NewBear Computing Store
A Division of NEWBURY LABORATORIES LTD.
SALES & SERVICE: BONE LANE NEWBURY BERKSHIRE RG14 5SH Tel: 0635 46898 Telex: 858815

DESIGN NOTE 22

7768 MON 1 BOARD

1. Introduction

This board has been produced as the first stage of expansion of the basic 7768 CPU , from a single board experimental machine to a full microcomputer.

When the excitement of toggling programs into the basic 7768 CPU board by hand has subsided a little, then users will wish to expand their system, not only by adding more memory (e.g. by using the 7768 4k RAM board), but also by;

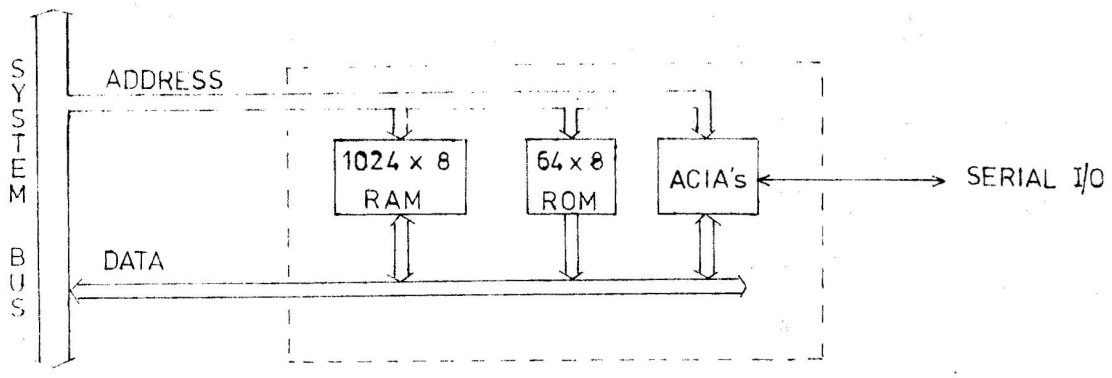
- Linking up to an external device such as a cassette tape recorder interface or a paper tape reader/punch combination so that programs may be stored permanently and re-loaded easily.
- Connecting any Teletype, VDU or similar terminal that may be available.
- Implementing some form of Monitor, or simple Operating System, to drive these I/O devices and generally aid in program loading and de-bugging.

The MON 1 board provides these facilities in a way which gives maximum flexibility for system growth, and for the particular requirements of the individual constructor.

2. Board Characteristics

Construction;	8.0" x 8.0" single sided PCB with gold plated 0.1" edge connector contacts. Compatible with 7768 CPU electrically and mechanically.
Bus;	Buffered on all lines. Compatible with 7768 system bus.
Power;	Requires +5V stabilised DC @ 0.55A typical. +12V DC @ 100mA and -12V DC @ 50mA may also be required depending on the type of serial I/O interface circuitry used.
Serial I/O:	Maximum of 2 transmit and 2 receive asynchronous serial ports. Standard data transmission rates from 50 to 9600 baud, crystal controlled frequencies. TTL, RS232C(V24) or '20mA' serial interfaces. Software selectable character of 7 or 8 data bits with or without parity, with 1 or 2 stop bits.
Memory;	1024 bytes of RAM, which may be write-protected, plus 32 or 64 byte ROM bootstrap.

3. General Description



The basic functional blocks on the board are;

- Two ACIA's (Asynchronous Communications Interface Adaptors) which allow the system to send and receive the type of serial asynchronous (stop-start) signals used by the majority of low speed computer peripheral equipment (cassette recorder interfaces, Teletypes, VDU's etc.)
  ACIA a (X5) is always fitted; ACIA b may be added later as the system grows.
  Buffer circuits in the ACIA serial transmit and receive lines convert between the MOS logic levels of the ACIA IC itself and TTL, RS232C(V24) or '20mA Current Loop' circuit interface levels as may be required by the peripheral device.
  A divider chain driven by the CPU 5MHz clock provides an accurate set of frequencies to drive the ACIA's and hence control their operating baud rate(s).
- A 1024 byte block of Read/Write memory (RAM) located at the top of memory address space (see Fig. 7). This may simply be used as additional memory in a basic system, or it can be used to hold the system Monitor program. According to the setting of a strap on the board, this 1k block of RAM may be 'Write Protected', so that the Monitor program held in it cannot be corrupted by faults in any other program running on the system.
- 32 or 64 bytes of Read Only Memory; implemented with easily programmed TTL 32 x 8 bit PROM's. This will normally hold a system 'Bootstrap' program which is run whenever the computer is switched on to load the system Monitor program from an external source (e.g. a cassette recorder) into the 1k RAM. (The board automatically disables the 'Wrtie Protect' feature during a Bootstrap load).

