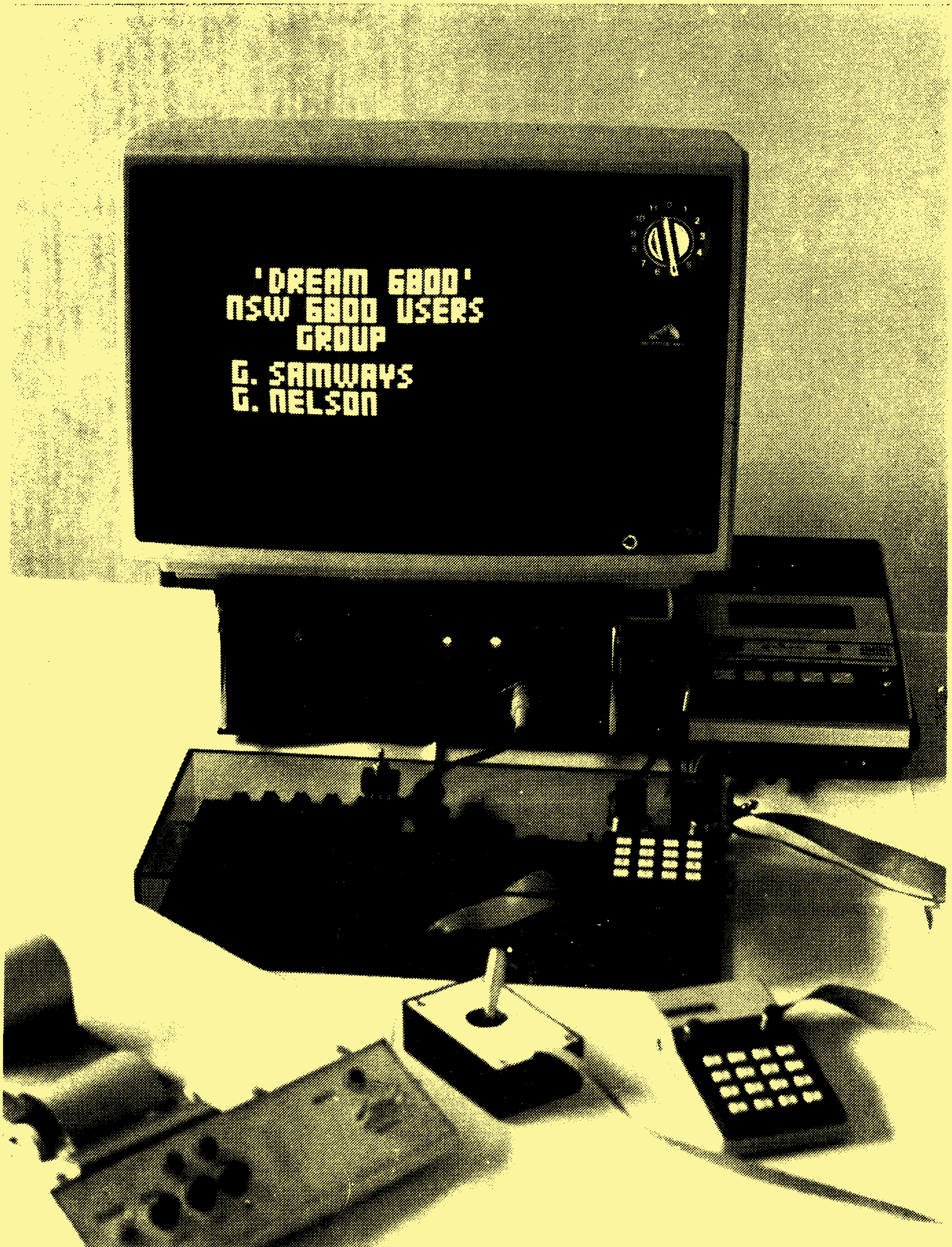
# NEWSLETTER No 2
# OCT '80



N.S.W. 6800 USERS GROUP

NEWSLETTER No.2 OCTOBER,1980

Since publication of our first newsletter, we have had quite a lot of feedback on how to improve our service. Lots of people have asked questions, so we will try to answer them all in the newsletter as it is impossible for us to reply to each of you personally.

Firstly, for those of you who thought that the cost was too high, GOOD NEWS. The response was much better than we had anticipated, so we were able to reduce our costs by having a larger number printed, so, COMMENCING NEXT ISSUE, THE PRICE WILL BE REDUCED TO $3-00, and if the demand keeps up, we may be able to reduce it even further in another couple of months.

Secondly, many of you asked who and what we were. The "group" consists of two people. Graeme Samways, an Electrical Degree Trainee, and Garry Nelson, a Timekeeper at a local colliery. Neither of us has a great deal of time to spare, and we are not out to make a fortune, just cover our costs. This is why we have asked for your help in some areas, such as filling out the address labels each month, sending us a tape if you submit a program, to save us punching it in by hand. Also, if you type the program in the manner set out in "How to Submit Programs", we can print from it, and Garry doesn't have to retype it.

The way the group was started is as follows. Graeme had written some programs, and submitted a dozen or so to "Electronics Australia" for consideration. When he didn't hear anything from them, he made further enquiries, and discovered that E.A. were scrutinising the programs and returning them as they didn't have the time or space to print them. This is not their fault of course, as they have to diversify their interests, and none of us would like to see E.A. devoted to one subject. (Unless it was the DREAM, of course.)

Graeme was not very happy with this and so, when Leo Simpson of E.A. suggested that he start a user group, as no one else seemed to have the time, we sat down to devise a streamlined method of distributing DREAM programs, and decided to "give it a go", using Graeme's programs and ideas to start it off.

