STA CRA	SELECT O/R A
7		BDC079	JSR ERASE	
A		7F002E	CLR VX	AND DISPLAY
D		7F002F	CLR VY	
0410		CE0252	LDX #LIST5	'SWITCH TO'
3		BD0280	JSR OUTSN	
6		7F002E	CLR VX	CR
9		8606	LDA #$06	
B		972F	STA VY	NEW LINE
D		CE0242	LDX #LIST3	
0420		BD0280	JSR OUTSN	'PROGRAM'

ADDRESS	LABEL	CODE	MNEMONIC	
0423		BDC2C4	JSR GETKEY	WAIT FOR KEY INPUT
6		BD02D8	JSR DMA OFF	THEN TURN OFF DISPLAY
9	PGM0	8604	LDA#$04	
B		B78042	STA O/R B	RESET ROM COUNTER
E		7F8042	CLR O/R B	(4040)
0431		CE0500	LDX #$0500	INITIALISE RAM COUNTER X
4	PGM1	A600	LDA 0,X	A= NEXT DATA BYTE
6		B78040	STA O/RA	SEND IT TO ROM
7		C603	LDB #$03	PREPARE TO PULSE PB1
B		B60209	LDA ROM	(PB1=O2+PB0 MUST BE HIGH FOR 2716)
E		810D	CMPA #$0D	IS ROM 2708
0440		2D06	BLT PGM2	YES SO SKIP 49MS
2		F78042	STB O/R B	PULSE HIGH
5		BD02EC	JSR 49MS	FOR 49 MS
8	PGM2	F78042	STB O/R B	
B		BD02DC	JSR 1MS	PULSE HIGH FOR 1MS
E		7F8042	CLR O/R B	TURN OFF PULSE
0451		C608	LDB #$08	
3		F78042	STB O/R B	INC ROM COUNTER VIA PB3
6		7F8042	CLR O/R B	(4040)
9		08	INX	INC RAM COUNTER
A		BC0209	CPX ROM	
D		26D5	BNE PGM1	IF ROM NOT DONE CONTINUE
F		F60209	LDB ROM	HAVING GONE ONCE THROUGH -
0462		C109	CMPB #$09	IS ROM 2708
4		260C	BNE PGM3	NO SO FINISH
6		FE0203	LDX C1	
9		08	INX	INC LOOP COUNTER
A		FF0203	STX C1	
D		8C0064	CPX #$0064	HAVE WE DONE 100 LOOPS
0470		26B7	BNE PGM0	IF NO - CONTINUE
2	PGM3	BD02D2	JSR DMA ON	YES
5		BDC079	JSR ERASE	CLEAR SCREEN
8		7F002E	CLR VX	
B		7F002F	CLR VY	DISPLAY
E		CE0252	LDX #LIST5	SWITCH TO
0481		BD0280	JSR OUTSN	
4		CE0235	LDX #LIST1	READ
7		BD0280	JSR OUTSN	
A		BDC2C4	JSR GETKEY	WAIT FOR KEY INPUT
D		7E02F8	JMP START	BACK TO START
0490	CHECK	4F	CLR A	ROUTINE TO CHECK CONTENTS
1		B78041	STA CRA	GET DDRA
4		B78040	STA DDRA	AND MAKE ALL INPUTS
7		8604	LDA #$04	
9		B78041	STA CRA	SELECT O/R A
C		8604	LDA #$04	USE PB2 TO
E		B78042	STA O/R B	RESET ROM COUNTER
04A1		7F8042	CLR O/R B	(4040)
4		CE0500	LDX #$0500	X = RAM COUNTER
7	CHECK1	B68040	LDA O/RA	GET BYTE FROM ROM

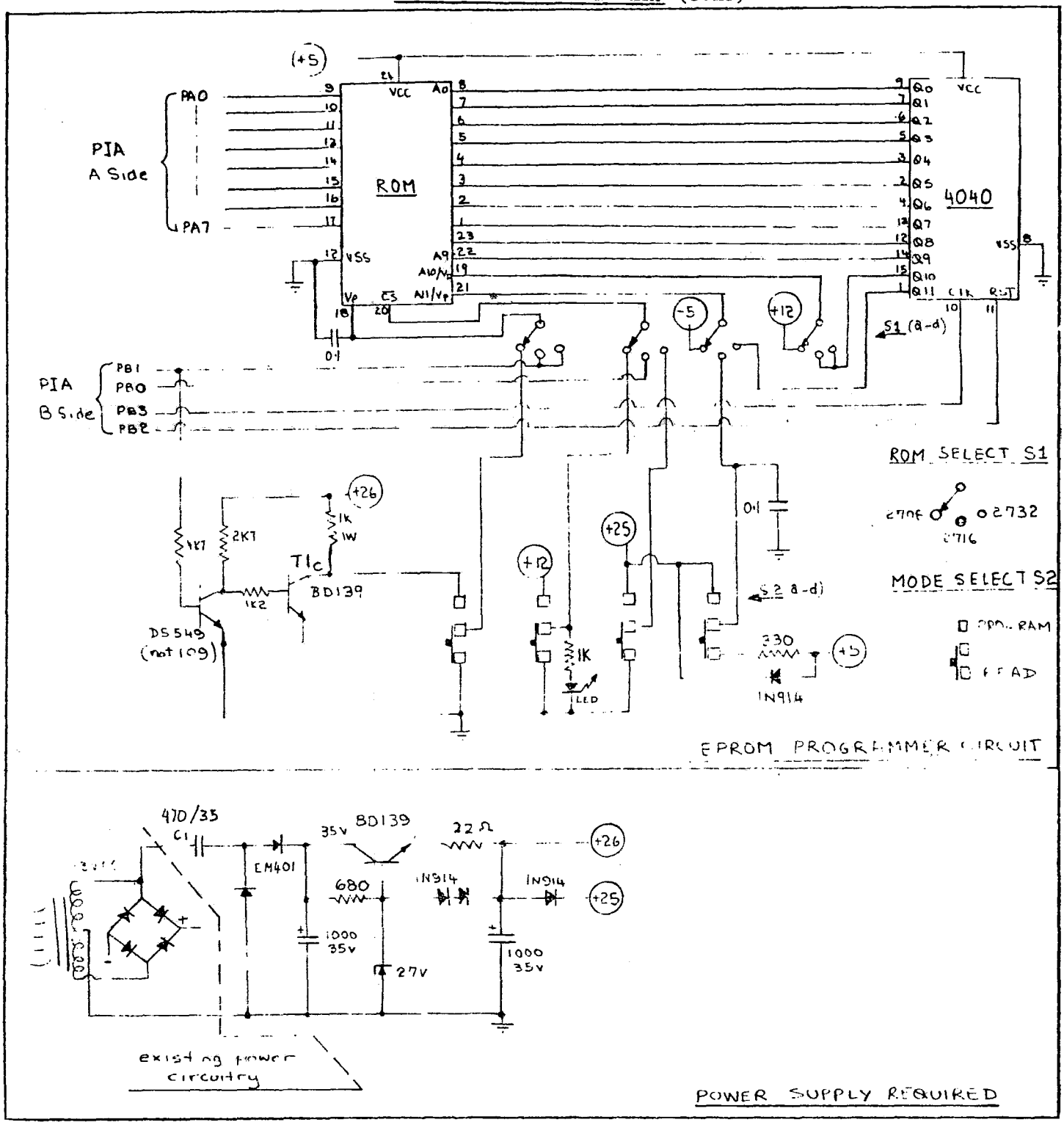
ADDRESS	LABEL	CODE	MNEMONIC	
04AA		E600	LDB 0,X	GET BYTE FROM RAM
C		11	CBA	ARE THEY EQUAL
D		2617	BNE CHECK3	NO SO GO TO FAIL DISPLAY
F		08	INX	IF SO, INC COUNTERS
04B0		C608	LDB #$08	
2		F78042	STB; 0/RB	
5		7F8042	CLR 0/RB	
8		BC0209	CPX ROM	SEE IF ALL LOCATIONS CHECKED
B		26EA	BNE CHECK1	IF NOT CONTINUE
D		BD02B2	JSR OK	ELSE DISPLAY OK
04C0	CHECK2	BDC2C4	JSR GETKEY	AND
3		7E02FB	JMP START	RETURN TO START
6	CHECK3	7E03E9	JSR NEW2	

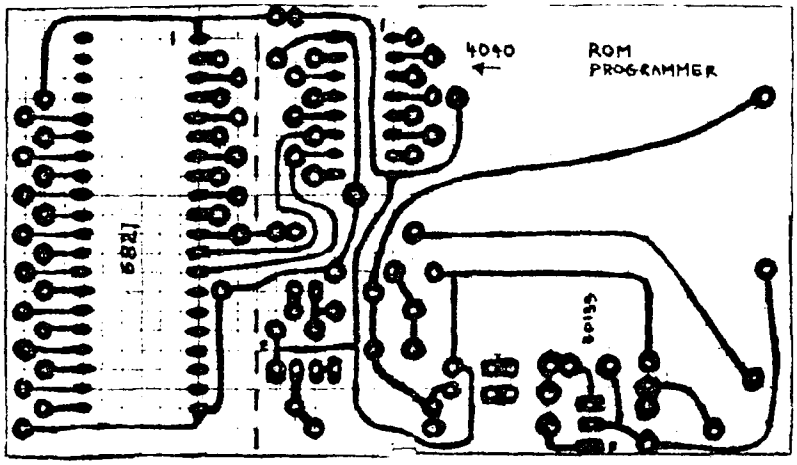


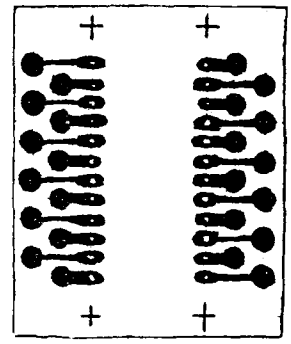




DREAM EPROM PROGRAMMER. (Cont)







****

DREAM MEMORY TESTER (0080 - 0100)

LINDSAY FORD,
"DREAMCARDS",
8 HIGHLAND COURT,
ELTHAM NORTH. 3095.

Are you having a problem with programs 'crashing' for no apparant reason? If you are sure that the load from the cassette was O.K., (start display loads perfectly every time), but the program crashes, then you may have an addressing fault on the expansion board, or a RAM chip with a faulty memory cell or cells.

To check your expansion RAM and addressing, key in the following program and RUN 0080, FN, 3.

It will take about two seconds to run, and then return to the moniter. Now check memory locations 00A8/9. If they contain '0000', then all is well and there is no addressing error or RAM bit 'stuck high'. Otherwise they will contain the address of the first defective memory location and you had better start checking the address lines, connecting wires, and RAM chips on the board.

To check for a RAM bit stuck LOW, change 0085 to '86FF', and 0097 to '81FF' and run it again.

ADDR.	INST.	MNEMONIC.	EXPLANATION.
0080	4F	CLR A	)
0081	97 A8	STA A $00A8	) Clear 00A8/9
0083	97 A9	STA A $00A9	)
0085	86 00	LDA A #00	)
0087	CE 00AA	LDX $#00AA	)
008A	A7 00	STA A $X	) Load system memory (00AA-0FFF)
008C	08	INX	) with constant.
008D	8C 1000	CPX $#1000	)
0090	26 F8	BNE $008A	)
0092	CE 00AA	LDX $#00AA	)
0095	A6 00	LDA A $X	)
0097	81 00	CMP A #00	) Check each byte of system
0099	26 09	BNE $00A4	) memory to see if constant
009B	08	INX	) retained.
009C	8C 1000	CPX $#1000	)
009F	26 F4	BNE $00A4	)
00A1	7E C360	JMP "Monitor"	)
00A4	DF A8	STX $00A8/9	) Store index of defective
00A6	20 F9	BRA $00A1	) byte at 00A8/9.
00A8	**		

****

THE CASE OF THE MYTHICAL HARDWARE ASSEMBLER

SID MOORBY, B.Sc.,
DREAM SERVICE,
[redacted]

The subject of this article is a non existant HARDWARE assembler sub-circuit in the m.p.u. chip. The sole purpose of the article is to lay to rest this GHOSTLY piece of hardware as the question of its existance has caused a fair amount of confusion amongst a large number of our non technical readers.

It is NOT the wish or intention of the writer to try to rubbish the work and efforts of another, for the articles which inadvertently caused this confusion were well written and very well presented, but, more important, were in reality correct, up to the point that the assembler being a hardware sub-circuit in the mpu. For there is indeed an assembler in the DREAM to convert Hexadecimal to Binary code, each key is assembled into what is termed (in computer terminology) a four bit nibble and stored in a register. If it is the first key of the two required to form a byte, it is shifted to occupy bits 7 - 4 and the next key goes into bits 3 - 0, BUT, it is NOT a hardware circuit that does this, it is in fact a software routine in the monitor eprom. Incidentally, it should be referred to as FIRMWARE, as it is not lost when all power is removed from the computer.

I cannot write out the routine here, (which is named "KEYPAD INPUT SERVICE ROUTINE", as to do so would be an infringement of Michael Bauer's Copyright on the CHIPOS manual, which is available from DREAMWARE. It is not expensive, and no DREAM owner should be without it.

While on the subject of assemblers, I must point out that Assembly Language, as used by Lindsay Ford in his articles, is only a means of documenting and commenting a program so that its operation can be followed and understood by others, as after you have finished writing your mnemonics, you must still go over it again to enter the correct op-codes for each instruction, which vary with the different addressing modes, calculate branch off-sets, etc.

On the other hand, true assembly language programming requires that an assembler program be resident in your computer and through the medium of an ASCII keybosrd, you actually type in the mnemonics of the instructions and the assembler selects the correct addressing mode by the use of the symbols #, $, *, and calculates branch offsets etc. It works in a very similar way to the Chip-8 interpreter, except that it stores the program in memory after translating it into binary code.

HAPPY DREAMING TO ONE AND ALL.

****

DREAM OWNERS AND READERS

As you probably know, this issue of the DREAMER may very well be the last one to be printed.

In view of this, I am thinking of starting a SOUTH AUSTRALIAN DREAM 6800 TECHNICAL NEWSLETTER, concentrating more on Machine Code and Assembly programming, such things as the development of almost full screen display and several other new developments.

HOWEVER, before I can start, which would be early in the new year, I must have some idea of the number of Dreamers who would be interested in a subscription to such a publication, with reader participation in submitting ideas, articles, programs, etc. I must have about 100 subscribers for it to pay for printing, registration and all costs in production and postage.

RATES are anticipated at $18-00 for a SIX MONTHS SUBSCRIPTION, but this will depend on the number of subscribers. With a lot it could be a fair bit less!

Will readers interested please write to:

SID MOORBY, DREAM SERVICE, [redacted]

****

TOWER OF HANOI (0200 - 0400)

K. SEMRAD,
[redacted]

All you have to do is transfer the seven discs from 1, to 2 or 3, one by one, but never put a larger one on top of a smaller one.

First key pressed is FROM, second key pressed is TO. E.G. If you want to move from 3 to 1, press key 3, then key 1.

All other keys are ignored. At the end of the game, press any key to restart.

```
0200  13CA 6100 6200 A080 F21E F155 4228 1218
0210  7001 7101 7202 1206 6200 6000 A08E F21E
0220  F055 421A 122A 7202 121C 6D00 6A00 6B1F
0230  A279 DAB1 7A08 3A40 1230 6A0A 6B1E A207
0240  DAB1 7BFF 3B0C 1240 7A16 3A4C 123C 6200
0250  A080 F21E F165 2354 2376 4228 1262 7202
0260  1250 6407 13C4 3E01 128C 6200 A080 F21E
0270  F065 3000 1280 7202 74FF 3400 126C 1298
0280  8300 6000 A080 F21E F055 12A0 3E02 1294
0290  620E 126C 4E03 129C 02F4 1262 621C 126C
02A0  6407 1348 3E01 12E4 620C A080 F21E F065
02B0  4000 12C0 72FE 74FF 3400 12AA 02F4 12A0
02C0  8030 A080 F21E F055 420C 12E2 421A 12E2
02D0  4228 12E2 7202 A080 F21E F065 8305 3F00
02E0  132E 12FE 3E02 12EC 621A 12AA 3E03 12BC
02F0  6228 12AA 8620 9721 C640 BDC2 E539 7D01

0300  6700 661C A08C F61E F065 8704 76F2 36F2
0310  1304 3000 7701 00E0 122C F000 6A00 6200
0320  2394 3A40 1320 23A6 F00A 00E0 1200 6A00
0330  6210 2394 3A40 1332 6A28 6B00 F829 DAB5
0340  6A3C F929 DAB5 1328 FE0A 89E0 12A4 FE0A
0350  88E0 1266 6507 6A09 6B11 8155 3F00 1368
0360  8154 8B14 8B14 00EE 8155 3F00 1372 6A1F
0370  1360 0A35 1362 A207 8CA0 7C02 4000 00E0
0380  DAB1 DCB1 7AFF 7C01 70FF 137C 9630 CE03
0390  EC7E C198 6B00 A3D0 F21E F065 038C DAB5
03A0  7A04 7201 00EE A0FC FD33 F265 6A00 6B00
03B0  F029 23BE F129 23BE F229 23BE 00EE DAB5
03C0  7A04 00EE 3707 134E 131C 6700 6001 1202
03D0  0000 0000 0000 0000 0000 0503 0601 0001
03E0  0102 0203 0200 0000 0000 0000 0304 0000
03F0  F3CE BBDE F6DE B6DE B7FA 56DA C546 0000
```

****

FUTURE DREAMING

I was disappointed to read the announcement of the final issue of the 'Dreamer' as this has been a vital support to DREAM users, but I can appreciate how taxing on time this venture is. Rather than see a good thing end, my friend and I are offering to take over the printing and editorship of the magazine. We are prepared to spend the time necessary to undertake such a venture.

The next six months subscription can be sent to the address below, as we can take over from February. We would urge subscribers to send new material too, as it is the subscribers who keep the newsletter going. (We will offer the same voucher system as before.)

Please include a S.A.E. so that we can send you an introductory letter telling you about ourselves and what we intend to do.

TO: ASHLEY EMERY, [redacted]

ADVERTISING

FOR SALE: DREAM 6800 with 3K RAM, 3A Power Supply, Joystick, 2K EPROM, Bags of software. Works well, looks good. (Selling because I now have a big system.) Price negotiable. Phone RAY SCHMIDT, [redacted] (H) or [redacted](W)

++++++++++++++

FOR SALE: CENTRONICS 779 PRINTER, the same as the one all the programs for the Dreamer are printed on. Has had very little use, complete with software and instructions to hook it up to your DREAM. (Or an APPLE.) $500-00 or offer. Phone GARRY on [redacted], or GRAEME on ([redacted].

++++++++++++++

FOR SALE: DREAM 6800. We believe this to be the most highly improved DREAM around. Comes complete with J.R. Components Expansion board, fitted with 6K RAM, Dreamsoft No.1 and No.2 EPROMS, Joystick, Invader Control, Sound Effects Generator, Indicator LED's, and everything else that has appeared in the Dreamer. All articles and Instructions for the DREAM and Add-ons are included, along with approximately 200 programs on tape. Included are a 3A Power Supply and Smoke Tinted case. Asking $350-00. Phone GRAEME [redacted]

++++++++++++++

FOR SALE: DREAM 6800. Similar to Graeme's, except it does not have a Sound Effects Generator, and is housed in a home made clear perspex box. $200-00 or offer, Phone GARRY, [redacted], NOTE: Only being sold to make way for a new colour computer, as 'the boss' says I can not have two.)

****

ANNOUNCING THE BIG ONE!

Wondering what to do with all that space in your expansion board memory? ------ Why not fill it with Dream Pontoon?

