

# DREAMER No.16

N.S.W. 6800 USERS GROUP,

\_\_\_\_\_

[illegible]

## ANYONE FOR TENNIS?

# IT'S HERE !!

## DREAMSOFT HARDWARE

AVAILABLE NOW ... DREAM 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 .....

We have come at last to the crossroads with the 'DREAMER'. Regretfully, we have decided not to continue with another six months subscription. There are several reasons for this, the falling off of the supply of new material each month, pressure of jobs and home life, the fact that production of a publication such as this demands a great deal of time and effort, so that it long ago ceased to be fun for us, and became just plain hard work.

We do have sufficient material on hand to produce another three, possibly four, issues. Rather than continue on a month by month basis, we have decided to combine all our outstanding material into one GIANT issue, to be released in January. This will save us time and money in printing and postage costs, which we will pass on to you.

We anticipate that it will be approximately 50-60 pages long, and it will be available for \$8-00, posted, which is a considerable saving over the cost of 3 or 4 single issues. For areas outside Australia, please add \$A2-00 for New Zealand, \$A3-00 for Other Areas, to cover Excess Postage.

In addition, because so many of you have asked for it, we are preparing a list of all our subscribers names and addresses, so that those of you who wish to keep in touch with each other, or find another DREAM owner in your district, may do so. This list will only be available to private purchasers of the January newsletter, for an extra \$2-00. If you do not wish to have your name and address included in this list, please drop us a line A.S.A.P., and we will leave it off.

We are also negotiating with a Sydney group at present, with a view to them taking over production of the newsletter. More news on this will be in the January issue.

Some of the things we will be bringing you will be:-

- A review on K. Semrad's new 2K EPROM. (See his ad. this issue.)
- A 'SIMPLE' Interpreter Language, to make your DREAM Multi-Lingual.
- Two Machine Code Utilities which enable you to step through a program to 'TRACE' errors.
- A revised (and much improved) version of the 'Times Tables' program, which gives multiple choices of formats.
- A 'Back to Front' 6800 Instruction Set to assist in analysing Assembly Language programs.
- A program to turn your DREAM into a Four Function Floating Point Calculator.
- A 'Wipe off Screen' routine.
- A Modified and Improved Joystick Routine.
- A 2K EPROM Programmer, and the Software to run it.
- A 'DREAM Memory Tester' Routine.
- An article on how to add a Hardware Encoded ASCII Keyboard to the DREAM.
- Also, Lindsay Fords's articles on Assembly Language programming have stirred up quite a bit of comment, (dare I say controversy?) and we will be including excerpts from some of the letters we have received on the subject.
- Plus, several new games, and some variations on previously published ones.
- Plus, articles on other subjects related to the DREAM.

In fact, it promises to be the best issue we have ever published, so don't miss it.

Included with this issue you will find a single label. To order the January DREAMER, please print your name and address on the label, and return it with your cheque or Money Order for;

\$ 8-00 (Newsletter Only)

\$10-00 (Newsletter and List of DREAM owners)

Plus: \$A2-00 Excess Postage New Zealand, \$A3-00 Other Areas.

TO : 27 Georgina Avenue, Keiraville. N.S.W. 2500. (Please also add -10c to cover Stamp Duty on all cheques sent from outside N.S.W.)

If you have 'Free Newsletter' vouchers, you may use THREE of these January, and all those who have articles and programs published in it, will not receive vouchers, but will have their \$8-00 refunded.

PLEASE, send your order in STRAIGHT AWAY, to give us some idea of how many to have printed. Also, if we get the orders in early enough, we will try and get it out to you before the New Year.

Back Issues will still be available for a limited time, at our normal rates. I.E. Issues No.'s 1 to 10, \$6-00 each. (Photo-Copies.)

No.'s 11 to 17, \$5-00 each. (While they last.) so, if you know someone who wants back issues, tell them to get their orders in quickly, before we run out.

Thank you all for your support over the past year and a half, particularly those who contributed material to help keep us going.

Our Best Wishes for the Festive Season, we hope Santa Claus brings you all something nice,

Garry and Graeme.

\*\*\*\*\*

# COMPUTER DICE

( 0200 - 0300 )

EDWARD PERATI,

To use these dice, pressing '1' will show one dice on the screen, giving a graphics display of the face. Pressing '2' will show two dice.

0200	6101	6202	E2A1	1222	E19E	1200	6A1B	6B0B
0210	2242	640A	F418	6410	F415	F507	3500	121A
0220	1200	6A0C	6B0B	2242	6A26	6B0B	2244	640A
0230	F418	640A	F415	F507	3500	1236	640A	F418
0240	1216	00E0	A2B0	C307	4300	1244	4301	1266
0250	4302	126E	4303	1278	4304	128A	4305	129A
0260	4306	12A4	1244	7A04	7B05	DAB2	00EE	DAB2
0270	7A08	7B0A	DAB2	00EE	7B0A	DAB2	7A04	7BFB
0280	DAB2	7A04	7BFB	DAB2	00EE	DAB2	7A08	DAB2
0290	7B0A	DAB2	7AF8	DAB2	00EE	228A	7A04	7BFB
02A0	DAB2	00EE	228A	7BFB	DAB2	7A06	DAB2	00EE
02B0	C0C0							

\*\*\*\*\*

# CHASE (Cont from Page 3.)

0500	5FCE	0000	A600	972E	A608	972F	DF12	CE03
0510	4137	C603	BDC2	26C6	99D1	3E27	1E33	DE12
0520	A600	AB10	A700	972E	A608	AB18	A708	972F
0530	DF12	CE03	4137	C603	BDC2	2633	DE12	08CB
0540	04D1	372F	BF7A	00A2	2736	3900	0000	0000
0550	CE00	80DF	12CE	0321	A600	08DF	14DE	12A7
0560	0008	8C00	A027	06DF	12DE	1420	EB86	06B7
0570	00A2	3900	CE00	00DF	A0DF	A286	0697	3639
0580	8608	B700	A25F	CE00	80DF	1237	BDC1	3298
0590	20DE	125F	E710	E718	8540	2706	8510	2708
05A0	2012	8504	271A	2024	C630	E100	2706	8601
05B0	A710	2022	C619	E108	2706	8601	A718	2016
05C0	C608	E100	2706	867F	A710	200A	C601	E108
05D0	27D6	06FF	A718	3308	CB04	D137	2FAB	3900
05E0	A320	8A00	8B10	7A01	7B01	8A84	8B94	DAB1
05F0	00EE	0000	0000	0000	0000	0000	0000	0000

\*\*\*\*\*

CHASE

( 0200 - 0600 )

Mr. K. BOLCH,

This program utilises M.J.B.'s Joystick Controller. The object is to clear all the dots while avoiding the monsters. You are the square with the dot in the centre. Clearing one of the dots on the extreme left or right hand side will halve the monster's speed. Note that 0080 - 00A2 is used as a scratch pad.

