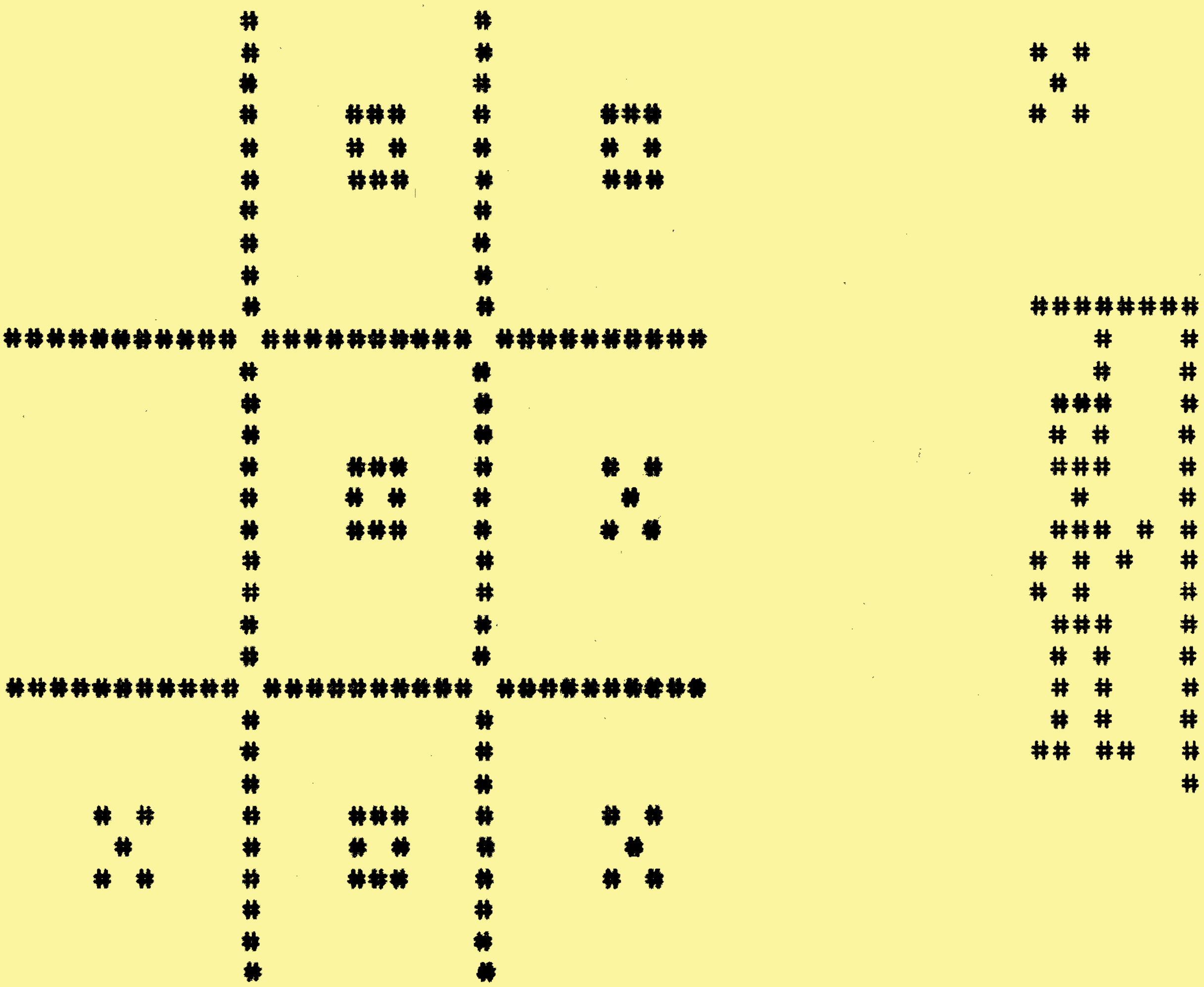
REGISTERED BY AUSTRALIA POST
Publication No.NBG4250.

DREAMER No.14
OCT, '81.

N. S. W. 6800 USERS GROUP,



Well, we said we would have a surprise for you, and we do. The HIGH RESOLUTION GRAPHICS mod. (At last) and it is now a practical conversion, being able to be switched in and out as required (under software control) so you do not have to throw away (or re-write) all your existing software. See M.J.B.'s article for full details.

ERRORS IN PROGRAM LISTINGS. IF you do find a boo-boo in one of our listings, or if a program does not work when you key it in, please first check that you have entered it correctly, then, if it still will not work, LET US KNOW. We want you to enjoy the games we give you, not give up in disgust because a program does not seem to work. We test them all before we print them, and they all run properly on our two DREAMs, so if you have received one that you can not make work, let us know so we can try to sort it out for you. Remember, we have no way of knowing unless you tell us!

HELP!! Our stream of programs and articles that we need to stay alive has dwindled to a trickle. If we are to carry on with DREAMER past December, into next year, we NEED LOTS MORE MATERIAL. So, if you have some programs that just need 'tidying up', or an article or two that is 'almost ready', PLEASE, finish them off and send them to us, so that we can keep on producing DREAMER, as our mailbag tells us that we are making a worthwhile contribution to helping people learn something about the computer they have built, and there is just nowhere else that information about the DREAM is available. We cannot write it all ourselves, due to time limitations, but we can collate and publish what you send in to us, for the benefit of ALL Dream addicts. (If you have sent in something that we have not used yet, don't be discouraged, we will get around to it. In the meantime, send us something else for consideration.)

(Continued Page 6.)

****

INVASION

GRAEME V. SAMWAYS.

Recently I took my DREAM to work to be used as part of an apprenticeship display. (A Lotto number selector, using big L.E.D. read outs.)

I just couldn't leave Dream Invaders at home, so they 'invaded' the electronics workshop. (At morning tea and lunch only, of course!)

The tradesmen were quite impressed and began by scoring in the 1000-1400 area. I then brought in the sound effects version. This nearly caused my downfall, because when they found out that the score stopped incrementing at 2500, they nearly killed me! I realised then and there that I had to write a routine that would allow the score to continue to function in some form, or I would die in the attempt.

I decided that what would be needed would be for the score to reset to zero at 2500, but leave the game speed and round score as they were.

This is what I finished with:-

```
           At 05F5 change 2703  7C  00A7
           to             0101  7E  0780     JMP  $0780
and add at    0780        26   05            BNE    #05  (already compared to 250)
              0782        7F   00A7          CLR  $00A7  (score = 0)
              0785        20   03            BRA    #03
              0787        7C   00A7          INC  $00A7
              078A        96   B5            LDAA $00B5  (Round #)
              078C        81   0A            CMPA   #0A  (after round 10)
              078E        2A   03            BPL    #03
              0790        7E   05FA          JMP  $05FA  (to speed check etc)
              0793        39                 RTS         (skip speed adj. etc)
```

This can be used with either the normal or sound effects versions. Now all you have to do is add 2500 to the displayed score if you get over 2500. It should also work at 5000, but they have so far only succeeded in scoring 4700, so we haven't yet proved this. But, it MUST work, for my sake, or I may be invaded by a hammer.

This modification leaves in all the warm up rounds, unlike the mod. we published in Dreamer No.11, which deletes the warm up rounds.

****

ERRATTA

Whoops, we did it again. Lindsay Ford has pointed out to us that the program listing for the 1K version of 'Strip Jack Naked' (I.E. without the graphics sequence added) has location 020C as 2400, and location 03FC as 2404, whereas both these should be 0000, or the program will crash. They are only changed to 2400 and 2404 if the graphics sequence (0400 - 0600) is added. Sorry 'bout that.

****

WANTED

Here is a selection of things that people have requested appear in the DREAMER. If you would like to try your hand at writing a program, or an article, but can not think of a subject, why not try one of the following?