The group activity consists of taking monthly subscriptions, setting out the newsletter, and printing and issueing it each month. Graeme does all the technical stuff, (Electronics, Programming etc) and Garry does the setting out, typeing, and writing stuff like this. Because of this, we are sure that you will understand that we cannot answer all your queries personally, as we just haven't got the time. If you need some help, we suggest you place an ad in the newsletter, (see "Advertising" further on), for someone close to you to help you. If you do require a personal reply from us, we will enclose it in your newsletter for the month, but please, don't be upset if it takes us a few weeks to get a reply to you. So now you know who to blame for the mistakes, if a circuit or program doesn't work, it is Graeme's fault, if something is typed or spelt incorrectly, it is Garry's fault. Also, all articles designed for beginners are tried out on Garry first, to see if they are simple enough for him to understand, as his only prior electronic experience prior to building his DREAM was self taught, and consisted of building a few projects up from kits, such as the "Basic Electronics" series from E.T.I., and progressing to such things as a Transistor Assisted Ignition for the car. (Most of them worked eventually, too.)

On with the questions. Some people have asked for information on how to use the system, and even basic electronics, while others want us to convert the DREAM to a "BASIC" system with 64K RAM, Dual Floppy Discs, etc. We do not have room to include the very basic electronics and logic, so we suggest if you need this type of help, obtain either "Digital Electronics" by B.Wand, (from Dick Smith) or "An Introduction to Digital Electronics" from E.A., or a similar reference. We will however be running a series of articles on "How to Use CHIPOS", the first one is in this issue.

The Memory Expansion in last months issue - some people felt we did not give enough information on how to build it. We did say, however, that as well as articles for beginners, we would be putting in some more technical articles for those who could handle them. We did feel that anyone who had built a DREAM would be able to follow the circuit given, but if you can't, we also mentioned that J.R.Components would be bringing out a kit for a memory expansion board, which will have full construction details etc. (We believe that this will have 8K RAM, an extra 1K EPROM, extra PIAs and full buffering. Watch for their ad in E.A.)

Program Explanations - Once again, time and space preclude us from printing fully expanded programs with an explanation of each step, but we will do it where ever possible, such as in the "Binary to Hexadecimal Quiz" this month, and other teaching type programs.

Distinguishing CHIPOS variables - Starting this month, in the fully expanded programs, all variables will be UNDERLINED thus, 0, F, etc., in the Explanation column, to distinguish them from Hex constants such as OF, 3E, etc. This will be a standard practice for all future issues.

Key functions - All programs we print will be modified to suit the "Digitran" keyboard, which we believe to be the most popular one in use.

Yearly Memberships - We haven't offered this yet as the length of time we continue to operate will depend on the number of programs and articles we receive, so keep them coming in. If the demand keeps up, we will make a decision on subscriptions in two or three months time.

Well, that's it for this month, if you have a suggestion, feel free to drop us a line. We don't mind criticism as long as it is constructive, and we don't mind you telling us if you think we are doing a good job. We have tried to make it as easy as possible for you to order each newsletter, you don't have to write us a letter, just PRINT your name and address on the enclosed label and put it in an envelope with your cheque or postal order for $3-00, (remember the price reduction) and we will send you the next one the first week in November. By the way, Michael Bauer has sent us some great stuff, and some of this will be in the November issue, so DON'T MISS IT!

'Bye for now, and,

Happy DREAMing,

Garry Nelson and Graeme Samways,

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

P.S. The photo on the front cover is of Graeme's DREAM. We will be telling you about the printer and the joystick in future issues.

The photo below shows how the TAPE LOAD LEDs described in IDEAS - A Bright One, in this issue were added to Graeme's system.

# HOW TO USE CHIPOS

## INTRODUCTION

CHIPOS is an "assembler" language. It works by stepping through the program, starting from 0200, and looking at each instruction's format, then executing that instruction before going on to the next.

The CHIPOS instructions cannot be used by the 6800 directly so when you "run" a CHIPOS program, (i.e., C000, FN, 3,) you run a machine code program from C000 (stored in the EPROM) which starts at 0200 and interprets and executes each instruction as it comes to it.

Over the next few months we will describe the uses of each instruction and groups of instructions.

This month we will describe:-
The GOTO instruction - 1MMM
The COMPUTED GOTO - BMMM
The GOTO SUBROUTINE - 2MMM
The RETURN instruction 00EE

## 1) GOTO (1MMM)

This instruction does what it says. That is, it sends the program to the location specified by the three hexadecimal digits after the 1. (MMM IS A GENERALISATION FOR ANY LOCATION.)

For example -

```
                1200  Says GOTO 0200 (i.e., The starting point.)
                1300  Says GOTO 0300
```

If you want to terminate a program without returning to the monitor, you can put the program into a loop by having a GOTO "Itself".
e.g., at Address Instruction

```
     0280      1280        this will cause it to go to location 0280
```

which says go to 0280. The computer will continue to do this until RESET is pressed.

## 2) COMPUTED GOTO (BMMM)

This does a similar thing in that it goes to the specified location plus the value of Variable 0. (0)

For example - you have 0 (Variable 0) = 80. Then you have a computed goto B300. This will result in the program going to location 0380. (i.e. 0300 + 80.) or, 0 = 58, instruction B2F0 says Goto 02F0 + 58, = Goto 0348.

This can be used for areas where a large number of GOTOs would be needed. Say you wanted to input a key then go to a different subroutine for each key entered.

```
       F00A   Wait for a key to be pressed then store the value in 0
       8004   Add 0 to itself. (i.e. double it)
       B300   Goto 0300 + 0 (Variable 0)

0300   1220   If key was 0 we go to 0300, then GOTO 1220
0302   1230   If key was 1 we go to 0302, then GOTO 1230
0304   1240   If key was 2 we go to 0304, then GOTO 1240
0306   1250   If key was 3 we go to 0306, then GOTO 1250
0308   1260   etc, etc,
```

## 3) GOTO SUBROUTINE (2MMM)

This instruction interrupts the flow of the program and then stores the location of the instruction AFTER the GOTO SUB in the scratch pad. It then goes to the location specified by the goto subroutine instruction and executes the subroutine from that point.