Dream Pontoon is that exciting card game Pontoon 21 translated into Chip 8. It has 4K of powerful logic that not only makes it a damned good player, but also results in a versatile game that can be played for hours without becoming boring.

IT FEATURES:
* Memory mapped card deck for absolute realism
* Fully floating player options (anything you can do your Dream can do better!)
* Probability based betting routines give high skill
* Automatic level of play settings and checksum

This is the biggest and most intelligent programme available for the Dream. To hell with Level II Basic, load this one up and see how smart a Dream can be.

Cassette and Instructions $17.50 Fully Commented Listing $7.50 Extra

Dream Rummy is an easy game to learn and great fun to play. High intelligence, memory mapped card deck, manual checksum and level of play settings give it reliability and realism. A bonus game of "Strip Jack Naked" is supplied free with this game - both require 2K, although "Strip Jack Naked" can be cut to 1K.

Cassette and Instructions $10.00 Commented Listing (Rummy only) $5.00 Extra

* DREAMCARDS
8 Highland Court, North Eltham 3095 Vic.
SOFTWARE THAT THINKS

ADVERTISING (Cont)

FOR SALE: DREAM 6800 Computer with JR 8K Expansion board and power supply. A DREAM to use. $230-00 the lot. N.HARLICK, [redacted]

++++++++++++++++

DREAMWARE
P.O. BOX 343
BELMONT, VIC., 3216

Please note: Stocks of Dream Invaders listings (i.e. 6800 Assembly language) are now depleted; NO FURTHER ORDERS WILL BE ACCEPTED. However, the Instructions plus HEX listing can still be supplied at $5.00 (sorry, no cassettes).

Also, the last print run of CHIPOS Manuals has sold out. Photocopies only available; $10.00

++++++++++++++++

DREAM USERS OF S.A.

The Dream Users of S.A. Group recently sent us a copy of their first newsletter, which contained an article on CHEAP MEMORY EXPANSION, which we thought would be of interest to a lot of readers, so, with their permission, we have reproduced the newsletter on the following two pages.

Anyone interested in joining this group should contact either Milton Collins, Phone [redacted] (day), [redacted] (evenings), or, D.A. Trabucco, (Chief Programmer), [redacted]

++++++++++++++++

A message from MJB

Many users wrote to express their views on the future direction of the DREAM. Most felt that the original concept of low cost and simplicity ought to be retained. A few wanted a brand new design, with even higher resolution (e.g. 256x256) and colour plus built-in sound effects. My personal view is that such a design could not compete favorably against commercial systems like the Atari 400, Commodore VIC-20, etc, because of the expense of ASCII keyboards; unless of course you were happy to stay with a hex keypad and use the CHIP-8 language (suitably extended). And why not?

Please note that plans to develop the "HIREM" monitor EPROM have been shelved, pending further deliberation. Let me emphasize, however, that the simple patches to CHIPOS (given in the "HI-RES" article, No.14) are sufficient to make the hi-res conversion worthwhile, and that many CHIP-8 programs will require only slight modification to work properly.

I'm sure all subscribers will join me in thanking Graeme and Garry for all their hard work in producing and distributing DREAMER, which has kept the DREAM alive for so long.

Keep an eye on "Marketplace" (and letters) in Electronics Aust, just in case there are any announcements about future DREAM-type developments. THE DREAM IS NOT OVER !

(Mike Bauer)

****

IT'S HERE !!

DREAMSOFT HARDWARE

AVAILABLE NOW ... DREAMSOFT EXPANSION BOARD

FEATURES :
* High-quality, double-sided, thru-hole-plated PCB
* Full 8K RAM space
* 6K EPROM space (3 x 2716)
* 2 PIAs
* Fully buffered Address & Data lines
* Single +5v supply (fully equipped board draws less than 1.5 Amps)

DREAMSOFTWARE

NO.1 PACKAGE — Pre-programmed 2K EPROM & Handbook
* Dreamtext
* Tape Load, Dump & Verify - displayed on screen
* Block Move
* Block Compare
* Branch offset calculation
* Supertypedream.

USE IT ON OUR NEW BOARD - SPACE PROVIDED (but it can be used on a JR expansion board, EA 4K RAM board or even a scrap of Veroboard.)

NO.2 PACKAGE — Pre-programmed 2K EPROM & Handbook
* Replaces your CHIPOS EPROM
* Data retained on screen after entry
* Displayed in 2-byte blocks
* Improved tape load
* Memory review (forwards & backwards)
* CHIP-8 Mnemonics displayed on screen by single keystroke
* 9-option command loop.

HOLIDAY PROJECT SPECIAL

Order your DREAMSOFT Expansion Board and BOTH Software Packages now and save a bundle!

BONUS OFFER

Every Order for all 3 products, Post Marked before Feb. 1 1982, will qualify for our Special of $75. A saving of $10 off our regular price!

- - - - - - - - - - - - - - - - Mail this coupon now -

To DREAMSOFT
P.O. BOX 139,
MITCHAM VIC. 3132

Please RUSH the following items.

QTY	ITEM		$
	DREAMSOFT No.1 PACKAGE (Resides 1800-1FFF)	@ $30	
	Instructions for installing the No.1 Package on the EA 4K RAM board	@ $5	
	DREAMSOFT No.2 PACKAGE (Resides C000-C7FF)	@ $30	
	DREAMSOFT EXPANSION BOARD	@ $25	

A CHEQUE/MONEY ORDER IS ENCLOSED
OR CHARGE MY BANKCARD No.496 .....
Expiry date .....
Cardholder signature .....

SEND TO: Name _____
Address _____
_____Postcode_____

DREAM USERS of S.A.

NEWSLETTER NUMBER ONE

In an attempt to bring our membership list up to date, the committee has decided to send a copy of this first newsletter to everyone who has given his address, even if we haven't seen him since.

If you wish to continue as a member of the DREAM users group please communicate with us, either by attending the next meeting or by phoning Milton Collins on [redacted].

A charge of $1 for each two newsletters will be made to cover costs.

PROGRAM FOR NEXT MEETING

(To be held on Monday 28-9-81 in room C204 of Regency Park Community College)

7.00 pm - General discussion; software exchange
- Experienced programmers assist beginners
- Demo. of subroutines available on the Dreamsoft EPROM

8.00 pm - Hardware fault-finding session using oscilloscopes and logic probes.

REPORT ON LAST MEETING (24-8-81)

Despite an initial mix up with room bookings a group of about twelve finally got together for a very productive meeting.

It was deciced to form a committee, and the following officers were elected:-

Chairman: Paul Marshall

Programmer: Dean Trabucco

Memsec.: Milton Collins

Many thanks to Ray Gatt for his rather startling demo. of the Sound Effects Generator - Dream Invaders will never be the same again, without sound.

Thanks also to Ashley Martin for his demo. of the Dreamsoft EPROM and its capabilities. We must do something about the automatic screen erase, since it is not always desirable.

The meeting finished with suggestions for activities at future meetings. If you have any suggestions, please pass them on to the committee.

CHEAP MEMORY EXPANSION

DREAM USERS of S.A.

By D. Trabucco

2114 RAM chips are now available for less than $3 but it seems hardly worthwhile in acquiring a JR board or spending hours wiring up a "veroboard" just for an extra 1k of RAM to run 2k programs which have appeared to date.

Providing you're not a "purist" then a technique called "piggy-backing" can provide the means of expanding the basic DREAM to 2k RAM at minimal cost.

Parts List

2 x 2114 - 1k x 4 RAM
1 x 7400 - Quad 2 input NAND gates

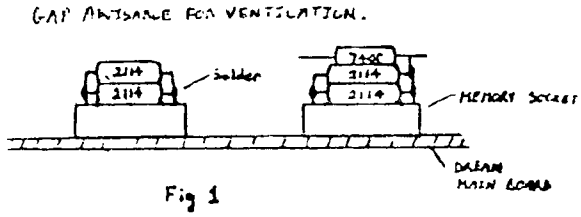


Fig 1

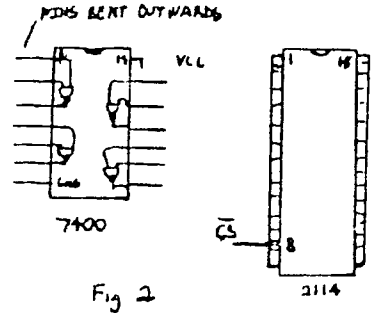


Fig 2

Procedure

1) Take 2114's and solder directly, pin for pin, on top of existing RAM (observing correct orientation), except for pin 8($\overline{cs}$) which is bent outwards. See figure 1 & 2.

2) Take 7400 and bond all pins outward excepting pin 14 (Vcc). The 7400 can then be placed on either RAM set and attached by soldering pin 14 to pin 18 of RAM. See figures 1 & 2.

3) Pin 7 (GND) of 7400 can be wired to pin 9 of RAM. The two $\overline{cs}$ (pin 8's) of RAM can be wired together.

4) Cut track between pin 6 IC 10b and 13 of 12d on main board.

5) Wire rest of 7400 and $\overline{cs}$ as per circuit diagram (figure 3). Drill hole in board taking care not to cut any tracks other than specified in (4), use it to pass wires through to underside of board.

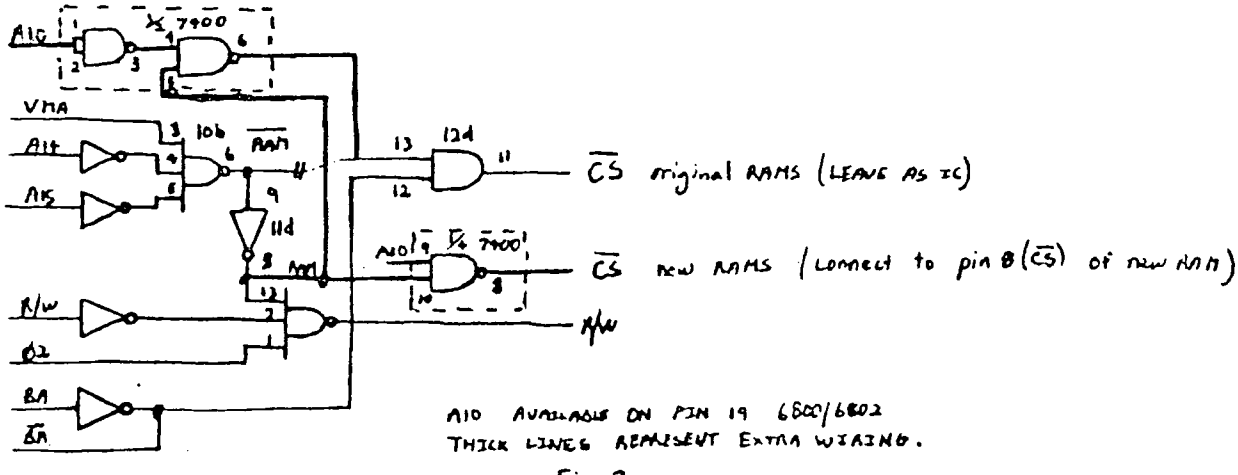


Fig 3

JOYSTICK SERVICE ROUTINE II

M. MEE
[redacted]

In reply to Michael Bauer's CHALLENGE and because I saw many uses for it, I wrote this program that will convert the joystick to a 'thrust' control. While this may be done simply in CHIPOS, it is awkward. Unlike a CHIPOS version, this machine code adaption of the Joystick Service Routine (See Nov.'80 Newsletter) is 'transparent'. Instead of mapping the joystick position onto the screen, it provides an increment in VC and VD that ranges from -2 to +2 (FE to 02 Hex). This can be used to control the thrust or acceleration of an object by adding VC and VD to the screen co-ordinates of that object. As most keyboard based programs operate this way, (e.g. 'Wipe-Off'), this also makes these programs easier to convert to joystick control.

Joystick Service Routine II is basically Michael Bauer's program with a different ending. It works by dividing 74 into 5 equal groups and consinging these groups the values -2, -1, 0, +1, +2. This value is returned as VC/VD. (See Assembly Language listing.) As this program does take longer to execute than its brother, it may be necessary to change location 0228 so that the joystick is sampled less often. Note that the rate of sampling is not as important now since there is a fair amount of travel between the different increments.

ASSEMBLY LANGUAGE LISTING (Of changes.)

```
0200   0204   CALL M/C AT 0204
0202   1268   GO TO 0268          (Note different address.)
```

As in Joystick Service Routine by M.J.Bauer.

(See November 1980 Newsletter.)

```
024F    96 3C       LDA    A VC      Fetch X-Coordinate
0251    BD 02 5E    JSR    CALC      Calculate increment
0254    D7 3C       STA    B VC      Store increment
0256    96 3D       LDA    A VD      Repeat for Y-Coordinate
0258    BD 02 5E    JSR    CALC
025B    D7 3D       STA    B VD
025D    3B          RTI              Finished !   Return
025E    C6 FD       LDA    B#-3      Initial value for Acc.B
0260    5C          INC    B         Increment Acc.B, Top of loop
0261    80 10       SUB    A#$10     Divides 74 into 5 parts
0263    2D 02       BLT    0267      Negative?
0265    20 F9       BRA    0260      No, try again
0267    39          RTS              Yes. Acc.B is new value for
                                     VC?VD.  Return.
```

HEX LISTING OF COMPLETE PROGRAM.

```
0200   02 04 12 68 C6 3B F7 80   13 C6 61 F7 80 12 C6 3F
0210   F7 80 13 CE 02 19 DF 00   39 7A 00 20 7A 00 21 7D
0220   80 12 7C 00 16 96 16 84   03 27 01 3B 86 FF 97 3C
0230   97 3D 7F 80 12 86 21 B7   80 12 C6 4A B6 80 12 46
0240   46 24 03 7C 00 3C 46 24   03 7C 00 3D 5A 26 ED 96
0250   3C BD 02 5E D7 3C 96 3D   BD 02 5E D7 3D 3B C6 FD
0260   5C 80 10 2D 02 20 F9 39
```

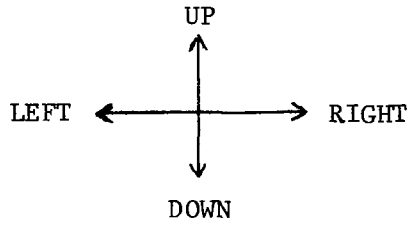
****

JOYSTICK SERVICE ROUTINE II

M. MEE
[redacted]

A DEMONSTRATION PROGRAM

This program, as the subtitle suggests, demonstrates the Joystick Service Routine II. It is very simple to operate. Load JSR II and type in the program shown below after it. Run as any CHIPOS program, i.e., COOO, FN. 3. A dot will appear in the top left hand corner. This dot will move according to the joystick position as shown below. Note that when the joystick is centred, the dot remains stationary; not necessarily in the centre of the screen. This is the difference between the two service routines.



JSR II DEMONSTRATION PROGRAM (In CHIPOS)

```
0268   00E0   ERASE                    Clear Screen
026A   6A00   VA = 00                  Initial Co-ordinates of Dot
026C   6B00   VB = 00
026E   6908   V9 = 08                  Time Delay. (Smaller for faster)
0270   A284   I = 0284                 dot is at this location
0272   8AC4   VA = VA + VC             Add increment to co-ordinates
0274   8BD4   VB = VB + VD
0276   DAB1   SHOW 1 AT VA , VB        Display dot
0278   F915   TIME = V9                Initialise time
027A   F807   V8 = TIME                Get time
027C   3800   SKF V8 = 00              Time yet?
027E   127A   GO TO 027A               No, try again
0280   DAB1   SHOW 1 AT VA , VB        Yes, remove dot
0282   1272   GO TO 0272               Get new increment
 0284   80**   DATA                    ...for dot.
```

****

AUSTRALIAN V.I.P. USER GROUP

TO ALL DREAM OWNERS

We appreciate that you will miss your regular monthly supply of games etc when the DREAMER ends, and are pleased to be able to offer you an associate membership in the A.V.U.G. for $10-00, (Six months period), or $20-00. (12 months period), which will entitle you to receive our monthly V.I.P. GAMES DIGEST for the relevant period. (Please note that as from 1/1/82 the amended Sales Tax regulations come into force and we will no longer SELL a newsletter, we will distribute programs and information FREE to financial members of the group.)

Each digest contains at least 5 new games, as well as tips, hints, modifications to existing games, etc.

Please direct enquiries or your order to,
THE EDITOR, A.V.U.G. GAMES DIGEST, P.O. BOX 41, BEXLEY. 2041.

****

DREAMSOUND COMPOSER (0200 - 0800)

Russ VERNON, [redacted]

This program uses Mike Bauer's DREAMSOUND generator to play music as he hinted in his article in DREAMER, ISSUE 8. This program allows you to type in encoded music, see a graphic display of what youhave typed, select a tempo, and then have DREAMSOUND play it for you.

The program is in machine language and resides between 0200 & 0600 in RAM. 0610 onwards is the storage area for notes and note lengths (SONGLIST). RAM 0030 - 0042 is used for storage of variables. Locations 0031,32 (POINTER) hold the position of the last note entered in the SONGLIST.

When run from 0200 the program enters the:

MONITOR CONDITION from which you may select;

0 Play current song,
1 Go to COMPOSE mode,
2 Reset the POINTER to 0610 & clear SONGLIST (fills it with #FF),
3 Change the tempo currently being displayed (type a 2 digit no.)

COMPOSE MODE (typing 1 in the MONITOR condition got you here)

0 Returns you to MONITOR above

1st entry: 1,2,3or4 selects one of 4 octaves,
5 selects a rest (then jumps to 3rd entry below)

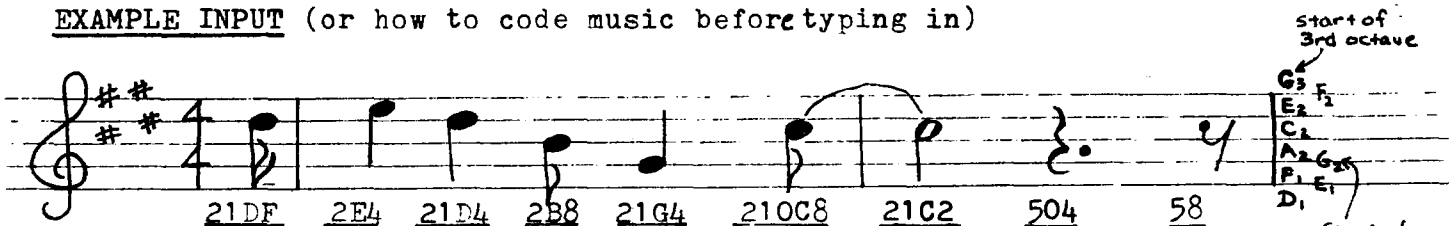
2nd entry: (selects the note, but first:)
0 slurs the note (ie. runs it into the next) } one, both or
1 makes the note a semitone sharp, and now } none can be
9,A,B,C,D,E,F select the note(9=G) used

3rd entry: (length of note, but again:)
0 produces a dotted note (i.e. length of note plus 1/2 again)
1 whole note 𝅝 ; 2 1/2 note 𝅗𝅥 ; 4 1/4 note ♩
8 1/8 note ♪ ; 9-F 1/16th note 𝅘𝅥𝅯

If a rest was selected as the first entry then:
1 whole note rest 𝄻 ; 2 1/2 rest 𝄼 ; 4 1/4 rest 𝄽
8 1/8 rest 𝄾 ; 9-F 1/16 rest 𝄿 .

After typing in the length, your note is displayed and you are returned to the COMPOSE mode above for the next note.