This arrangement; of using a short Bootstrap program stored in ROM to load the full system monitor, means that the fixed element (in ROM) is kept to a minimum and is therefore not likely to need changing, whereas the system monitor proper can easily be altered as the system grows, or when the user wishes to experiment with different features.

4. Circuit Description

X23,29 buffer and invert the least significant ten address lines (A0-A9) from the system bus. These ten buffered address signals ($\overline{A0}$-$\overline{A9}$) are then connected to the ten address inputs of each 1024 x 1 bit Random Access Memory (X15-22) to select the desired bit in each IC. The inversion of A0-A9 caused by X23,29 does not matter as it is present during both Read and Write operations,so effectively cancells.

The four most significant address lines (A12-15) are connected to Nand gate X14 so that pin 8 of this IC goes low - enabling one of the two '1 out of 4 decoders' in X14 - only when these address lines are all at logic '1'; corresponding to addresses in the range Fxxx (hex). When X13 pin 15 is low, then one of its four outputs (pins 9,10,11,12) will go low, according to the state of the address lines A10,11;

A11	A10	X13 pin 9	10	11	12	Address range	Use
0	0	1	1	1	0	F000 - F3FF	Sel of 256 byte RAM on CPU
0	1	1	0	1	1	F400 - F7FF	Input/Output addresses
1	0	1	1	0	1	F800 - FBFF	Reserved for VDU memory
1	1	0	1	1	1	FC00 - FFFF	1k RAM on MON 1 board

When pin 12 is low , the output pin 6 of X14 is forced high, to select the 256 word memory on the CPU card itself. For a large system pins 1 & 2 of X14 would be connected to pins 4 & 5. However, where only the CPU and MON 1 cards are fitted, then pins 1 & 2 of X14 should be connected to the address line A15, so that the 256 word CPU board memory is also selected when A15 is low. This allows it to appear in address space 0000 - 00FF, where the short (Direct) address instruction mode can be used.

Pin 10 of X13 going low signifies an I/O operation; as discussed later.

When pin 9 of X13 goes low, then the second 'one out of four decoder' in X13 is enabled, and one of its outputs (pins 4,5,6,7) will go low according to the state of the system R/W (Read/Write) line and the board $\overline{BOOT}$ input. This latter input is connected to 0V via a single pole on/off switch which is closed for 'Bootstrap Load', otherwise open.

$\overline{BOOT}$ input	R/W	X13 pin 4	5	6	7	Use
0V	0 (Write)	1	1	0	1	Bootstrap mode write
+5V	0 "	1	1	1	0	Normal mode write
0V	1 (Read)	0	1	1	1	Bootstrap mode read
+5V	1 "	1	0	1	1	Normal mode read

When pin 4 goes to 0, X9 pin 9 is forced to '1', enabling the two X9 gates feeding the PROM's X3,4. If address line A5 is at '0', then X9 pin 3 will go to '0', enabling X3, whereas if A5 is '1', $\overline{A5}$ will be '0', X9 pin 3 will be '1', and X9 pin 6 will go to '0', to enable X4.

The 1024 x 1 bit RAM's (X15-22) are enabled via X11,12 whenever X13 pin 5 is low (a read from memory in normal mode), or X13 pin 6 is low (a write from CPU into memory in Bootstrap mode). If pin 9 of X11 is strapped to pin 7 of X13, then X15-22 will also be enabled when X13 pin 7 goes low (Write from CPU in normal mode). However, if X11 pin 9 is connected to X11 pin 10 instead, then in

normal mode the RAM's X15-22 will not be affected by a 'write from CPU' operation.

X10 pin 11 pulls the R/W inputs to X15-22 low (= write) during a write operation (R/W input to board low) when the E timing input to the board is high. These R/W and E board input signals are buffered and inverted as necessary by parts of X10,29.