0200	0204	1264	063B	F780	1306	61F7	8012	063F
0210	F780	130E	0219	DF00	397A	0020	7A00	217D
0220	8012	7C00	1696	1684	0327	013B	86FF	973C
0230	973D	7F80	1286	21B7	8012	064A	B680	1246
0240	4624	037C	003C	4624	037C	003D	5A26	ED96
0250	3C80	0A2C	014F	973C	963D	800A	2C01	4F47
0260	973D	3B10	0574	6708	6204	22C0	2280	0500
0270	2400	23E0	4500	126C	75FF	2280	2400	126C
0280	6A07	8A12	4A01	2290	0450	4F01	13A0	00EE
0290	6A07	8A02	4A00	2484	00EE	7EC0	0000	0000
02A0	6800	6900	4C00	1200	4D00	697F	4D1F	6901
02B0	4C3F	12BA	00EE	68FF	00EE	6801	00EE	0000
02C0	034D	6400	6012	6109	A31D	D013	0550	6E99
02D0	0500	6308	6E00	6801	6900	6500	00EE	96A3
02E0	9B34	8164	2305	7C00	A180	64CE	0038	97A3
02F0	D6A1	D734	7EC1	E000	0000	0000	0000	0000

0300	FF00	4400	1F5F	1F5F	1F00	4400	1F5F	1F5F
0310	1F00	4400	1F5F	1F5F	1F00	4400	FFE0	A0E0
0320	8003	0303	0335	3535	3501	0911	1901	0911
0330	1901	0101	01FF	FFFF	FF00	0000	0000	0000
0340	00E0	E0E0	9638	9BA1	97A1	9738	397F	00A0
0350	CE03	00DF	12CE	0100	DF14	DE12	A600	08DF
0360	12DE	14C6	07A7	0008	5A26	FA08	8C01	E827
0370	04DF	1420	E5CE	0100	A600	841F	8B20	A700
0380	0808	0808	0808	0886	80A7	0008	8C01	E826
0390	E739	24AC	6F80	2402	FF0A	029A	0000	0000
03A0	25E0	A31D	D013	D013	4F01	1430	040C	7401
03B0	4458	23C0	6F01	FF18	00EE	0000	0000	0000
03C0	371C	7704	471C	13CC	24AC	12C0	3601	76FF
03D0	670C	7201	13C8	0000	0000	0000	0000	0000
03E0	3900	13EC	3800	13F6	6301	00EE	4D00	69FF
03F0	4D1F	6901	00EE	4C00	68FF	4C3F	6801	00EE

0400	8F60	FF15	FF07	3F00	1404	00EE	9630	8105
0410	2705	8133	2701	3996	3181	0127	0D81	0927
0420	0981	1127	0581	1927	0139	0620	9735	3900
0430	72FF	4200	1392	24AC	12C0	0000	0000	0000
0440	0000	0000	0000	0000	0000	0000	0000	0000
0450	9630	972E	9631	972F	0603	BDC2	2196	309B
0460	3881	3626	014A	8102	2601	4C97	3097	2E96
0470	319B	3926	014C	811A	2601	4A97	3197	2FC6
0480	037E	C221	6308	6900	6800	4C00	149C	4D00
0490	14A4	4C3F	14A0	4D1F	14A8	00EE	68FF	00EE
04A0	6801	00EE	69FF	00EE	6901	00EE	00E0	6305
04B0	6B0B	D3B3	0000	0000	02DE	FA29	6B0A	6328
04C0	D3B5	73FC	F929	D3B5	73FC	A038	F433	FA29
04D0	D3B5	73FC	F929	D3B5	73FC	F829	D3B5	6300
04E0	F229	D3B5	6FFF	1402	0000	0000	0000	0000
04F0	0000	0000	0000	0000	0000	0000	0000	0000

## ASSEMBLY LANGUAGE AND THE ROLLING CHECKSUM

### PART 3

by Lindsay R. Ford,  
"Dreamcards",  
8 Highland Crt.,  
Eltham Nth., 3095 Vic.

*In Parts 1 & 2 of this article I've told you a bit about the basics of Assembly Language and shown you how to write a checksum routine. This last article explains some of the instructions we used in the previous Part, polishes up the routine a little and shows you how to make it fully automatic. Read on MacDuff - if you've grasped the material in the last two parts then this one will make you a fair dinkum Assembly Language Programmer!!*

---

Have you managed to figure out the answer to that question I asked you last month- the one about why I had used Accumulator B for the Checksum total when the display routine in the Eprom only works with the contents of Accumulator A? If you worked your way through the display sequence I gave you using Michael's book the answer should have been fairly obvious - before the display routine in the Eprom can be accessed it is necessary to white out the display window and set the cursor position, both functions that require Accumulator A set to a fixed value. If the checksum total was in this Accumulator at the time this was done then it would be lost, hence I used Accumulator B for the calculation and then left the total there right up until we got to the display routine. This sort of trick is frequently required with Assembly Language as the Accumulators are the only practical places to store data and there's only two of them (in Chip 8 you have 16!)

Anyway, before we go on with the finishing touches to the routine, let's have a look at the various instructions we used in the display sequence last month;

- 'BD' - JSR This is the "Jump to Subroutine" instruction we talked about. The four digit operand contains the address of the subroutine and the names given in the comments are the subroutine names given by Michael Bauer in his commented listing of the Interpreter. Note that an assembly language subroutine must end in a '39' (in the same way that a Chip 8 subroutine must end in an '00EE') if control is to be returned to the main programme once the subroutine has been executed.
- '86' - LDA A This "Load Accumulator A" instruction sets Accumulator A equal to the 2 digit operand following it. Have a look at the explanation of how the LDX instruction works given in the first part of this article - the same applies here.
- '17' - TBA "Transfer Accumulators" does exactly what it says, in this case setting Accumulator A equal to Accumulator B.
- '20 FE' - BRA The 'BRA' instruction is similar to 'BNE' except that it "Branches Always" (irrespective of the Zero Status Flag etc.) It is therefore similar to a simple "goto" instruction, but is used where you want the programme to be relocatable. If you bothered to work out the branch offset in this particular case you will have noticed that it branches endlessly to itself. Why have we done this when Michael Bauer's book says that Assembly Language programmes should be terminated with a "Jump to Monitor" command ('7E C360')? The reason is that Michael's book was written before the "Dreamsoft No. 2" Eprom became available and this device responds immediately control is returned to the monitor by blanking the display and replacing it with an address prompt. As we want the display retained but do not want the programme to go any further we simply insert this Branch instruction, causing an endless loop that can only be broken by pressing 'reset'.
- '7E' - JMP This instruction wasn't in our programme, but as I referred to it above I'd better describe it. It is a "Jump" instruction that

causes the programme to go to the location specified in the four digit operand (in the same way that this would happen in Chip 8 with a 'lxxx' "goto" instruction). Like the JSR instruction, it is capable of addressing 64K. '\$C360' is the Monitor entry point in the Interpreter.

### Accessing From Chip 8

In the last two months we've designed ourselves a lovely self-contained Assembly Language programme that we can run with an '0080 FN 3', but what if we want to access it direct from Chip 8 so that it can form part of our main Chip 8 programme? Well we could use the Chip 8 'Do Assembly Language Subroutine' instruction 'OMMM' (if I get any letters saying "but my keyboard doesn't have an 'M' key ..." I'll personally strangle the writers - 'MMM' simply refers to a memory location), but this causes problems of its own. To begin with, we would have to put this instruction low down in the programme to be tested and this in turn would require us either to re-write it or to delete one of its instructions to make room. The other hassle is that;

THE CHIP 8 "DO ASSEMBLY LANGUAGE SUBROUTINE" INSTRUCTION 'OMMM' WILL NOT ADDRESS A SUBROUTINE LOCATED IN MEMORY BELOW \$0100 (I've emphasised this as many Dreamers will not be aware of this limitation of the instruction).