EXAMPLE INPUT (or how to code music before typing in)



21DF	2E4	21D4	2B8	21G4	210C8	21C2	504	58
oct 2 sharp note D 1/16 long	noteE 1/4 note		1/8 note		oct2 sharp & slurred C,1/8 long		rest dotted 1/4 long	rest 1/8 long

This is the start of"The Long and Winding Road" listed below, after the COMPOSER has translated 21DF etc. into note values and lengths. Type in DREAMSOUND COMPOSER 0200-0600, and then the SONGLIST. GO from 0200 and type "0" to hear it. (set tempo to 08 first).

SAVING SONGS

SONGLISTS can be saved in the usual manner with the TAPE commands (FN1 & FN2). Use FN0 to set start & end of dump (0610, Pointer).

NOTE: The screen buffer wraps around when about twelve notes are input. At this stage, check your inputs so far, then type '0', then '1', to clear the screen. (This also updates the POINTER value.)

If you know someone else with a DREAM, then it could play an accompaniment to enrich the sound. Adjustment of the 1K VCO tune pot. on the DREAMSOUND board will enable the system/s to be tuned to pitch if required.

```
0200   C6 05 BD 04 8A 7F 80 22  7F 80 23 B7 80 22 C6 3C
0210   F7 80 21 BD C2 87 BD 04  EC BD 02 39 27 3D 4A 27
0220   40 4A 27 05 4A 27 2D 20  ED CE 06 10 DF 31 86 FF
0230   7E 04 50 01 01 01 01 01  01 BD C2 C4 4D 39 CE 80
0240   20 C6 38 E7 01 C6 FF E7  00 C6 3C E7 01 A7 00 C6
0250   34 E7 01 39 BD C3 90 97  30 20 BB 7E 03 00 7E 02
0260   16 BD 03 49 8D D3 2F AE  81 05 27 52 81 06 27 4B
0270   22 A4 97 33 BD 03 96 5F  D7 35 D7 39 BD 02 39 26
0280   05 7C 00 39 20 F6 81 01  26 05 7C 00 35 20 ED 81
0290   09 2D 83 97 3A 80 09 48  9B 35 D6 33 5A 27 04 8B
02A0   0E 20 F9 8B A2 97 37 20  03 7E 02 16 DE 36 A6 00
02B0   DE 31 A7 00 08 DF 31 7E  04 90 A3 7E 02 16 7F 00
02C0   3A DE 31 6F 00 08 DF 31  8D 04 7E 04 99 96 BD 04
02D0   20 27 10 54 81 02 27 0B  54 81 04 27 06 54 81 08
02E0   27 01 54 D7 34 96 3B 27  03 54 DB 34 DE 31 96 39
02F0   27 01 50 E7 00 08 DF 31  39 03 A6 39 01 01 01 01

0300   86 00 BD 02 3E 7F 80 22  CE 06 10 A6 00 81 FF 26
0310   05 7F 80 22 20 93 B7 80  22 A6 01 BD 03 2A 08 08
0320   20 E9 BD C0 79 C6 14 D7  2F 39 16 2A 01 50 96 30
0330   D7 3D 5F DB 3D 4A 2E FB  D7 20 7D 00 20 26 FB A6
0340   01 4D 2B 03 7F 80 22 39  39 BD 03 22 01 C6 00 D7
0350   2E 86 0B BD 05 66 86 00  BD 05 7B 86 08 BD 05 66
0360   86 07 BD 05 66 86 09 BD  05 66 01 01 01 01 01 01
0370   C6 1A D7 2F C6 00 D7 2E  86 01 BD 05 66 86 09 BD
0380   05 66 D6 2E CB 04 D7 2E  BD 05 56 C6 00 D7 2E D7
0390   2F 39 01 01 01 01 96 33  BD 05 7B C6 06 D7 2F D6
03A0   2E C0 04 D7 2E 39 BD 04  5B 01 BD 05 66 BD 04 32
03B0   96 35 27 11 7C 00 2E CE  04 42 C6 07 BD 05 8D 96
03C0   2E 80 06 20 04 96 2E 80  04 97 2E 86 0E 97 2F 96
03D0   34 C6 FD 5C 48 26 FC CB  0E 17 BD 05 66 96 2E 81
03E0   3B 26 02 86 FF 4C 97 2E  96 39 27 12 86 0B 97 2F
03F0   CE 04 40 C6 02 BD 05 8D  96 2E 80 02 97 2E 96 3B

0400   27 0F 86 0E 97 2F 96 2E  80 02 97 2E 86 16 BD 05
0410   66 96 2E 4C 97 2E 7F 00  2F 39 03 70 39 03 4D 39
0420   7F 00 3B BD 02 39 26 05  7C 00 3B 20 F6 C6 10 81
0430   01 39 96 2E 81 37 2D 03  4F 97 2E 39 C0 41 E1 63
0440   88 F8 10 58 70 D8 70 D0  40 00 D8 A8 A8 A8 88 00
0450   A7 00 08 8C 07 FF 26 F8  7E 02 16 96 3A 27 03 80
0460   09 39 86 06 97 2F 97 3C  96 34 C6 FF 5C 48 26 FC
0470   C0 03 58 CB 80 D7 41 DE  40 BD 05 6D BD 03 FE 39
0480   E1 70 E8 70 51 A8 88 BA  89 BA D7 36 5A D7 40 39
0490   BD 02 CE BD 03 A6 7E 02  64 BD 04 5B 7E 02 64 58
04A0   42 72 7F E2 E4 5F A0 93  DF F3 44 D1 DD 57 F7 FB
04B0   C3 73 DF F1 F9 CD EA 49  E7 D9 EB 7B 42 C2 DB C3
04C0   84 E3 C1 13 47 41 61 41  F7 B7 F3 D2 FD E5 C0 5B
04D0   9A E4 51 DB 5B 49 F4 80  10 75 F5 19 C3 E1 7A 5D
04E0   4B EB C9 F0 A9 EF C2 F2  A3 FA 76 A2 BD C0 79 C6
04F0   00 D7 2E D7 2F CE 04 4A  C6 05 BD 05 8D 86 00 BD
```

```
0500   05 7B 86 07 BD 05 66 86   08 BD 05 66 86 09 BD 05
0510   66 86 00 BD 05 7B 86 0A   BD 05 66 C6 07 D7 2F C6
0520   00 D7 2E 86 09 BD 05 66   86 05 BD 05 66 CE 04 4A
0530   C6 05 BD 05 8D 86 0B BD   05 66 86 00 BD 05 7B BD
0540   05 87 96 30 44 44 44 44   BD 05 7B 96 30 84 0F BD
0550   05 7B C6 1A D7 2F CE 00   31 FC 00 2E BD C3 C8 CE
0560   00 32 BD C3 C8 39 D6 2F   D7 3C CE 05 DA BD C1 9A
0570   C6 05 BD C2 24 D6 2E CB   04 20 1F D6 2F D7 3C BD
0580   C1 93 C6 05 BD C2 24 D6   2E CB 04 20 0D 96 2F 97
0590   3C 01 01 BD C2 26 C6 06   DB 2E D7 2E D6 3C D7 2F
05A0   39 01 89 8A 8B 8B 8C 8D   8D 8E 8F 90 91 92 92 93
05B0   94 95 97 98 99 9A 9A 9C   9E A0 A2 A4 A4 A6 A9 AB
05C0   AE B1 B4 B4 B7 BA BD C0   C3 C6 C6 C9 CC CF D2 D5
05D0   D9 DD DD E1 E5 E9 ED F1   F1 F6 F6 CE B7 DF D7 DD
05E0   F2 4F D6 DD F3 CF 93 4F   B6 DE E9 2E 49 2E BB DE
05F0   93 DE F2 48 49 7A C5 47   F7 80 F7 92 FC 92 D9 26

0600   D9 A6 88 FA 51 A8 08 00   01 10 01 01 FF FF FF FF
0610   A0 01 A2 04 A0 04 99 02   95 04 9C FE 9C 08 9C FC
0620   95 02 99 FE 99 F8 9C 02   9E 04 9C FE 9C 10 00 04
0630   9C 04 A0 02 A2 04 99 01   95 01 9C 10 00 02 8E 02
0640   97 02 97 02 99 04 9C 02   99 FE 99 10 00 04 9C 04
0650   A0 02 A2 04 99 FE 99 02   95 02 9C 08 00 04 9C 02
0660   9C 02 A0 04 A0 FF A2 03   A2 0C 00 08 A2 01 A2 01
0670   A2 FE A2 03 A2 01 A6 01   A2 01 9C FE 9C 03 91 01
0680   95 01 99 01 91 FE 91 02   95 02 99 08 A2 01 A2 01
0690   A2 FE A2 03 9C 01 A6 01   A2 01 9C FE 9C 03 97 01
06A0   95 01 99 01 91 FE 91 02   95 02 99 06 99 02 A2 04
06B0   A0 04 99 02 95 04 9C FE   9C 08 9C 04 95 02 99 FE
06C0   99 08 9C 02 9E 04 9C FE   9C 10 00 04 9C 04 A0 02
06D0   A2 04 99 01 95 01 9C 10   8E 02 97 02 97 02 99 04
06E0   9C 02 99 FE 99 10 00 04   9C 04 A0 02 A2 04 99 FE
06F0   99 02 95 02 9C 08 00 04   00 04 9C 02 9C 02 A0 04

0700   A0 FF A2 03 A2 10 00 04   FF FF 00
```

****

500 BAUD CASSETTE

(0080 - 0100)

ROWAN McKENZIE,

[redacted]

After experimenting with different Baud rates, I found that 500 Baud was the highest reliable speed to load and save programs at.

To use the program, store the start and end data at 0002 as with the normal CHIPOS monitor, then run from 0080. Pressing key 1 will load data from cassette at 500 Baud, while key 2 will save data.

If you would like to experiment with other Baud rates, change the data at 00D8.

The program is fully relocatable to allow any section of memory to be saved or loaded.

```
0080   BD C2 C4 81 01 27 13 81   02 26 F5 BD C3 41 A6 00
0090   8D 2D 08 9C 04 26 F7 7E   C3 60 BD C3 41 8D 0A A7
00A0   00 08 9C 04 26 F7 7E C3   60 BD C3 25 A6 00 2B FC
00B0   8D 24 C6 09 0D 69 00 46   8D 1A 5A 26 F7 20 12 BD
00C0   C3 25 36 6A 00 C6 0A 8D   0B A7 00 0D 46 5A 26 F6
00D0   32 DE 12 39 8D 00 37 C6   7B 5A 01 26 FC 33 39 01
```

****

AN OPEN LETTER TO THE EDITORS

First of all, we would like to take this opportunity to thank Graeme and Garry for their fine efforts in producing the 'DREAMER' over the last 16 months. You did a great job, Fellas and it's a great pity that it has to finish.

We understand that there may be a successor and if so we would wish him well because, like you, we are interested in maintaining the concept of the DREAM. It is, despite its low cost, potentionally a very powerful computer and we have a number of projects in the pipeline which will enhance its capabilities even further.

Some possibilities are:-

* Morse Code generation from an ASCII keyboard input.
* Telephone Answering.
* light show controller.
* Full Machine Code disassembler.
* Teletext decoding.
* Automatic labelling and loading of tapes.
* Model Railway controller.
* Burglar Alarm systems.

We have already written some of these, but in the immediate future we are planning a few other surprises, like:

1. A PROM board containing up to 16 chips (may be 2708s or 2716s) and which is accessed via a PIA (such as one on our new expansion board). Thus a very large amount of software may be stored and copied into a working RAM area for execution.
2. A Customised EPROM Programming service. Readers will be able to send us tapes and we will put the data on EPROMS.
3. We are already well underway in the writing of a universal programmable controller which will complement DREAMTEXT and allow users to control a whole range of external devices against a real time clock.
4. We also have plans to publish a few sensing circuits which will let the computer monitor or measure parameters in the outside world, such as temperature, light, sound, water level etc.
5. A Hardware-encoded ASCII keyboard.
6. Another Software package to tie all this together.

We feel sure that there must be many enthusiasts, who, like ourselves, want to keep the DREAM alive and share ideas. To this end we would like to invite all your readers to put their names on our mailing list so that we can keep them up to date with our developments free of charge.

That's the way we're going at the moment but if you or your readers have other ideas please let us know, we are always open to suggestions.

We wish you luck for the future but, please, don't buy anything with a Z-80 in it!

Regards.....

Graham Leadbeater and Graeme Hollis

DREAMSOFT.

****

CALCULATOR PROGRAM (0080 - 0700)

MARTIN HEAD
[redacted]

Here it is! A 'useful' program for Dreamers who want to come back from outer space, or whatever. This is full Machine Language (gasp!) and converts the Dream into a four function floating point calculator.

Key functions are:-

A0 = Addition
A2 = Multiplication
A4 = Division
A6 = Subtraction
A8 = Enter
AA = +/-
AC = Clear Entry
AE = Exchange

HOW TO RUN THE PROGRAM.

1. Key in address 04C0 and press Function 3. (The screen should now go blank.)

The calculator uses a 'homebrew' form of Reverse Polish Notation (like Hewlett Packard's....) which differs in that the results of calculations must be entered if they are to be used again. In general, RPN works like this;

If you want to add 2 and 3 you key, 2, Enter, (A8), 3, Add. (A0).
To evaluate 6 multiplied by 4, you key, 6, Enter, (A8), 4, Multiply. (A2).
To evaluate 8 divided by 4, you key 8, Enter, (A8), 4, Divide. (A4).
I will illustrate this with a simple series of calculations on the computer.

2. Try entering a number. e.g. -2.4×10^{-3}. Do it like this;
   Key 2 D 4 AA EE 03 AA A8
   (2 . 4 +/- Exp 03 +/- Enter.)
   After pressing A8, the number should be displayed on the screen.

3. Put in another number and multiply the two. e.g. 1.2×10^{5}.
   Key 1 D 2 EE 05 CC A2
   (1 . 2 Exp 05 Mult.)
   (CC lets you see that you have put in the right number before you actually multiply them, it is not vital.)
   After pressing A2 the correct answer should appear:-

$-2.4 \times 10^{-3} \times 1.2 \times 10^{5} = -2.88 \times 10^{2}$.

4. Let's try dividing -2.88×10^{2} by 2: First we must enter it (press A8) and now press
   2 D A4
   (2 . Divide.)
(2 = 2.0×10^{0}, and on this system, it is not necessary to specify the exponent as being 00, although the calculator works only in scientific notation.)
The correct answer should appear on screen : -1.44×10^{2}.

Getting the hang of it? I think that it is impossible to crash the program once it is entered correctly, so don't hesitate to experiment further with it.

PITFALLS.

1. Numbers must be entered in scientific notation.
   E.G. Enter .1 as 1.0×10^{-1} etc.
2. Errors, such as division by 0 or a multiplication beyond the range of the program cause it to reset itself.

MORE TITBITS.

If you would like a photo-copy of the (long),(handwritten) listing of this program, plus details of which subroutines do what, and how you may call them in programs of your own, send me a cheque or money order for $10-00, at the above address.

```
0080   BD C0 79 86 08 BD C3 E0  C6 03 CE 04 00 BD C2 26
0090   BD C3 E0 B6 03 C0 27 04  4F BD 04 F0 01 86 04 97
00A0   2E CE 03 C0 A6 02 BD C3  D2 86 03 E6 06 26 04 09
00B0   4A 26 F8 97 FF CE 03 C2  86 0A 97 2E 08 BD C3 C8
00C0   7A 00 FF 2C F7 BD C3 DC  A6 00 84 0F 26 09 D6 2E
00D0   C0 08 D7 2E BD C3 D2 7A  00 2E 86 0E BD C3 D2 B6
00E0   03 C1 27 05 96 2E BD 04  F0 BD C3 DC B6 03 CF BD
00F0   C3 CA 39 00 00 00 00 00  00 00 00 00 07 A4 00 FF

0200   86 04 C6 02 01 F7 02 0C  0C C6 09 66 0B 7C 02 0C
0210   5A 01 26 F7 4A 26 EB E6  01 A6 0F 5D 27 09 8B 99
0220   19 26 09 6F 01 20 05 8B  01 19 25 03 A7 0F 39 7E
0230   C3 60 FF 02 4F C6 09 A6  0A 26 05 09 5A 26 F8 39
0240   8B 99 19 40 8B 99 A7 0A  09 A6 0A 5A 26 F5 CE 03
0250   F0 39 FF 02 64 C6 09 0C  A6 0A A9 1A 19 A7 0A 09
0260   5A 26 F5 CE 03 C0 39 CE  03 C0 A6 00 88 01 A7 00
0270   CE 03 C0 A6 01 A1 11 26  08 A6 0F A1 1F 27 14 2D
0280   0C 6D 01 26 03 CE 03 D0  BD 02 00 20 E3 6D 01 26
0290   F4 20 F5 A6 00 A1 10 26  06 BD 02 52 7E 02 C4 4D
02A0   26 03 CE 03 D0 BD 02 32  CE 03 C0 6F 00 BD 02 52
02B0   25 05 BD 02 32 6C 00 A6  10 27 09 CE 03 D0 BD 02
02C0   32 CE 03 C0 A6 02 84 F0  27 03 BD 02 00 7E 03 94
02D0   FF 02 DC C6 10 6F 0F 09  5A 26 FA CE 03 D0 39 FF
02E0   02 F0 CE 03 EF C6 30 A6  00 A7 10 09 5A 26 F8 CE
02F0   03 C0 39 86 04 C6 0A F7  02 FE 0C C6 09 69 01 01

0300   7A 02 FE 5A 26 F7 4A 26  EC 7E 06 00 A6 11 A1 21
0310   26 0C A7 01 A6 1F AB 2F  19 24 1C 7E C3 60 4D 2E
0320   19 A6 2F E6 1F 8B 99 19  40 8B 99 1B 19 25 08 8B
0330   99 19 40 8B 99 6C 01 A7  0F 39 A6 1F E6 2F 20 E5
0340   CE 03 C0 BD 02 DF BD 02  D0 BD 03 0C A6 10 A8 20
0350   A7 00 86 08 E6 26 26 05  09 4A 4A 26 F7 4C 97 FE
0360   CE 03 C0 E6 22 27 07 BD  02 52 6A 22 20 F5 CE 03
0370   D0 BD 02 00 CE 03 E0 BD  02 F3 7A 00 FE 26 E1 CE
0380   03 C0 A6 02 84 F0 27 03  BD 02 00 A6 02 26 04 6F
0390   0F 6F 01 39 A6 02 26 03  BD 02 F3 39 17 A6 01 88
03A0   01 A7 01 39 FF 03 BA 86  10 97 FB A6 00 E6 10 A7
03B0   10 E7 00 08 7A 00 FB 26  F2 CE 03 C0 39 3F 67 4A
03C0   01 00 01 44 00 00 00 00  00 00 00 00 00 00 00 02
03D0   00 00 00 00 00 00 00 00  00 01 00 00 00 00 00 14
03E0   01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 02
03F0   00 00 00 00 00 00 00 00  00 02 00 00 00 00 00 14

0400   00 00 80 00 00 E0 00 00  09 60 28 82 06 41 81 32
0410   CE 03 C0 A6 00 88 01 A7  00 39 CE 03 C0 7E 02 D0
0420   CE 03 C0 7E 03 A4 CE 03  C0 BD 04 B0 5D 26 03 7E
0430   C3 60 BD 03 A4 BD 02 DF  BD 02 D0 CE 03 D0 BD 04
0440   B0 5D 26 01 39 CE 03 C0  6D 2F 27 06 A6 21 88 01
0450   A7 21 BD 03 0C A6 10 A8  20 A7 00 BD 02 DF CE 03
0460   D0 BD 02 D0 CE 03 C0 86  01 A7 12 86 0E 97 FE CE
0470   03 E0 BD 05 D0 5D 27 1A  CE 03 F0 BD 02 32 CE 03
0480   E0 BD 02 52 CE 03 C0 BD  02 52 CE 03 F0 BD 02 32
```

```
0500   05 7B 86 07 BD 05 66 86  08 BD 05 66 86 09 BD 05
0510   66 86 00 BD 05 7B 86 0A  BD 05 66 C6 07 D7 2F C6
0520   00 D7 2E 86 09 BD 05 66  86 05 BD 05 66 CE 04 4A
0530   C6 05 BD 05 8D 86 0B BD  05 66 86 00 BD 05 7B BD
0540   05 87 96 30 44 44 44 44  BD 05 7B 96 30 84 0F BD
0550   05 7B C6 1A D7 2F CE 00  31 FC 00 2E BD C3 C8 CE
0560   00 32 BD C3 C8 39 D6 2F  D7 3C CE 05 DA BD C1 9A
0570   C6 05 BD C2 24 D6 2E CB  04 20 1F D6 2F D7 3C BD
0580   C1 93 C6 05 BD C2 24 D6  2E CB 04 20 0D 96 2F 97
0590   3C 01 01 BD C2 26 C6 06  DB 2E D7 2E D6 3C D7 2F
05A0   39 01 89 8A 8B 8B 8C 8D  8D 8E 8F 90 91 92 92 93
05B0   94 95 97 98 99 9A 9A 9C  9E A0 A2 A4 A4 A6 A9 AB
05C0   AE B1 B4 B4 B7 BA BD C0  C3 C6 C6 C9 CC CF D2 D5
05D0   D9 DD DD E1 E5 E9 ED F1  F1 F6 F6 CE B7 DF D7 DD
05E0   F2 4F D6 DD F3 CF 93 4F  B6 DE E9 2E 49 2E BB DE
05F0   93 DE F2 48 49 7A C5 47  F7 80 F7 92 FC 92 D9 26

0600   D9 A6 88 FA 51 A8 08 00  01 10 01 01 FF FF FF FF
0610   A0 01 A2 04 A0 04 99 02  95 04 9C FE 9C 08 9C FC
0620   95 02 99 FE 99 F8 9C 02  9E 04 9C FE 9C 10 00 04
0630   9C 04 A0 02 A2 04 99 01  95 01 9C 10 00 02 8E 02
0640   97 02 97 02 99 04 9C 02  99 FE 99 10 00 04 9C 04
0650   A0 02 A2 04 99 FE 99 02  95 02 9C 08 00 04 9C 02
0660   9C 02 A0 04 A0 FF A2 03  A2 0C 00 08 A2 01 A2 01
0670   A2 FE A2 03 A2 01 A6 01  A2 01 9C FE 9C 03 91 01
0680   95 01 99 01 91 FE 91 02  95 02 99 08 A2 01 A2 01
0690   A2 FE A2 03 9C 01 A6 01  A2 01 9C FE 9C 03 97 01
06A0   95 01 99 01 91 FE 91 02  95 02 99 06 99 02 A2 04
06B0   A0 04 99 02 95 04 9C FE  9C 08 9C 04 95 02 99 FE
06C0   99 08 9C 02 9E 04 9C FE  9C 10 00 04 9C 04 A0 02
06D0   A2 04 99 01 95 01 9C 10  8E 02 97 02 97 02 99 04
06E0   9C 02 99 FE 99 10 00 04  9C 04 A0 02 A2 04 99 FE
06F0   99 02 95 02 9C 08 00 04  D0 04 9C 02 9C 02 A0 04

0700   A0 FF A2 03 A2 10 00 04  FF FF 00
```