Tri-state buffers X7,8 provide a path for information to enter and leave the system data bus. If information is to be read from the 1024 byte RAM (X15-22), the ROM's (X3,4) or either ACIA, then X11 pin 6 goes low to enable the card data output buffers.

$\overline{IO}$ (X11 pin 12) goes low when X13 pin 12 goes low as long as $\overline{A2}$ and $\overline{A3}$ are both high (Address bus lines A2 & A3 0). This signal enables the data output buffers via X12,11, and is also connected to the $\overline{CS2}$ (Chip Select 2) input of both ACIA's.

The ACIA's transfer information to and from the system over their 8 bidirectional data lines (X5,6 pins 15-22). The E input to X5,6 (pin 14) ensures correct timing of the data transfer, while the direction of data flow is controlled by R/W (pin 13). The 'RS' (Register Select, pin 11) input selects the 'data' or 'Control/ Status' registers within the ACIA for connection to the data bus. For an information transfer to take place, the CS0 & CS1 inputs must be high, and the $\overline{CS2}$ input low. $\overline{CS2}$ is fed from X11 pin 12 as previously described, while CS0 is permanently high. CS1 is connected to address line A1 (X6) or $\overline{A1}$ (X5) so that X5 is selected when address bus line A1 is at 0, X6 when it is at 1.

The $\overline{IRQ}$ (Interrupt Request) outputs of X5,6 are buffered by the open collector inverters (part of X1) to drive the $\overline{IRQ}$ input of the CPU card.

The transmit and receive halves of each ACIA require clock inputs (TX Clk & RX Clk) normally at 16 x the serial data baud rate - although the ACIA's may also be programmed to work with clock inputs at 1 x or 64 x the baud rate. Simple RC filters (e.g. C3, R14) and schmidt trigger buffers are provided to reduce any noise that may be present on the clock signals. The values shown are nominal, and users experiencing difficulties with noisy clock inputs may increase the capacitance values, especially if the clock frequencies involved are relatively low.

The transmit serial data from each ACIA (pin 6) is fed to a conditioning circuit (e.g. Q1 etc.) to convert it to TTL, RS232C or 20mA current loop signals as required. Similar conversion circuits (e.g. Q3 etc.) are provided for the received serial data signal.

Each ACIA also has a data link control output ($\overline{RTS}$; Request To Send) and two inputs ($\overline{CTS}$; Clear To Send,and $\overline{DCD}$; Data Carrier Detect). These may be useful in some applications and are therefore brought out (un buffered) to the card edge connector. If they are not used, then the inputs ($\overline{DCD}$ & $\overline{CTS}$) must be connected to 0V for proper operation of the ACIA.

Further information on the characteristics and use of ACIA's is given in Appendix 3, and the user is also advised to read the manufacturer's data sheet for these devices.

X24,25 & 26 divide the CPU 5MHz clock to obtain a series of frequencies which are nominally 16 x the 'standard' data rates of 300,600,1200,2400,4800 & 9600 baud.(Due to practical difficulties the actual frequencies produced are 0.16% higher than nominal,

e.g. the '16 x 9600Hz'output is actually 16 x 9615 Hz, but this is well within the acceptable frequency tolerance for all peripheral devices). For those wishing to operate at lower baud rates X28 may be set to divide the 16 x 1200 Hx signal by any integer from 1 to 16 (see Fig 4). These 'standard' frequencies are available at the board connector for wiring to the TX Clk, RX Clk board inputs as required.

X26,28 are four stage binary counters with an output pin (15) which goes to 1 when the '1111' state is reached. This output is inverted and fed to the counter synchronous load input so that at the next clock pulse the state of the input pins (A,B,C,D) is loaded into the counter. Thus the counter counts cyclically between the number at the A B C D inputs and '1111'. Since the ABCD inputs to X26 are 1100, then it will count in the sequence;

Q_A	Q_B	Q_C	Q_D
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1
1	1	0	0

Note that there are 13 counts in a complete cycle, and during this time the QC output changes from 0 to 1 twice. Thus the input to the next divider stage (X25 pin 1) is effectively 5 MHz ÷ 13/2

5. Construction

Refer to Figs 3 - 6

The close track spacing necessary, particularly around X3-6 and X15-22, means that great care has to be taken to avoid accidental short circuits. The constructor should take care to make good soldered joints using the minimum amount of solder while ensuring that a joint is made all round the component pin or wire end. Use of a small soldering iron with a small, clean, bit is essential. Also, before fitting any component, feel all over the track side of the board for any loose swarf left from the board drilling process, and examine it carefully for unetched copper 'bridges' between tracks.