- A CHESS program
- DRAUGHTS
- A LIGHT PEN
- A 'WESTERN GUNFIGHT' game
- More 'Joystick' programs
- A FLIGHT SIMULATOR game
- A MORSE CODE DECODER
- An EPROM PROGRAMMER
- Radio Amateur orientated programs
- More 'Serious' programs

****

ASSEMBLY LANGUAGE AND THE ROLLING CHECKSUM

PART 1

By Lindsay R. Ford,
"Dreamcards",
[redacted]

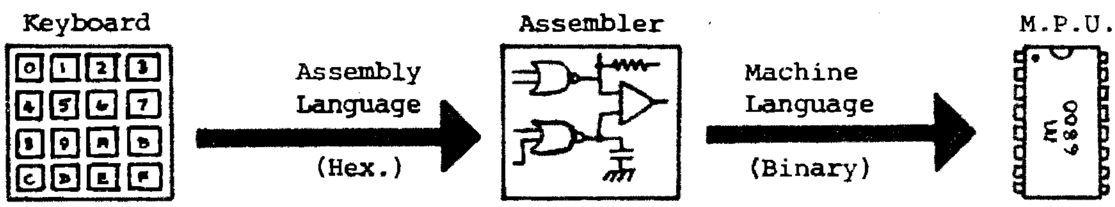
CAUTION: *The Surgeon-General warns that reading this article may be injurious to your ignorance. Mental defectives, psychopaths and those whose only interest in the Dream is running programmes written by other people should skip this article entirely. Otherwise sit down, turn off the telly and read on slowly and carefully. What we are going to try to do is to teach you Assembly Language programm--ing by developing a programme as you learn.*

So you've mastered Chip 8, but you're reluctant to try 'machine code' as it looks so complicated? Well you should stop being so negative - if that 2 Gigabytes of dynamic memory you're carrying around between your ears can't master what a Chip 8 Eprom can do in 1K, then you might as well trade in your brain on a TRS 80!

Of all the programming languages available 'machine code' offers by far the greatest speed and programme efficiency, but it can look pretty frightening because of the awful terminology that goes with it. All too often expressions such as "LDA A" and "CPX $#03D0" that are supposed to explain 'machine code' functions only serve to further cloud their meanings. Luckily these are problems we can get over by trying out the various instructions, but before we do so let's get some of the ground rules straight.

ASSEMBLY LANGUAGE CONCEPTS;

To begin with, let's stop talking about 'machine code'. This expression refers to the binary data that is stored in the computers memory, but you actually key it into the computer in hexadecimal form. These hex expressions have to be translated into binary 'machine code' by a part of the microprocessor chip (the "M.P.U.") before they can be used or stored. 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.



Now assembly language routines can be 'commented' to allow the user to understand what each particular instruction in the programme 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 the location in memory at which the material following it commences, but what of the rest? Were we to look at a book on 6800 Assembly Language programming we would see that the expression 'CE' is a recognised instruction (ie: a code which forces the computer to take some definite action) and that the '0200' following it is an operand (ie: the data on which the computer is forced to act). This may seem a little confusing at first, but if you stop and think you will see that Chip 8 works in much the same way. Take the Chip 8 instruction '6A00' - in this case '6A' is the instruction (it forces the computer to set Variable A to a value dictated by the two digits following it) and '00' is the operand (as it is the data on which the computer acts when it sets Var. A).

Now our computer book would not only tell us that 'CE' was a recognised instruction - it would also tell us that it means "Load Index Register". In an effort to shorten instruction definitions such as this, however, they have been reduced to a set of standard mnemonics (shorthand forms of the full definition) and in this case the expression "LDX" would be used instead of "Load Index Register".

Having got this far, we're still confronted with the expression $#0200 and although this obviously tells us something about the operand, the symbols are somewhat mysterious As you will commonly encounter three of these symbols, let's set them out;

- **$** - means that the numbers following it relate to a memory location. Thus an operand mnemonic preceded by this sign may indicate that the instruction jumps or branches to the particular location or that it does something to the data stored at that location.
- **#** - means that the numbers following it are data. An operand mnemonic preceded by this sign indicates that something is done with the numbers themselves (such as adding them to something else or storing them somewhere).
- ***** - means that a number ordinarily occupies the location at which it is placed, but for the purposes of the particular programme or instruction the actual value of that number is irrelevant.

At this point you're probably rather confused as in our example the '0200' was precceed by two symbols that seem to be inconsistent. After all, isn't the $ sign telling us that '0200' is a memory location and the # sign telling us that it is data? How can it be both?

The answer to this one lies in the instruction itself - as the Index Register is that part of the MPU that points it to the relevant memory location to be dealt with, then the instruction is really taking the data in the operand (ie: the number '0200') and putting it in the Index Register so that it will be treated by the computer as a memory location. Now do you understand the '$#" ? If you don't (or if anything else so far is vague) then you'd better go back to the start and read it again. These concepts are fundamental to understanding Assembly Language, so if you're not too clear on them you'll be lost before much longer.

MACHINE LANGUAGE CONCEPTS:

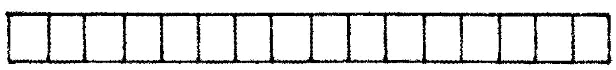
When we were programming in Chip 8 we could happily sit back and talk to the interpreter (that's the device that translates Chip 8 instructions into Assembly Language so that the assembler can recognise them) without knowing too much about what was happening in the computer itself. The Assembly Language programmer can't afford such luxuries, however, as his instruction set deals with the microprocessor direct.

If you have ever bothered to read a commented Assembly Language routine you will have seen that the instructions deal mainly with manipulation of data and addresses in things called Accumulators, Registers and Flags. There's no need to be put off by these expressions as all they refer to are small 'chunks' of memory within the microprocessor chip itself, each one having a different function. In the 6800 and 6802 MPU's there are 6 of these 'chunks' of memory capable of storing more than one 'bit' of data;

	Accumulator A	(8 bits)
	Accumulator B	(8 bits)
Dedicated to specific tasks	Index Register X	(16 bits)
	Programme Counter PC	(16 bits)
	Stack Pointer SP	(16 bits)
	Status Register	(8 bits)

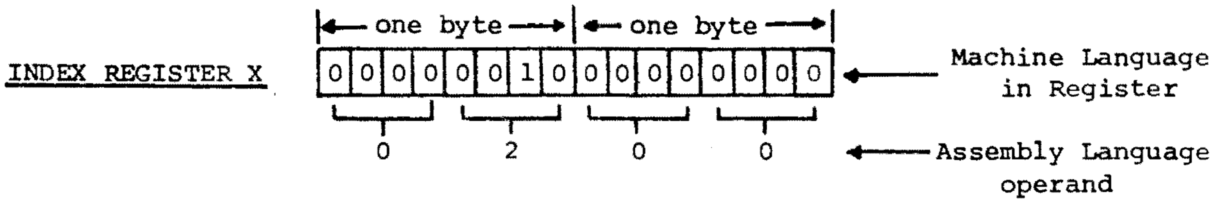
There are also a number of 'single bit' Flags used to indicate whether particular conditions have or have not been met within the MPU (you can liken these to warning lights as when they're 'on' they tell you something has happened).

You can better imagine how the 'chunks of memory' I've listed above function if you think of them as being little groups of boxes in which information is stored, each box being referred to as a bit. If we look at our earlier example about the Index Register we could represent it in diagramatic form as 16 bits (boxes) in a line;

INDEX REGISTER X 

Now each of these boxes is capable of being full or empty, but no other state is permitted (although the state of any given box may be irrelevant to the instruction or programme in question, in which case we will denote its contents with a *). Let's call the full state '1' and the empty state '0'.

You will remember that the Index Register was described as being that part of the MPU that points to the memory location to be dealt with, so what happens when a programme meets the 'LDX $#0200' instruction referred to in our previous example? If you recall that the computer cannot deal directly with hexadecimal numbers then you have probably guessed that the operand is translated into binary by the assembler and all that happens then is that these binary digits are stored in the register;



From the above you can see that each hex digit requires 4 bits of memory to hold its binary equivalent and (for the sake of convenience) we refer to the 8 bits of memory required to hold a two digit hex number as a byte. If you contemplate this for a while you will notice that the Index Register, Programme Counter and Stack Pointer can all hold 2 bytes each (just enough room for a 4 digit memory address) whereas the Accumulators and Status Register only hold 1 byte. This may tell you something about the function of those parts of the MPU, but we'll go into it in greater detail when we start working on an actual routine.

6800/6802 ADDRESSING CONVENTIONS;

Before we look at actual instructions, though, there's one area of Assembly Language that causes tremendous confusion amongst beginners and that's addressing options. In Chip 8 these options are extremely limited (for instance, you may address a routine either by a '1XXX' "goto" instruction or a 'BXXX' "computed goto" instruction, a feature that is not available on any of the other Chip 8 instructions). In Assembly Language, however, a wide range of options exists , the particular instruction and operand being different in each mode.