****

500 BAUD CASSETTE (0080 - 0100)

ROWAN McKENZIE,
[redacted]

After experimenting with different Baud rates, I found that 500 Baud was the highest reliable speed to load and save programs at.

To use the program, store the start and end data at 0002 as with the normal CHIPOS monitor, then run from 0080. Pressing key 1 will load data from cassette at 500 Baud, while key 2 will save data.

If you would like to experiment with other Baud rates, change the data at 00D8.

The program is fully relocatable to allow any section of memory to be saved or loaded.

```
0080   BD C2 C4 81 01 27 13 81  02 26 F5 BD C3 41 A6 00
0090   8D 2D 08 9C 04 26 F7 7E  C3 60 BD C3 41 8D 0A A7
00A0   00 08 9C 04 26 F7 7E C3  60 BD C3 25 A6 00 2B FC
00B0   8D 24 C6 09 0D 69 00 46  8D 1A 5A 26 F7 20 12 BD
00C0   C3 25 36 6A 00 C6 0A 8D  0B A7 00 0D 46 5A 26 F6
00D0   32 DE 12 39 8D 00 37 C6  7B 5A 01 26 FC 33 39 01
```

****

```
0280    7901 3903 1232 6F00   6E00 EEA1 1392 6E02
0290    EEA1 13A0 6E01 EEA1   139A 0000 0000 0000
02A0    6900 3F01 1232 3A00   12D0 3B0D 12D0 6610
02B0    F618 76FF 6E04 20F6   3602 12B0 00E0 235A
02C0    F618 7601 6E04 20F6   3613 12C0 00E0 1600
02D0    670F 6604 F618 0300   DAB5 77FF 3700 12D4
02E0    6708 6E08 20F6 0300   230A 77FF 3700 12E2
02F0    00E0 610F 227,6 71FF  3100 12F4 00E0 1600

0300    C604 D721 C640 BDC2   E539 A38B DAB5 00EE
0310    6A10 6B08 A376 DAB5   7A08 A37B DAB5 7A08
0320    A380 DAB5 7A08 A385   DAB5 00EE 8282 8282
0330    FE82 8282 823F 2020   203E 2020 203F 8F08
0340    0808 0F09 0808 88C1   2222 22C2 0282 4221
0350    F008 0808 0808 0808   F000 6A0C 6B0B A32C
0360    DAB9 7A08 A335 DAB9   7A08 A33E DAB9 7A08
0370    A347 DAB9 126E CEA4   A4A4 CEEE 8A4E 2AEA
0380    EE84 4424 E4EE 8ACA   8CEA 00A4 0840 0490
0390    0000 23BA 7B01 23BA   12A0 23BA 7AFF 1396
03A0    23BA 7BFF 1396 0080   8000 8080 6B0D 6A00
03B0    A3A7 DAB5 00EE 6B0C   6A39 A3C0 DAB5 00EE
03C0    043C E03C 0400 0080   D341 6301 C407 7407
03D0    D341 00EE D341 7401   7301 D341 1258 D341
03E0    74FF 13D8 D781 6701   C807 7815 D781 00EE
03F0    D781 7801 7701 D781   126C D781 78FF 13F4

0400    ABB8 01C0 0CEE AE22   AA90 0140 0A8A A422
0410    AA90 01C0 0EEE A422   AA90 0140 0A8A A400
0420    5390 0140 0CEA E422   0000 0000 0000 0000
0430    0038 1DDD 00AE EE00   0028 1515 00A8 AA00
0440    0038 1DDD 00EE EA00   0028 1915 00A8 CA00
0450    0028 15D5 C0AE AE00   0000 0000 0000 0000
0460    0001 DDDD DDC0 8000   0001 1549 4940 8000
0470    0001 1DC9 C940 8000   0001 1509 4940 0000
0480    0001 D509 5D40 8000   0000 0000 0000 0000
0490    0000 0000 0000 0000   0000 0000 0000 0000
04A0    0000 0000 0000 0000   0000 0000 0000 0000
04B0    3832 3BB2 82AE ECEE   282A 2AAA 82AA AA8A
04C0    383A 2AAB 82AA AAEE   282A 2AA9 02AA AA8C
04D0    2833 BBB1 014E ACEA   0000 0000 0000 0000
04E0    0000 0000 0000 0001   0000 0000 0000 0000
04F0    FFFF FFFF FFFF FFFF   FFFF FFFF FFFF FFFF

0500    EEEA EE0E E06E EEE0   A84A AA04 A08A A880
0510    EE4A EA04 A04E E8E0   C84A CA04 A028 A880
0520    AE4E AA04 E0C8 AEE0   0000 0000 0000 0000
0530    EEEC EFA8 EEEE EEE0   A8AA 8AA8 8AA8 8440
0540    E8EA EAB8 CAEC E440   A8AA 8A90 8AC8 8440
0550    AEAC EA92 8EA8 EE40   0000 0000 0000 0000
0560    0000 8EEE EEE0 0000   0000 8488 A880 0000
0570    0000 848E A8E0 0000   0000 8488 A880 0000
0580    0000 EEEE AEE8 0000   0000 0000 0000 0000
0590    0000 0000 0000 0000   0000 0000 0000 0000
05A0    0000 0000 0000 0000   AEA0 0EEA 8C00 EEE0
05B0    AAA0 08AA 8A00 AA40   EAA0 08AA 8A00 AA40
05C0    4AA0 08AA 8A00 AA40   4EE0 0EEE EC00 AE40
05D0    0000 0000 0000 0000   CEEA E001 C00E EE80
05E0    AA4A 8001 400A A480   AE4A E001 C00A E480
05F0    AC4A 8001 400A A480   CAE4 E001 400A AEE0
```

```
0600   3A2C 1640 060C 66FF   F618 1200 CE04 00FF
0610   0635 CE05 00FF 0637   CE01 00FF 0639 FE06
0620   35A6 00BC 0637 270C   08FF 0635 FE06 39A7
0630   0008 20E7 3905 0005   0002 0000 0000 0000
0640   0000 0000 064C 66FF   F618 1200 CE05 00FF
0650   0675 CE06 00FF 0677   CE01 00FF 0679 FE06
0660   75A6 00BC 0677 270C   08FF 0675 FE06 79A7
0670   0008 20E7 3906 0006   0002 0004 0404 0404
```

****

PRESS-A-PATTERN (0100 - 0C00)

J. MARCHINGTON,
[redacted]

Although not strictly a game, this program should prove to be a lot of fun for those whose DREAMS are equipped with at least 3K of memory. This is necessary on account of the considerable amount of data (locations 03A4 to 0BA3) required to provide the graphics.

Running from C000 will clear the screen. Pressing any one of the sixteen keys will produce an individual pattern, peculiar to the particular key pressed. Pressing the same key again will erase the pattern.

As all the designs are perfectly symmetrical, care must be taken when entering the data from the listing to ensure that no mistakes occur. Key through each of the sixteen patterns by pressing each key twice (once to display then again to erase) and inspect carefully for symmetry.

The object of the exercise is to press a number of keys in turn to produce other patterns. For example, keying in 0, 2, 5, and B should show a formation of nine squares, similar in appearance to Rubik's Cube. On the other hand, pressing keys 1, 2, 4, 6, 7 and 9 should display a large cross with horizontal and vertical arms.

The idea is to form specific patterns in as few key presses as possible. Remember to use any one key once only in forming a pattern, since the use of the same key a further time will effectively nullify its entry. You can form new patterns from two or more you have already found by combining the various key entries used for the individual designs and eliminating key presses when they 'double up'. For example, combining the nine squares with the cross, (the examples given above,) would result from the keying in of 0, 1, 4, 5, 6, 7, 9, and B.

Try your hand at forming the following patterns. A large Diamond; A very small square in the centre of the screen; A six-pointed star; A small cross at each of the four corners (i.e. four small crosses symmetrically placed).Enter, if you still have the facility, the data from 0100 to 01C7 and try to recreate those two designs.

Incidently, it is possible to produce 2 to the power of 16, minus 1 patterns, or 65,535 different designs. If you discount the sixteen patterns actually given in the program data , this still leaves 65,519 different displays. You will be amazed at the variety obtainable. Incredible in colour, if we had it!

```
0100   0000 0000 0000 0000   0000 0000 0000 0000
0110   0000 0000 0000 0000   0000 0000 0000 0000
0120   0000 0000 0000 0000   0000 0000 0000 0000
0130   0000 0000 0000 0000   00F1 8F00 00F0 0F00
0140   00F3 CF00 00F0 0F00   00F7 EF00 00F0 0F00
0150   00FF FF00 00F0 0F00   001F F800 000F F000
0160   003F FC00 000F F000   007F FE00 000F F000
0170   00FF FF00 000F F000   00FF FF00 000F F000
0180   007F FE00 000F F000   003F FC00 000F F000
0190   001F F800 000F F000   00FF FF00 00F0 0F00
01A0   00F7 EF00 00F0 0F00   00F3 CF00 00F0 0F00
01B0   00F1 8F00 00F0 0F00   0000 0000 0000 0000
01C0   0000 0000 0000 0000   FF
```

PRESS-A-PATTERN (Cont)

```
0200    00E0 F00A 4000 1246   4001 125A 4002 126E
0210    4003 1282 4004 1296   4005 12AA 4006 12BE
0220    4007 12D2 4008 12E6   4009 12FA 400A 130E
0230    400B 1322 400C 1336   400D 134A 400E 135E
0240    400F 1372 1202 2386   A3A4 238E A3C4 238E
0250    A3E4 238E A404 238E   1202 2386 A424 238E
0260    A444 238E A464 238E   A484 238E 1202 2386
0270    A4A4 238E A4C4 238E   A4E4 238E A504 238E
0280    1202 2386 A524 238E   A544 238E A564 238E
0290    A584 238E 1202 2386   A5A4 238E A5C4 238E
02A0    A5E4 238E A604 238E   1202 2386 A624 238E
02B0    A644 238E A664 238E   A684 238E 1202 2386
02C0    A6A4 238E A6C4 238E   A6E4 238E A704 238E
02D0    1202 2386 A724 238E   A744 238E A764 238E
02E0    A784 238E 1202 2386   A7A4 238E A7C4 238E
02F0    A7E4 238E A804 238E   1202 2386 A824 238E

0300    A844 238E A864 238E   A884 238E 1202 2386
0310    A8A4 238E A8C4 238E   A8E4 238E A904 238E
0320    1202 2386 A924 238E   A944 238E A964 238E
0330    A984 238E 1202 2386   A9A4 238E A9C4 238E
0340    A9E4 238E AA04 238E   1202 2386 AA24 238E
0350    AA44 238E AA64 238E   AA84 238E 1202 2386
0360    AAA4 238E AAC4 238E   AAE4 238E AB04 238E
0370    1202 2386 AB24 238E   AB44 238E AB64 238E
0380    AB84 238E 1202 6A10   6B00 6108 00EE 239E
0390    239C 239C 239C 7AE0   7B08 00EE F11E DAB8
03A0    7A08 00EE F0F0 F0F0   0F0F 0F0F 0000 0000
03B0    0000 0000 0000 0000   0000 0000 0F0F 0F0F
03C0    F0F0 F0F0 0000 0000   0000 0000 F0F0 F0F0
03D0    0F0F 0F0F 0F0F 0F0F   F0F0 F0F0 0000 0000
03E0    0000 0000 0000 0000   0000 0000 0F0F 0F0F
03F0    F0F0 F0F0 F0F0 F0F0   0F0F 0F0F 0000 0000

0400    0000 0000 0F0F 0F0F   F0F0 F0F0 0000 0000
0410    0000 0000 0000 0000   0000 0000 F0F0 F0F0
0420    0F0F 0F0F 0000 0000   0000 0000 0103 070F
0430    1F3F 7FFF 80C0 E0F0   F8FC FEFF 0000 0000
0440    0000 0000 0103 070F   1F3F 7FFF 0000 0000
0450    0000 0000 0000 0000   0000 0000 80C0 E0F0
0460    F8FC FEFF FF7F 3F1F   0F07 0301 0000 0000
0470    0000 0000 0000 0000   0000 0000 FFFE FCF8
0480    F0E0 C080 0000 0000   0000 0000 FF7F 3F1F
0490    0F07 0301 FFFE FCF8   F0E0 C080 0000 0000
04A0    0000 0000 0000 0000   0000 0000 F0F0 F0F0
04B0    F0F0 F0F0 0F0F 0F0F   0F0F 0F0F 0000 0000
04C0    0000 0000 FFFF FFFF   0000 0000 F0F0 F0F0
04D0    0000 0000 0F0F 0F0F   0000 0000 FFFF FFFF
04E0    0000 0000 0000 0000   FFFF FFFF 0000 0000
04F0    F0F0 F0F0 0000 0000   0F0F 0F0F 0000 0000

0500    FFFF FFFF 0000 0000   0000 0000 F0F0 F0F0
0510    F0F0 F0F0 0F0F 0F0F   0F0F 0F0F 0000 0000
0520    0000 0000 0000 0000   0000 0000 0103 070F
0530    0F07 0301 80C0 E0F0   F0E0 C080 0000 0000
0540    0000 0000 0000 0000   183C 7EFF 0103 070F
0550    1F3F 7FFF 80C0 E0F0   F8FC FEFF 0000 0000
0560    183C 7EFF FF7E 3C18   0000 0000 FF7F 3F1F
0570    0F07 0301 FFFE FCF8   F0E0 C080 FF7E 3C18
0580    0000 0000 0000 0000   0000 0000 0103 070F
```

```
0590   0F07 0301 80C0 E0F0   F0E0 C080 0000 0000
05A0   0000 0000 FFFF FFFF   FFFF FFFF 0080 C0E0
05B0   F0F8 FCFE 0001 0307   0F1F 3F7F FFFF FFFF
05C0   FFFF FFFF 7F3F 1F0F   0703 0100 FFFF FFFF
05D0   FFFF FFFF FFFF FFFF   FFFF FFFF FEFC F8F0
05E0   E0C0 8000 0001 0307   0F1F 3F7F FFFF FFFF
05F0   FFFF FFFF FFFF FFFF   FFFF FFFF 0080 C0E0

0600   F0F8 FCFE FFFF FFFF   FFFF FFFF FEFC F8F0
0610   E0C0 8000 7F3F 1F0F   0703 0100 FFFF FFFF
0620   FFFF FFFF 0F0F 0F0F   F0F0 F0F0 F0F0 F0F0
0630   0F0F 0F0F 0F0F 0F0F   F0F0 F0F0 F0F0 F0F0
0640   0F0F 0F0F F0F0 F0F0   0F0F 0F0F 0F0F 0F0F
0650   F0F0 F0F0 F0F0 F0F0   0F0F 0F0F 0F0F 0F0F
0660   F0F0 F0F0 0F0F 0F0F   F0F0 F0F0 F0F0 F0F0
0670   0F0F 0F0F 0F0F 0F0F   F0F0 F0F0 F0F0 F0F0
0680   0F0F 0F0F F0F0 F0F0   0F0F 0F0F 0F0F 0F0F
0690   F0F0 F0F0 F0F0 F0F0   0F0F 0F0F 0F0F 0F0F
06A0   F0F0 F0F0 0000 0000   0000 0000 FFFF FFFF
06B0   FFFF FFFF FFFF FFFF   FFFF FFFF 0000 0000
06C0   0000 0000 FFFF FFFF   FFFF FFFF 0000 0000
06D0   0000 0000 0000 0000   0000 0000 FFFF FFFF
06E0   FFFF FFFF FFFF FFFF   FFFF FFFF 0000 0000
06F0   0000 0000 0000 0000   0000 0000 FFFF FFFF

0700   FFFF FFFF 0000 0000   0000 0000 FFFF FFFF
0710   FFFF FFFF FFFF FFFF   FFFF FFFF 0000 0000
0720   0000 0000 0000 0000   0000 0000 FF7F 3F1F
0730   0F07 0301 FFFE FCF8   F0E0 C080 0000 0000
0740   0000 0000 80C0 E0F0   F8FC FEFF 0103 070F
0750   1F3F 7FFF 80C0 E0F0   F8FC FEFF 0103 070F
0760   1F3F 7FFF FFFE FCF8   F0E0 C080 FF7F 3F1F
0770   0F07 0301 FFFE FCF8   F0E0 C080 FF7F 3F1F
0780   0F07 0301 0000 0000   0000 0000 0103 070F
0790   1F3F 7FFF 80C0 E0F0   F8FC FEFF 0000 0000
07A0   0000 0000 0F0F 0F0F   FFFF FFFF F0F0 F0F0
07B0   FFFF FFFF 0F0F 0F0F   FFFF FFFF F0F0 F0F0
07C0   FFFF FFFF FFFF FFFF   0F0F 0F0F FFFF FFFF
07D0   F0F0 F0F0 FFFF FFFF   0F0F 0F0F FFFF FFFF
07E0   F0F0 F0F0 0F0F 0F0F   FFFF FFFF F0F0 F0F0
07F0   FFFF FFFF 0F0F 0F0F   FFFF FFFF F0F0 F0F0

0800   FFFF FFFF FFFF FFFF   0F0F 0F0F FFFF FFFF
0810   F0F0 F0F0 FFFF FFFF   0F0F 0F0F FFFF FFFF
0820   F0F0 F0F0 FFFF FFFF   FFFF FFFF FEFC F8F0
0830   E0C0 8000 7F3F 1F0F   0703 0100 FFFF FFFF
0840   FFFF FFFF FEFC F8F0   E0C0 8000 0103 070F
0850   1F3F 7FFF 80C0 E0F0   F8FC FEFF 7F3F 1F0F
0860   0703 0100 0080 C0E0   F0F8 FCFE FF7F 3F1F
0870   0F07 0301 FFFE FCF8   F0E0 C080 0001 0307
0880   0F1F 3F7F FFFF FFFF   FFFF FFFF 0080 C0E0
0890   F0F8 FCFE 0001 0307   0F1F 3F7F FFFF FFFF
08A0   FFFF FFFF 0103 070F   1F3F 7FFF 0080 C0E0
08B0   F0F8 FCFE 0001 0307   0F1F 3F7F 80C0 E0F0
08C0   F8FC FEFF 7F3F 1F0F   0703 0100 FEFC F8F0
08D0   E0C0 8000 7F3F 1F0F   0703 0100 FEFC F8F0
08E0   E0C0 8000 0001 0307   0F1F 3F7F 0080 C0E0
08F0   F0F8 FCFE 0001 0307   0F1F 3F7F 0080 C0E0
```