The use (or not) of IC sockets is largely a matter for personal preference (although sockets should be fitted in the X3,4 positions to allow for any possible changes to the bootstrap program). Poor quality sockets must be avoided as they can cause many hard to trace faults. Provided that the constructor is sure of the quality of the IC's he is using, is confident of his ability to solder tham in the right way round first time, and is using a low leakage soldering iron, then there is no reason why the IC's should not be soldered directly into the board. In case of trouble, remove a suspect IC by first cutting the body free from all leads, then remove the IC leads from the PCB one at a time. This procedure ruins the IC but does least damage to the board.

The best order of construction is;

First fit all the straps (not the ribbon cables at this time) using sleeved wire where appropriate.One way of getting thin sleeved wire is to take a length of thin solid cored insulated wire, strip off the insulation for about ½" from each end, then, grasping the inner wire firmly at each end with pliers and/or a vice, pull until you can feel the copper wire stretch and flow slightly. This operation reduces the diameter of the wire slightly so that it will slide freely within the insulation, and it also removes the 'spring' from the wire so it may be formed more easily.

Next, form and fit the ribbon cables as shown in Figs 3 & 5, followed by the miscellaneous resistors, capacitors, diodes and transistors.

Finally, fit the IC's; with X5,6, & 15-22 being fitted last.

After assembly clean the track side of the board thoroughly with a small stiff brush (a very hard toothbrush is ideal) and examine it carefully for short circuits caused by bent leads, excess solder on joints, solder splashes etc.

6. Testing

a) First check with an ohm meter for short circuits between tracks connecting X15-22, also between OV & +5V. Some reading is to be expected due to the internal resistances within IC's, but use the meter on its lowest ohms range to detect true short circuits. This stage is most important as it can reveal faults which would otherwise prove extremely difficult to locate.

b) Strap A to B so that the 256 word CPU board memory will respond to addresses below 8000.
Strap F to D to remove the Write Protection from X15-22.
X3,4 should <u>not</u> be fitted at this stage.

c) Check the wiring between the MON 1 and CPU cards, then switch on while carefully monitoring the +5V supply.(Note that the + and - 12V supplies are not needed at this stage so it might be prudent to leave them un connected). Leave the equipment switched on for a few minutes while checking all components for signs of overheating.

d) Test the operation of the Control Panel Load, Reset & Data switches, they should appear to operate as they did before the MON 1 card was added to the system. In fact, however, the Control Panel is now loading into and examining the top 256 words of the 1k memory on the MON 1 board, rather than the 256 word CPU board memory.(The Load logic forces the high 8 address lines to '1's). One slight difference is that setting FF on the address switches now accesses a RAM location rather than the Data Switch & Display Register.

e) Load and run the following program;

```
Address     Contents
FF 00       08          START   INX
FF 01       26 FD               BNE START
FF 03       4C                  INC A
FF 04       B7 00 FF            STA A DISPLAY
FF 07       20 F7               BRA START

FF FE       FF 00               Program start address
```

This is the FLASHER Version 1 from the CPU manual modified to run on a system incorporating the MON 1 board; it increments the display about once per second. The main differences from the original version are;

- Since the system now has more than 256 words of memory, we have to specify the full 16 bit memory addresses, as in the STA A DISPLAY instruction.
- Program addresses are given in the listing as the full 16 bits (4 hex digits) although when loading and examining via the control panel only the least significant 8 bits (two right hand hex digits) are set up - the CPU Load logic automatically sets the most significant digits to FF
- When the MPU RESET is applied, the CPU looks at addresses FFFE and FFFF for the program starting address. Since the system now uses all 16 address bits (having more than 256 words of memory) and as address FFFF no longer corresponds to the Data Switch Register, we have to load the program starting address into locations FFFF & FFFE before operating RESET.