## 4) RETURN FROM SUBROUTINE (00EE)

When this instruction is encountered in a program the computer looks for the location stored by the GOTO SUBROUTINE instruction in the scratch pad, then it goes to this address, deletes the address from the scratch pad, then continues to execute the program from the retrieved address. E.G. Say you want to display 4 digits which are stored in 0, 1, 2, & 3, in a line with a one dot space between them.

```
ADDR  PROG  EXPLANATION
0200  6000  0 = 00
0202  6101  1 = 01
0204  6202  2 = 02     Set up variables
0206  6303  3 = 03

0208  6A00  A = 00     X coordinate = 00
020A  6B00  B = 00     Y coordinate = 00

020C  F029  Set index pointer to display the value of 0
020E  2300  Go to subroutine at 0300  (Display subroutine)
0210  F129  Set index pointer to display the value of 1
0212  2300  Go to subroutine at 0300
0214  F229  Set index pointer to display the value of 2
0216  2300  Go to subroutine at 0300
0218  F329  Set index pointer to display the value of 3
021A  2300  Go to subroutine at 0300
021C  F000  Return to monitor (CHIPOS)

(End of main program.)
SUBROUTINE
0300  DAB5  Display digit at locations specified in A, B.
0302  7A04  Add 4 to A. (Move 4 dots to the right, ready to display
            the next digit.
0304  00EE  Return to the location after the go to subroutine instruction.

(End of subroutine)
```

G.V.Samways.

(To be continued next month)

********************

## ERRATA

Please note the following Boo-Boos on last months memory expansion design. No.1. The heading should read "1 to 5K", NOT 1 to 8K. If you use more than 5K (i.e. the original 1K plus 4K expansion) you will overload the 6800 chip and it may suffer from an internal haemorrhage or something equally nasty. Being restricted to 5K should not really worry you, as the CHIPOS language can only address the first 4k anyway.

No.2. Pin 10 of the 2114 was shown as going to $R/\overline{W}$, which is Pin 6 of the second expansion bus socket. This is not correct. Pin 10 of the 2114 should go to $\overline{WE}$, which is derived from Pin 10 of the existing 2114 sockets.

********************

## ATTENTION ALL RADIO AMATUERS.

It would appear that quite a number of our subscribers are radio amatuers. If you would like to contact other DREAMers on the air, send us your particulars in the following format when you order your next newsletter.

NAME CALLSIGN TIME AND FREQUENCY FOR CONTACT

We will print all those received in the DECEMBER newsletter, then print an updated list each three months after that.

********************

## BINARY - HEXADECIMAL QUIZ (0200 - 0300)

G. V. Samways,
[redacted]

As this is the first of our "teaching" programs we have expanded the program to include explanations so that you can follow the instructions through and see what each one does. We have also adopted the practice of underlining all Hexadecimal VARIABLES thus; 0, F, etc. This helps to separate these and make them easily distinguishable from CONSTANTS 00, OF, etc.

The program is designed to teach you how to convert BINARY numbers to HEXADECIMAL and vice versa.

Two 3 digit scores are displayed. The left hand one is the computer's score, the right hand one is yours. In the centre just under the scores, either a four digit Binary number, or a Hex. number is displayed. You simply enter the Hex. or Binary equivalent to the displayed number. If you are right, the correct answer is displayed instantly, if you are wrong you get a bleep, then the correct answer is shown underneath. After about two seconds the score is incremented and a new number selected at random.

| SECTION | ADDRESS | PROGRAM | EXPLANATION |
|---|---|---|---|
| SET UP | 0200 | 6C00 | C = 00. (Computer score = 00) |
| | 0202 | 6D00 | D = 00. (Your score = 00) |
| | 0204 | 00E0 | Erase (Clear the screen) |
| | 0206 | 6A08 | A = 08 (X coordinate = 08) |
| | 0208 | 6B00 | B = 00 (Y coordinate = 00) |
| | 020A | A080 | Index = 0080 (Set index) |
| | 020C | FC33 | Store 3 digit decimal equivalent of C at 0080 |
| | 020E | 22AC | Go to Subroutine at 02AC (Display score) |
| | 0210 | 6A28 | A = 28 (X coordinate = 28) |
| | 0212 | A080 | Index = 0080 (Set Index) |
| | 0214 | FD33 | Store 3 digit decimal equivalent of D at 0080 |
| | 0216 | 22AC | Go to subroutine at 02AC (Display score) |
| SET UP NUMBER | 0218 | CE0F | E = Random number less than 10. (i.e. a random Hex. number between 00 and 0F) |
| SET 0 TO 8 BIT | 021A | 6008 | 0 = 08 |
| | 021C | 80E2 | 0 = 0 + E. Filter out all but 8 bit. |
| | 021E | 3000 | Skip next instruction if 0 = 00. (Skip if 8 bit 0) |
| | 0220 | 6001 | 0 = 01 If 8 bit 1, 0 = 01 |
| SET 1 TO 4 BIT | 0222 | 6104 | 1 = 04 |
| | 0224 | 81E2 | 1 = 1 + E. Filter out all but 4 bit. |
| | 0226 | 3100 | Skip next instruction if 1 = 00. (Skip if 4 bit 0) |
| | 0228 | 6101 | 1 = 01 If 4 bit 1, 1 = 01 |
| SET 2 TO 2 BIT | 022A | 6202 | 2 = 02 |
| | 022C | 82E2 | 2 = 2 + E. Filter out all but 2 bit. |
| | 022E | 3200 | Skip next instruction if 2 = 00. (Skip if 2 bit 0) |
| | 0230 | 6201 | 2 = 01 If 2 bit 1, 2 = 01 |
| SET 3 TO 1 BIT | 0232 | 6301 | 3 = 01 |
| | 0234 | 83E2 | 3 = 3 + E. Filter out all but 1 bit. |
| | 0236 | 3300 | Skip next instruction if 3 = 00. (Skip if 1 bit 0) |
| | 0238 | 6301 | 3 = 01 If 1 bit 1, 3 = 01 |