If you have a look at Michael Bauer's "Chip 8" book you will see a table of Assembly Language instructions and, omitting parts relating to processor cycles and other such things we're not concerned with here, a typical example is as follows;

OPERATIONS	MNEM.	IMMEDIATE	DIRECT	INDEXED	EXTENDED	INHERENT
Load Index Reg.	LDX	CE	DE	EE	FE	
Decrement Index Reg.	DEX					09

Now the five columns containing instructions are the different addressing modes of each particular type of instruction and they operate as follows;

a) Immediate Addressing:

In this mode the operand following the instruction is treated as data and acts directly on the particular sub-circuit involved. Thus in our 'CE 0200' (LDX $#0200) example the computer goes no further than the '0200' when it looks for the material the instruction commands it to load into the Index Register. Similarly, in the case of an Accumulator instruction addressed in this mode, the instruction looks to its operand for the data to act on but only two digits are used as that is the limit of the capacity of an Accumulator.

b) Direct Addressing;

This particular mode is used to address memory locations in the first page of memory (ie: below $0100). As these locations can all be specified with a two digit operand this is all that is used. For example, the instruction 'DE 80'

(LDX $0080) would load the Index Register with the data contained in memory locations $0080 and (as the Index Register holds 2 bytes) $0081.

c) Indexed Addressing:

The indexed mode also looks to memory for the data on which to act, but derives the address of the byte/s concerned by adding the 2 digit operand to the contents of the Index Register at the time the instruction is encountered. This may seem a little curious in the context of the LDX instruction, but it is of far greater importance with instructions dealing with Accumulators. To stay with our LDX example for the time being, 'EE 80' (LDX $X+80) would load the Index Register with the contents of the two bytes of memory starting at an address 80 (Hex.) bytes above the one held in the register when it first reached this instruction.

d) Extended Addressing:

This mode uses a four digit operand, but treats it as an address rather than as pure data. Thus 'FE 0200' (LDX $0200) would load the Index Register with the 2 bytes of data held in memory commencing at $0200. It is important to contrast this mode with "Immediate Addressing" or total confusion will result. You should also remember that for data in memory below $0100 the proper choice is the "direct" mode.

e) Inherent Addressing:

This mode is only employed with instructions that are so specific in their function that no operand is required (ie: the operand is 'inherent' in the instruction itself). An example of this is the DEX instruction which serves to reduce the contents of the Index Register by 01 each time it is encountered.

f) Branch Addressing:

This particular mode of addressing only applies to the special instructions in the 6800/6802 Assembly Language set known as Branch instructions (they function a little like Chip 8 "goto" instructions, but only operate where various specified conditions exist). The address to be branched to is derived from the 2 digit operand with a branch offset calculation which we'll deal with later in this article.

To summarize the main addressing modes in the form of a short "look-up" table;

a)	IMMEDIATE - Data contained in operand
b)	DIRECT - 2 digit operand points to memory below $0100
c)	INDEXED - Add 2 digit operand to contents of Index Register for address
d)	EXTENDED - Address above $00FF in 4 digit operand
e)	INHERENT - No operand
f)	BRANCH - 2 digit operand specifies branch offset

This completes our brief tutorial in the basics of Assembly Language and so next month we'll try and apply what we've learned in writing a routine. In the meantime you'd better make sure your parachute is buckled before you leave the safety of the 'plane!! If there is anything we've talked about so far that you've the slightest doubt over then read and re-read it until it's completely clear. The remainder of this article will assume you have this basic knowledge, so you'll be lost if you don't.

****

From Page 1.

NEXT MONTH, we will have, Part 2 of Lindsay Ford's Assembly Language articles, another program to help you understand Binary to Hexadecimal conversions, plus a swag of games and all our other usual bits and pieces.

Until then,

KEEP ON DREAMING,

GARRY and GRAEME

****

'HI-RES' DISPLAY MOD.

M.J. Bauer
[redacted]

In the editorial of DREAMer No 10, Graeme suggested that I had abandoned the idea of higher resolution graphics due to problems with hardware design. Well, this is really not the case at all; (and I think he knows this, but he thought he might be able to provoke some action from me. The trick worked, Graeme!)

The facts are these: (1) the hardware modification exists (I developed it ages ago) to give a 128 x 64 dot display. It's a somewhat messy modification requiring quite a bit of PCB track cutting and rewiring, but not too difficult for an experienced enthusiast. But I was very reluctant to release the modifications because (2): NONE of the programs written for the DREAM, including the CHIPOS monitor, will run on a modified version! A new EPROM is required which contains modified display routines. A DREAM-6800 which has a modified 128x64 display (as given here) will not function at all with the original CHIPOS EPROM. Even when you have a suitable EPROM installed for use with the extended graphics, none of your existing programs (games, etc) will run properly, because of the new screen format. Some CHIP-8 programs might work in the upper LHS of the screen, but most programs will go completely haywire! For example, DREAM INVADERS would require a complete rewrite, not just a few simple patches, to work on a 128x64 grid.

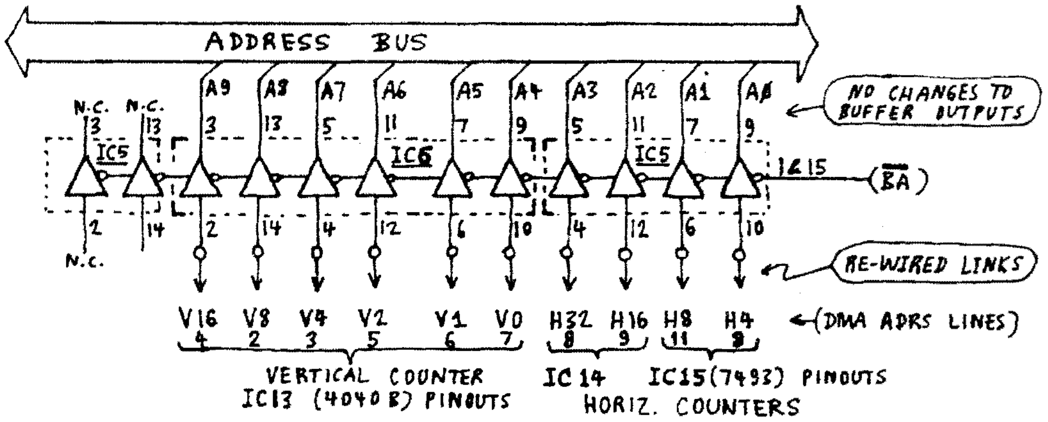
So, before you embark on any hardware mods, you should convince yourself that your standard DREAM-6800 has outlived its usefulness! Further, it is recommended that you do not attempt this upgrade unless you are reasonably experienced and have a good understanding of how the hardware works (because you might have to trace a fault).