f) Since the display (and switch register) are effectively the highest addresses of the 256 word CPU board memory, and since we have now set the system so that this 256 word memory will respond to addresses in the ranges 0000 to 7000 and F000 to F3FF,

the display (and switch register) will respond to addresses

00FF, 01FF, 02FF - - - 7FFF and F0FF, F1FF, F2FF, F3FF

Change the address part of the STA -- instruction in the above program (locations FF05,FF06) to check that the addresses given above are valid.

g) Connect a temporary wire between pin 20 ($\overline{\text{BOOT}}$) and 0V, then examine the contents of memory via the control panel. All locations should appear to contain FF (all '1's) as long as X3,4 are not plugged in.
Try to load a known pattern (other than FF) into memory; the display should show the pattern for as long as the Load switch is held depressed, but should revert to FF when the Load switch is released. Now, without switching off, or operating any of the control panel switches, cut the temporary wire linking pin 20 to 0V. The display should now show the data previously loaded.

h) Reconnect the temporary wire between pin 20 and 0V, plug in X3,4. Now read and check (via the control panel) the contents of these ROM(s). Note that due to the partial address decoding used, the same information will be read regardless of the setting of the two mist significant address switches (A6 & A7).
Disconnect the temporary wire.

i) Strap U,V,W,X to give the desired frequency on edge connector pin 13 (16 x 1200/n Hz). Check the frequencies on all outputs (connector pins 13-19). Note that if viewed on an oscilloscope, a slight jitter will be visible on the higher frequency outputs.

k) Test the ACIA's and their buffers by linking $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ of each to 0V, and connecting one of the ACIA clock divider outputs to the TX Clk & RX Clk ACIA inputs (board connector pins 60,61,62,63).

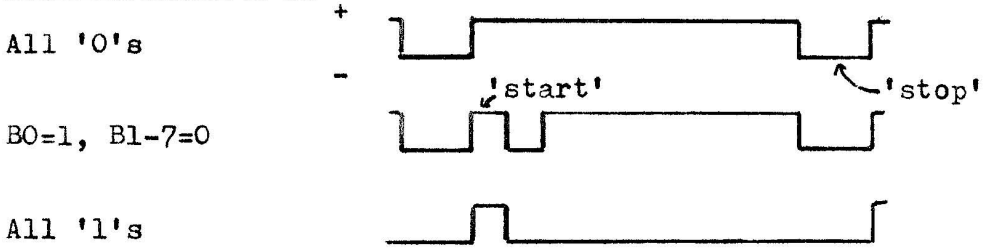
Test ACIA (a) (X5) first by temporarily looping TX DATA to RX DATA (edge connector pins 66,70 & 72 connected together, no connection to pin 67) and ensure that both TX and RX Data buffers are set up for the same interface levels (TTL or RS232 or 20mA). Load and run the following program;

```
Address    Contents
 FF 00      CE F4 00  START LDX #F400   point @ ACIA a
 FF 03      86 03           LDA A #3
 FF 05      A7 01           STA A X,1   reset ACIA
 FF 07      86 11           LDA A #11
 FF 09      A7 01           STA A X,1   set ACIA control reg
 FF 0B      A6 01     LOOP  LDA A X,1   get ACIA status reg
 FF 0D      85 02           BIT A #2    TX buffer empty ?
 FF 0F      27 04           BEQ TSTRX   wait for it
 FF 11      96 FF           LDA A SWREG
 FF 13      A7 00           STA A X     transmit sw reg
 FF 15      A6 01     TSTRX LDA A X,1
 FF 17      85 01           BIT A #1    RX buffer full ?
 FF 19      27 F0           BEQ LOOP    & round again
 FF 1B      A6 00           LDA A X     get rec data
 FF 1D      97 FF           STA A DSPLAY
 FF 1F      20 EA           BRA LOOP    & round again

 FF FE      FF 00                       start address
```

This program transmits the setting on the control panel switch register via the ACIA, The serial signal is then received by the same ACIA, and displayed. Typical waveforms to be found on the edge connector pins 66,70,72 are shown below for RS232 interfaces, the waveforms will be inverted for 20mA or TTL interfaces.

Switch Reg Setting



ACIA(b) (X6) can be tested similarly; change the first instruction of the test program to

```
CE F4 02    START LDX # F402 point @ ACIA b
```

7. Diagnostics

To be performed with the board removed from the system and supplied with +5V, +12V & -12V.