## BINARY - HEXADECIMAL QUIZ (CONT)

```
SECTION   ADDRESS   PROGRAM   EXPLANATION

CHOOSE    023A      C401      4 = Random number either 0 or 1.
MODE      023C      4401      Skip next instruction if 4 ≠ 01
          023E      1286      Go to 0286

HEX TO    0240      6A1E      A = 1E  (X coordinate = 1E)
BINARY    0242      6B08      B = 08  (Y coordinate = 08)
          0244      FE29      Set index to point to show E
          0246      DAB5      Display the random Hex.number
          0248      6A14      A = 14  (X coordinate = 14)
          024A      6B10      B = 10  (Y coordinate = 10)
          024C      6900      9 = 00  Reset accumulative score
          024E      22C2      Go to subroutine at 02C2 (Get binary key)
          0250      3800      Skip next instruction if 8 = 00.(Skip if
                              input key = 00.)
          0252      7908      Add 8 to 9 if key input was 1.
          0254      22C2      Go to subroutine at 02C2 (Get binary key)
          0256      3800      Skip next instruction if 8 = 00.
          0258      7904      Add 4 to 9 if key input was 1.
          025A      22C2      Go to subroutine at 02C2 (Get binary key)
          025C      3800      Skip next instruction if 8 = 00.
          025E      7902      Add 2 to 9 if key input was 1.
          0260      22C2      Go to subroutine at 02C2 (Get binary key)
          0262      3800      Skip next instruction if 8 = 00.
          0264      7901      Add 1 to 9 if key input was 1.
(9 now equals the Hex equivalent of the Binary input.)

COMPARE   0266      59E0      Skip if 9 = E. (Does Binary input = the random number?)
          0268      126E      Go to 026E (If 9 ≠ E)
          026A      7D01      D = D + 1. (Increment your score.)
          026C      1274      Go to 0274
          026E      7C01      C = C + 1. (Increment computers score.)
          0270      6420      4 = 20  (Set bleep time.)
          0272      F418      Bleep for 640 ms.  (20 x 4 ms.)
          0274      6A14      A = 14  (X coordinate = 14)
          0276      6B18      B = 18  (Y coordinate = 18)
          0278      22E0      Go to subroutine at 02E0 (Display 4 bit binary number)

DELAY     027A      6480      4 = 80  (Set time delay)
          027C      F415      Set timer at 4
          027E      F407      Get timer value
          0280      3400      Skip next instruction if 4 = 00.(Time is up)
          0282      127E      Go to 027E (Return to check timer value)
          0284      1204      Go to 0204 (Return to start)

BINARY    0286      6A14      A = 14  (X coord. = 14.)
TO HEX    0288      22DE      Go to subroutine at 02DE (Display 4 digit binary no.)
          028A      F90A      9 = key input. (Get Hex key)
          028C      F929      Set index to show 9
          028E      6A1E      A = 1E  (X coordinate = 1E)
          0290      6B10      B = 10  (Y coordinate = 10)
          0292      DAB5      Display inputed Hex key
          0294      59E0      Skip if 9 = E. (Skip if input key = Hex number)
          0296      129C      Go to 029C
          0298      7D01      D = D + 1  (Increment your score)
          029A      12A2      Go to 02A2
          029C      7C01      C = C + 1  (Increment computers score)
```

## BINARY - HEXADECIMAL QUIZ (CONT)

| SECTION | ADDRESS | PROGRAM | EXPLANATION |
|---|---|---|---|
| BINARY | 029E | 6420 | 4 = 20 (Set bleep time) |
| TO HEX | 02A0 | F418 | Bleep for 640 ms. (20ms x value of 4) |
| (CONT) | 02A2 | 6A1E | A = 1E (X coordinate = 1E) |
| | 02A4 | 6B18 | B = 18 (Y coordinate = 18) |
| | 02A6 | FE29 | Set index to display correct Hex. number |
| | 02A8 | DAB5 | Display correct number |
| | 02AA | 127A | Go to 027A (Go to delay) |
| | | | |
| SUBROUTINES | | | |
| DISPLAY | 02AC | F265 | Load 0, 1, 2 from index |
| SCORE | 02AE | F029 | Set index to show 0 |
| | 02B0 | 22BC | Go to subroutine at 02BC (Display) |
| | 02B2 | F129 | Set index to show 1 |
| | 02B4 | 22BC | Go to subroutine at 02BC (Display) |
| | 02B6 | F229 | Set index to show 2 |
| | 02B8 | 22BC | Go to subroutine at 02BC (Display) |
| | 02BA | 00EE | Return from subroutine |
| | | | |
| DISPLAY | 02BC | DAB5 | Display 5 byte number from index |
| | 02BE | 7A04 | A = A + 4 (Increment X coordinate by 4.) |
| | 02C0 | 00EE | Return from subroutine |
| | | | |
| GET KEY | 02C2 | 64FE | 4 = FE (Set 4 to detect all but 0, 1, keys) |
| (BINARY) | 02C4 | F80A | 8 = key (Input key) |
| | 02C6 | 8482 | 4 = 4 and 8 (Detect key higher than 1.) |
| | 02C8 | 3400 | Skip next instruction if 4 = 00. (Skip if key 0 or 1 ) |
| | 02CA | 12D0 | Go to 02D0 |
| | 02CC | 22D6 | Go to subroutine at 02D6 (Display digit) |
| | 02CE | 00EE | Return from subroutine |
| | 02D0 | 6420 | 4 = 20 (Set bleep time) |
| | 02D2 | F418 | Bleep for 640 ms. |
| | 02D4 | 12C4 | Go to 02C4 (Get another key) |
| | | | |
| DISPLAY | 02D6 | F829 | Set index to show 8 |
| BINARY | 02D8 | DAB5 | Display 5 bytes from index |
| | 02DA | 7A07 | A = A + 7 (Increment X coordinate by 7) |
| | 02DC | 00EE | Return from subroutine |
| | | | |
| DISPLAY | 02DE | 6B08 | B = 08 (Set X coordinate to 08) |
| 4 BINARY | 02E0 | F029 | Set index to show 0 |
| BITS | 02E2 | 22D8 | Go to subroutine at 02D8 (Display binary) |
| | 02E4 | F129 | Set index to show 1 |
| | 02E6 | 22D8 | Go to subroutine at 02D8 (Display binary) |
| | 02E8 | F229 | Set index to show 2 |
| | 02EA | 22D8 | Go to subroutine at 02D8 (Display binary) |
| | 02EC | F329 | Set index to show 3 |
| | 02EE | 22D8 | Go to subroutine at 02D8 (Display binary) |
| | 02F0 | 00EE | Return from subroutine |

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

# ACEY DUECEY (0200 - 0400)

G.V.Samways,
[redacted]

In this game the computer selects two random numbers between 0 and 255. These are displayed in the middle left and right of the screen, the larger on the left. The difference is calculated and displayed top left, your score,(in $'s) is displayed top right.