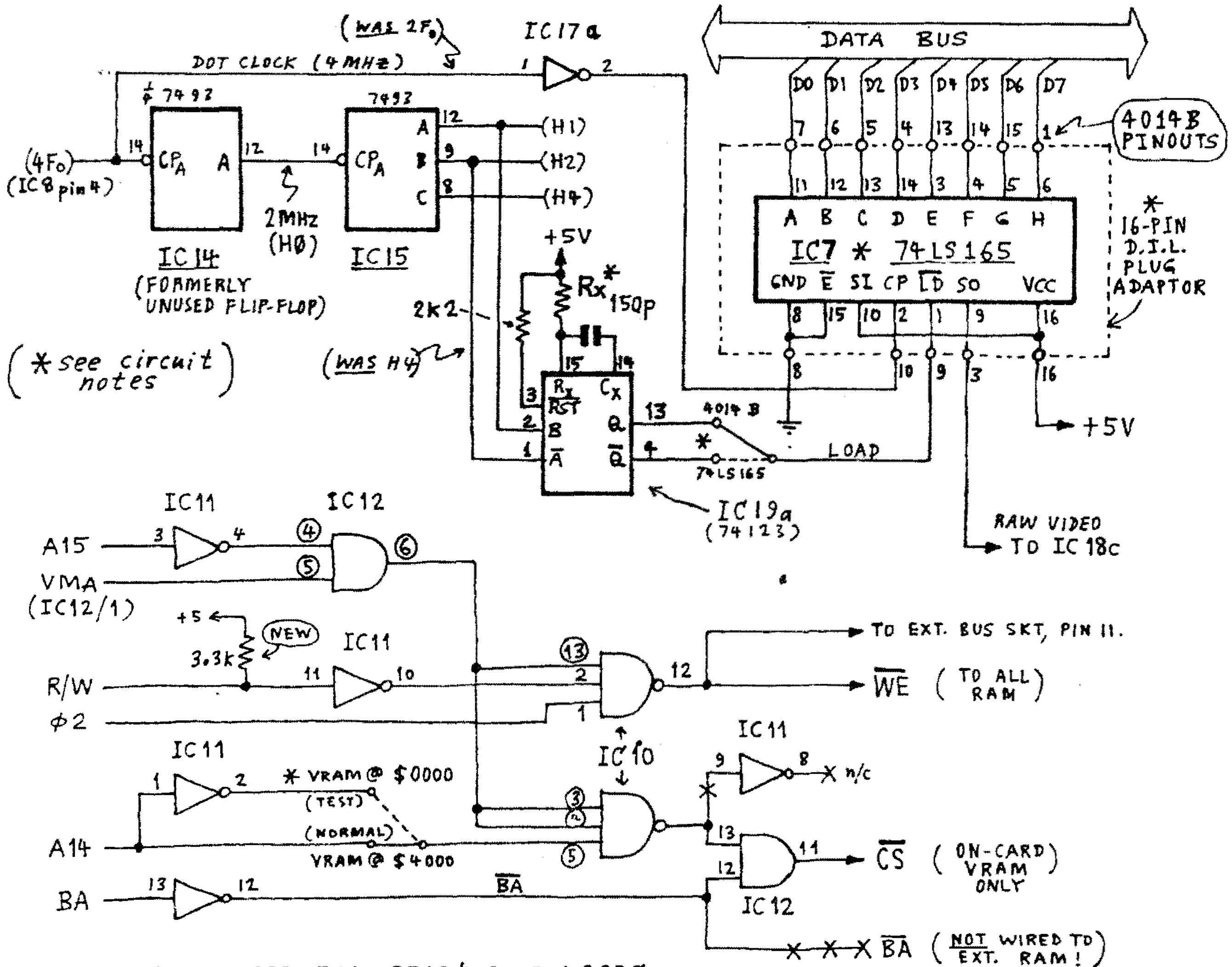
POST SCRIPT

Having performed the conversion and tested it myself (including the EPROM patches), I am now a lot more enthusiastic about it than before. (The article was scheduled for the August issue, but there were bugs in it.) It was an unexpected pleasure when the 4014 was seen to operate at 4 MHz! The 74LS165 works too, but the dot widths are not as consistent as the 4014 because the latter is synchronous loading (a very minor point indeed).

If there is sufficient demand, I could be persuaded to write a new operating-system EPROM for hi-res DREAMs, which would contain a 'software switch' to select the desired display format – the old 64 x 32, or the new 128 x 64. Thus, the modified system would be able to run CHIP-8 programs written for the standard Dream or the 'hi-res' Dream (simply by using a different command for 'GO'). However, machine-code programs which used the display would, in general, NOT be transportable from one system to the other without extensive modification. This probably won't be of major concern. The new operating-system EPROM – let's call it 'HIDIOS' (HI-res Dream Improved Operating System) – would use a 2716 (2K) and would incorporate several enhancements to the original CHIPOS.

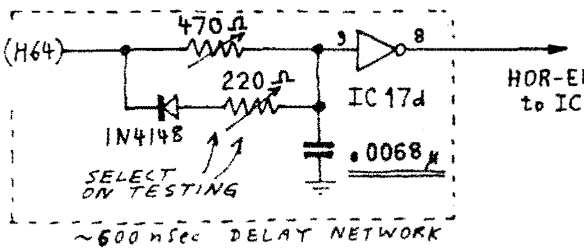
CIRCUIT MODIFICATIONS
FOR DOUBLE-RESOLUTION DISPLAY ON DREAM-6800
(124 H x 64 V DOTS)



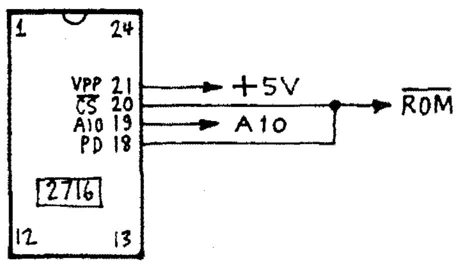


* MODIFIED RAM READ/WRITE LOGIC

(n = NEW CONNECTION)



~600 nSec DELAY NETWORK



WIRING CHANGES FOR 5V 2716 (2K) EPROM (OPTIONAL)

MJB-09/81

CIRCUIT NOTES

(1) The 'dot clock' freq. is doubled to 4.00 MHz (6875/pin 4).

(2) The video memory buffer (VRAM) is now all of the on-board 1K, which has been relocated to $4000 - $4400. (NB: The RAM space from $4400 - $8000 is not available for expansion.)

(3) At least 1K of external RAM is required at $0000 for user programs and scratch. 4K - 8K is recommended. The E/A or J-R expansion boards may be used, provided they are modified so that the '$\overline{BA}$' signal (Bus Available) does NOT select the lowest 1K block; (see note elsewhere).

(4) The original 4014 shift-register could well perform satisfactorily at 4 MHz (mine did), so try it first. The 4014 requires a 300 nSec (± 50 nSec) positive-going LOAD pulse. Use Rx = 4.7k. It is strongly recommended to wire 8 x 10k pullup resistors on the data lines (D0-D7). Also, tie 4014 pin 11 to Vcc.

(5) If your 4014 can't hack the pace, then a 74LS165 may be substituted, as shown on the circuit. The pinouts are different, so the 74LS165 must be piggy-backed onto a 16 pin DIL plug and jumper-wired using (preferably) 30 guage 'Kynar' wire. In this case, the LOAD pulse required is about 125 nSec (± 50 nSec) negative-going. Use Rx = 1k in the one-shot (IC19).

(6) The 'horizontal-enable' delay network will no doubt need to be adjusted. Use trimpots as shown. This is not a critical adjustment, but it's nice to have only the wanted part of the display showing. The ideal test pattern is a rectangular border, one dot thick, around the extremities of the display. Write a CHIP-8 program to do it.

(7) If you have already modified your Dream board (e.g. to take a RAM expansion board), then first put it back to its original form. Then rewire it as shown. Note that $\overline{WE}$ (Write Enable, to all RAM) is still available for use by external RAM. N.B: $\overline{CS}$ must now only be used to select the on-card 1K of RAM. $\overline{WE}$ should now be routed to pin 11 on the expansion bus socket (formerly $\overline{BA}$). $\overline{BA}$ is no longer required for any expansion of memory.

(8) Special note for use of J-R expansion board: (i) $\overline{CS}$ and $\overline{BA}$ signals must be disconnected; (ii) the 74LS08 pins 2,5&12 ($\overline{BA}$) are now tied high; (iii) the 74LS155 pin 2 wired to A15, and pin 14 wired to A14; (ensure pins 2 & 14 not shorted).

TRACKS TO BE CUT and LINKS TO BE REMOVED (at specified pins):-

( ) IC8 (6875 skt) pin 5; use PCB track for 4 MHz.
( ) IC19 (74123) pins 1,2 & 3.
( ) IC6 (74LS367) pin 14.
( ) IC15 (7493) pin 14.
( ) Only if using 74LS165; IC7 (4014 skt) pin 9.
( ) IC10 (74LS10) pins 3 & 13.
( ) Remove links from IC10 pins 4 & 5.
( ) Remove links from IC 5 pins 4, 6, 10 & 12.
( ) Remove links from IC 6 pins 2, 4, 6, 10 & 12.

It is recommended to make all new connections on the underside of the PCB, with 30G 'Kynar' (wire-wrap) wire, for neat and reliable results. Take care not to nick the wire when stripping insulation.

Changes to the CHIPOS EPROM (Refer to your CHIPOS manual)

The subroutine 'DISLOC' computes the 16 bit address of the byte to be altered during a display operation. (In general, two such bytes adjacent to each other will be altered during execution of a SHOW instruction, because any arbitrary pattern byte will overlap the boundary of two adjacent video RAM locations. Or, put another way, the X coordinate of a symbol to be displayed will not, usually, correspond with a V-RAM byte boundary.) DISLOC builds up the 16 bit address 'BLOC' (Buffer LOCation) given the symbol coords 'VX' and 'VY' as input parameters; (VX is passed via acc-B). The resulting BLOC is moved to the X-reg for use by subr 'SHOWUP'.