Test signals to be applied to board connector;

'0' = direct connection to OV.
'1' = connection to +5V via 1k ohm resistor.

Measurements made with 20kohm/Volt or better meter;

'0' = 0 to 0.4V
'1' = +2.4 to +5V Anything inbetween is wrong.

a) Address Buffers X23,29

Output of inverter should be opposite to input;

Check that X29 pin 10 is '1' when '0' applied to A0 (con. pin 21)
" " " " " " '0' " '1' " " " " " "
" " " " 4 " '1' " '0' " " A1 " " 22
etc.

b) R/W & E Buffers X10,29 (parts)

Check that X10 pin 9 is '1' and X10 pin 8 is '0' when '0' is applied to R/W (con. pin 4)

Check that X10 pin 9 is '0' and X10 pin 8 is '1' when '1' is applied to R/W.

Check that X10 pin 1 is '1' and X10 pin 3 is '0' when '0' is applied to E (con. pin 6)

Check that X10 pin 1 is '0' and X10 pin 3 is '1' when '1' is applied to E.

Check that X10 pin 11 is '0' only when R/W is '0' and E is '1' - and that any other combination of R/W & E inputs makes X10 pin 11 = '1'

c) Nand Gate X14 (first half)

Ensure that A12-A15 (con. pins 33-36) are all '1's, then check that X13 pin 15 is '0'.

Connect each of A12-A15 in turn to '0'; in each case X13 pin 15 should be '1'.

d) One Out Of Four Decoder X13 (first half)

With A12-A15 all '1' (hence X13 pin 15 = '0') check X13 pins 9 - 12 for combinations of inputs on A10-11;

A10	A11	X13 pin; 9	10	11	12
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

↑ pin 9 (anal) set to 0 by A10 to A15 all '1'

e) Nand Gate X14 (second half)

Set inputs A10-15 all to '1'; check that '256SEL' (con. pin 5) is '0'. Change A10,11 inputs to '0'; '256SEL' should go to '1'.

If A is strapped to C, then also set inputs A10-14 to '1', A15 to '0'; '256SEL' should be '1'.

f) One Out Of Four Decoder X13 (second half)

Set A10-A15 all to '1'. Check that X13 pin 1 is '0'.

Check X13 outputs (pins 4 - 7) for combinations of $\overline{\text{BOOT}}$ (con. pin 20) and R/W (con. pin 4) inputs;

$\overline{\text{BOOT}}$	R/W	X13 pin;	4	5	6	7
0	0		1	1	0	1
0	1		0	1	1	1
1	0		1	1	1	0
1	1		1	0	1	1

g) X15-22 CE Drive (X11,12 parts)

Check X15 pin 13 for combinations of inputs $\overline{\text{BOOT}}$ and R/W;

$\overline{\text{BOOT}}$	R/W	X15 pin 13
0	0	0
0	1	1
1	0	0 if F strapped to D, else 1.
1	1	0

h) ROM Select (X9 part)

Set $\overline{\text{BOOT}}$ input to '0' (low), R/W input to 1 (so X13 pin 4 is '0'). With A5 (con pin 26) set to '0', check that X3 pin 15 is 'o',X4 pin 15 is '1'.
With A5 set to '1', check that X3 pin 15 is '1', X4 pin 15 is '0'.
With R/W input set to '0', check that X3 pin 15 and X4 pin 15 are both '1' for A5 = 'o' and also for A5 = '1'.

i) Data Input Buffers (X7,8 & 12, parts)

Set R/W (con. pin 4) to '0', A10 to '1', A11 to '0', A12-15 to '1', then check that X12 pin 8 is '0'. Also check that whatever input is applied to the data bus lines D0 - D7, the same pattern is present on lines BD0-BD7 (X7 pin 17 etc.)

j) Data Output Buffers (X7,8,11,12 ,parts)

First check that these buffer outputs can be set to the high impedance state by applying '0' to board E input (which should cause X11 pin 6 to go to '1') then check each data bus line D0-D7 as follows;

- temporarily connect data bus line to 0V via 1kohm resistor, the voltage across the resistor should be less than 0.1V.
- remove the resistor from 0V and temporarily connect it to +5V. The voltage across the resistor should be less than 0.1V.