The answers to both of these problems are fairly simple. Let's take as an example our 2K programme "The Auto-Seducer" (\$0200 - \$07FF) and say that at \$0200 it has the Chip 8 instruction '6A00'. Then;

#A				#B			
\$0200	2800	Gosub	\$0800	\$0200	2800	Gosub	\$0800
\$0800	080C	Gosub (Ass. Lang.)	\$080C	\$0800	080C	Gosub (Ass. Lang.)	\$080C
2	6064	} Beep for 2.0 sec		2	6064	} Beep for 2.0 sec	
4	F018			4	F018		
6	00E0		Clear the display	6	00E0		Clear the display
8	6A00	Set A = 00		8	6A00	Set A = 00	
A	00EE	End		A	00EE	End	
\$080C	BD 0080	JSR	\$0080	\$080C - 29	Checksum Routine (relocated without change)		
F	39	END					

In each of these examples we have used the 'OMMM' instruction within its range by making it call a subroutine at \$080C, but in #A it merely calls an Assembly Language "Jump to Subroutine" instruction that re-directs the computer back down to the Checksum at \$0080, whereas in #B we have moved the Checksum itself up to \$080C so that it can be addressed by the 'OMMM' instruction direct. Neither alternative is preferable, the one you choose depending on where you have the memory space available to conveniently locate the Checksum. In either event, though, you must replace the '20 FE' BRA instruction at the end of the Checksum with a '39' END instruction to allow it to work as a subroutine to the Chip 8 programme.

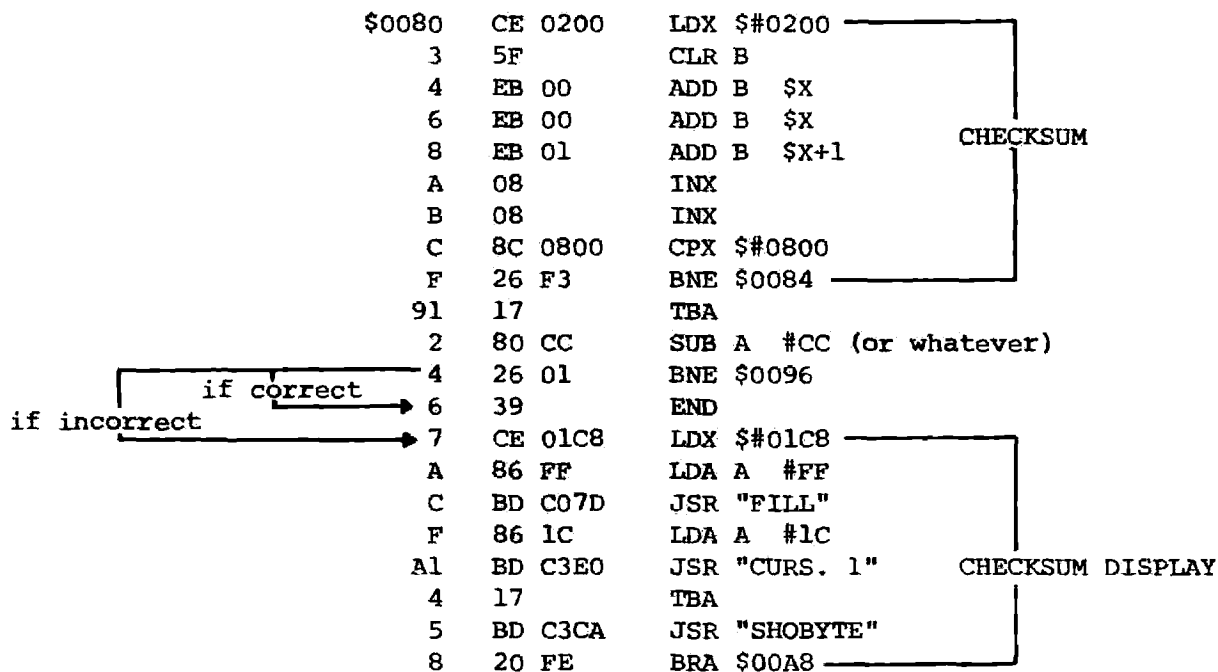
What we achieve with either of these suggested Chip 8 routines is that the Checksum runs and then the display is held for 2.0 seconds whilst a 'beep' is produced. If the displayed total indicates an error you can press the 'reset' button to stop further programme execution, otherwise the display will blank, Variable A will be set to #00 as it would have been had the instruction at \$0200 not been removed to make way for the '2800' "Gosub" instruction and control will be returned to the main programme.

### The Automatic Checksum;

Now there's something a little crude about this routine in that the computer should be able to work out for itself if there is a Checksum error without you needing to watch displays and press reset buttons. The way we can achieve this is simply by checking the total after the Checksum is done, but before it is displayed. To do this we first store the Checksum total (which at that stage is in Accumulator B) in Accumulator A, then subtract what we know to be the correct total (say in this case it's #CC). If the result is zero then the Checksum must indicate a correct result and we can use a Branch instruction to end the routine then and there and return control to the Chip 8 programme. If the result of the subtraction is other than zero then an error must have occurred and our branch instruction can be told to throw us into the routine to display the Checksum total (this is still in Accumulator B, so it's not affected by the subtraction) and then halt further execution with an endless loop.

This is how we do it;

# THE AUTOMATIC ROLLING CHECKSUM



Wow!! That all started to get a little complicated towards the end (it was the logic, not the Assembly Language), but we've now got ourselves a working Checksum. To address it you still use the Chip 8 routines suggested earlier, but there's no longer any need for the "beep" and "clear display" instructions at \$0802-7 as the whole thing is now fully automatic. Now I'm not going to suggest uses for the Checksum (these should be fairly obvious), but one thing I would say is this: how about using the non-automatic routine (the one you start with an '0080 FN 3') to total the material above \$0200 in any programme you supply to the "Dreamer" - that way Dreamers who want to try your programme can key it in and then use the same routine to check their programming before they try to run it.

In any event, this series of articles was not written to show you how a Checksum works - I could have told you that in a single page! The idea behind the whole exercise was to strip away some of the air of mystery that surrounds Assembly Language and get you on your way to understanding it. After a bit of head-scratching and some bad language (not of the computer type) you should have some of the basics of the language and should also know how the following instructions work:

LDX	INX	DEX	CPX	LDA	CLR	ADD	SUB	TBA
BRA	BNE	JSR	JMP	OMMM (Chip 8)	END			

Now that's not bad considering how hard you probably thought it was to begin with. Of course you've still got a way to go before you can really say that you "know" the language (we haven't covered the Programme Counter or Stack Pointer in any detail, and the Status Register is still very mysterious), but at least now you should have enough of a grounding to start working it out for yourself. What you should do at this point (if you haven't already done so) is to buy a copy of Michael Bauer's "Chip 8" book and a good Assembly Language text such as Lance Leventhal's "6800 Assembly Language Programming" (Osborne & Assoc. 1978) as all your remaining questions will find their answers in one or the other.