PRESS-A-PATTERN (Cont)

```
0900    F0F8 FCFE FF7F 3F1F   0F07 0301 FEFC F8F0
0910    E0C0 8000 7F3F 1F0F   0703 0100 FFFE FCF8
0920    F0E0 C080 0000 0000   0000 0000 0F0F 0F0F
0930    F0F0 F0F0 F0F0 F0F0   0F0F 0F0F 0000 0000
0940    0000 0000 0F0F 0F0F   F0F0 F0F0 0F0F 0F0F
0950    F0F0 F0F0 F0F0 F0F0   0F0F 0F0F F0F0 F0F0
0960    0F0F 0F0F F0F0 F0F0   0F0F 0F0F F0F0 F0F0
0970    0F0F 0F0F 0F0F 0F0F   F0F0 F0F0 0F0F 0F0F
0980    F0F0 F0F0 0000 0000   0000 0000 F0F0 F0F0
0990    0F0F 0F0F 0F0F 0F0F   F0F0 F0F0 0000 0000
09A0    0000 0000 0000 0000   0000 0000 FFFF FFFF
09B0    FFFF FFFF FFFF FFFF   FFFF FFFF 0000 0000
09C0    0000 0000 FFFF FFFF   FFFF FFFF 0F0F 0F0F
09D0    FFFF FFFF F0F0 F0F0   FFFF FFFF FFFF FFFF
09E0    FFFF FFFF FFFF FFFF   FFFF FFFF FFFF FFFF
09F0    0F0F 0F0F FFFF FFFF   F0F0 F0F0 FFFF FFFF

0A00    FFFF FFFF 0000 0000   0000 0000 FFFF FFFF
0A10    FFFF FFFF FFFF FFFF   FFFF FFFF 0000 0000
0A20    0000 0000 FFFF FFFF   F0F0 F0F0 F0F0 F0F0
0A30    0103 070F 0F0F 0F0F   80C0 E0F0 FFFF FFFF
0A40    0F0F 0F0F F0F0 F0F0   0103 070F FFFF FFFF
0A50    FFFF FFFF FFFF FFFF   FFFF FFFF 0F0F 0F0F
0A60    80C0 E0F0 0F07 0301   F0F0 F0F0 FFFF FFFF
0A70    FFFF FFFF FFFF FFFF   FFFF FFFF F0E0 C080
0A80    0F0F 0F0F F0F0 F0F0   FFFF FFFF 0F07 0301
0A90    F0F0 F0F0 F0E0 C080   0F0F 0F0F 0F0F 0F0F
0AA0    FFFF FFFF 0103 070F   1F3F 7FFF 0000 0000
0AB0    0000 0000 0000 0000   0000 0000 80C0 E0F0
0AC0    F8FC FEFF 0000 0000   0000 0000 FEFC F8F0
0AD0    E0C0 8000 7F3F 1F0F   0703 0100 0000 0000
0AE0    0000 0000 0000 0000   0000 0000 0080 C0E0
0AF0    F0F8 FCFE 0001 0307   0F1F 3F7F 0000 0000

0B00    0000 0000 FF7F 3F1F   0F07 0301 0000 0000
0B10    0000 0000 0000 0000   0000 0000 FFFE FCF8
0B20    F0E0 C080 F0F0 F0F0   0F0F 0F0F 0F0F 0F0F
0B30    FFFF FFFF F0F0 F0F0   FFFF FFFF 0F0F 0F0F
0B40    F0F0 F0F0 0F0F 0F0F   FFFF FFFF FFFF FFFF
0B50    F0F0 F0F0 FFFF FFFF   0F0F 0F0F F0F0 F0F0
0B60    FFFF FFFF FFFF FFFF   0F0F 0F0F F0F0 F0F0
0B70    FFFF FFFF 0F0F 0F0F   FFFF FFFF FFFF FFFF
0B80    F0F0 F0F0 0F0F 0F0F   F0F0 F0F0 FFFF FFFF
0B90    0F0F 0F0F FFFF FFFF   F0F0 F0F0 F0F0 F0F0
0BA0    0F0F 0F0F 4090 A1C5   B1B8 B030 4C43 84C0
```

****

FOR SALE : 2K PROGRAMMED EPROM

Like the original CHIPOS, except;
FN 0 changed to display address and data plus three lower and one higher address and data, Increment and Decrement address. PLUS: FN 4 - Move, FN 5 - Compare, FN 6 - Branch Offset Calculator, FN 7 - Tape Verify, FN 8 - Load Coded 9searches for program on tape), FN 9 - Dump Coded, FN A - Verify Coded, FN B - Run M/C, FN C - Clear. PLUS, 7 NEW COMMANDS. Display ASCII, Roll UP or DOWN, Complement all or part of screen, 1200 Hz Tone, Time Delay up to 1½ Hrs., Fill screen with anything, EX.OR. ONLY $17-50, from,
K. SEMRAD [redacted]

****

2K EPROM PROGRAMMER.

K. SEMRAD,
[redacted]

The 2516 or 2716 is easy to program, requires only 5V supply and has twice the capacity of the 2508.

I built it out of parts I had at home, the only thing I had to buy was the ZIF socket.

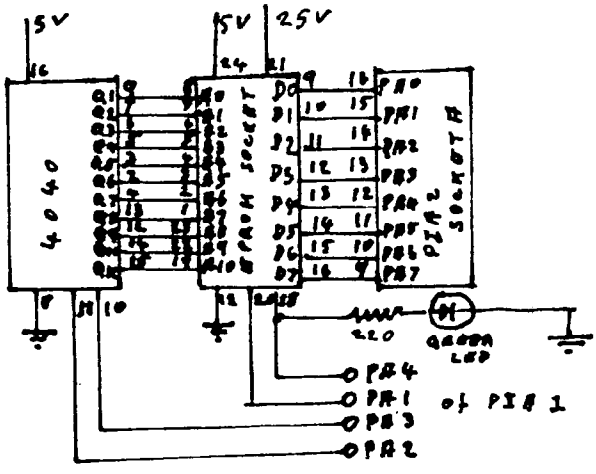
Hardware:

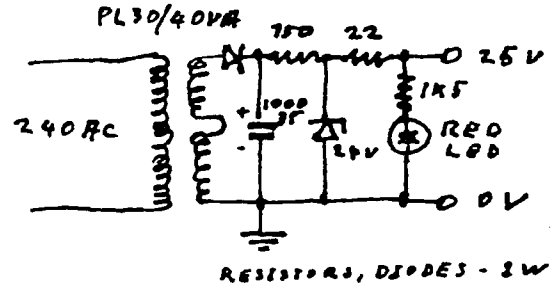
The EPROM address lines are conected to the outputs of a 4040 CMOS counter. The EPROM data lines go to PIA 2 port A. $\overline{CS}$ and $\overline{CE}$ go to the PIA socket on the mainboard. The 25V power supply can probably be improved, mine works.

Software:

On switch on, pin 18 and 20 of the EPROM are both low, then PIA 2 is set for input and the CMOS counter is reset, setting all address lines of the EPROM to zero. Data from the EPROM is then read in, inverted and if it equals zero, the address of the EPROM and the Indexregister are both incremented until the Indexregister equals 0800 (2K). That was to check if the EPROM is erased.
Now the PIA 2 Port A is set for output, the 4040 reset and the starting address of the program to be burnt in, is loaded into the Indexregister. Data is then transfered to the PIA 2 PA, pin 18 of the EPROM set high for 50 msec, the addresses incremented until the address equalls the address stored in 0004.
The third part verifies the data in the EPROM.

After loading start and finish in 0002 and 0004, go from 0300 (or whereever the software has been stored). Almost immeadiatly the green LED will come on and stay on for 100 sec. You should then see 0300 displayed on the screen. If an address between the starting and finishing address is displayed, the EPROM has not been programmed properly. If an address between 0000 and 0800 is displayed the EPROM was not erased(the green LED did not come on and that address was displayed immeadiatly.





Software for EPROM programmer (2K)

Check if erased

```
0300    5F        CLRB
0301    8D6C      BSR   set PIA2PB for input
0303    CE0000    LDX   set counter to zero
0306    8605      LDAA
0308    B78012    STAA  bit 0, bit 2 reset IC 4040
```

(Continued on next page)

```
030B    8601      LDAA
030D    B78012    STAA   bit 0, ready to read EPROM
0310    B68020    LDAA   EPROM data into Acc A
0313    43        COMA   invert data
0314    2666      BNE    if not zero, display address and stop
0316    8609      LDAA
0318    B78012    STAA   bit 0, bit 3 incr EPROM address
031B    08        INX    incr counter
031C    8C0800    CPX    finished ?
031F    26EA      BNE    no, repeat

                                                  Program EPROM
0321    8637      LDAA
0323    BDC2FE    JSR    kill DMA/display
0326    C6FF      LDAB
0328    8D45      BSR    set PIA2PB for output
032A    8607      LDAA
032C    B78012    STAA   bit 0, bit 1 EPROM write, bit 2 reset 4040
032F    DE02      LDX    load starting address
0331    A600      LDAA   get data
0333    B78020    STAA   transfere to port
0336    8613      LDAA
0338    B78012    STAA   bit 0, bit 1 write, bit 4 burn in
033B    860F      LDAA   )
033D    BDC2F3    JSR    )
0340    4A        DECA   ) wait for 50 msec
0341    26FA      BNE    )
0343    860B      LDAA
0345    B78012    STAA   bit 0, bit 1 write, bit 3 incr EPROM addr.
0348    08        INX    incr addr of prog tobe burnt in
0349    9C04      CPX    finished ?
034B    26E4      BNE    no, repeat

                                                  Verify program
034D    5F        CLRB
034E    8D1F      BSR    set PIA2PB for input
0350    8605      LDAA
0352    B78012    STAA   bit 0, bit 2 reset IC 4040
0355    DE02      LDX    load starting address
0357    8601      LDAA
0359    B78012    STAA   bit 0, ready to tead EPROM
035C    A600      LDAA   load data from RAM
035E    B18020    CMPA   compare with data in EPROM
0361    2619      BNE    if not equal, display address and stop
0363    8609      LDAA
0365    B78012    STAA   bit 0, bit 3 incr EPROM address
0368    08        INX    incr address of program
0369    9C04      CPX    finished ?
036B    26EA      BNE    no, carry on
036D    200F      BRA    yes, go to end

                                                  Subroutine to
036F    4F        CLRA                            set PIA2PB
0370    CE8020    LDX
0373    A701      STAA   DDR
0375    E700      STAB   input or output
0377    8604      LDAA
0379    A701      STAA   Data Buffer
037B    39

037C    DF06      STX    store address for display
037E    7EC360    JMP    return
0381    *
```

****

SIMPLE INTERPRETER (0D80 - 1000)

ROWAN McKENZIE,
[redacted]

This 'SIMPLE' interpreter requires a DREAMSOFT EPROM, a spare PIA, an ASCII keyboard and at least 2K of RAM. The program originally came from Electronics Today International, January 1979, and has been re-written to suit the DREAM 6800.

Although the 'SIMPLE' language is not an ideal language for the DREAM, reasonably good programs may be written. I would suggest that you get hold of a copy of the ETI article, as I will only give a brief description of the statements available to the user.

Programs are written in a similar way to the BASIC language. No line numbers are used, as programs run from the first line to the last. (Unless the program contains a JUMP statement.) Single letters are used to represent statements, which shortens programs considerably.

My version was written for 4K of memory, but this can easily be relocated by changing the underlined instructions in the listing.

The statements available for writing programs are as follows;

T - Type.

All characters following the letter 'T', on the same line, will be printed on the screen. Any characters may follow the 'T', except the special editing characters. These will be described later.

A - Accept.

The computer will wait for a key to be pressed, then store its ASCII value in a character buffer.

M - Match.

The character following this will be matched with the character in the character buffer. A flag will be set to 'yes' if the characters are the same. If they are not the same, it will be set to 'no'.

Y - Yes.

If the flag is set to 'yes', the next statement on the line will be executed. Otherwise the next line will be executed.

N - No.

This is the same as 'Y' except the next statement on the line will be executed if the flag is set to 'no'.

E - End.

Returns control to the text editor - (explained later). This ends the program.

J - Jump.

This allows the program to jump to a specified line. To use this, a '*' is placed at the beginning of the line which will be jumped to. The J must be followed by a number. The interpreter will then search the program for the line beginning with that '*'.

```
E.G.    * T HI
          T THERE
        * T !!!!
          J2
          E
```

The J2 will cause a jump to the 3rd line as this is the second '*'.

S - Subroutine.

This is used the same way as the Jump statement. The subroutine called must start with a '*'.

R - Return.
This is placed at the end of a subroutine.

K - Keep.
Places the character in the character buffer in one of nine variables. The number following the K corresponds to the desired variable.

G - Get.
This is the reverse of the keep statement. The desired variable is placed in the character buffer.

P - Print.
This statement will print the ASCII character in the character buffer, on the screen.

L - Load.
The character following this statement is placed into a counter.

I - Increment.
Increments the counter by 1.

D - Decrement.
Decrements the counter by 1.

X - Exchange.
The character in the counter is exchanged with the character in the character buffer.

B - Random.
This will generate a random number between 0 and 8. (Sorry about the 'B', but R is already used for Return.)

C - Comment.
The interpreter will ignore the rest of this line. Any comment may follow the C.

To enter a program, run from 0EB5. This is the entry point to the text editor. The text editor uses a set of special function characters which can not be used in a program. These are:-

- Return editor to the 1st line of the program.
& - Display the next line. This is used to list a program.
% - Used to replace or erase a line. To replace a line, list the program up to the line before the relevant line, then enter the new one. The new line must be smaller or equal in length to the original. When the new line has been typed, end it with a '%' instead of a carriage return. To erase a line, use the same procedure but don't enter any new line.
$ - Will run the program from the current line. To run from the start, use # then $.

More than one statement may be placed on a line. The only condition being that if a Type statement is used, it must be the last on the line.
To correct typing errors, use Back Space, then enter the correct character. Complete lines can be changed before carriage return has been pressed, by typing several back spaces.
To separate statements on a line, any character with an ASCII value less than 41 (the letter 'A') may be used. e.g. !.,;:- or statements may be placed without separating characters, although this makes it difficult to read.
The counter which is available to the programmer may contain any

character. If a letter is present and an Increment statement is executed, the next letter in the alphabet will replace it. In other words, the ASCII value of the character will be incremented, whether it is a letter, number, or line feed.

If the interpreter finds an illegal character or more than 64 characters on a line, a '?' will be displayed, along with the rest of that line, and the program will end.

This interpreter is very limited as the DREAM only allows 16 characters on a line, although more may be used if the lines are written 'on top' of each other. This allows up to 64 characters per line. (NOTE: The Type statement is still limited to 16 characters if the user is to understand what is being typed!)

For a full description of the SIMPLE language, I suggest you refer to the ETI article which also includes some excellent examples of what can be done with this interpreter.

```
0D80   1B 32 34 36 38 30 2D 08   00 57 52 59 49 50 5D 0D
0D90   00 53 46 48 4B 3B 5C 00   00 58 56 4E 2C 2F 00 00
0DA0   5A 43 42 4D 2E 20 00 00   41 44 47 4A 4C 40 7F 00
0DB0   51 45 54 55 4F 5B 0A 00   31 33 35 37 39 3A 5E 00
0DC0   1B 22 24 26 28 30 3D 08   00 57 52 59 49 50 7D 0D
0DD0   00 53 46 48 4B 2B 7D 00   00 58 56 4E 3C 3F 00 00
0DE0   5A 43 42 4D 3E 20 00 00   41 44 47 4A 4C 60 7F 00
0DF0   51 45 54 55 4F 7B 0A 00   21 23 25 27 29 2A 7E 00

0E00   37 DF 38 BD 1C BB 8D 4B   36 BD 1C BB 32 81 08 26
0E10   27 DE 38 09 DF 38 A6 00   81 4D 27 06 C6 04 81 57
0E20   26 02 C6 06 96 3A 10 97   3A A6 00 BD 18 A4 96 3A
0E30   10 97 3A 01 01 01 20 CB   33 DE 38 37 36 BD 18 A4
0E40   D6 3B C1 1F 26 0A 86 0A   BD 18 A4 86 E6 BD 18 A4
0E50   32 33 39 CE 80 20 86 04   C6 FF 6F 01 6F 00 A7 01
0E60   6F 03 E7 02 A7 03 6F 02   E1 00 27 16 BD C2 F3 CE
0E70   80 20 86 7F A7 02 86 07   E1 00 26 09 0D 66 02 4A
0E80   2A F6 7E 0E 53 48 48 48   E6 00 4A 4C 54 25 FC C6
0E90   FF E7 02 64 02 E6 01 C4   40 1B CE 0D 7F 4C 08 4A
0EA0   26 FC A6 00 C6 FF CE 80   20 6F 02 E1 00 26 FC BD
0EB0   C2 F3 C6 00 39 BD C0 79   CE 00 01 DF 3A CE 02 00
0EC0   DF 3D BD 0E 00 80 26 27   4B 4C 27 11 4C 27 47 4C
0ED0   27 E3 8B 23 A7 00 08 81   0D 27 2D 20 E5 C6 FF D7
0EE0   3C 37 C6 40 A6 00 81 0D   27 17 7D 00 3C 27 04 6F
0EF0   00 20 03 BD 0E 3B 08 5A   26 EA 86 3F BD 0E 3B 20

0F00   BC 86 0D BD 0E 3B 08 33   86 D6 BD 0E 3B 7D 00 3D
0F10   27 04 20 AC 20 43 A6 00   08 81 5A 24 2F 80 41 2D
0F20   F5 DF 38 CE 0F 32 08 4A   2C FC A6 00 36 86 0F 36
0F30   DE 38 39 DF ED 8D A7 51   4C BD 4C AC 60 B1 A0 7A
0F40   89 4C E6 4C 76 5E 56 4C   4C 4C 94 88 86 3F BD 0E
0F50   3B 09 DF 3D 20 03 7F 00   3D 7F 00 3C 20 83 DF 35
0F60   8D 67 CE 02 00 86 2A A1   00 27 03 08 20 F9 08 7A
0F70   00 37 2A F3 20 A0 DE 35   20 9C A6 00 08 90 34 26
0F80   04 C6 4F 20 91 5F 20 8E   86 4F 11 27 89 86 0D 7F
0F90   00 37 20 D3 37 96 33 D6   34 97 34 D7 33 33 20 E6
0FA0   A6 00 97 33 08 20 DF 7A   00 33 20 DA 7C 00 33 20
0FB0   D5 DF 31 8D 14 96 34 A7   00 DE 31 20 C9 DF 31 8D
0FC0   08 A6 00 97 34 DE 31 20   BD A6 00 80 31 81 09 24
0FD0   2C 97 37 CE 00 F7 DF 38   9B 39 97 39 DE 38 39 BD
0FE0   0E 00 97 34 20 A0 96 34   BD 0E 3B 20 99 DF 12 BD
0FF0   C1 32 84 07 8B 31 97 FF   DE 12 20 EF 01 7E 0F 4E
```

****

MAZE (0080 - 0400)

A. GROVES,
[redacted]

A mouse is positioned at the TOP RIGHT HAND corner of the maze. The object of the game is to guide the mouse through the maze to the block of cheese in the BOTTOM LEFT HAND corner. To move UP, use Key 1, DOWN, key 9, LEFT, key 4, and RIGHT, key 6. If the walls or any part of the maze are hit, the mouse returns to the start. You have 5 turns to reach the cheese within a random time limit of 0 - 120 seconds. If you reach the cheese, the game restarts automatically.