The old 64x32 display format consisted of 256 bytes of video RAM, which meant that DISLOC only had to compute the low order byte of the buffer location, BLOC, while the high order byte remained constant at 01. The new 128x64 format requires 1K of RAM. Therefore, ten bits are required to specify a given RAM location, so DISLOC must now compute the high order byte as well.

Relocated routines:-	
SHOW2	C22C
SHOWUP	C262
DISLOC	C271

CHIPOS PATCHES for 'Hi-res'

```
C07B  40 00
C081  44 00
C3B6  43 90
C3E3  3A

C220  29 7F 00 3F  DE 26 C4 0F  26 02 C6 10  37 DF 14 A6
C230  00 97 1E 7F  00 1F D6 2E  C4 07 27 09  74 00 1E 76
C240  00 1F 5A 26  F5 D6 2E 8D  28 96 1E 8D  15 D6 2E CB
C250  08 8D 1E 96  1F 8D 0B 7C  00 2F DE 14  08 33 5A 26
C260  CB 39 16 E8  00 AA 00 E7  00 11 27 04  86 01 97 3F
C270  39 96 2F 84  3F 58 44 56  44 56 44 56  44 56 8A 40
C280  97 1C D7 1D  DE 1C 39
```

TESTING

Just in case there is something wrong with the way you have your expansion RAM board wired, I recommend that you first test your 'hi-res' system without it. This is possible by temporarily locating the 1K Video RAM at $0000. Simply jumper the A14 line to IC10/pin5 (instead of A14; see circuit). Astute readers will realize that the system's scratch area and stack ($0000 - $0080) will actually be visible at the top of the display window, so you will be able to observe the bit patterns changing when you use 'memod'. Notice the 2 bytes toward the top left of the display counting in binary? These are the relative-time-clock (RTC) counters ($0020,21). Of course with the old CHIPOS EPROM, nothing sensible will be displayed, but you can still use memod 'blindly' to write stuff into the display buffer (now at $0080 - $0400); try it. If all goes well, jumper the VRAM back to $4000, plug in your expander board and use memod again to check writing data into the video buffer (at $4000). If that works, plug in a re-programmed EPROM, and your new system is ready for use. The 128 x 64 graphics format is quite respectable and should inspire lots of practical applications. □

HI-RES DREAM-6800 SURVEY COUPON (No obligation)

□ I would be interested in purchasing a 'HIDIOS' EPROM, should it become available (at say $30 incl. documentation), since I am intending to (or already have) convert my Dream-6800 to 'hi-res'.

(Mail to: M.J. Bauer, [redacted] Vic [redacted]

DREAM PONTOON REVIEW.

The 'Dreamcards' advertisement claims that their new game, "Dream Pontoon", is the biggest and most intelligent program available in the Chip-8 language - and they're not kidding!! Not only does it turn the DREAM into a highly skilled card player, it takes the game much further than many of the Basic language versions being sold, by allowing the computer to play either as the 'banker' or 'challenger'.

If you are not too sure how Pontoon is played (and even if you have never been that keen on card games) you will still find 'Dream Pontoon' well worth getting just to see how much in the way of brains can be crammed into 4K of Chip-8.

The 'Dream Pontoon' package contains a cassette, an instruction booklet (which is of good quality and easy to follow) and a leaflet describing some alterations that can be made to the program to allow it to cater for any taste. The optional commented listing (all 27 pages of it) is well worth getting too, as it gives the best breakdown of Chip-8 logic routines that we have seen.

The game starts off with a fully automatic checksum that makes sure that it has loaded O.K. and then prompts you to set the level of skill with which the DREAM is to play. It also 'scrambles' the random number generator to ensures that your DREAM can not cheat.

Each player then gets a 'kitty' of $1,000 and either you or the computer is elected 'Banker'. One card is dealt to each player from the memory-mapped deck and the idea is for the challenger to buy more cards from the banker to reach a total as close as possible to 21. When the challenger has finished buying, the banker begins taking cards until he (it?) feels close enough to 21 to compare hands. If either player goes over the total then he loses his money, otherwise may the best hand win.

The game progresses like this until one player bankrupts the other, but there are a lot of surprises on the way. For instance, special combinations of cards exist that increase the payout to the winner or result in the 'Bank' changing hands. All these special combinations come from the accepted Pontoon rules and they have been fully implemented in the 'Dreamcards' version.

You will discover that the most demanding task in the game is to try to work out what to pay for each card, as there are about 40,000 possible combinations. On it's 'high skill' setting the DREAM is lethal at figuring out these bets, but the 'low skill' setting and the various "de-tuning" routines suggested in the program notes allow even beginners to play against it with some success.

In a review like this we could not possibly cover all of the things that go to make up the game, but two other items could not pass without comment. We were rather surprised when on one occasion in the heat of the moment the DREAM offered to sell the bank for $140, and later when it became abusive after a bad bit of play!! You would expect this in a game against another human, but from a computer......?

So, have another look at the DREAMCARDS advertisement in this issue. The game is tremendous value for money, and is the most complex and absorbing game that we have seen written for the DREAM to date, and you really need to keep your wits about you to win against the computer on the 'high skill' levels, (although Garry's children still prefer 'Dream Invaders') but be warned, if you buy it you could easily become addicted, lock yourself away with your DREAM and never be seen again!!

****

HEXADECIMAL TO BINARY CONVERSION AND THREE LOGICAL FUNCTIONS.

J. MARCHINGTON,
[redacted]

Here is a program designed to assist those who find the conversion of Hexadecimal digits to Binary numbers something of a problem.

In this particular program, two pairs of hexadecimal digits are keyed in, one pair at a time. After keying in the first pair, which will appear in the central top section of the display area, the equivalent binary byte (noughts and ones) will appear to the right. The second pair of hex digits may be keyed in at this stage and will appear below the first pair. As before, their binary equivalent will appear to the right, below the first byte.

After a short delay, a block displaying the word OR will appear, followed by the logical ORed value of the two pairs of hex digits. The corresponding eight bits will be displayed to the right. The ANDed and EXclusive ORed values of the two hex pairs are computed on the next two lines, together with their respective binary numbers.

To produce a new set of equivalents, it is only necessary to press the first hex digit key. This will clear the screen of the data from the previous exercise.

As the program is entirely in Chip-8, it should be run in the usual way. (I.E., C000, FN, 3.).

```
0200    6A15 6B00 F10A 00E0   8510 F129 DAB5 7A04
0210    F20A 8620 F229 DAB5   22DE 22EE 7A07 22F4
0220    6A15 7B06 F30A 8530   F329 DAB5 7A04 F40A
0230    8640 F429 DAB5 22DE   22EE 7A07 22F4 22DE
0240    22EE 6A00 7B07 6007   A342 2330 22DA 22EA
0250    7A0D 7B01 8C10 8C31   85C0 233A 8C20 8C41
0260    86C0 FC29 DAB5 22DE   22EE 7A07 22F4 22DE
0270    22EE 6A00 7B05 A350   2330 22DA 22EA 7A0D
0280    7B01 8C10 8C32 85C0   233A 8C20 8C42 86C0
0290    FC29 DAB5 22DE 22EE   7A07 22F4 22DE 22EE
02A0    6A00 7B05 A35E 2330   2332 22DA 22EA 7A05
02B0    7B01 8C10 8C31 8D10   8D32 8CD5 85C0 233A
02C0    8C20 8C41 8D20 8D42   8CD5 86C0 FC29 DAB5
02D0    22DE 22EE 7A07 22F4   1200 6E0C 12E0 6E26
02E0    FE15 FE07 3E00 12E2   00EE 6E0C 12F0 6E26
02F0    FE18 00EE 670F 8752   6808 231C 6707 8752

0300    6804 231C 6703 8752   6802 231C 6701 8752
0310    6801 231C 8560 4A30   12F4 00EE 8785 4F01
0320    1326 6900 1328 6901   F929 DAB5 7A04 00EE
0330    DAB7 7A08 F01E DAB7   00EE FC29 DAB5 7A04
0340    00EE FF88 AAA8 A98A   FF80 8080 8080 8080
0350    0088 AA8A AAAA FF78   98A8 A8A8 98F8 008A
0360    BA8D BA8A FF07 E2EA   EAEA A2FF E020 A020
0370    60A0 E000
```

****

NOUGHTS AND CROSSES. (0090 - 0400)