Then set E,R/W A10-A15 and $\overline{\text{BOOT}}$ board inputs to '1'; X11 pin 6 should go to '0'. Check that the output of each buffer (X7 pins 3,5 etc.) is the same as its input (X7 pins 2,4, etc.) Note that the signal present on the buffer input is coming from memory, which will be holding a random pattern; by trying different patterns on the board address inputs A0-A9 it should be possible to find a memory cell containing a '1' and another containing a '0', thus testing both states of the output buffer. Alternatively, if the memory IC's X15-22 are fitted in sockets then they can be removed and '1' or '0' signals applied to the data output buffer inputs (X7 pins 2,4 etc.)

Finally, change A11 input to '0', and apply '0' to board A2,3 inputs. Check that X11 pin 12 and X11 pin 6 are both '0'.

k) Memory IC's X15-22

First check the correct operation of the top 256 words by loading and reading test patterns with the system control panel switches. Once it has been established that these memory locations are working then the remainder of memory is best checked via the system keyboard, using a monitor such as BUG-1.

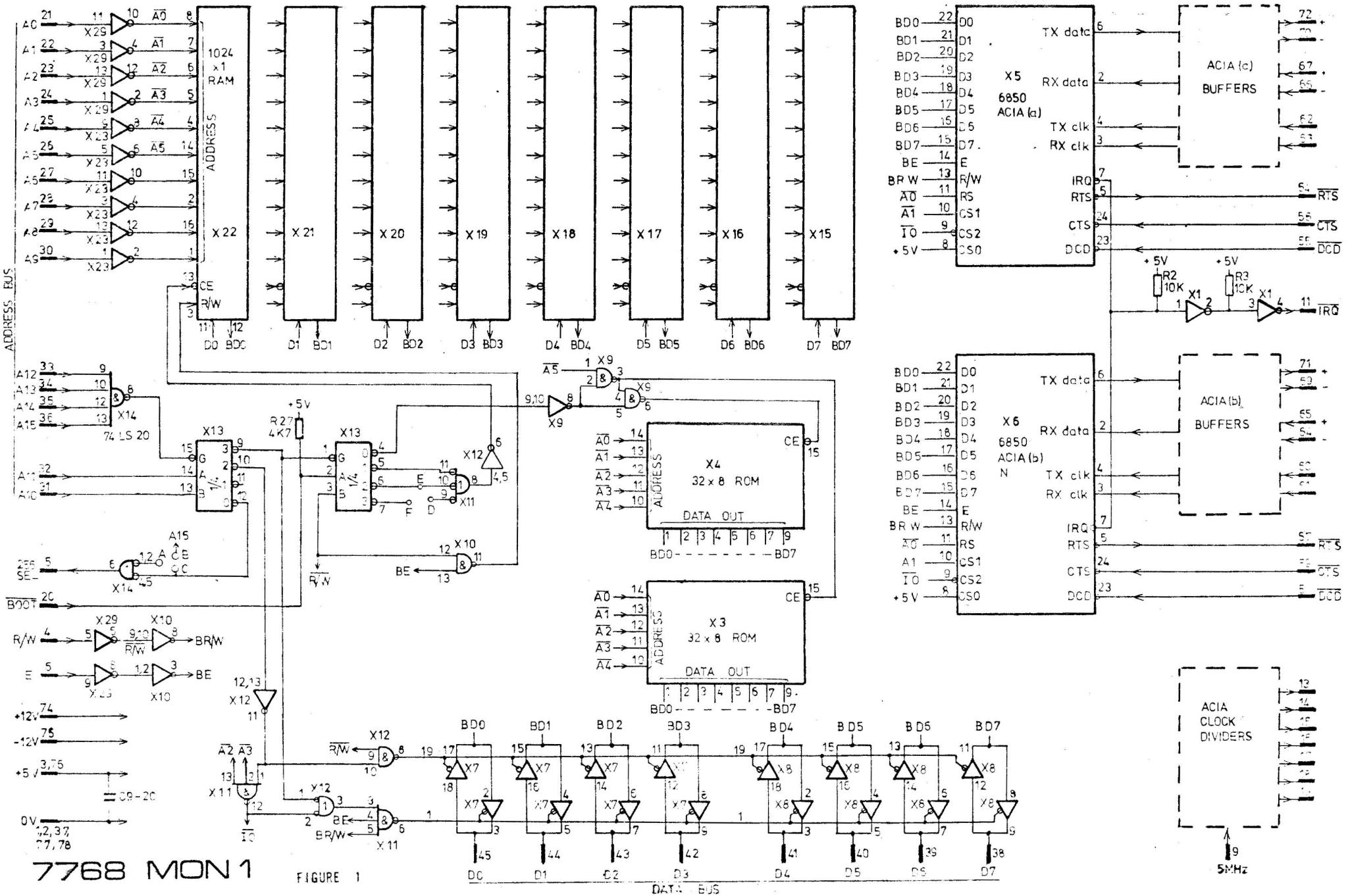
l) ACIA Clock Dividers (X24-27)