As mice are of low intelligence, the mouse will pause from time to time to work out the right direction. Pressing a key while the mouse is stopped will have no effect. This function can be changed to normal movement by changing location 02B4 from C608 to 6608. If you do not reach the cheese you will be devoured by the cat hiding in the maze.

Key functions can be found at the following locations. UP - 02A3, DOWN - 02A9, LEFT - 0297, RIGHT - 029D.

```
0080   00E0 6B00 20A0 6B1F  20A0 6B3C 20B0 6A3F
0090   20B0 A0CD 6A04 6B13  DAB4 2202 10DC 0000
00A0   6A00 A0AE DAB1 7A08  3A40 10A2 00EE FF80
00B0   6B01 A0AF DAB1 7B01  3B1F 10B2 00EE C0C0
00C0   00E0 0000 0000 0700  0000 003C 0080 8080
00D0   8000 0404 0400 0000  8080 8080 A0BE 6A02
00E0   6B1C DAB2 1270 40A0  4000 40A0 7840 A078
00F0   7F21 E1A1 FF1F 1133  20E0 B0F0 7838 78FF


0200   1080 A0C1 6A03 6B01  DABB 7A0D DABB 7A09
0210   DABB 7A09 DABB 7A0B  DABB 7A05 DABB 0000
0220   6A01 6B10 DABB 7A04  DABB 7A09 DABB 7A06
0230   DABB 7A0D DABB 7A07  DABB 7A07 DABB 7A08
0240   DABB A0CC 6A08 6B01  DABF 7A10 DABF 7A0D
0250   DABF 7A05 DABF 0000  6A11 6B10 DABF 7A08
0260   DABF 7A0F DABF 7A08  DABF 7A07 DABF 00EE
0270   C578 6700 6800 0000  22D6 6A39 6B02 A0E6
0280   DAB3 4500 12EE 4705  12EE 0000 0000 A0E6
0290   DAB3 6C00 6D00 6E04  EEA1 6CFF 6E06 EEA1
02A0   6C01 6E01 EEA1 6DFF  6E09 EEA1 6D01 8AC4
02B0   8BD4 DAB3 C608 F615  F607 3600 12B8 4A03
02C0   135A 4F00 12F6 6920  F918 22D6 7701 22D6
02D0   A0E6 DAB3 127A A382  F733 6303 6403 F265
02E0   F229 D345 00EE 75FF  0000 6800 00EE 6950
02F0   F918 0000 1302 7801  4808 22E6 1282 0000

0300   0000 00E0 A0ED 6A18  6B0E DAB3 A0F1 6A1C
0310   6B0A DAB7 2328 00E0  6960 F918 A0F8 6A1C
0320   6B0A DAB8 2344 1326  6640 F615 F607 3600
0330   132C 00EE CAAA EAAA  CEEE AAEE C8A8 4040
0340   4000 4000 A334 6A03  6B0A DAB5 7A08 A339
0350   DAB5 7A08 A33E DAB5  00EE 6950 F918 00E0
0360   A0EA 6A20 6B0E DAB3  A0BE 6A1B 6B0F DAB2
0370   2328 00E0 A0ED 6A1C  6B0D DAB4 2344 2328
0380   1200 0000 0400 0000  0000 0000 0000 0000
```

DOGFIGHT (0200 - 0400)

G. CLARE,
[redacted]

You are in the cockpit of your fighter plane. To bring the enemy into your sights, use keys:-

8 to bank to the LEFT
A to bank to the RIGHT
D to DIVE
5 to CLIMB
F to FIRE

The number of shots remaining is shown TOP RIGHT, and the score at TOP LEFT. (Score 5 points for every hit.) Hits are indicated by a flash and dispersing debris.

There is a time limit, indicated by the white line moving from Left to Right at the top of the screen. The game is over when the line reaches the right hand side of the screen, or when there are no remaining shots.

To restart the game, press key '0'.

N.B. You can fire at the same time as you are pressing any other key.

```
0200   2302 6564 671F 6810   22CA 22E8 22BE 6B00
0210   CA3F 8DA0 8EB0 C91F   CC07 A2F4 DAB2 4C00
0220   7DFF 3C01 122A 7DFF   7EFF 4C02 7EFF 3C03
0230   1236 7D01 7EFF 4C04   7D01 3C05 1242 7D01
0240   7E01 4C06 7E01 3C07   124E 7DFF 7E01 13A4
0250   0000 E0A1 7DFD E1A1   7D03 E2A1 7EFD 1320
0260   0000 A2F4 DAB2 8AD0   8BE0 4900 1216 79FF
0270   121A 22E8 75FF 22E8   22BE A2F6 6F00 D781
0280   3F00 1298 D781 3500   1262 A2F4 DAB2 F00A
0290   3000 128E 00E0 1200   22CA 7605 22CA 22BE
02A0   D781 7BFE 6010 A2F7   DAB7 DAB7 4000 12FE
02B0   70FF 12A8 A2F4 7B02   DAB2 2346 131A 640D
02C0   6008 610A 6205 630F   00EE A080 F633 F265
02D0   6301 6402 F029 D345   7304 F129 D345 7304
02E0   F229 D345 A2F6 00EE   A080 F533 F265 6334
02F0   12D2 0000 10FE 807C   FEFE FEFE FE7C 6004

0300   12B4 23E0 671C 680D   A30E D787 00EE 8244
0310   0000 0044 8200 F418   1272 3500 120E 128E
0320   6300 0326 1340 BDC2   87A6 0084 0826 0FC6
0330   0FBD C289 A600 8480   2604 C601 D733 3900
0340   3300 1316 1262 7B04   A37E 7AFE DAB9 7A08
0350   A387 DAB6 7AF8 A37E   DAB9 7A08 A387 DAB6
0360   7B04 A37E 7AF8 DAB9   7A08 A387 DAB6 7AF8
0370   A37E DAB9 7A08 A387   DAB6 F718 138E 0024
0380   4080 0000 4000 0480   4000 0800 8000 22BE
0390   00EE 7BFE 0398 00EE   CE80 2086 38A7 0386
03A0   30A7 0339 E4A1 7E03   6100 03AE 13D8 7A00
03B0   8827 0139 8618 9788   CE02 F6C6 0186 0097
03C0   2F96 8A97 2EBD C224   7C00 8A86 3C91 8A27
03D0   0139 8601 9731 3900   3100 128A 6106 1252
03E0   6600 03E6 00EE 8618   9788 86FE 978A 3900
```

MODIFIED T.V. TYPEWRITER - VIDEOTELETYPEWRITER.

(0080 - 0300)

R.G. DAVIS,
[redacted]

I liked the VIDEOTELETYPEWRITER program in July DREAMER, but thought it a lot of trouble to work out the display using it as a 64 X 28 grid, so I tried the T.V. TYPEWRITER, and using MEMOD from 0100, copied the contents down on a piece of paper, adding the extra 00 Bytes after each 8 Bytes and then loading the VIDEOTELETYPEWRITER program and adding the new display listing. Well, there had to be a better way!

I modified the program to do away with adding the extra bytes after each 8 bytes, and then I used a program by BRUCE MITCHELL, (MODIFIED T.V. TYPEWRITER), that stored the Video page at 0300 - 03FF, then loaded VIDEO-TELETYPEWRITER now modified to read from 0300, very much easier.

I then rang Bruce to say I was going to send the program in to the DREAMER, and he said it was alright to send his listing in with mine for anyone who could make use of it.

So I had another look at it and found that by relocating some of the listing, I could fit the VIDEOTELETYPEWRITER program at 0080 - 00B5, then added an extra key function, so that now you enter the program, use it as T.V. TYPEWRITER to get the message as you want it on the screen, then press 'A' to load it to 0300 - 03FF, then press '9' to get into the VIDEOTELETYPEWRITER. As easy as that! To change the number of lines, change 009E (3B1F) to the number of lines you wish it to read, and you now get the full video page. The T.V.TYPEWRITER program described above can still be used as a T.V. Typewriter, press 'A' to store the contents of the screen at 0300, then key in another page of lettering and recall the stored page from 0300 by pressing key 'B'.

```
0080   6A00 6B00 6C00 A300   FC1E DAB1 20A6 4A38
0090   1098 7A08 7C01 1086   6A00 7B01 7C01 3B1F
00A0   1086 00E0 1080 6D02   FD18 6D05 FD15 FD07
00B0   3D00 10AE 00EE CE01   00DF DCCE 0200 DFDE
00C0   CE03 00DF DA7E 00E0   CE03 00DF DCCE 0400
00D0   DFDE CE01 00DF DA7E   00E0 0200 0400 0400
00E0   DFDA DEDC A600 9CDE   270A 08DF DCDE DAA7
00F0   0008 20EC 3900 95B3   4800 84A5 CD4F 9DFD


0200   6A00 6B00 602F 0266   DAB5 F00A 12E0 8100
0210   F00A 8101 602F 0266   DAB5 8010 62C0 8122
0220   41C0 1234 4016 1250   4020 1254 0266 DAB5
0230   7A04 1204 8200 64F0   8242 640F 8042 42C0
0240   8A05 42D0 8B04 42E0   8B05 42F0 8A04 1204
0250   A25C 1256 A25D DAB5   7A06 1204 F8A8 A8A8
0260   A850 0000 0000 9630   810F 2203 7EC1 9380
0270   10CE 027E 7EC1 9896   3048 4848 4897 3039
0280   F6CE B7DA E92E F492   B75A F248 B7FA B6DE
0290   F6DE 93DE 5EDE BBDE   C546 492E F6DA 56DA
02A0   BFDA B55A 4BDA F11E   0024 2A22 88A8 8000
02B0   0BA0 0380 1550 1110   0820 1C70 419E FFFE
02C0   4384 AAAA 5555 6D80   4124 9380 5D74 E1C6
02D0   E31C 120A 02DD 120A   0000 DF00 A67E 00B8
02E0   400A 12F0 400B 12F4   4009 10A2 0277 120E
02F0   02FA 120A 02FD 120A   0000 7E00 B67E 00C8
```

MATH TABLES MK.2

(0080 - 0400)

DARREL WOOLNOUGH,
[redacted]

This is a rewrite of my earlier 'Math Tables' program published in the March '81 issue of 'DREAMER'. It incorporates many improvements and modifications and now offers four variations on the same theme instead of the original two. The rewrite was inspired by a letter from a fellow Dreamer contributor, Mr.Terry Mackrell, a,N.Z. School teacher, who is interested in such programs.

Some of the changes are cosmetic - after Cooo, FN, 3, the program name is now displayed for a couple of seconds followed by 'WHICH ?'. The latter is designed as a prompt to enter the required table via the keyboard.

Four variations are now possible. The 'RANDOM' and 'SEQUENTIAL' modes of the original program have been retained and what I call an automatic or 'AUTO' version of each has been added. To recap briefly, 'RANDOM' mode selects random multipliers in the selected table between the limits 0 - 12. The child enters the answer via the keyboard and is allowed three attempts. The new program displays 'TRY AGAIN' for a first wrong answer, 'LAST TRY', for the next, and finally 'SORRY' after three efforts. The new program then displays the complete problem plus answer together with 'RIGHT' for a couple of seconds before moving on to the next multiplier. The original program simply showed the correct answer alone and very briefly. In addition, the original program would only show a correct answer after all the digits were entered and would never show an incorrect one. The Mk.2 version shows the answer digit by digit as it is entered thus enabling a parent or teacher to immediately spot where the child is having difficulty. If it is desired to allow more than three attempts, the value of Variable 8 should be changed accordingly. In the 'RANDOM' program this is located at 0203. The display 'TRY AGAIN' appears for all wrong answers except the final two.

In the 'AUTO' versions, the problem and answer are both displayed for a period before moving on to the next. In the 'AUTO SEQUENTIAL' program for example, if 4 times table was selected, 4 X 0 = 0 would display together with 'RIGHT' followed after a delay by, 4 X 1 = 4, continuing up to 4 X 12 = 48, at which stage the program resumes at 4 X 0 = 0.

The 'AUTO RANDOM' program is identical except that the multipliers are random between 0 and 12.

In my opinion, the latter two variations, particularly the Auto Sequential, would be ideal for a learner who could repeat the table 'parrot fashion' following the computers lead. After a suitable period, the child would progress to the standard sequential mode, and finally, to the random mode when fully confident.

To change tables for all variations, simply hold Key 'F' until the prompt appears. My nine year old has progressed from mediocre math marks at school to near top of his class simply by practising with the DREAM.

PROGRAM CHANGES

MEMORY LOCATION	RANDOM	SEQUENTIAL	AUTO SEQUENTIAL	AUTO RANDOM
0202/03	As per	6900	6900	6800
0204/05		6803	6800	
0206/07		8490	8490	
0208/09	program	1212	1212	
02F0/F1		6701	6701	6701
02F2/F3		6E70	6E70	6E70
0308/09	listing.	13F4	13F4	
030A/0B		6701	6701	6701

****

```
0080    6C0F 6D08 A0E2 2314   A3D0 2314 A3C4 2314
0090    A3BE DCD5 6C08 7D0D   A3C4 2314 A3D0 2314
00A0    A0E7 2314 A3EE 2314   A0EC 2314 A3E2 DCD5
00B0    6A80 FA15 FA07 3A00   10B4 FD18 00E0 6A08
00C0    6B0E A0F1 230E A3BE   230E A3B2 230E A0F6
00D0    230E A3BE 230E 7A03   A0FB DAB5 F30A 00E0
00E0    1202 88D8 A888 88F0   88F0 88F0 F880 F880
00F0    F888 88A8 A8F9 F880   8080 F8F8 0878 0040

0200    1080 6803 640C C01F   8405 4F00 1204 8940
0210    8490 6500 6A03 6B06   6C0C 6D15 600F E09E
0220    1224 10BE A3FC F333   F265 3100 1230 1234
0230    F129 DAB5 7A06 F229   DAB5 7A06 A3A0 DAB5
0240    7A08 A3FC F433 F265   3100 124E 1254 F129
0250    DAB5 7A06 F229 DAB5   7A06 A3A6 DAB5 7A08
0260    0000 0000 0000 0000   8534 74FF 3400 1268
0270    A3FC F533 F265 3800   127C 12C2 3000 1286
0280    3100 129A 12AE F60A   360F 1290 00E0 10BE
0290    A3FC F629 230E 5600   131A F60A 360F 12A4
02A0    00E0 10BE A3FC F629   230E 5610 131A F60A
02B0    360F 12B8 00E0 10BE   A3FC F629 230E 5620
02C0    131A A3AC 2314 A3B2   2314 A3B8 2314 A3BE
02D0    2314 A3C4 2314 3800   130A 3000 12E4 3100
02E0    12E8 12EC F029 230E   F129 230E F229 230E
02F0    6704 6E30 FE15 FE07   3E00 12F6 77FF 6E04

0300    FE18 3700 12F2 00E0   1202 6702 12F2 DAB5
0310    7A06 00EE DCD5 7C07   00EE 78FF 4801 137A
0320    3800 1354 A3E2 2314   A3E8 2314 A3AC 2314
0330    A3AC 2314 A3CA 2314   6E40 FE15 FE07 3E00
0340    133C 6E10 FE18 00E0   1210 0000 0000 0000
0350    0000 0000 6C03 A3C4   2314 A3AC 2314 A3CA
0360    2314 7C04 A3D0 2314   A3D6 2314 A3D0 2314
0370    A3B2 2314 A3DC 2314   1338 6C05 A3EE 2314
0380    A3D0 2314 A3E2 2314   A3C4 2314 7C04 A3C4
0390    2314 A3AC 2314 A3CA   2314 1338 0000 0000
03A0    8850 2050 8800 00F8   00F8 0000 F888 F890
03B0    9000 2020 2020 2000   F880 9888 F800 8888
03C0    F888 8800 F820 2020   2000 8850 2020 2000
03D0    2050 F888 8800 F880   9888 F800 88C8 A898
03E0    8800 F880 F808 F800   F888 8888 F800 8080
03F0    8080 F800 7901 390D   1204 1202 0003 0200
```

****

ERRATTA

The 'Modification to KALAH' contained in the last issue has TWO errors in the new listing to be added from 0380.

038B should be 86
03CA should be E0

****

TRACE

(0300-04B8)
or anywhere

Edward Perati,
[redacted]

"TRACE" is a machine code program which enables you to single step through your machine code program, giving you a display of all registers after each instruction ("Trace" incorporates Michael Bauer's Dreambug Routine). These registers are displayed as: CCR, B, A, X, PC.

* The start address of your machine code program must be placed at Location $0002-$0003 inclusive.

* A jump, 7E 030A must be placed at location $0080-$0082.

* As "Trace" is self modifying, it must be run in RAM.

* As "Trace" works by inserting a 3F at the next instruction, the program being traced must also be in RAM. If calls are made to routines in ROM, unpredictable things may occur.

* If you require your program to stop, a software interrupt ($3F) is considered a terminator by "Trace" and returns control to monitor.

* If the reset key is pressed while stepping through your program, the location of the next instruction must be changed back to its original data as "Trace" will have inserted a $3F there. If the instruction that was going to be executed at the time of pressing reset was a branch or jump, the $3F will be found at the branch or jump address.

* To run "Trace", simply key in its start address and press Function 3, the first instruction of your program will be executed and depressing any other hex key will cause the next instruction to be executed and so on.

Relocation

Assuming "Trace" is already on cassette, simply place a new start and end address at location $0002-$0005 inclusive, load the tape and make the following changes. If you haven't got it on cassette yet, just load the program where you wish and make the following changes.