If you are still to be convinced that Assembly Language is of use to you after having read these articles, then let me throw down a challenge; I'll give a free copy of "Dream Rummy" to anyone who can write a Chip 8 routine that duplicates what the Checksum routine here does in less than double its length (that's fair, isn't it?) If your Chip 8 version can check 2K of programme in under 5 seconds I'll throw in "Dream Pontoon". When you've given this a go, pause to think on why people who buy "Basic" computers need so much memory and a lot of spare time!!

Happy Dream-Assembling

\*\*\*\*\*

TIME SQUASH

( 0200 - 0400 )

ARTHUR HAND,  


How long can you keep the 10 balls in play? 30 seconds, 1 minute? Or can you last for 10 minutes? The timer on the right of the screen will show you how long. Move your racquet LEFT - Key 4 (address 0267) or RIGHT - Key 6 (address 026D) to keep the ball in the court. The number of remaining balls is shown under the timer.

Press any key to start or restart the game. After 10 minutes, (if you can last that long!) the timer will reset to 00:00. If you can last for two minutes you are doing well. The timer is reasonably accurate, considering that it is only counting each cycle in the program, up to 33, (21 Hex) which equals about one second. Adjustments in the program count can be made at 0263.

```

0200 6A0A 6B00 A3A4 DAB1 7A01 3A26 1206 6A0A
0210 6B01 DAB1 7B01 3B1E 1212 6A25 6B01 DAB1
0220 7B01 3B1E 121E 630A 6400 6500 6600 A3C8
0230 6000 F055 2310 2326 2350 A3A5 6716 681E
0240 D781 13B2 0000 8A70 8B80 6901 CE01 3E01
0250 6902 6C01 3E01 6CFF 6DFF 8A94 A3A4 DAB1
0260 7401 4421 22FA 6904 E9A1 2332 6906 E9A1
0270 2340 A3A4 DAB1 8AC4 8BD4 DAB1 3F01 1290
0280 4B1E 12B2 4A0A 12E0 4A25 12EA 4B00 12F0
0290 3B1F 1260 DAB1 236A 22FC 2350 73FF 2350
02A0 4300 12AA A3A5 D781 123A 2378 F90A 00E0
02B0 1200 0390 4A0A 12D8 4A25 12D2 9A70 12D2
02C0 8970 7901 9A90 12D2 6DFF 6C01 1260 0000
02D0 0000 6CFF 6DFF 1260 6C01 6DFF 1260 0000
02E0 238A 6C01 4B00 6D01 1260 238A 6CFF 12E4
02F0 238A 6D01 1260 0000 0000 6400 2326 7501

```

```

0300 353C 1326 2310 6500 7601 13A8 2310 2326
0310 A3C4 6928 6E0A F633 F265 F129 D9E5 7904
0320 F229 D9E5 00EE A3C4 6932 6E0A F533 2318
0330 00EE 4706 00EE A3A5 D781 77FF D781 00EE
0340 4726 00EE A3A5 D781 7701 D781 00EE 0000
0350 A3C4 F333 F265 6930 6E18 F129 4100 1362
0360 D9E5 7904 F229 D9E5 00EE 6E00 238A 0390
0370 7E01 3E0F 136C 00EE 6E00 0390 0390 0390
0380 238A 7E01 3E0C 137A 00EE 6902 F918 00EE
0390 C602 D721 C640 F780 127D 0021 26FB C601
03A0 F780 1239 80F0 0000 460A 6600 2310 2326
03B0 00EE A3C8 F065 4001 1246 FF0A 6001 A3C8
03C0 F055 1246

```

\*\*\*\*\*

ERATTA

Did you have trouble making 'SLIDE' (July Dreamer) work? If you did, it was our fault. Somehow, we left the last byte off the program listing. It should end;

0380 137C 00EE \*\*\*\* \*\*

\*\*\*\*\*

1. I'm confused about the new 'hi-res' RAM address decoding scheme; how does it work?

The address space is divided into 4 main sections (each 16k) by decoding A14 and A15, thus:

A15	A14	BLOCK SELECTED	
0	0	\$0000	User RAM
0	1	\$4000	Video RAM buffer
1	0	\$8000	I/O devices (PIAs, etc)
1	1	\$C000	Monitor EPROM

The original DREAM circuit provided on-board decoding for 3 of these blocks, excluding \$4000-7FFF. The device-select logic has been modified to decode this block for the on-board Video RAM (all 1K), but there were too few gates available to provide a select line (RAM) for the 'user RAM' at \$0000. Therefore the external user RAM board must perform this function on its own. Any standard RAM board designed for use by the Motorola 6800 bus will have its own address decoding, but the simplified expander boards knocked up specially for the DREAM probably don't. The minimum requirement is that A14 and A15 be decoded to give an enable signal, where  $E = A14 \cdot A15$ . (see also E.A. July 79, p85 and ans to Q8 and Q5). The RAM expansion circuit given in the DREAM INVADERS instructions will work with the hi-res system, simply by disconnecting BA.

2. What will the 'HIDIOS' monitor EPROM contain?

Well, for a start, its name has been changed to "HIRESM" (HI-Res Extended Monitor), many will be relieved to hear. But, let me emphasize that it doesn't exist yet. So far, the response to the survey coupon has been encouraging, but only about 10% of the circulation of DREAMer. If this figure reaches 20%, then "HIRESM" will almost certainly come into being. If so, it will definitely contain the following:-

- \* A software switch, to swap between 64x32 and 128x64 display format.
- \* A faster CHIP-8 interpreter, with powerful new instructions added.
- \* More monitor commands, improved 'memod', user prompting, etc.
- \* Alphanumeric display routines (9 lines x 32 chars), scrolling.
- \* Built-in program de-bugging and development aids.

3. Will the 'DREAMSOFT' EPROMs (1 and 2) be at all compatible with the new hi-res system?

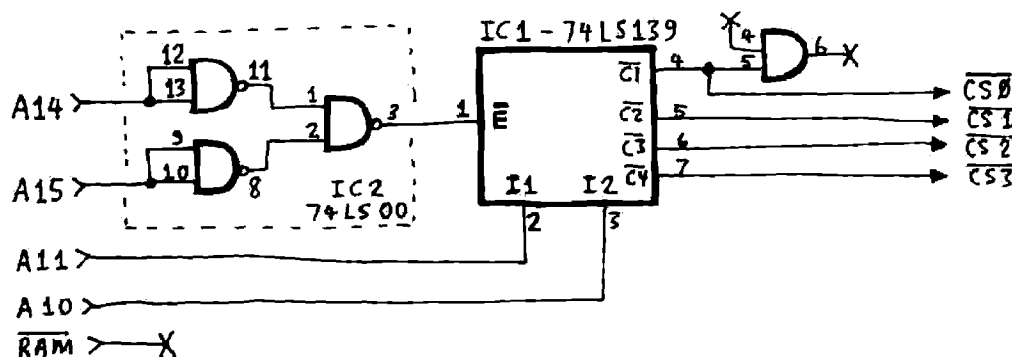
Absolutely not. It is anticipated that the majority of DREAM owners will stick with the standard format, while a minority of enthusiasts who like to dabble with hardware and machine code (i.e. experienced 'hackers'), will branch off into the nebulous realms of hi-res.

4. Will DREAM INVADERS run on the hi-res system?

No. It will require extensive patching. I will probably get around to adapting D.I. for hi-res, but no guarantees, O.K? (If so, it will be published, or distributed free of charge; the patch data, that is, not the whole program!)