To play, enter 0, a third random number will be displayed. If this number is between the first two, YOU WIN. If it is not, YOU LOSE. For a winning hand, 256 is added to your score, and the difference, (displayed top left) is subtracted. If you lose, the difference is subtracted from the score. From this you will see that the larger the "difference" number is, the more chance you have of winning, but you will score less.The smaller the "difference" number, the greater the risk of losing, but you score more if you win. If you feel that the "difference" is too small, pushing any other key will select two new random numbers. When you go broke, (which is very easy), your score is erased and the computer bleeps. Enter F to restart the game.

```
0200    6300  6400  6A38  6B00    8830  2390  2390  6A10
0210    6B17  A3E8  DAB7  6A18    6B18  6000  6100  A3F0
0220    234C  6105  A3F2  234C    6100  A3F4  234C  0000
0230    CCFF  CDFF  8EC0  8ED5    4E00  1230  3F00  1248
0240    8ED0  8DC0  8CE0  8ED5    6A08  6B0C  88C0  2390
0250    2390  6A38  88D0  2390    2390  6A08  6B00  88E0
0260    2390  2390  FF0A  3F00    12F0  C9FF  6A20  6B0C
0270    8890  2390  2390  6A38    6B00  8830  2390  8840
0280    2390  7301  6A38  8830    2390  3800  6300  8484
0290    8840  2390  6700  87E5    A3FD  F733  6A18  6B18
02A0    A3F0  23A8  A3F2  23A8    A3F4  23A8  6100  A3FA
02B0    F133  8890  88C5  3F00    1310  8890  88D5  4F00
02C0    1310  6506  75FF  23B8    3500  12C4  6A18  6B18
02D0    A3F0  23A8  A3F2  23A8    A3F4  23A8  6F38  FF15
02E0    FF07  3F00  12E0  6A20    6B0C  8890  2390  2390
02F0    6A08  6B00  88E0  2390    2390  6A08  6B0C  88C0

0300    2390  2390  6A38  88D0    2390  2390  FB18  1230
0310    6F1C  FF18  6506  6600    75FF  2350  3500  1318
0320    3601  12CC  033A  6F10    FF15  FF07  3F00  132A
0330    6F0F  EF9E  1324  00E0    1200  C610  D721  C640
0340    7EC2  E500  0000  0000    0000  0000  F155  13AA
0350    A3F0  F51E  F065  8100    A3FA  F51E  F065  4600
0360    1378  3100  1376  6109    6601  8105  8010  A3F0
0370    F51E  F055  00EE  71FF    6600  8710  8105  3F00
0380    136C  6601  8170  710A    8105  136C  0000  0000
0390    A3F7  F833  F265  F129    DAB5  7A04  F229  DAB5
03A0    7AF4  8800  00EE  0000    F165  F029  DAB5  7A04
03B0    F129  DAB5  7A04  00EE    A3F0  F51E  F065  8104
03C0    A3FA  F51E  F065  8104    A3F7  F133  F265  A3F0
03D0    F51E  8020  F055  00EE    0000  0000  0000  0000
03E0    0000  0000  0000  0000    20F8  A0F8  28F8  2000  ← $
03F0    0000  0000  0000  0000    0000  0000  0000  0000
```

* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

## MINE FIELD

(0200 - 0400)

G.V. Samways,
[redacted]

A grid is displayed and you have to move from the top left hand corner to the bottom right hand corner. Ten mines have been placed randomly in the grid. If you hit one, your score is incremented by one, a marker is left which you cannot pass, and the cursor is returned to the start ready for you to try again.

To move, you enter:-

$9 UP

$4 LEFT A6RIGHT

D1 DOWN

There is no guarantee that you will be able to make it to the bottom corner. Good luck!!)

```
0200   6A00  6B00  2300  6A00    6B1E  2300  6A00  6B02
0210   2310  6A3E  6B02  2310    6A02  6B06  2320  7A08
0220   3A3A  121A  6400  C201    CA07  4201  1232  4A07
0230   1228  8AA4  8AA4  8AA4    7A02  3201  7A04  CB03
0240   4200  1248  4B03  123E    8BB4  8BB4  8BB4  7B02
0250   3200  7B04  6300  A080    F31E  9340  126C  7302
0260   F165  5A00  1256  5B10    1256  1226  4414  1370
0270   80A0  81B0  F155  7402    1226  6A02  6B02 2382 A32C
0280   DAB4  FC0A 137A 6900  3C0509    128E  4B02  1282  3C0804
0290   1296  4A02  1282  3C0A06    129E  4A3A  1282  3C0B01
02A0   12A6  4B1A  1282  2330    6300  A080  F31E  F165
02B0   7302  4316  12CA  5A00    12AA  5B10  12AA  6820
02C0   F818  7701  A356  DAB4    127A  6900  A32C  2330
02D0   3A3A  1282  3B1A  1282    6840  F815  F807  3820
02E0   12DC  F818  A2FC  F733    F265  6A1C  6B10  00E0
02F0   F029  235A  F129  235A    F229  1360  0000  0000

0300   A30C  DAB2  7A08  4A40    00EE  1302  FFFF  C0C0
0310   A30E  DAB2  7B02  4B1E    00EE  1312  0F0F  0F0F
0320   A31C  DAB4  7B08  4B1E    00EE  1322  0060  6000
0330   DAB4  4C0502 7BFF  4C0802    7AFF  4C0A06 7A01  4C0D01
0340   7B01  7901  DAB4  4904    00EE  6F10  FF15  FF07
0350   3F00  134E  1330  9000    0090  DAB5  7A0406 00EE
0360   235A  6880  F815  F807    380002 1366  ~~00E0~~ 13 74 ~~1200~~ 0000
0370   6700  127A  ~~0000~~  0000    0000  0000  0000  0000
                   0060  F818            1200  F60A  320F  1284
        *     *     *     *     *     *     *     *     *     *     *
0380   1284  6010  FC18  6B02    00EE
```