M. K. DOWSON.

A game for 2 players, (0 & X) only. The computer draws a grid, and the players take alternate turns, using keys 0,1,2, 4,5,6, 8,9,A, in the usual fashion. All games automatically restart, with alternate player to last winner going first. The computer keeps cumulative game scores which can be reset to 0 by a restart from C000.

```
0090    6432 6502 D453 7508   A0E6 D450 20A2 20B2
00A0    00EE 664F F618 F615   F607 3600 10A8 00E0
00B0    00EE 6416 6506 A3BA   D453 740A A089 20D2
00C0    6416 6516 A3BD D453   740A A08C 20D2 20A2
00D0    00EE F265 F029 D455   7404 F129 D455 7404
00E0    F229 D455 00EE FF11   1171 5171 2175 A9A1
00F0    7151 5151 D901


0200    A089 6107 23E8 6C00   6B00 6D00 2228 6118
0210    222A 6071 223C 6102   620B 222C 6102 6216
0220    222C 6072 223C 1242   610D 6200 6321 A269
0230    D121 7201 73FF 3300   1230 00EE A232 F055
0240    00EE A281 6000 2256   A28D 6004 2256 A299
0250    6008 2256 1268 6203   F055 6103 F11E 72FF
0260    7001 3200 1258 00EE   A080 6109 23E8 6001
0270    70FF A26F 3001 7002   F055 8E00 FA0A 6021
0280    4A21 22AA 4A21 22BC   4A02 22CE 4A04 22E0
0290    4A21 22F2 4A21 2304   4A08 2316 4A09 2328
02A0    4A21 233A 4D09 120A   126E A281 F055 6406
02B0    651A 234C A080 F055   235E 00EE A285 F055
02C0    6412 651A 234C A081   F055 235E 00EE A289
02D0    F055 641E 651A 234C   A082 F055 235E 00EE
02E0    A28D F055 6406 650F   234C A083 F055 235E
02F0    00EE A291 F055 6412   650F 234C A084 F055

0300    235E 00EE A295 F055   641E 650F 234C A085
0310    F055 235E 00EE A299   F055 6406 6504 234C
0320    A086 F055 235E 00EE   A29D F055 6412 6504
0330    234C A087 F055 235E   00EE A2A1 F055 641E
0340    6504 234C A088 F055   235E 00EE 3E01 1358
0350    6001 A3BA D453 00EE   6006 A3BD 1354 7D01
0360    23C0 8014 8024 23B0   23C0 8034 8064 23B0
0370    23C0 8044 8084 23B0   23C0 8344 8354 8030
0380    23B0 23C0 8674 8684   8060 23B0 23C0 8144
0390    8174 8010 23B0 23C0   8254 8284 8020 23B0
03A0    23C0 8244 8264 8020   23B0 4D09 23C6 00EE
03B0    4003 23CC 4012 23DA   00EE E0A0 E0A0 40A0
03C0    A080 F865 00EE 00E0   20B2 00EE 7C01 A089
03D0    FC33 A3BD 2090 6D09   00EE 7B01 A08C FB33
03E0    A3BA 2090 6D09 00EE   6000 F055 F01E 71FF
03F0    3100 13E8 00EE
```

****

MULTI-PILE NIM (0080 - 0400)

J. FISHER.
[redacted]

This is an entirely different form of 'NIM' to that described in the January 1981 issue of 'Dreamer'. It offers a greater degree of skill and it is not player versus player, but player versus computer.

In this version of NIM there are up to 15 piles, of up to 15 objects, and you are only allowed to take the objects from one pile in a single turn. However, you may take as many objects as you wish from that pile. There are enough messages put on the screen to make this game almost self-explanetory.

POINTS TO NOTE: 1. The computer will always take first turn, so if you want first turn, put the original number of objects in each pile as they would be after your first turn.

2. The board is displayed after the computer's turn with a white band in empty piles. The computer is then waiting for your turn.

3. When the game is over, press any key except F to restart.

4. The program uses 0080 - 008F as workspace.

```
0090    0C18 1710 1B0A 1D1E   150A 1D12 1817 1C2D
00A0    8011 0A2F 110A 2912   2F20 1217 800C 110E
00B0    0A1D 2D80 1118 202F   160A 1722 2F19 1215
00C0    0E1C 2E80 1718 2712   172F 1912 150E 8019
00D0    1215 0E2E 801D 0A14   0E2E 8000


0200    6B00 6C00 A0B4 2364   FE0A 4E00 1208 00E0
0210    A0C4 2364 6D00 7D01   FD29 6B00 6C2C DCB5
0220    F00A DCB5 A07F FD1E   F055 5DE0 1216 00E0
0230    6A00 6D00 7D01 A07F   FD1E F065 234A 8A80
0240    5DE0 1234 3A00 1266   CD0F 88D0 88E5 3F00
0250    1248 A080 FD1E F065   4000 1248 70FF A080
0260    FD1E F055 12AC 6901   8890 8894 78FF 88A5
0270    3F00 127A 8890 8984   1268 6D00 A07F 8890
0280    7D01 FD1E F065 8802   4800 127C 234A 8080
0290    A07F FD1E F055 6D00   7D01 A07F FD1E F065
02A0    3000 12AC 5DE0 1298   A0A1 1338 6B00 6D00
02B0    6C00 7D01 A07F FD1E   F065 3000 12CA A351
02C0    DCB1 7C08 3C20 12C0   12D8 A253 6800 7801
02D0    DCB1 7C02 5800 12CE   7B02 5DE0 12B0 6B00
02E0    6C24 A0CF 2364 FD0A   4D00 1358 88E0 88D5
02F0    3F01 1358 A07F FD1E   F065 4000 1358 6C3C

0300    FD29 DCB5 8800 6B08   6C24 A0D5 2364 F90A
0310    4900 1358 8895 4F00   1358 8080 A07F FD1E
0320    F055 00E0 6D00 7D01   A07F FD1E F065 3000
0330    1230 5DE0 1326 A090   00E0 6B00 6C00 2364
0340    F00A 400F F000 00E0   1200 8900 89A1 80A2
0350    68FF 8805 8892 00EE   00E0 6B10 6C24 A0AD
0360    2364 12AC 0388 038F   4080 00EE 4016 137C
0370    4020 1380 039B DCB5   7C04 1366 A3AE 1382
0380    A3AF DCB5 7C06 1366   FE00 26FF 03AC 39FE
0390    03AC A600 9730 08FF   03AC 3996 3081 0F22
03A0    037E C193 8010 CE03   B27E C198 00AD F8A8
03B0    A8A8 A850 F6CE B7DA   E92E F492 B75A F248
03C0    B7FA B6DE F6DE 93DE   5EDE BBDE C546 492E
03D0    F6DA 56DA BFDA B55A   4BDA F11E 0024 2A22
03E0    88A8 8000 0BA0 0380   1550 1110 0820 4124
03F0    419E
```

****

BIG SQUARE PUZZLE

TONY HORNCASTLE,
[redacted]

The object is to move the big square, (block 2) out of the 'gate' (the dotted line on the right), in as few moves as possible.

To enter a move, key in the block number followed by the direction. The key functions, and the memory locations they are stored in, are,
5 - Down, (035D), 8 - Left, (0365), A - Right, (0359), D - Down, (0361).

Any invalid move, (outside the bounds or another piece in the way), is disallowed and must be re-entered.

```
0200    1340 88D0 6B07 6906   8895 3F01 6B0F 00EE
0210    6C07 4D02 6C0F 4D06   6C0F 00EE A33E 8510
0220    6900 8400 6800 D451   7401 7801 58B0 1226
0230    7501 7901 59C0 1222   00EE 8400 7402 3B07
0240    7404 8510 7501 3C07   7504 FD29 D455 00EE
0250    A3E0 6000 6100 6200   6308 6400 6518 6610
0260    6700 6810 6918 F955   A3EA 6010 6108 6218
0270    6308 6418 6510 6620   6700 6820 6918 F955
0280    6D01 22AA 221C 223A   7D01 3D0B 1282 A33F
0290    6428 6500 6617 D451   D461 7501 7601 3508
02A0    1296 22E4 6E00 22B8   00EE A3DE FD1E FD1E
02B0    F165 2202 2210 00EE   6431 6519 A3FD FE33
02C0    F265 F029 D455 7404   F129 D455 7404 F229
02D0    D455 00EE A33E D451   89F0 D451 00EE 6940
02E0    F918 00EE A33E 6428   6509 D451 7502 3517
02F0    12EA 00EE 223A 221C   8020 8130 221C 223A

0300    A3DE FD1E FD1E F155   4700 1310 7701 00EE
0310    3D02 131C 3018 131C   4108 6701 22B8 7E01
0320    22B8 00EE 4700 1344   4704 133C 4701 22E4
0330    22DE 22AA 8200 7208   8310 13DC 132A 80E0
0340    2250 6700 FD0A 4D00   13D8 8AD0 690B 8A95
0350    3F00 13D8 22AA FA0A   4A0A 138C 4A05 13A0
0360    4A0D 13C4 3A08 13C4   4000 13D8 8200 72F8
0370    8400 74FE 8310 8510   22D4 3900 13D8 4C07
0380    13DC 750E 22D4 3900   13D8 13DC 8200 82B4
0390    4227 13D8 8200 7208   8400 84B4 7401 1374
03A0    4100 13D8 8310 73F8   8510 75FE 8200 8400
03B0    22D4 3900 13D8 4B07   13DC 740E 22D4 3900
03C0    13D8 13DC 8310 83C4   431F 13D8 8310 7308
03D0    8510 85C4 7501 13AC   22DE 1344 22F4 1324
03E0    0000 0008 0018 1800   1010 1000 1808 1818
03F0    2008
```