All machine-code programs which write data directly into the video RAM buffer (\$0100-0200) will need to be modified to run under HIRESM.

- (i) Remove the 74LS08 and replace it with a 74LS00.
- (ii) Cut track at IC2/pin6 & connect track to IC2/pin5, as shown.
- (iii) Disconnect RAM, which is no longer available from the DREAM board.
- (iv) Make new connections with A14 & A15, as shown.
- (v) Disconnect BA. (This pin11 on the exp. skt may now be used for WE.)



Yes; I recommend reserving this space for use by utilities that reside in RAM along with user programs, i.e. relocaters, dumpers, I/O drivers & service routines, de-buggers, etc.

8. Why isn't the space \$4400-8000 free for expansion?

9. Can the 6802 be used in a hi-res modified DREAM?

10. Why don't suppliers of EPROM software offer their products as a listing only? Surely this would result in a much lower price?

Many people ask this question. What you are paying for, in fact,

is not so much the EPROM itself, nor the labour in burning it in, but the INFORMATION and the effort which went into developing the program. You are also paying, indirectly, on behalf of all the rogues who COPY software. Computer software, video tapes, records, cassettes, books and magazines would all be cheaper if consumers took the copyright laws more seriously. (Would you beleive?)

11. How about a new PCB for the HI-RES DREAM?

This is a distinct possibility. I have been thinking along the lines of a hi-res (128x64) display, with HI-REM monitor ROM, single 5 volt power supply, improved clock circuit, improved tape I/O, etc. Couple this with any of the expander boards around, and you would have a very respectable system for a very modest outlay. There is still nothing around that offers the newcomer to micro technology as much as the DREAM-6800 for his money, despite E.T.I's feeble attempt to clone the RCA Cosmac VIP!

12. What about colour expansion for the DREAM?

Adding colour is neither simple nor cheap, especially if you want control over the colour of each individual pixel (which is desirable in a colour computer, I think). Therefore, colour conflicts with the original design objectives of simplicity and low-cost. (So does the connection of an ASCII keyboard, by the way.)

---

A SOUND IDEA

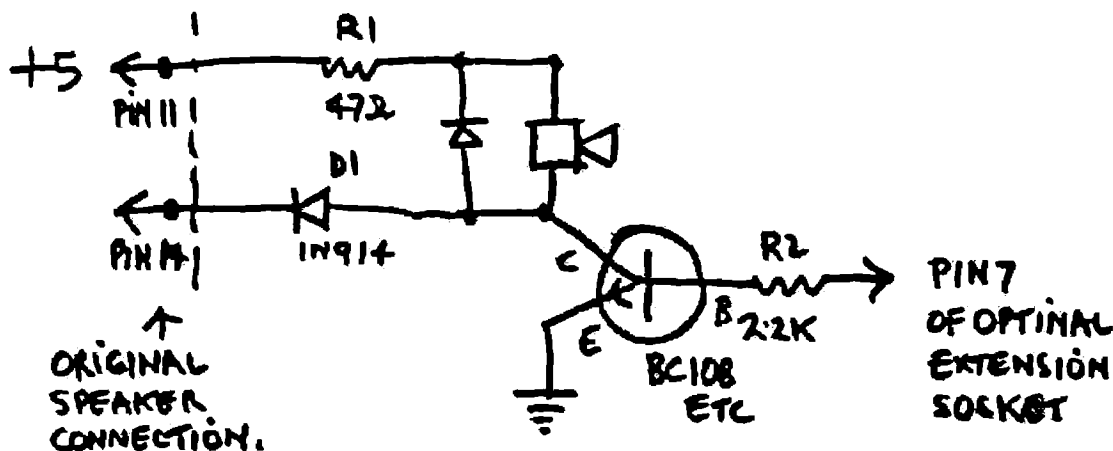
FRANK REES,  
[REDACTED]

By adding R1, R2, ( $\frac{1}{2}$  Watt Resistors), 1 Diode, and 1 Transistor, you can have inbuilt sound for your DREAM. (See circuit diagram below.)

Connection to Pin 7 of PLA B connects PB1 to base of transistor, which drives speaker, allowing you to play all the Music programs printed to date, and also play Organ and others to come. A Tag Strip or small piece of Veroboard can be used to mount parts where convenient.

INSTANT SOUND EFFECTS. If you load 'Astro Fighter' you will find that you have instant sound effects. While not loud, it is very effective, and is caused by constant polling of location 8012 by the Joystick Routine.

If more information is required, send a note with your question and a S.S.A.E. to Frank Rees at the above address.



\*\*\*\*\*

## 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]

+++++

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

DREAMSOFT,  


If you have our No.1 Software Package, a Teletype, and an ASCII Keyboard, then this program will make your DREAM into a Word-Processor.

Many business letters use standard wording. Suppose, for example, that someone wants to tell a customer that the item ordered is out of stock. A 'Form Letter' would be used and the typist would use the same tired old words that she has used hundreds of times before, and just change the 'variables' like the date, the recipient's name and order number, the item ordered and the date that stocks are expected.

This program allows you to generate Form Letters, save them on tape, and print them out as often as needed and all you need type each time are the variables.

Let's take it for a test run.....

You have to write dozens of letters to your friends, thanking them for the lousy wedding presents (or whatever).

You load the program and run from 0680, the words 'FORM LETTER START DATA?' come up on the screen. You use the hex keypad to enter the start address of RAM where the letter is to be stored. (Say 0200) The screen now displays ' 0 = RECD 1 = PLAY' You operate key '0' for the 'create' mode.

The screen blanks out and the Teletype starts up, you type the letter on the ASCII keyboard and it is printed on the Teletype (letters and figures shifts are inserted automatically), when you come to a variable you hit 'ESC' and type in the variable. At the end of the variable you hit 'EOT' (Control 'D') (ASCII '04') and continue with the letter.

If you make a mistake you can erase it by hitting 'Backspace' (ASCII '08').

At the end of the letter you hit 'EOT' and the screen comes on and displays the last RAM address used. (This is so you will know where to start your next letter.) You hit any HEX key now and the program re-starts. Enter '0200' again and select the 'Play' mode.

The Printer starts churning out the letter (without mistakes) and stops at the first variable. You type in the variable on the ASCII keyboard and terminate it with an 'EOT'. The letter continues up to the next variable and the program waits again.

At the end of the letter the last address is displayed as before. Any Hex key re-starts the program as before.

Now you can have the computer print the letter as many times as you like - and for each one you would type only the addressee's name and the type of gift.

When you finish, dump the data from 0200 to the end address onto a tape and save it for your next wedding.

The program is easily re-located, just change the data at 0681/2. This is the start address of the "FORM LETTER" message and will be Hex 70 bytes on from the program's start address.

It will not take you long to realise that the ASCII keyboard routine published in E.A. has it's limitations; E.G. the "wait for key release" feature means that after you release a key you must wait for the teletype to digest that character before you hit another key. This severely limits typing speed.

At DREAMSOFT we use hardware-encoded keyboards based on the General Instruments AY-5-2376 chip. This gives 2-Key rollover, N-Key lockout, Capitals lock (not Shift Lock), entire 128 ASCII codeset, no processor overhead and virtually no software is required.

The result is a keyboard that is much more versatile and pleasant to use and it seems to make less mistakes!

We are preparing an article on this subject and hope to publish it soon in the DREAMER.