9

# MINE FIELD SMALL

(0200 - 0400)

G.V. Samways,

[redacted]

This is the same as "Mine Field" except on a smaller scale. You are now half size in a bigger grid, and there are 100 mines. (The computer may take 5 - 10 seconds after drawing the grid to determine the position of the mines.) You can save a lot of work by loading Mine Field, then stepping through the memory and modifying the program where necessary.

↑ ← 4 6 →
9 ↓

```
0200    6A00  6B00  2300  6A00      6B1F  2300  6A00  6B01
0210    2310  6A3F  6B01  2310      6A01  6B01  2320  7A04
0220    3A3D  121A  6400  C201      CA0F  4201  1232  4A0F
0230    1228  8AA4  8AA4  7A01      3201  7A02  CB07  4200
0240    1246  4B07  123C  8BB4      8BB4  7B01  3200  7B02
0250    6300  A080  F31E  9E40 1500 1268  7302  F165  5A00
0260    1252  5B10  1252  1226 6764 4464  1276  80A0  81B0
0270    F155  7402  1226  6700      6A01  6B01  0000  A30E
0280    DAB1  FC0A  6900  3005 3C09 128E  4B0102 1282 3C08
0290    1296  4A0182 1282 3C0A06    129E  4A3D  1282  3C0D01
02A0    12A6  4B1D  1282  2330      6300  A080  F31E  F165
02B0    7302  4366  12CA  5A00      12AA  5B10  12AA  6820
02C0    F818  7701  A356  DAB2      1278  6900  A30E  2330
02D0    3A3D  1282  3B1D  1282      6840  F815  F807  3820
02E0    12DC  F818  A2FC  F733      F265  6A1C  6B10  00E0
02F0    F029  235A  F129  235A      F229  1360  0000  0000

0300    A30C  DAB1  7A08  4A40      00EE  1302  FF80  8040
0310    A30D  DAB1  7B01  4B1F      00EE  1312  0000  3030
0320    A31C  DAB4  7B04  4B1D      00EE  1322  FF18  FF00
0330    DAB1  4C0509 7BFF 4C0804    7AFF  4C0A06 7A01 4C0D01
0340    7B01  7901  DAB1  4992 04 02 00EE 6F10  FF15  FF07
0350    3F00  134E  1330  40C0      0000  DAB5  7A04  00EE
0360    235A  6880  F815  F807      380001 1366 00E0 15 1200 0000
0370    0000  0000  0000  0000      0000  0000  0000  0000
                    00E0  F818      1200. FC0H  3C0F  1284
0380    1362  6C10  FC18  00EE
* * * * * * * * * * * * * * * * * * * * * * * * * * *
        254
```

## ONE CHECK (0200 - 0400)

G. V. Samways,
[redacted]

(From a "Basic" program by D. Ahl.)

In this game or puzzle, 48 checkers are placed on the two outside spaces of a standard 64 square checker board. The object is to remove as many checkers as possible by making diagonal jumps. (As in standard checkers.)

It is easy to remove 30 to 39,
a challenge to remove 40 to 44,
and a substantial feat to remove 45 to 47.

Enter the "X" coordinate (0-7), then the "Y" coordinate (0-7), of the checker you wish to move, check your position with the cursor, then move by entering:-

| | | | |
|---|---|---|---|
| 4 | UP LEFT | 6 | UP RIGHT |
| C | DOWN LEFT | E | DOWN RIGHT |

Entering any other key will cancel the move and allow you to select a new set of coordinates.

```
0200    A2CC  6A00  22C0  7A04    3A20  1204  A2DC  6A00
0210    22C0  6A04  22C0  6A18    22C0  6A1C  22C0  6B00
0220    22D0  6B04  22D0  6B18    22D0  6B1C  22D0  6900
0230    632C  640C  A2BC  F933    F265  F029  22E0  F129
0240    22E0  F229  22E0  6608    6708  1360  87C2  4700
0250    1256  F618  1248  8CC4    8CC4  6708  1368  87D2
0260    3700  1252  8DD4  8DD4    A358  DCD4  8EF0  3E00
0270    1370  DCD4  4E00  1252    8AC0  8BD0  22E8  4E00
0280    1252  22E8  3E00  1252    A2DC  DCD4  6F00  8CA5
0290    4F00  7C08  7CFC  8CA4    6F00  8DB5  4F00  7D08
02A0    7DFC  8DB4  DAB4  DCD4    632C  640C  F029  22E0
02B0    F129  22E0  F229  22E0    7901  1230  0000  0000
02C0    6BFC  7B04  DAB4  3B1C    12C2  00EE  0000  2000
02D0    6A04  7A04  DAB4  3A14    12D2  00EE  0070  7070
02E0    D345  7304  00EE  0000    3804  1308  4A00  1354
02F0    4A04  1354  4B00  1354    4D04  1354  7CFC  7DFC

0300    DCD4  8EF0  DCD4  00EE    3806  1320  4A18  1354
0310    4A1C  1354  4B00  1354    4B04  1354  7C04  12FE
0320    380C  133A  4A00  1354    4A04  1354  4B18  1354
0330    4B1C  1354  7CFC  7D04    1300  380E  1354  4A18
0340    1354  4A1C  1354  4B18    1354  4B1C  1354  7C04
0350    7D04  1300  6E00  00EE    0050  0050  0000  0000
0360    * * * * * * *  *  *  * *  *  * *  *  *  *  *  *
        FC0A  3C0F  124C  1376    FD0A  3D0F  125E  1376

0370    F80A  380F  1272  00C0    1200
```

## SPIRAL TRIAL. (0200 - 0400)

P. E. MARSTON.

Computer draws spiral.