Example:

"Trace" start address	$0E00	"Trace" start address	$0E00
add	$01B4	add	$0138
first result	$0FB4	second result	$0F38

Place the first result into memory location specified by the second result, i.e. location $0F38-$0F39 contain $0FB4.

The jump at location $0080 must also be changed.

Example:

"Trace" start address	$0E00
	$000A
result	$0E0A

Therefore, locations $0080-$0082 contain 7E 0E0A.

TRACE 2

(0300-048B)
or anywhere

Edward Perati,
[redacted]

"TRACE 2" is a machine code program which enables you to single step through your machine code program, giving you a display of all registers after each instruction ("Trace 2" incorporates Michael Bauer's Dreambug Routine). These registers are displayed as: CCR, B, A, X, PC.

* The start address of your machine code program must be placed at Location $0002-$0003 inclusive.

* A jump, 7E 030A must be placed at location $0080-$0082.

* As "Trace 2" is self modifying, it must be run in RAM.

* As "Trace 2" works by inserting a 3F at the next instruction, the program being traced must also be in RAM. If calls are made to subroutines, they will be executed without being traced and upon returning to the main program, "Trace 2" will continue tracing your program.

* If you require your program to stop, a software interrupt ($3F) is considered a terminator by "Trace 2" and returns control to monitor.

* If the reset key is pressed while stepping through your program, the location of the next instruction must be changed back to its original data as "Trace 2" will have inserted a $3F there. If the instruction that was going to be executed at the time of pressing reset was a branch or jump, the $3F will be found at the branch or jump address.

* To run "Trace 2", simply key in its start address and press Function 3, the first instruction of your program will be executed and depressing any other hex key will cause the next instruction to be executed and so on.

Relocation

Assuming "Trace 2" is already on cassette, simply place a new start and end address at location $0002-$0005 inclusive, load the tape and make the following changes. If you haven't got it on cassette yet, just load the program where you wish and make the following changes.

Example:

"Trace 2" start address	$0E00	"Trace 2" start address	$0E00
add	$0187	add	$0110
first result	$0F87	second result	$0F10

Place the first result into memory location specified by the second result, i.e. location $0F10-$0F11 contain $0F87.

The jump at location $0080 must also be changed.

Example:

"Trace 2" start address	$0E00
add	$000A
result	$0E0A

Therefore, locations $0080-$0082 contain 7E 0E0A.

TRACE (0300 - 0500)

```
0300   DE 02 DF 30 8D 6F DE 30   6E 00 CE 00 33 9F 34 C6
0310   07 31 5A 26 FC A6 00 36   09 A6 00 36 9E 34 B6 80
0320   13 97 16 8A 08 B7 80 13   86 02 BD C3 E0 CE 01 C8
0330   86 FF BD C0 7D 30 C6 04   37 BD C3 C8 86 01 BD C3
0340   DE 08 33 5A 26 F2 86 FF   BD C3 DE 08 BD C3 C8 86
0350   02 BD C3 DE 08 BD C3 C8   08 BD C3 C8 DF 10 BD C2
0360   87 7D 80 11 2A FB BD C2   F3 BD C2 F3 CE 01 C8 4F
0370   BD C0 7D 20 02 20 34 96   16 B7 80 13 CE 2F FF 09
0380   26 FD 96 38 DE 32 A7 00   DF 30 8D 1F 96 36 81 01
0390   27 16 96 37 81 01 26 0F   9F 34 C6 07 31 5A 26 FC
03A0   30 EE 00 9E 34 8D 50 3B   7E C3 60 7F 00 36 7F 00
03B0   37 A6 00 81 8E 27 4B 81   8C 27 47 01 CE 27 43 84
03C0   F0 81 F0 27 3D 81 B0 27   39 81 70 27 35 97 38 CE
03D0   00 05 C6 60 10 27 3B 96   38 CB 20 09 26 F6 81 20
03E0   27 30 81 90 27 2C 81 D0   27 28 DE 30 A6 00 81 3F
03F0   27 4B 81 39 27 4C 08 DF   32 A6 00 97 38 86 3F A7

0400   00 39 DE 30 A6 00 81 7E   27 3D 81 BD 27 39 08 08
0410   20 E4 DE 30 A6 00 81 6E   27 56 81 20 27 6C 81 8D
0420   27 68 81 AD 27 4A C6 0E   97 38 86 22 91 38 27 07
0430   4C 5A 26 F8 08 20 BF B7   04 B4 7E 04 AD 86 01 97
0440   36 39 86 01 97 37 39 DE   30 08 EE 00 20 A9 96 3A
0450   0C 9B 38 97 3A 24 05 96   39 4C 97 39 39 20 98 96
0460   3A 0C 90 38 97 3A 2C 05   96 39 4A 97 39 39 20 86
0470   08 A6 00 97 38 9F 34 31   31 31 31 31 32 97 39 32
0480   97 3A 9E 34 8D C8 DE 39   20 D3 08 A6 00 85 FF 2B
0490   0B 08 DF 39 97 38 8D B6   DE 39 20 C1 84 7F 16 86
04A0   80 08 DF 39 10 97 38 8D   B6 DE 39 20 ED 31 31 32
04B0   34 34 34 06 20 D4 08 20   B5
```

++++++++++++

TRACE 2. (0300 - 0500)

```
0300   DE 02 DF 30 8D 6F DE 30   6E 00 CE 00 33 9F 34 C6
0310   07 31 5A 26 FC A6 00 36   09 A6 00 36 9E 34 B6 80
0320   13 97 16 8A 08 B7 80 13   86 02 BD C3 E0 CE 01 C8
0330   86 FF BD C0 7D 30 C6 04   37 BD C3 C8 86 01 BD C3
0340   DE 08 33 5A 26 F2 86 FF   BD C3 DE 08 BD C3 C8 86
0350   02 BD C3 DE 08 BD C3 C8   08 BD C3 C8 DF 10 BD C2
0360   87 7D 80 11 2A FB BD C2   F3 BD C2 F3 CE 01 C8 4F
0370   BD C0 7D 20 02 20 1F 96   16 B7 80 13 CE 2F FF 09
0380   26 FD 96 38 DE 32 A7 00   DF 30 8D 0A 96 36 81 01
0390   27 01 3B 7E C3 60 7F 00   36 A6 00 81 8E 27 47 81
03A0   8C 27 43 81 CE 27 3F 84   F0 81 F0 27 39 81 B0 27
03B0   35 81 70 27 31 97 38 CE   00 05 C6 60 10 27 33 96
03C0   38 CB 20 09 26 F6 81 20   27 28 81 90 27 24 81 D0
03D0   27 20 DE 30 A6 00 81 3F   27 3B 08 DF 32 A6 00 97
03E0   38 86 3F A7 00 39 DE 30   A6 00 81 7E 27 2C 08 08
03F0   20 E8 DE 30 A6 00 81 6E   27 49 81 20 27 5F C6 0E

0400   97 38 86 22 91 38 27 07   4C 5A 26 F8 08 20 CB B7
0410   04 87 7E 04 80 86 01 97   36 39 DE 30 00 EE 00 20
0420   BA 96 3A 0C 9B 38 97 3A   24 05 96 39 4C 97 39 39
0430   20 A9 96 3A 0C 90 38 97   3A 2C 05 96 39 4A 97 39
0440   39 20 97 08 A6 00 97 38   9F 34 31 31 31 31 31 32
0450   97 39 32 97 3A 9E 34 8D   C8 DE 39 20 D3 08 A6 00
0460   85 FF 2B 0B 08 DF 39 97   38 8D B6 DE 39 20 C1 84
0470   7F 16 86 80 08 DF 39 10   97 38 8D B6 DE 39 20 ED
0480   31 31 32 34 34 34 06 20   D4 08 20 B5
```

****

ASCII TYPEWRITER

(0200 - 0400)

E. WITTE,
[redacted]

This program is similar to the one shown in E.A. July,1979, except that it uses the world-wide standard ASCII code. To use it, simply RUN from C000 and key in your ASCII characters. Destructive backspace is not included, so you will have to erase letters, in the same manner as in the original T.V. TYPEWRITER.

```
0200  22 06 22 0C 12 02 6A 00 6B 00 00 EE 60 1F 02 5E
0210  DA B5 F0 0A 02 67 F1 0A 81 01 60 1F 02 5E DA B5
0220  80 10 41 00 00 EE 41 0A 12 F0 41 0D 13 24 41 08
0230  13 34 41 09 13 32 41 0C 13 38 41 1B 12 06 62 3F
0240  80 22 40 0D 12 52 40 17 12 56 02 5E DA B5 7A 04
0250  00 EE A3 3F 12 58 A3 40 DA B5 7A 06 00 EE 96 30
0260  CE 02 6E 7E C1 98 39 96 30 48 48 48 48 97 30 39
0270  F7 8E B7 DE D7 DD F2 4E D6 DD F3 CE 93 4E F6 CE
0280  B7 DA E9 2E F4 92 B7 5A F2 48 B7 FA B6 DE F6 DE
0290  93 DE 5E DE BB DE C5 46 49 2E F6 DA 56 DA BF DA
02A0  B5 5A 4B DA F1 1E 69 26 25 48 C9 2C B5 00 FF FE
02B0  00 00 41 24 00 5A 5D 74 CD 66 B1 1A F7 B6 00 24
02C0  29 22 89 28 15 50 0B A0 88 00 03 80 80 00 91 12
02D0  F6 DE 49 24 F3 9E E7 9E 3E D8 E7 CE F7 CE 24 9E
02E0  F7 DE E7 DE 08 20 88 20 2A 22 1C 70 88 A8 41 9E
02F0  6A 00 7B 06 4B 1E 02 FE 4B 1E 7B FA 00 EE CE 01

0300  00 DF F4 CE 02 00 DF F6 CE 01 30 A6 00 08 DF F2
0310  DE F4 A7 00 08 DF F4 DE F2 9C F6 26 EE CE 01 C8
0320  4F 7E C0 7D 6A 00 7B 06 4B 1E 02 FE 4B 1E 7B FA
0330  00 EE 7A FE 7A FC 00 EE 03 3C 00 EE BD C0 79 F8
0340  A8 A8 A8 A8 50 00 00 00 00 00 00 00 00 00 00 00
0350  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0370  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

****

ASCII TO BAUDOT PRINTER

(0300 - 0400

E. WITTE.

This routine accepts an ASCII character from 0031 (Chip-8 Variable 1) and outputs its Baudot equivalent on a Baudot printer. It is a Machine Code sub-routine and at the beginning of each program, 0040 must be cleared. Below is the listing, and a demonstration program which accepts ASCII via a Hexadecimal keyboard and outputs it on a Baudot printer.

```
0300  C6 01 D7 20 96 31 C6 07 F7 80 12 CE 03 4F 08 4A
0310  2A FC A6 00 2A 10 7D 00 40 26 1B 7C 00 40 36 86
0320  FE 8D 13 32 20 10 8A 80 7D 00 40 27 09 7F 00 40
0330  36 86 F6 8D 01 32 7D 00 20 26 FB 44 24 04 C6 07
0340  20 02 C6 05 F7 80 12 C6 01 D7 20 4D 26 E8 39 01
0350  80 80 80 80 80 80 80 80 80 80 C4 80 80 D0 80 80
0360  80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
0370  C8 80 80 80 74 5A 80 4A 5E 64 FA 62 58 46 78 7A
0380  6C 6E 66 42 54 60 6A 4E 4C 70 5C 5C 80 7C 80 72
0390  72 C6 F2 DC D2 C2 DA F4 E8 CC D6 DE E4 F8 D8 F0
03A0  EC EE D4 CA E0 CE FC E6 FA EA E2 80 80 80 80 80
03B0  4A C6 F2 DC D2 C2 DA F4 E8 CC D6 DE E4 F8 D8 F0
03C0  EC EE D4 CA E0 CE FC E6 FA EA E2 80 80 80 80 80
```

(See bottom of next page for Demonstration program.)

FOWL PLAY (0080 - 0400)

ROGER GRAHAM,
[redacted]

This is another graphics-orientated program along similar lines to 'Big Dog'.(Dreamer No.15, Nov.81.) Strictly for fun, RUN C000, FN, 3.

```
0080   80C0 E0F0 F8FC 7F7F  3F3F 1F1F 0F07 0301
0090   FFFF FFFF FFFF FFFF  FFFF FF3E 1CFC FCFC
00A0   FCFC FCF8 F0E0 C080  1CFF FEFA FF1C 0808
00B0   0808 0818 0F0A 1E1F  1F37 FF3F 1F1B 0301
00C0   80C0 E0FF FFFF FF7F  3F1F 0F07 030F FFFF
00D0   FFFF FFFF FFFE F8F0  E002 0508 1A[illegible] 78F8
00E0   F8F0 F0F0 E0E0 C080  1010 18F0 F0C0 C0C0
00F0   C0E0 7030 2030 70E0  4070 7040 50E0 4060

0200   6A00 6B00 A080 DAB0  7A08 7B06 A090 DABD
0210   7A08 A09D DABB 7A06  A0A8 DAB8 7AF2 7B0D
0220   A0B0 DAB5 225E 6A20  6B00 A0B5 DABB 7A08
0230   7B06 A0C0 DABD 7A08  7B01 A0CC DABD 7A08
0240   7BF9 A0D9 DABF 7AF6  7B14 A0E8 DAB4 225E
0250   13A2 6C05 FC15 FC07  3C00 1256 00EE 6C32
0260   1254 6C64 1254 A0ED  DAB3 00EE 6005 6D18
0270   6E30 6A20 6B04 2266  FD18 2266 2252 2266
0280   F018 2266 2252 2266  FD18 2266 2252 2266
0290   F018 2266 2252 2266  FE18 2266 00EE 6701
02A0   6100 6A16 6B06 A0A8  DAB8 7BFF DAB8 2252
02B0   DAB8 7B01 DAB8 F718  7101 3108 12A8 00EE
02C0   6A02 6B11 A08F DAB1  2252 7BFE A3DE DAB3
02D0   2252 7BFF A3DC DAB4  2252 A3E1 DAB4 2252
02E0   7BFF A3D8 DAB5 2252  A3EC DAB5 2252 7AFF
02F0   A3E6 DAB6 2252 7AFF  A3E5 DAB7 2252 7B01

0300   A3F1 DAB7 7A08 7B03  A3E7 DAB2 2252 7AF8
0310   7BFE 6200 2324 7B01  2324 2252 7201 320B
0320   1314 00EE A3FA DAB6  7A08 7B02 A3F8 DAB2
0330   7AF8 7BFE 00EE 7A01  A3D2 DAB6 7AFF A3F4
0340   DAB2 00EE 2336 7A01  7BFF 2336 2252 2336
0350   7A01 7BFF 2336 2252  2336 7A01 2336 2252
0360   2336 7A01 7B01 2336  2252 2336 7A01 7B01
0370   2336 F718 00EE 7B02  A0F0 DAB4 7AFF DAB4
0380   7A06 7BFF A0F4 DAB5  7A01 DAB5 225E 7BFF
0390   A0F9 DAB6 7AF9 7B02  A0F0 DAB4 7A02 7BFE
03A0   00EE 226C 229E 6800  6900 22C0 2376 2344
03B0   7902 3914 13AE 225E  7804 8980 3814 13AA
03C0   22C0 2376 229E 226C  229E 226C 00E0 2262
03D0   1200 FFFD FF72 2030  2040 4020 1010 0804
03E0   0220 2010 0818 0000  80C0 603E 4080 8040
03F0   213C 4080 0080 413E  8080 3E7F FFFF 7F3E
```

**********"

ASCII TO BAUDOT PRINTER (Cont) (0200 - 0230)

DEMONSTRATION PROGRAM

```
0200   0210 F00A 0218 F10A
0208   8101 0300 1202 0000
0210   7F00 4039 0000 0000
0218   9630 4848 4848 9730
0220   3900 0000 0000 0000
```

****

THREE IN A ROW

(0200 - 0320)

K. SEMRAD,
[redacted]

This is a variation of the game 'Connect 4'. You have to try and get three marbles of your colour in a horizontal, vertical, or diagonal row.

Keys correspond to the columns, other keys are ignored, except 'C', which restarts the game.

```
0200    A268 6A00 6B01 DAB1   7B01 3B1E 1206 7A08
0210    3A38 1204 7AF8 DAB1   7AFF 3AFF 1216 60FF
0220    6100 A080 F11E F055   7101 3158 1222 6E00
0230    6A38 6B0F 2294 6B1A   F00A 4000 1238 6106
0240    8105 3F01 1238 8004   8004 8004 8100 8A00
0250    A08C F11E F065 40FF   1266 7101 7BFC 3BFE
0260    1250 0308 1236 7AF8   80E0 A08C F11E F055
0270    029C 2294 3C01 1282   F00A 300C 1278 00E0
0280    1200 6A38 6B0F 2294   6DFE 3E02 7D04 8ED4
0290    2294 1236 A303 FE1E   DAB3 00EE 7F00 3CDE
02A0    2609 DFFC A600 97FE   8601 8D0A 8607 8D06
02B0    8608 8D02 8609 DEFC   1648 094A 26FC A600
02C0    91FE 260C 1708 4A26   FCA6 0091 FE27 3039
02D0    1708 4A26 FCA6 0091   FE26 0D17 4808 4A26
02E0    FCA6 0091 FE27 1839   1748 084A 26FC A600
02F0    91FE 2701 3908 5A26   FCA6 0091 FE26 037C

0300    003C 393E 223E 3E3E   C610 D721 C640 7EC2
0310    E5
```

****

WIPE OFF ROUTINE FOR VIDEO DISPLAYS

(0300 - 0330)

BRUCE MITCHELL,
[redacted]

Since I use 'Alpha Display' quite a lot for Videotape identification and title sequences, I have written a few extra effects which can be called up as subroutines using the appropriate command.

This particular one gives a top-to-bottom 'wipe-off' effect. It is completely relocatable. The speed of wipe is adjustable by altering 021D. (A value of 02 seems smoothest.)

It should also be quite compatible with DREAMTEXT.