****

ASTEROID BELT. (0200 - 0300)

J. FISHER,
[redacted]

Steer a 2 by 2 square, using keys 5, 8, A, D, from the top left corner to the bottom right corner of the screen with as few crashes into the asteroids or the border as possible. The only problem is that once started, you can not stop. The speed of your spaceship may be set after the border is displayed by keying 0 (Fast) to F (Slow). The position of the asteroids may be such that it is impossible to get through. If this happens, just reset the game. Congratulations to anybody whose reactions are fast enough to play this game on speed 0. I have seen it done once, but after only 200 attempts! At the other end of the scale, I have only seen it done once on F, because it is so slow that people die of boredom on the way! Recommended speed for beginners is about 8. When you have successfully completed the course, the number of crashes is displayed. Any key except 'F' will restart the game.

```
0200   6919 6305 6408 650A  660D A2EF 6D00 22D4
0210   6D1F 22D4 A2EE 6C00  22E0 6C3F 22E0 6700
0220   6800 FE0A A2F0 FE1E  F065 8E00 A2EC 6C01
0230   6D01 DCD2 6C3D 6D1D  DCD2 A2EE CC3F CD1F
0240   DCD1 3F01 124A DCD1  123C 7801 3864 123C
0250   A2EC 6C3D 6D1D DCD2  6C01 6D01 6A00 6B00
0260   FE15 E39E 126A 6A00  6BFF E49E 1272 6AFF
0270   6B00 E59E 127A 6A01  6B00 E69E 1282 6A00
0280   6B01 DCD2 8CA4 8DB4  DCD2 4F01 129E 3C3D
0290   1296 4D1D 12AC F807  3800 1296 1260 F918
02A0   DCD2 7701 6C01 6D01  DCD2 125C F918 00E0
02B0   6C18 6D0D A300 F733  F265 F029 DCD5 7C05
02C0   F129 DCD5 7C05 F229  DCD5 F80A 480F F000
02D0   00E0 120A 6C00 DCD1  7C08 3C40 12D6 00EE
02E0   6D01 DCD1 7D01 3D1F  12E2 00EE C0C0 80FF
02F0   0203 0405 0608 0A0D  1014 1920 2832 4050
```

****

ADVERTISING

If you would like some help, can offer some help, have something to sell, or would like to buy something, send it in to us with a fee of $1-00, and we will print it in two newsletters. THIS OFFER ONLY APPLIES TO PRIVATE ADVERTISERS and we would ask you to keep them reasonably short, something like the ones below. Commercial enterprises who wish to advertise in the DREAMER are invited to contact us for details of rates, etc.

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

****

TOUCH DOWN

(0080 - 0400)

D. TOON.

Your Star Fighter has lost all power to the engines. You must pilot it back to the 'Mother-Ship' using only its retro rockets, but your on-board navigation computer is damaged, and automatically stops you each time you release the thrust button. It will also not allow you to continue in the same direction, or retrace the last move, due to retro overload.

Use either the directional joystick, or keys
~~5~~ 9 - Up, (0255), ~~8~~ 4 - Left, (0235), ~~A~~ 6 - Right, (023F), or ~~D~~ 1 - Down, (025F).

```
0080   E028 4410 3800 D372   73FE D372 6C00 123E
0090   D372 7302 D372 6D00   1248 D372 77FF D372
00A0   6C00 125E D372 7701   D372 6D00 1268 0000
00B0   F715 F707 3700 10B2   00EE 6B00 20D0 6B1F
00C0   20D0 6A00 20E2 6A3F   20E2 1226 0000.0000
00D0   6A00 A09D 6C00 DAB1   7A08 7C01 3C08 10D6
00E0   00EE 6B01 A207 6C00   DAB1 7B01 7C01 3C1E
00F0   10E8 00EE 00E0 6002   03B6 6764 20B0 12C8

0200   12D6 6A00 6B05 A080   6D00 CC3F 8AC4 DAB1
0210   7D01 3D05 120A 7B04   3B1D 1208 6A1C 6B00
0220   A081 DAB2 10BA 631C   671D A083 D372 6F00
0230   6C80 6D80 6E08 EEA1   1086 4C00 124E 6E0A
0240   EEA1 1090 4D00 124E   3F00 1270 12E2 0000
0250   6C80 6D80 6E05 EEA1   109A 4C00 122E 6E0D
0260   EEA1 10A4 4D00 122E   3F00 1270 12EA 0000
0270   3700 128C 331C 128C   22AC 6800 6310 6710
0280   F318 20B0 7801 3810   127C 10F4 22C0 6800
0290   6708 6308 F318 0000   02A2 7801 3810 1290
02A0   12B4 C608 D721 C640   BDC2 E539 00E0 6001
02B0   03B6 00EE 00E0 6000   03B6 6764 20B0 12CA
02C0   00E0 6003 03B6 00EE   02CE 00E0 1202 7C00
02D0   937A 0089 3900 02DA   12C8 7F00 937F 0089
02E0   3900 6E03 EEA1 12F2   1234 6E03 EEA1 12F2
02F0   1254 02A2 6608 F618   02A2 F618 02A2 00E0

0300   6003 03B6 02A2 F618   02A2 F618 02A2 1200
0310   0757 D91C 1DDD C920   0455 5510 1154 8920
0320   0655 5D18 1554 8920   0455 5510 1154 8000
0330   0475 59DC 11DC 8920   EEEE EEEA 8EEE EE64
0340   8AA8 AA4A 8A44 AA84   8AAA EE4A 8E44 AA44
0350   8AAA CA4A 8A44 AA20   EEAE AA4E EA4E EAC4
0360   3ABB B877 7755 DD45   22AA 2845 5255 5545
0370   12BB 3865 5255 5D85   0AA2 3045 5255 5940
0380   3BA3 B847 7229 D545   AEAE 1DDD D5DC 3B90
0390   AAA8 1554 9514 2290   EEAE 1D54 9DDC 2A90
03A0   AAA8 1554 9518 2A80   AA4E 155C 95D4 3B90
03B0   0101 0101 0101 CE03   1096 3001 270B C628
03C0   085A 2702 20FA 4A20   F2FF 03F8 C628 085A
03D0   26FC FF03 FAFE 03F8   FF03 FCCE 0160 FF03
03E0   FEFE 03FC A600 BC03   FA27 0C08 FF03 FCFE
03F0   03FE A700 0820 E739   0360 0388 0388 0188
```

****

EDUCATIONAL MULTIPLICATION. (0200 - 0400)

PAUL FLYMEN,
[redacted]

This program allows an operator to try to multiply two randomly generated numbers displayed on the screen. The computer multiplies them and stores the answer. The player keys in what they think the correct answer is. If the answer is correct, the computer acknowledges it with a tick, and displays it. If it is incorrect, then a cross flashes to signify 'try again'.