Timer starts counting down from the moment the player appears in the middle of the spiral.

Player moves through spiral with keys 4, 6, 9, 1.

A crash sends player back to start. The number of crashes are recorded and shown R. H. S.

Game stops when player reaches home, the time left being his score.

Game also stops when 5 crashes are recorded or when timer = 000.

```
0200  6A00  6B01  A2F8  22FC  3A3E  1206  2302  3B19
0210  120C  2308  3A04  1212  230E  3B03  1218  22FC
0220  3A3B  121E  2302  3B17  1224  2308  3A07  122A
0230  230E  3B05  1230  22FC  3A35  1236  2302  3B15
0240  123C  2308  3A0A  1242  230E  3B07  1248  22FC
0250  3A32  124E  2302  3B13  1254  2308  3A16  125A
0260  230E  3B09  1260  22FC  3A29  1266  2302  3B11
0270  126C  2308  3A18  1272  230E  3B0B  1278  22FC
0280  3A27  127E  2302  3B0F  1284  2308  3A1B  128A
0290  230E  3B0D  1290  6564  6700  6800  2314  231E
02A0  6A20  6B0D  A2F8  DAB1  4500  1348  4705  1348
02B0  4A01  1348  A2F8  DAB1  6C00  6D00  6E04  EEA1
02C0  6CFF  6E06  EEA1  6C01  6E01  EEA1  6DFF  6E09
02D0  EEA1  6D01  8AC4  8BD4  DAB1  6604  F615  F607
02E0  3600  12DE  4F00  134E  6920  F918  231E  7701
02F0  231E  A2F8  DAB1  12A0  8000  0000  DAB1  7A01
0300  00EE  DAB1  7B01  00EE  DAB1  7AFF  00EE  DAB1
0310  7BFF  00EE  A356  F533  6301  2328  00EE  A356
0320  F733  6332  2328  00EE  641B  F265  F029  D345
0330  7305  F129  D345  7305  F229  D345  00EE  2314
0340  75FF  2314  6800  00EE  6950  F918  134C  7801
0350  480A  233E  12A8  0000
```

## 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 of inclusion in the newsletter. ALL CONTRIBUTORS OF ARTICLES AND PROGRAMS PRINTED WILL RECEIVE TWO MONTHS NEWSLETTERS FREE OF CHARGE. Along with a listing of the program 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 20sec High tone, then 20sec 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 dealing in tapes, but if you submit a program, and wish to sell tapes, just state this after your program explanation, and detail your charges etc.

Programs submitted for consideration must be typed on A4 in BLACK 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) Details of cassette cost etc. (If applicable)

5) 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, and saves us a great deal of work if they do not have to be retyped before printing.

That's all there is to it, so send us in your favourites, and don't forget, for each one we use, you get two months newsletters free of charge.

*********************

## BACK COPIES OF NEWSLETTERS

Copies of all newsletters from No.1, September 1980, are available at a cost of $4-00 each, from:-

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

***********************

# HOW TO RUN MACHINE CODE PROGRAMS

G. V. Samways,
[redacted]

The following three short programs generate different sequences of High and Low tones. They are designed to show the use of machine code programs and subroutines.

## TONE 1.

```
ADDR.   PROG.    EXPLANATION

0200    C03F     Get random number less than 3F, store in 0
0202    F018     Bleep for this time at 2400 Hz
0204    C03F     Get random number less than 3F
0206    0210     Go to machine code subroutine at 0210 for 1200 Hz tone
0208    1200     Go to 0200. (Return to start.)
020A    0000     No op.
020C    0000     No op.
020E    0000     No op.
0210    F60030   Load Accumulator "B" from 0030.(I.E.with value stored in 0 )
        D721     Store Accumulator "B" at 0021 (Duration of tone)
        C640     Load Accumulator "B" to 0040 (1200 Hz tone)
        7EC2E5   Go to C2E5(Go to EPROM for rest of subroutine)
```

## TONE 2

```
0200    6000     Set 0 = 00
0202    7001     Add 1 to 0
0204    F018     Bleep for 0 x 20 milliseconds at 2400 Hz
0206    0210     Go to machine code subroutine at 0210
0208    400F     If 0 ≠ (does not equal) F skip next instruction
020A    1200     Go to 0200. (Return to start.)
020C    1202     Go to 0202. (return to increment 0)
020E    0000     No op.
0210    F60030   )
        D721     )
                   Machine code subroutine
        C640     )
        7EC2E5   )
```

## TONE 3

```
0200    6000     0 = 00
0202    6120     1 = 20
0204    7001     Increment 0
0206    71FF     Decrease 1 by 01
0208    F018     Bleep for 0 x 20 mSec. at 2400 Hz
020A    0220     Go to machine code subroutine at 0220
020C    3101     If 0 = 01 skip next instruction
020E    1204     Go to 0204. (Return to increment.)
0210    F018     Bleep for 0 x 20 mSec. at 2400 Hz.
0212    0220     Go to machine code subroutine at 0220.
0214    70FF     Decrement 0
0216    7101     Increment 1
0218    3001     If 0 = 01 Skip next instruction
021A    1210     Go to 0210. (Return to bleep.)
021C    1200     Go to 0200. (Return to start.)
021E    0000     No op.
0220    F60031   )
        D721     )
                   Machine code subroutine
        C640     )
        7EC2E5   )
```

# IDEAS

## A BRIGHT ONE

We have received several ideas for putting indicator LEDs on the tape input, but they have all been illegal as they all place the LED from the output of a TTL chip to Ov.(See Figure 1.)(This includes the Applied Technology and J.R. Components indicator on the 6802 board.)When you run a LED from a TTL output to ground, you draw at least 10-20 mA, but the high level output current of most 74 series I.C.'s, (including the 7474 and 7404), is only 400 uA. This places 50 times the absolute maximum current on the I.C.(If you are having trouble with your tape input we suggest you remove the LED and replace the 7474.(I.C.21)). The low level output current however, can supply up to 8 mA, (7474, 74LS74, 74LS04), or 16 mA (7404), so if you use the small 3mm diameter LEDs and limit the current to under 8 mA, you can run the LEDs directly.