# FORM LETTER (Cont)

**PROGRAM LISTING:** This listing was prepared using a new Machine Code Disassembler supplied to us by DREAMSOFT for evaluation, which we believe they will shortly be offering for sale. As you can see, it produces a similar format to their Chip-8 Disassembler, so we have given you the listing in this way to allow you to follow the program through and see how it works. G. & G.

<u>ADDR.</u>	<u>INSTR.</u>	<u>MNEMONIC.</u>	<u>YOUR COMMENTS</u>
0680	CE 06 F0	LDX #06F0	
0683	BD 19 04	JSR 1904	
0686	BD 18 00	JSR 1800	
0689	CE 1D 7A	LDX #1D7A	
068C	BD 19 04	JSR 1904	
068F	BD C2 C4	JSR C2C4	
0692	36	PSH A	
0693	01	NOP	
0694	01	NOP	
0695	01	NOP	
0696	BD 19 A4	JSR 19A4	
0699	32	PUL A	
069A	4D	TST A	
069B	26 18	BNE TO 06B5	
069D	BD C7 F0	JSR C7F0	
06A0	DE 02	LDX 0002	
06A2	81 08	CMP A #08	
06A4	26 05	BNE TO 06AB	
06A6	09	DEX	
06A7	DF 02	STX 0002	
06A9	20 F2	BRA TO 069D	
06AB	A7 00	STA A 00, X	
06AD	81 04	CMP A #04	
06AF	27 27	BEQ TO 06D8	
06B1	8D 0E	BSR TO 06C1	
06B3	20 E8	BRA TO 069D	
06B5	DE 02	LDX 0002	
06B7	A6 00	LDA A 00, X	
06B9	81 04	CMP A #04	
06BB	27 1B	BEQ TO 06D8	
06BD	8D 02	BSR TO 06C1	
06BF	20 F4	BRA TO 06B5	
06C1	08	INX	
06C2	DF 02	STX 0002	
06C4	81 1B	CMP A #1B	
06C6	27 03	BEQ TO 06CB	
06C8	7E 19 56	JMP 1956	
06CB	BD C7 F0	JSR C7F0	
06CE	81 04	CMP A #04	
06D0	26 01	BNE TO 06D3	
06D2	39	RTS	
06D3	BD 19 56	JSR 1956	
06D6	20 F3	BRA TO 06CB	
06D8	BD C0 79	JSR C079	
06DB	86 3F	LDA A #3F	
06DD	BD C2 FE	JSR C2FE	
06E0	DE 02	LDX 0002	
06E2	86 20	LDA A #20	
06E4	97 2E	STA A 002E	
06E6	97 2F	STA A 002F	
06E8	BD 1B 32	JSR 1B32	
06EB	BD C2 C4	JSR C2C4	
06EE	20 90	BRA TO 0680	
06F0	18 D6 46 4F 52 4D 20 4C	45 54 54 45 52 D6 04 00	

## FIRST STEP IN PROGRAMMING....

The flowchart is the most useful device for developing a computer program. Not essential, I will admit, but it does have very real advantages.

The flowchart allows the programmer to plan the sequential flow of the program, hence the name flowchart. It is important when programming to jot your ideas down on paper in easy to read statements, then arrange them in logical order. The graphical layout of the flowchart helps you to build this logical sequence, with branches and loops becoming glaringly obvious, and what's more, the instructions are written in English.

If you wanted to de-bug a problem of logical sequence, or perhaps later include sound effects etc., the flowchart can be re-arranged with ease. Very much easier than decompiling and changing a program written in some computer language.

To illustrate the use of the flowchart, two simple programs have been included, one in CHIP-8, the other in ASSEMBLY LANGUAGE. Each flowchart is very simple, but remember it serves to show how the program works and how the computer will execute the steps logically. IN ORDER TO KEEP IT SIMPLE, DETAILED INSTRUCTIONS MAY BE EXCLUDED FROM A FLOWCHART.

General instruction statements are placed in rectangles. The program always starts at the top of the page and works down vertically.

Decisions (conditional branches) within the program are contained in diamond shapes, which offer alternative paths to follow. Note that if conditions are not met, the program will not branch out, but continue down the page.

The lines which connect the shapes, indicate the logical flow of the program, the arrows show direction.

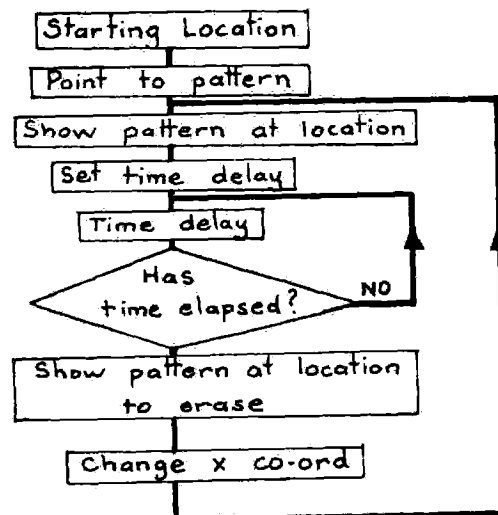
Once drawn the flowchart can be translated into any computer language.

The programs presented here have been fully expanded, to show how the language was used and how translation from the flowchart was effected. Regardless of the language used, extra operational steps are required to expand the statements in the flowchart.

Sub-routines in the second program are called from the DREAMSOFT EPROM No. 1. It is therefore necessary to have the DREAMSOFT No. 1 EPROM fitted in order to run this program. My congratulations to the DREAMSOFT BOYS ON THIS VERY WORTHWHILE addition to the DREAM. I have found the documentation supplied with the No. 1 package very useful in Assembly Language programming.

#### PROGRAM 1.     CHIP-8

This program flies a helicopter across the screen. Could be a useful routine in the development of a game program.