The answer never exceeds 256 in order to simplify the problem for youngsters.

```
0200    2300 231A 6A18 6B00  230C 8430 8540 2300
0210    4300 1200 6A18 6B06  230C 73FF 4300 122A
0220    6F00 8544 3F00 12AE  121A 8350 6A14 6B10
0230    A080 F333 F265 6C30  4000 1250 F80A 5800
0240    126C F80A 5810 126C  F80A 5820 126C 1256
0250    4100 1248 1242 6930  F918 6A2A 6B10 A33A
0260    DAB5 FC15 FC07 3C00  1264 12BC 6A2A 6B10
0270    6D05 A334 22C2 0000  FC07 3C00 1278 22C2
0280    0000 FC07 3C00 1282  7DFF 12B2 F029 4000
0290    1296 DAB5 12A8 7A04  F129 4100 12A0 DAB5
02A0    7A04 F229 DAB5 12CA  7A04 F129 129E 00E0
02B0    1200 4D00 1238 6E10  0340 1272 6A14 6B10
02C0    128C DAB5 6C10 FC15  00EE 6CFF FC15 FC07
02D0    3C00 12CE 12AE FFFF  FEFC FDE7 DFBC ACE6
02E0    7F7E 7F22 7F5B 7F2F  7D26 5E0D 1A07 0FD8
02F0    FEEC FDAC EE8F BC8E  FEFC FFC5 FFFE F7AB

0300    6300 C33F A080 F333  F265 00EE F129 3100
0310    DAB5 7A04 F229 DAB5  00EE 6B0E 6A14 A332
0320    DAB1 7A01 3A20 131E  6B06 6A10 A334 DAB5
0330    00EE 8000 8850 2050  8800 0408 10A0 4000
0340    F600 3ED7 21C6 407E  C2E5
```

****

IDEAS

EXTENSION TO A LONG ONE A. GROVES.

Another way to provide better access to the EXPANSION BUS and I/O SOCKETS is to make an external socket board on a piece of veroboard, with 16 pin sockets.

Make up jumper cables out of ribbon cable with crimp type DIP headers on one end, the other end of the cable is soldered directly to the tracks on the rear of the veroboard for each respective socket.

A rectangular section 8 - 10 cm. long can be cut out of the side of the case adjacent to the I/O socket to bring out all connections. The extension board can be attached to the base of the DREAM by two right angle brackets and mounted outside the case.

Make the veroboard wider than the cutout section in the case, this will provide support when plugging in external devices.

****

BLOCK SHIFT AND CHIP-8 PROGRAM ADJUSTER

(0100 - 0200)

J. FISHER,
[redacted]

This program is not as convenient to use as the EDITOR in the January 1981 issue of 'Dreamer', as all it can do is shift blocks of data around. However, it has the big advantage that it will adjust the addresses in the Chip-8 instructions, 0MMM, 1MMM, 2MMM, AMMM, and BMMM, accordingly. The program and its scratchpads also do not use RAM that would normally be required for the programs you want to adjust. Instead, it uses RAM between 0000 and 0080 for the scratchpads, and the display buffer for the program. The program can be relocated without change if you have RAM to spare.

Using MEMOD, enter the start address and the end address of the block of data to be moved at 0002 and 0004 the way you would for putting programs on cassette. Next enter the starting address of the place you want the data to be moved to at 0030. Then enter starting and ending addresses of up to 8 sections of RAM which ARE IN CHIP-8 and need their addresses adjusted. Enter them in groups of 4 bytes starting at 0040, i.e., First start address at 0040, First end address at 0042, second start address at 0044, second end address at 0046 and so on, up to eighth start address at 005C and eighth end address at 005E. IF you do not want to alter the maximum 8 blocks of Chip-8 instructions, then enter FFFF as the start address of the first block of RAM not required. Lastly, RUN,the program from 0100. I.E., RST, 0100, FN, 3.

To show you that the program has finished its run, (it may take a few seconds) the 4 digit address of the Chipos monitor will be changed to FFFF.

```
0800    96 31 90 03 97 33 96 30   92 02 97 32 CE 00 40 DF
0810    34 08 08 EE 00 DF 38 DE   34 EE 00 DF 36 8C FF FF
0820    27 54 E6 00 C4 F0 C5 E0   27 0C C1 A0 27 08 C1 20
0830    27 04 C1 B0 26 2D A6 00   84 0F A7 00 91 02 05 20
0840    26 08 A6 01 91 03 25 7C   A6 00 91 04 22 12 26 06
0850    A6 01 91 05 24 6E A6 01   9B 33 A7 01 A6 00 99 32
0860    1B A7 00 08 08 DF 36 9C   38 26 B7 DE 34 08 08 08
0870    08 8C 00 60 26 99 96 32   2A 1E DE 02 DF 34 DE 30
0880    DF 36 DE 34 A6 00 08 DF   34 DE 36 A7 00 08 DF 36
0890    DE 34 9C 04 26 EE 20 24   96 05 9B 33 97 37 96 04
08A0    99 32 97 36 DE 04 DF 34   DE 34 09 A6 00 DF 34 DE
08B0    36 09 A7 00 DF 36 DE 34   9C 02 26 EC CE FF FF DF
08C0    06 7E CC 00 A6 00 20 98
```

NOTE: This program should prove useful for those who do not have any memory expansion. You will note that the program listing produced on the printer shows the program located at 0800. This is because the program which controls our printer uses the display area of RAM for messages etc, which would corrupt the 'Block Shift' program. Because it is fully relocatable however, it will work equally well from either location. (Or any other you may wish to use.) Don't forget that if you locate it anywhere other than 0100, you must run it from that location, not 0100.

Graeme & Garry.

****

HOW TO SUBMIT PROGRAMS

To remain in operation, we need a constant supply of new programs, and articles about the DREAM 6800. If you can write an article on modifications you have made to your DREAM, or the use you are making of it, or if you have written any games, or utility programs, we invite you to submit them to us for consideration. ALL CONTRIBUTORS OF PROGRAMS PRINTED WILL RECEIVE VOUCHERS FOR TWO FREE NEWSLETTERS. CONTRIBUTORS OF ARTICLES AND IDEAS PRINTED WILL RECEIVE FROM ONE TO THREE VOUCHERS, BASED ON THE GENERAL INTEREST CONTENT OF THE ARTICLE, AND THE AMOUNT OF WORK THAT HAS GONE INTO IT. Along with the listing for all programs submitted, we will need a tape recording, with at least twenty seconds of High and Low "leader" on it. We need a leader to align our tape heads, and tune the DREAM input port. To do this you first must record 20 Sec High tone, then 20 Sec Low tone. The High tone is normal leader, and can be recorded normally. To get the Low tone, load in the following Machine Code program.

```
0200    8640  Accumulator A = 40
0202    B78012  Store in PIA output port.
0205    20FE  Branch back 2 bytes from 0207
0207    0000
```

This will produce a continuous Low tone when run 0200, FN, 3. After 20 seconds press RESET to return to normal. Then load your program. We need the electronic copy so we can test the program and verify the listing BEFORE printing, to eliminate program errors and increase the enjoyment of other users.

We will not be able to enter into correspondence, but will print corrections or improvements where necessary. We will not be selling tapes.

Programs submitted for consideration should be typed, for clarity, and set out in the following format:-

1) Program name and memory location.

2) Your name and address. (If you do not wish to receive any correspondence from other users, omit your address.)

3) The program explanation. (Don't forget key functions)

4) The program listing, typed single space. (If in doubt, have a look at the way the programs in this issue have been typed, and copy the format)

Following the guidelines set out above lets us check out the programs submitted quickly and easily. If you do not have access to a typewriter, we will accept a handwritten listing, providing it is LEGIBLE, and accompanied by a tape. However, if we cannot read your writing, and the tape will not load, or has 'bugs' in it, there will be no way we can check the program, and it will not be considered.

That's all there is to it, so send us in your favourites, and don't forget, for each one we use, you get vouchers for two newsletters free of charge. Should you be a prolific programmer, and accumulate some surplus vouchers, or have already paid a subscription to the newsletter, we will redeem the vouchers at a rate of six vouchers for $15-00.

****

PRICE STRUCTURE

The cost of this newsletter is $3-50 per issue. An advance subscription is available at reduced cost. Please write for details of cost and length of time remaining in current subscription period.

BACK ISSUES. Copies of all newsletters from No. 11, JULY 1981, are available at a cost of $4-00 each. Supplies of No.s 1 to 10, inclusive, are exhausted, but we are able to supply photo-copies of these at a cost of $6-00 each, posted. Please allow extra delivery time for back orders of No.s 1 to 10, due to the time required to have photo-copies made. Post all orders to; N.S.W. 6800 USERS GROUP, [redacted]

(Please add -10c to all CHEQUES sent from outside N.S.W., to cover Stamp Duty charged by N.S.W. Government. This is only required on cheques and does not apply to Money Orders etc.)

****