I have designed the following circuit (See Figure 2) as it gives both High and Low level indication. This circuit will draw 7 mA, which is well within specifications. The circuit allows you to tune the input port for the best dump. You can test the circuit by connecting the tape output to the tape input while the computer is not running. The High tone is present at the output, (the leader tone) so the GREEN LED will glow. If you run the following machine code program the RED LED will glow, and the LOW tone should be heard.

```
ADDRESS   PROGRAM  EXPLANATION

0200      8640     Load accumulator A to 40
0202      B78012   Store this at the PIA B port
0205      20FE     Branch always back 2 Bytes.(i.e.,to 0205, Itself.)
0207      0000
```

To run this program (and other machine code programs) enter the 4 digit starting address, (0200) then FN, 3. You should then hear the LOW tone and the RED LED should glow.The IN914s are to prevent reverse voltages being applied to the LEDs.

FIGURE 1.

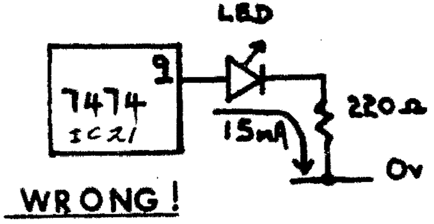
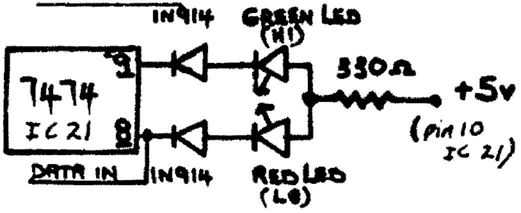


FIGURE 2.



G.V.Samways.

********************

## A HOT ONE

Have you noticed that your EPROM and 2114 RAM chips (or your 6802 if you have one) are getting HOT? This can cause software failure during the warmer months, or after prolonged use.It was first noticed on my system by tape load failure. To alleviate this problem you can Heat Sink your EPROM, 2114s etc. I used very light gauge aluminium (0.7mm or 1/32"). See Figure 3 for dimensions. If your EPROM has a raised programming window, you may need to cut a hole in the centre of the heatsink so it fits around the window. You should also allow ventilation into your DREAM.

When you have cut out the heatsink, colour it black with a Texta or paint, then mark Pin 1, and the I.C.number.(2708, 2114, 6802 etc)Don't forget to put CHIPOS on the EPROM heatsink.(Letraset makes a neat job.)

## IDEAS - CONT

Fold it up along the lines to about 45 degrees, (See Figure 4.) Clean the centre area underneath the heatsink (where it will make contact with the I.C.) with emery or wet and dry. Apply some heatsink compound to the underside of the heatsink in the centre, (Not too much,) then apply some Super Glue or Araldite to each corner. (See Figure 5.) Don't allow the heatsink compound to mix with the glue. Press the heatsink onto the I.C. until the glue is dry, making sure that the Pin 1 mark is over Pin 1. You should also cover the window of the EPROM if the heatsink does not achieve this, to prevent U.V. rays from entering and erasing your CHIPOS program.

FIGURE 3

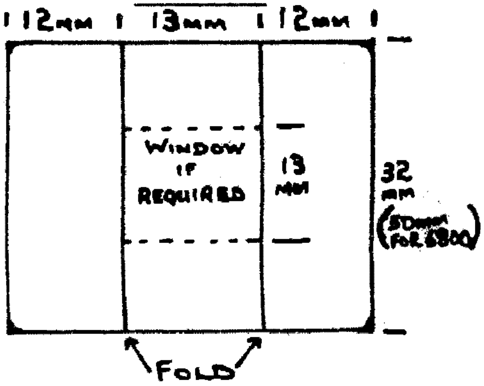


FIGURE 4

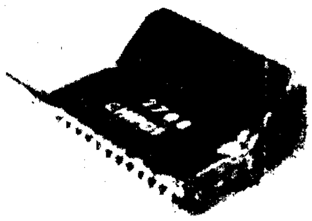
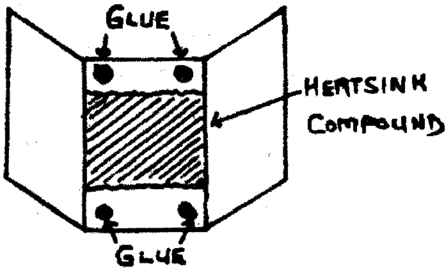


FIGURE 5



G. V Samways.

******************

## CAN YOU HELP?

Mr. R. Moroney [redacted] 4051, writes that his DREAM has turned into a nightmare. He purchased a kit and assembled it, but the beastie will not work. All it does when you switch it on is emit a continuous bleep. If there is anyone in the Brisbane area with the technical knowhow to help him sort it out, he would appreciate the help.

******************

## ADVERTISING

We have had a few requests for this, so starting next month we will be including a section for you to advertise in. 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 one above.)Commercial enterprises who wish to advertise in the newsletter are invited to contact us.

******************

## NEXT MONTH

Next months newsletter will contain at least the following:-

- Four games programs, including "Tic Tac Toe" and a "Poker Machine", plus two others, we haven't decided which ones yet.
- More IDEAS
- The next instalment of "How to use CHIPOS"
- Wait for it - A JOYSTICK CONTROLLER, from Michael Bauer, and it is really something. DON'T MISS IT.

******************