DATA STORED AT 021A     F820   7C70   8800

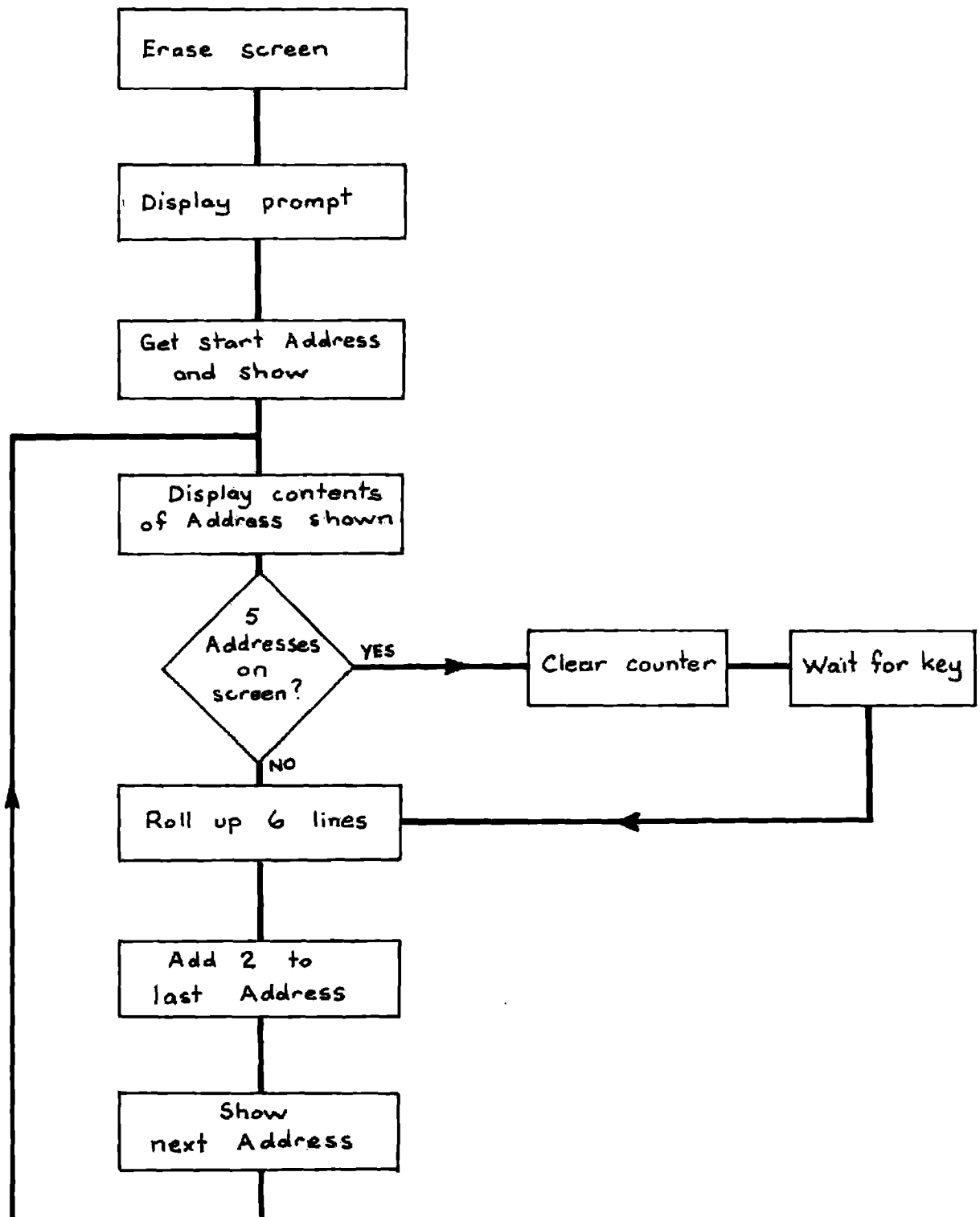
0200	6A30	VA=30	} Set Starting Co-ords (30,00)
0202	6B00	VB=00	
0204	A21A	I=21A	} Point to pattern
0206	DAB5	SHOW 5 @ A , B	
0208	0000	NOP	
020A	6608	V6=08	} Time delay
020C	F615	TIME=V6	
020E	F607	V6=TIME	
0210	3600	SKF V6=00	Has time elapsed?
0212	120E	GOTO 020E	
0214	DAB5	SHOW 5 @ A , B to erase	
0216	7AFF	VA=VA + FF	} Subtract 1 from x co-ord
0218	1206	GOTO 0206	

PROGRAM 2. ASSEMBLY LANGUAGE (M/C)

This program is relocatable and can rapidly list memory contents from any given address, in the traditional scrolled screen format.

Run from starting address (e.g. 0200 F N 3)

Type in any address to be listed from, then any key will continue listing.



LIST MEMORY

(0200 - 0300)

P. E. MARSTON.

ASSEMBLY LANGUAGE (M/C) Relocatable. USES DREAMSOFT No. 1 EPROM.

Key in start address of Memory to be listed.

Hit any key to continue listing.

<u>ADDR</u>	<u>INSTR</u>	<u>MNEMONIC</u>	<u>EXPLANATION</u>
0200	BD C079	JSR C079	Erase screen
0203	7F 003A	CLR \$003A	} Set Co-ords (00,00)
0206	7F 003B	CLR \$003B	
0209	CE 1E0C	LDX #\$1E0C	Locate prompt
020C	BD 1904	JSR 1904	Display prompt
020F	86 05	LDA \$05	} Set max. count
0211	97 42	STA \$42	
0213	7F 0043	CLR \$0043	Clear counter
0216	86 08	LDA \$08	} Set Co-ords (08,18)
0218	C6 18	LDB \$18	
021A	97 2E	STA \$2E	
021C	D7 2F	STB \$2F	
021E	BD 1818	JSR 1818	Get X and show
0221	DF 40	STX \$40	
0223	DE 40	LDX \$40	
0225	EE 00	LDX \$00,X	
0227	86 20	LDA \$20	} Set Co-ords (20,18)
0229	C6 18	LDB \$18	
022B	97 2E	STA \$2E	
022D	D7 2F	STB \$2F	
022F	BD 1B32	JSR 1B32	Display X
0232	7C 0043	INC \$0043	Increment counter
0235	96 43	LDA \$43	} Compare counter with max. count
0237	91 42	CPA \$42	
0239	27 16	BEQ 0251	Branch if same
023B	BD 1919	JSR 1919	Roll up 6 lines
023E	DE 40	LDX \$40	} Add 2 to last address
0240	08	INX	
0241	08	INX	} Set Co-ords (08,18)
0242	86 08	LDA \$08	
0244	C6 18	LDB \$18	
0246	97 2E	STA \$2E	
0248	D7 2F	STB \$2F	
024A	DF 40	STX \$40	
024C	BD 1B32	JSR 1B32	Display contents of X
024F	20 D2	BRA 0223	Branch always
0251	7F 0043	CLR \$0043	Clear counter
0254	BD 1C2A	JSR 1C2A	Wait for any key
0257	20 E2	BRA 023B	Branch always

Alterations to the  
DAY OF THE WEEK CALCULATOR

BRUCE BRODIE,  
[REDACTED]

The listing shown below from 0080 - 0100 changes the Day of the week calculator (Dreamer No.9, May '81), from the 20th to the 19th Century.

```
0080 01 86 04 00 01 05 00 03 06 01 04 06 E5 85 E5 25
0090 E7 38 28 28 28 28 F9 A9 A9 A9 A9 CE 4A 4A 4A CA
00A0 E5 45 45 45 47 38 20 38 20 38 A9 A9 A9 A9 51 CC
00B0 0A CA 0A CC E5 45 47 45 45 28 28 28 28 38 E7 85
00C0 E7 86 85 38 10 10 10 38 E7 85 E7 25 E5 38 10 10
00D0 10 10 00 20 40 80 00 00 00 30 00 00 CE AA AE AA
00E0 CA EE 48 4E 48 4E 38 08 18 00 10 00 0E 3D 01 00
00F0 17 00 04 3D 01 EB 00 EB 00 00 00 00 00 00 00 00
```

The alterations below, from 02A9 to 02B9 inclusive, allow you to restart the above program without having to key 'C'. The 'F' key to return to the monitor has also been removed from the program.

```
02A0                                C6 88 D7 20 7D 00 20
02B0 26 FB 7E 02 00 01 01 01 01 01
```

\*\*\*\*\*

Modification to  
KALAH

K. SEMRAD,  
[REDACTED]

This mod. allows you to play Kalah, (Dreamer No.6, Feb. '81) against the computer. Press 'C' to play against the computer, or 'B' for two players.