```
0300    86 FF         LDA A #FF
0302    CE 01 00      LDX   #0100
0305    A7 00         STA A  00,X
0307    08            INX
0308    8C 01 08      CPX   #0108
030B    26 F8         BNE TO 0305
030D    CE 01 00      LDX   #0100
0310    C6 08         LDA B #08
0312    A6 00         LDA A  00,X
0314    A7 08         STA A  08,X
0316    6F 00         CLR    00,X
0318    08            INX
0319    5A            DEC B
031A    26 F6         BNE TO 0312
031C    86 02         LDA A #02
031E    97 20         STA A  0020
0320    7D 00 20      TST    0020
0323    26 FB         BNE TO 0320
0325    8C 01 F8      CPX   #01F8
0328    26 E6         BNE TO 0310
032A    7E C0 79      JMP    C079
```

****

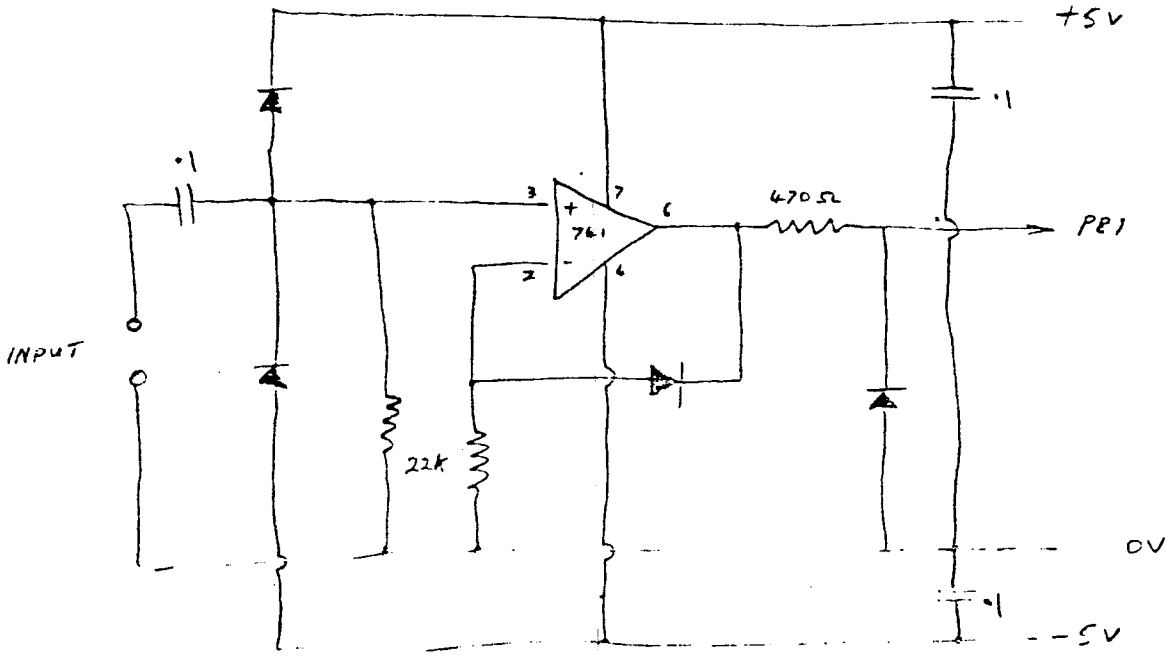
FREQUENCY COUNTER (0200 - 0300)

DREAMSOFT
[redacted]

This program lets you use the DREAM as an Audio Frequency Counter. It will measure frequencies in the range 2Hz - 18.5KHz with less than 0.2% error. It does not depend on either of our software packages and so can be run on a standard DREAM. The program is written in Chip-8 but most of the work is done by Machine-Code subroutines, thus it is started from C000.

The signal to be measured is fed into PB1 via an interface circuit similar to the DREAM Tape Input. This amplifies the signal and makes it into a TTL compatible square wave. The DREAM's own tape input circuit could be used with a switch to change from Tape Input to Frequency Measurement, but we found this to be less than ideal. Most experimenters will have their spare PIA lines and power supplies wired out to a socket so a separate interface 'plug-in' will be the best way to do it.

Here is the circuit we used. Note the over-voltage protection diodes, the larger input coupling capacitor to extend low frequency response, and the 470 ohm resistor in the output to avoid conflict between the 741 output and the PIA before the latter gets re-configured as an input.



The program switches off the screen DMA during counting to allow the highest possible frequency to be measured, thus we blank the screen, count the pulses arriving on PB1 for one second, convert this number to decimal, put it into the display, turn the screen on for one second viewing time, then repeat the process.

The most obvious application will be checking your computer's own Tape Output frequencies. Just plug the 'Tape Out' into the counter input to read the 2400Hz tone and change the data at 024D from '01' to '00' to measure the 1200Hz tone.

The accuracy of the counter is adequate for most applications, such as checking modems, RTTY tones, engine tachometer, telephone experiments, etc., but it is not good enough for tuning musical instruments. This is because the '20mS' interrupt timer in the DREAM is <u>not</u> 20mS but actually 19.968mS. (T.V. line rate 15625Hz divided by 312 lines for a non-interlaced picture = 50.080128Hz frame rate.) Thus when we count for '1 second' it is really 998.4 mS. The error is not noticable at low frequencies and you can compensate using this table to give greater accuracy if it is required.

FREQUENCY COUNTER (Cont)

Actual Frequency	Indicated by DREAM
10 Hz	10 Hz
20	20
100	100
200	200
1000	998
2000	1996
10000	9965
15625	15550
18500	18420

For top and bottom end checks ; the line oscillator of a T.V. set, locked into a broadcast signal, will be exactly 15625 Hz, and the mains are 50 Hz. (No kidding?)

```
0200   023D 0251 00E0 6A10  6B10 F029 DAB5 7A04
0210   F129 DAB5 7A05 F229  DAB5 7A04 F329 DAB5
0220   7A04 F429 DAB5 7A08  A238 DAB5 1202 2710
0230   03E8 0064 000A 0001  AEA2 E4A8 AECE 8012
0240   863B A701 8675 A700  863F A701 8601 A700
0250   3986 3FB7 8013 8632  9720 7D00 2026 FBC6
0260   37F7 8013 CE00 00C6  0297 207D 0020 2712
0270   F580 1227 F67D 0020  2708 F580 1226 F608
0280   20E9 DFC0 CE00 30DF  C2CE 022E 96C0 D6C1
0290   7F00 C4E0 01A2 0025  057C 00C4 20F5 EB01
02A0   A900 36DF C0DE C296  C4A7 0032 08DF C2DE
02B0   C008 088C 0238 26D8  3900 0000 0000 0000
```

****

2K CHIPOS EPROM

The 2K pre-programmed EPROM advertised by KEITH SEMRAD in last month's Dreamer looked very interesting, so we thought that you might like to know a bit more about it than the brief rundown in the ad permits.
Here then is a description of the functions available:-

At 'switch-on', "FN" is displayed.

Key 0 Will display "0 MD"
Key in address, that address plus three lower and one higher address will appear on the screen, with the data stored at those addresses.
Modify data with Hex. keys.
Increment addresses with FN 1.
Decrement addresses with FN 0.
Return to Function mode with FN, FN.

1 Will display "1 LD
BEG"
Key in start address of program to be loaded,
"END+1" will be displayed.
Key in last plus one address of program, start tape and hit any key.

2 Will display "2 DP
BEG"
Acts as in '1'.

3 Will run CHIPOS program from 0200.

4 Will display "4 MV
BEG"
Key in start address of program to be moved
"END+1" will be displayed
Key in address, block is moved and returns to Function mode.

5 Will display "5 CP
BEG"
As in 4 if O.K., or else
"* ERROR XXXX *" is displayed.

6 Will display "6 BR"
Key in least significant byte of address to branch from
"XX TO" will be displayed
Key in least significant byte of address to branch to
"YY = ZZ" will be displayed, and return to Function mode.

7 Will display "7 VE
BEG"
As in 1 if O.K., else
"*ERROR XXXX*" is displayed.

8 Will display "8 LD
CODE"
Key in code of program wanted (2 digits, 00-FF)
Start tape, hit any key.
"NOW XX" will be displayed if the program on tape is not the one wanted, else the program will be loaded.

9 Will display "9 DP
BEG"
Key in start address of program to be dumped
"END+1" will be displayed
Key in last plus one address of the program to be dumped
"CODE" will be displayed
Key in code, start tape and press any key.

A Will display "A VE
CODE"
Key in code, then as in 7.

B Will display "B
BEG"
Key in starting address of M/C program, then press any key

C Will display "C CL
BEG"
Key in first address of memory to clear
"END+1" will be displayed
Key in last plus one address of memory to clear, press any key.

NEW COMMANDS

FX17	As FX18 but 1200 Hz
FX16	Time delay, for X see table.
FX30	Roll screen up or down, for X see table.
FXEE	Complement screen, part or full, see table.
FXFC	Fill screen with value of X
FXDA	Display ASCII characters, roll, complement, etc, see table.
8XY3	EXORs X,Y, result stored in X.

2K CHIPOS EPROM (Cont)

LOOK-UP TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Ø	100	SP	0	@	P	RU 1	RU16	RD 1	RD16	H=00	H=61	V=00	V=24	C 1	C17
1	Ø	200	!	1	A	Q	RU 2	RU17	RD 2	RD17	H+ 2	H- 2	V+ 1	V- 1	C 2	C18
2	Ø	500	"	2	B	R	RU 3	RU18	RD 3	RD18	H+ 4	H- 4	V+ 2	V- 2	C 3	C19
3	Ø	1s	#	3	C	S	RU 4	RU19	RD 4	RD19	H+ 6	H- 6	V+ 3	V- 3	C 4	C20
4	EOT	2s	$	4	D	T	RU 5	RU20	RD 5	RD20	H+ 8	H- 8	V+ 4	V- 4	C 5	C21
5	Ø	5s	%	5	E	U	RU 6	RU21	RD 6	RD21	H+10	H-10	V+ 5	V- 5	C 6	C22
6	Ø	10s	&	6	F	V	RU 7	RU22	RD 7	RD22	H+12	H-12	V+ 6	V- 6	C 7	C23
7	Ø	20s	'	7	G	W	RU 8	RU23	RD 8	RD23	H+14	H-14	V+ 7	V- 7	C 8	C24
8	Ø	½m	(	8	H	X	RU 9	RU24	RD 9	RD24	H+16	H-16	V+ 8	V- 8	C 9	C25
9	Ø	1m	)	9	I	Y	RU10	RU25	RD10	RD25	H+18	H-18	V+ 9	V- 9	C10	C26
A	Ø	2m	*	:	J	Z	RU11	RU26	RD11	RD26	H+20	H-20	V+10	V-10	C11	C27
B	Ø	5m	+	;	K	[	RU12	RU27	RD12	RD27	H+22	H-22	V+11	V-11	C12	C28
C	Ø	10m	,	<	L	\	RU13	RU28	RD13	RD28	H+24	H-24	V+12	V-12	C13	C29
D	Ø	½H	-	=	M	]	RU14	RU29	RD14	RD29	H+26	H-26	V+13	V-13	C14	C30
E	Ø	1H	.	>	N	^	RU15	RU30	RD15	RD30	H+28	H-28	V+14	V-14	C15	C31
F	Ø	1½H	/	?	O	—	RU16	RU31	RD16	RD31	H+30	H-30	V+15	V-15	C16	C32

TABLE ABBREVIATIONS

Ø	Nothing
EOT	Must be last on table, end.
SP	Space, moves 4 dots right.
RU	Roll Up X lines.
RD	Roll Down X lines.
C	Complements screen from line X.

EXAMPLE:

0200	A212	0210	1200	0220	EFF6	0230	6060
0202	6000	0212	A0C0	0222	0460	0232	6004
0204	F0DA	0214	A84E	0224	6060		
0206	F10A	0216	4557	0226	6060		
0208	A223	0218	2043	0228	6060		
020A	F1DA	021A	4F44	022A	6060		
020C	F00A	021C	4553	022C	6060		
020E	00E0	021E	138F	022E	6060		

We feel that the extra functions and new commands contained in this EPROM will make a worthwhile addition to your system, and that is very reasonably priced, at $17-50.

Available from Keith Semrad, [redacted]

[redacted]

****

NICK PYNER

Here is a back to front 6800 instruction set to assist in analysing assembly language programs. "Box" refers to the area on the standard instruction sheet where the code may be found viz:

1. addressing modes
2. index register and stack
3. jump and branch
4. condition code register

Code	Mnemonic	Mode	Box	Code	Mnemonic	Mode	Box	Code	Mnemonic	Mode	Box
01	NOP	Inhr	3	2F	BLE	Rel	3	5A	DECB	Inhr	1
06	TAP	"	4	30	TSX	Inhr	2	5C	INCB	"	1
07	TPA	"	4	31	INS	"	2	5D	TSTB	"	1
08	INX	"	2	32	PULA	"	1	5F	CLRB	"	1
09	DEX	"	2	33	PULB	"	1	60	NEG	Indx	1
0A	CLV	"	4	34	DES	"	2	63	COM	"	1
0B	SEV	"	4	35	TXS	"	2	64	LSR	"	1
0C	CLC	"	4	36	PSHA	"	1	66	ROR	"	1
0D	SEC	"	4	37	PSHB	"	1	67	ASR	"	1
0E	CLI	"	4	39	RTS	"	3	68	ASL	"	1
0F	SET	"	4	3B	RTI	"	3	69	ROL	"	1
10	SBA	"	1	3E	WAI	"	3	6A	DEC	"	1
11	CBA	"	1	3F	SWI	"	3	6C	INC	"	1
16	TAB	"	1	40	NEGA	"	1	6D	TST	"	1
17	TBA	"	1	43	COMA	"	1	6E	JMP	"	3
19	DAA	"	1	44	LSRA	"	1	6F	CLR	"	1
1B	ABA	"	1	46	RORA	"	1	70	NEG	Ext	1
20	BRA	Rel	3	47	ASRA	"	1	73	COM	"	1
22	BHI	"	3	48	ASLA	"	1	74	LSR	"	1
23	BLS	"	3	49	ROLA	"	1	76	ROR	"	1
24	BCC	"	3	4A	DECA	"	1	77	ASR	"	1
25	BCS	"	3	4C	INCA	"	1	78	ASL	"	1
26	BNE	"	3	4D	TSTA	"	1	79	ROL	"	1
27	BEQ	"	3	4F	CLRA	"	1	7A	DEC	"	1
28	BVC	"	3	50	NEGB	"	1	7C	INC	"	1
29	BVS	"	3	53	COMB	"	1	7D	TST	"	1
2A	BPL	"	3	54	LSRB	"	1	7E	JMP	"	3
2B	BMI	"	3	56	RORB	"	1	7F	CLR	"	1
2C	BGE	"	3	57	ASRB	"	1	80	SUBA	Imm	1
2D	BLT	"	3	58	ASLB	"	1	81	CMPA	"	1
2E	BGT	"	3	59	ROLB	"	1	82	SBCA	"	1

Code	Mnemonic	Mode	Box	Code	Mnemonic	Mode	Box	Code	Mnemonic	Mode	Box
84	ANDA	Imm	1	AC	CPX	Indx	2	D6	LDAB	Dir	1
85	BITA	"	1	AD	JSR	"	3	D7	STAB	"	1
86	LDAA	"	1	AE	LDS	"	2	D8	EORB	"	1
88	EORA	"	1	AF	STS	"	2	D9	ADCB	"	1
89	ADCA	"	1	B0	SUBA	Ext	1	DA	ORAB	"	1
8A	ORAA	"	1	B1	CMPA	"	1	DB	ADDB	"	1
8B	ADDA	"	1	B2	SBCA	"	1	DE	LDX	"	2
8C	CPX	"	2	B4	ANDA	"	1	DF	STX	"	2
8D	BSR	Rel	3	B5	BITA	"	1	E0	SUBB	Indx	1
8E	LDS	Imm	2	B6	LDAA	"	1	E1	CMPB	"	1
90	SUBA	Dir	1	B7	STAA	"	1	E2	SBCB	"	1
91	CMPA	"	1	B8	EORA	"	1	E4	ANDB	"	1
92	SBCA	"	1	B9	ADCA	"	1	E5	BITB	"	1
94	ANDA	"	1	BA	ORAA	"	1	E6	LDAB	"	1
95	BITA	"	1	BB	ADDA	"	1	E7	STAB	"	1
96	LDAA	"	1	BC	CPX	"	2	E8	EORB	"	1
97	STAA	"	1	BD	JSR	"	3	E9	ADCB	"	1
98	EORA	"	1	BE	LDS	"	2	EA	ORAB	"	1
99	ADCA	"	1	BF	STS	"	2	EB	ADDB	"	1
9A	ORAA	"	1	C0	SUBB	Imm	1	EE	LDX	"	1
9B	ADDA	"	1	C1	CMPB	"	1	EF	STX	"	1
9C	CPX	"	2	C2	SBCB	"	1	F0	SUBB	Ext	1
9E	LDS	"	2	C4	ANDB	"	1	F1	CMPB	"	1
9F	STS	"	2	C5	BITB	"	1	F2	SBCB	"	1
A0	SUBA	Indx	1	C6	LDAB	"	1	F4	ANDB	"	1
A1	CMPA	"	1	C8	EORB	"	1	F5	BITB	"	1
A2	SBCA	"	1	C9	ADCB	"	1	F6	LDAB	"	1
A4	ANDA	"	1	CA	ORAB	"	1	F7	STAB	"	1
A5	BITA	"	1	CB	ADDB	"	1	F8	EORB	"	1
A6	LDAA	"	1	CE	LDX	"	2	F9	ADCB	"	1
A7	STAA	"	1	D0	SUBB	Dir	1	FA	ORAB	"	1
A8	EORA	"	1	D1	CMPB	"	1	FB	ADDB	"	1
A9	ADCA	"	1	D2	SBCB	"	1	FE	LDX	"	3
AA	ORAA	"	1	D4	ANDB	"	1	FF	STX	"	3
AB	ADDA	"	1	D5	BITB	"	1				

POKER MACHINE

(0080 - 0400)

TONY HORNCASTLE,
[redacted]

This is a 3-Reel poker machine. Any three of a kind pays. One or more reels may be held provided none were held last spin and there was no payout.

To play:-

Key 0 to spin reels.
Key 1 to hold reel 1.
Key 2 to hold reel 2.
Key 3 to hold reel 3.
Key C to cancel holds.

```
0090    A083 F255 A3B6 3500   F265 20A8 A083 F265
00A0    20A8 A3B6 F255 00EE   6308 6414 A3B0 3000
00B0    D343 631E 3100 D343   6331 3200 D343 00EE
00C0    6F00 EFA1 00EE 7F01   C9FF 3F04 10C2 6F0C
00D0    EF9E 10C0 00EE

0200    A248 6303 2278 7302   333D 1204 A24A 6401
0210    6303 D340 7313 334F   1212 A25A 6300 641A
0220    D345 A260 7308 D345   A266 7308 D345 A26C
0230    632A D345 A272 7308   D345 6D29 6E00 2282
0240    0000 6500 0000 12FE   C000 8080 8080 8080
0250    8080 8080 8080 8080   8080 EE4A 4A4A 4E00
0260    EE4A 4E4A 4A00 8080   8880 E000 EEAA EE8A
0270    8A00 A0A0 E840 4000   6400 D342 6411 D342
0280    00EE 6316 641A A080   FD33 F265 F029 D345
0290    7304 F129 D345 7304   F229 D345 6338 FE33
02A0    F265 F129 D345 7304   F229 D345 00EE A3BA
02B0    6700 F91E 7701 370A   12B2 D34A 00EE 6800
02C0    3500 22AE C01E 6906   B2CA 79FF 79FF 7900
02D0    79FF 7900 79FF 7900   7900 79FF 7900 7900
02E0    79FF 7900 7900 7900   7900 22AE 6F06 FF15
02F0    FF07 3F00 12F0 7801   3807 12C2 00EE 6000

0300    6100 6200 6404 3000   1312 6309 89A0 22BE
0310    8A90 3100 131E 631C   89B0 22BE 8B90 3200
0320    132A 632F 89C0 22BE   8C90 2282 6600 4500
0330    134E 5AB0 134E 5BC0   134E 80A0 80A4 B340
0340    7619 7619 7605 7605   7605 7605 7605 7DFF
0350    6E00 2282 4600 1372   0000 2282 7D01 7E01
0360    2282 0000 0000 0000   0000 6F06 FF18 5E60
0370    135A 4D00 13AE 6000   6100 6200 2090 20C0
0380    3E00 13A4 3503 13A4   3F01 1390 6001 137C
0390    3F02 1398 6101 137C   3F03 13A0 6201 137C
03A0    4F0C 1376 6503 8504   8514 8524 1304 F100
03B0    A0E0 A000 0000 0000   0000 1818 3C3C 3C3C
03C0    FFE7 E781 0000 18FF   FFFF FF18 0000 0000
03D0    00FF 00FF 00FF 0000   0000 542A 542A 542A
03E0    0000 0018 1818 E7E7   1818 1800 087F 4848
03F0    7F09 097F 0800 00E7   E7E7 1818 E7E7 E700
```