CHANGE: 0276 to A080, 0284 to A086, 0288 to A08D, 0290 to 13DE,  
02A2 to A07F, 02B2 to A07F, 02BA to A07F, 0318 to A080,  
0322 to A0FC, 0356 to 1386, 036E to A07F.

THEN ADD:

```
0380 8080 8080 8080 6201 6106 A085 F11E F065
0390 9020 13C8 7201 71FF 3100 138A 6200 6106
03A0 A086 F11E F065 8205 3F01 13C6 8204 7201
03B0 71FF 3100 13A0 6101 A086 F11E F065 3000
03C0 13C8 7101 13B8 8204 8810 6080 F015 F007
03D0 3000 13CE 1358 3000 1386 F80A 1358 FE0A
03E0 3E0B 13F0 A3FC F165 A356 F155 0001 1292
03F0 3E0C 13DE A3FA F165 13E8 13D6 F80A 13D6
```

\*\*\*\*\*

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 (searches 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, EXOR. ONLY \$17-50, from,  
K. SEMRAD, [REDACTED]

\*\*\*\*\*

DARREL WOOLNOUGH,

Thus far I have not seen in the 'Dreamer' any programs involving competition between two players, each using the keyboard to control separate object movement such as the separate racquets in this game. I can only assume that the reason for this is the way keys inter-react when simultaneously pressed - that is unless my Dream has a bug in it and is the only one to react this way. I imagine it is due to the crossbar nature of the keyboard causing confusing data on the PIA inputs.

Notwithstanding these limitations, I have devised a fast competitive game of tennis enabling two players to compete against each other rather than against the computer.

After starting (C000, FN, 3.), the outline of a court plus net, racquets and ball appear together with scores set to zero at each end. The Left Hand player serves with key '0'. The same key controls the UPWARD movement of his racquet, whilst key 'C' is DOWN, key '8' IN, and key '4' OUT. The equivalent keys for the second (Right Hand) player are 3, F, B, and 7. The locations to change to suit your keyboard are, 0279, 027F, 0285, 028B, 0291, 0297, 029D, 02A3.

The only gentleman's agreement which must be adhered to because of the nature of the keyboard as already discussed, is that once a player has hit the ball, he must not move his racquet until his opponent has hit the ball, whereupon he resumes racquet control. Contrary to what one would first believe, this restriction tends to make the game faster and I believe more exciting. If by error a player should forget or in heat of battle try to move his racquet out of turn causing his opponent to miss the ball, the offending player should allow a free point to be scored against him. Perhaps an extra penalty point should also apply.

If the ball is in the upper half of the court, the computer decides on a 50% chance basis whether the return will be straight down the line or cross court to the lower opposing half of the court. A shot from the lower court could be straight or cross court to the opposing upper court. A player can rush the net, (easier on a straight return) but he should beware that he should stop before contact with the ball, or it could pass straight through. This does not apply to cross court movement, where a running shot can be played with no problems.

Scoring is straightforward, five points to a game, and six games to a match. Once a win has occurred, (WIN appears on winner's side of the court) the game re-commences automatically after a delay. The game can be speeded up a little by changing the data at location 026C/D from 6A01 to 1276.

Borg, watch out! (Or should that be McEnroe?)

0200	6A07	6B00	A39C	7A01	DAB1	3A39	1206	7B01
0210	DAB1	3B1F	120E	7AFF	DAB1	3A07	1216	7BFF
0220	DAB1	3B00	121E	6A20	6B01	DAB1	7B0A	DAB1
0230	7B09	DAB1	7B0A	DAB1	6900	6B00	22C4	6D00
0240	6E00	22E6	630E	6418	A39C	D344	6535	6604
0250	D564	6700	681A	D781	6A0C	EA9E	1258	6A04
0260	FA18	6C02	60FF	D781	87C4	8804	6A01	FA15
0270	FA07	3A00	1270	D781	6A00	EA01	23B2	6A0C
0280	EA01	23BE	6A08	EA01	23CA	6A04	EA01	23D6
0290	6A03	EA01	23E2	6A0F	EA01	23EE	6A0B	EA01
02A0	2080	6A07	EA01	208C	6A07	97A0	1356	6A39
02B0	97A0	135A	9370	235E	9570	238C	1266	D781
02C0	22C8	7B01	22CC	00EE	6004	F018	A3FA	FB33
02D0	F265	F229	6002	6101	3B05	1306	7901	6B00
02E0	12CC	D781	22EC	7E01	22F0	00EE	6004	F018
02F0	A3FA	FE33	F265	F229	603C	6101	3E05	133A

## TENNIS (Cont)

```
0300 7D01 6E00 12F0 D015 230C 00EE A3FA F933
0310 F265 F229 6002 611A D015 3906 00EE 600A
0320 6108 A3A0 D015 7006 A3A6 D015 7004 A3AC
0330 D015 60A0 F018 00E0 1200 D015 2340 00EE
0340 A3FA FD33 F265 F229 603C 611A D015 3D06
0350 00EE 6024 1320 22E2 1394 22BE 1394 3F01
0360 00EE 6C02 6A04 FA18 6A10 8A85 3F00 137A
0370 CA01 3A00 1384 6000 00EE CA01 3A00 1388
0380 6000 00EE 60FF 00EE 6001 00EE 3F01 00EE
0390 6CFE 1364 A39C D344 D564 1244 8080 8080
03A0 8888 8A88 F800 E040 4040 E000 86C8 A898
03B0 8800 D344 4402 7402 74FE D344 00EE D344
03C0 441A 74FE 7402 D344 00EE D344 431D 73FE
03D0 7302 D344 00EE D344 4309 7302 73FE D344
03E0 00EE D564 4602 7602 76FE D564 00EE D564
03F0 461A 76FE 7602 D564 00EE 0000 0100 0000

0080 D564 4523 7502 75FE D564 00EE D564 4537
0090 75FE 7502 D564 00EE
```

\*\*\*\*\*

## MICRO SHORT

( 0200 - 0210 )

GRAEME V. SAMWAYS.

This tiny program takes whatever signal is on the TAPE INPUT and puts it on the TAPE OUTPUT. In effect, a 'short circuit' between the two. What can you use it for? Copying tapes, tape to tape.

0200	86 36	LDAA	\$36	
0202	BD C2 FE	JSR	C2FE	Turn off display and RTC
0205	CE 80 12	LDX	8012	Set X to Tape I/O point.
0208	A6 00	LDAA	X,0	Get Tape In Bit 7
020A	49	ROLA		Rotate to Carry Flag
020B	49	ROLA		Rotate Carry Flag to Bit 0
020C	A7 00	STAA	X,0	Store at Tape Out Bit 0
020E	20 F8	BRA	0208	Loop

To escape from the program, hit RESET.

\*\*\*\*\*

## CLEARANCE

DREAM INVADERS: There are still a few copies of this popular, fast moving action game, based on Space Invaders. Sorry, no cassettes left. Instructions (incl. 2K RAM expansion notes) plus HEX listing...\$5.00. Fully commented 6800 Assembly listing (18 pages) .....\$5.00.

CHIPOS MANUAL: Due to continuing demand, a new print run has been made, but increased printing costs have forced the price up a bit to \$6.00. (Foreign orders, please add \$1.00) The manual contains a fully commented source listing of the CHIPOS EPROM, plus valuable data for the machine-code programmer or student thereof, etc.

Send to: 'DREAMWARE', PO Box 343, BELMONT, VIC., 3216.