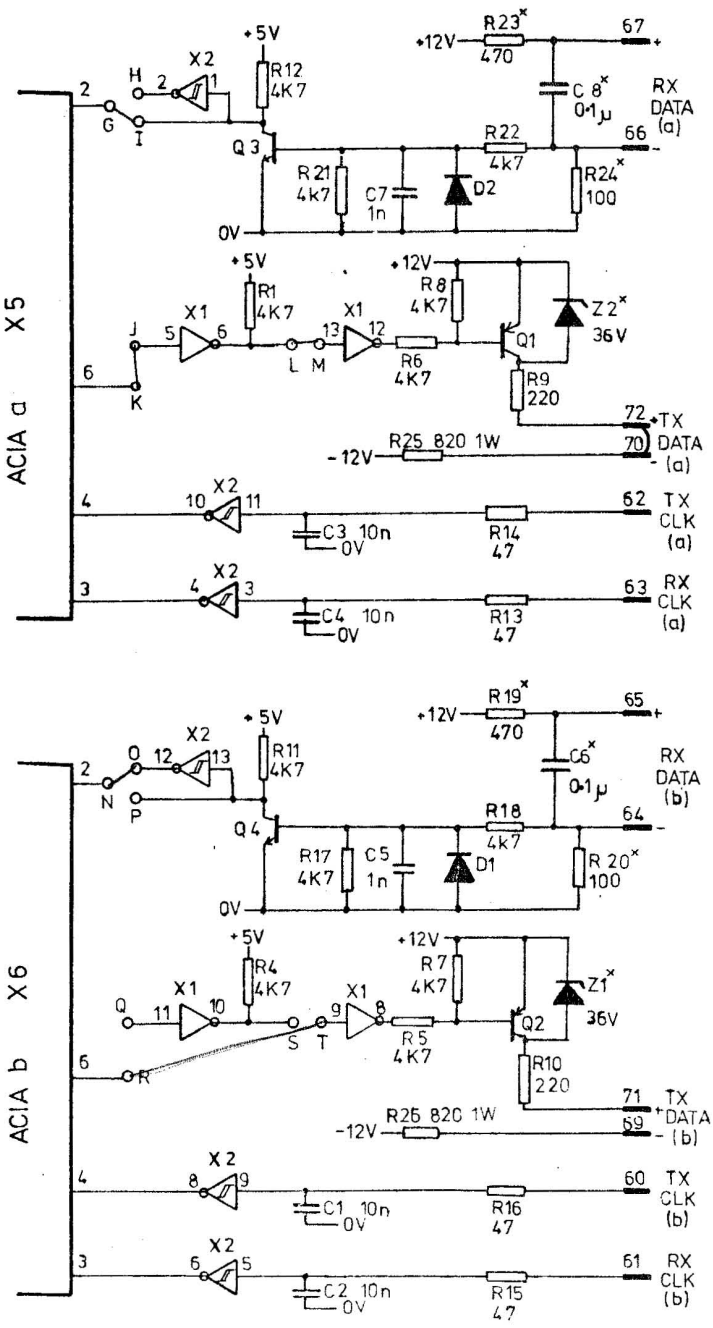
If a 'scope is not available, then applying a TTL compatible audio frequency signal to the '5MHz' input will allow you to check the operation of the dividers by monitoring their outputs with an audio amplifier or headphones.

m) ACIA Clock Buffers

Apply '0' to TX CLK(a) input (con. pin 62) and check that X2 pin 10 is '1'.
Apply '1' to TX CLK(a), check that X2 pin 10 is '0'.

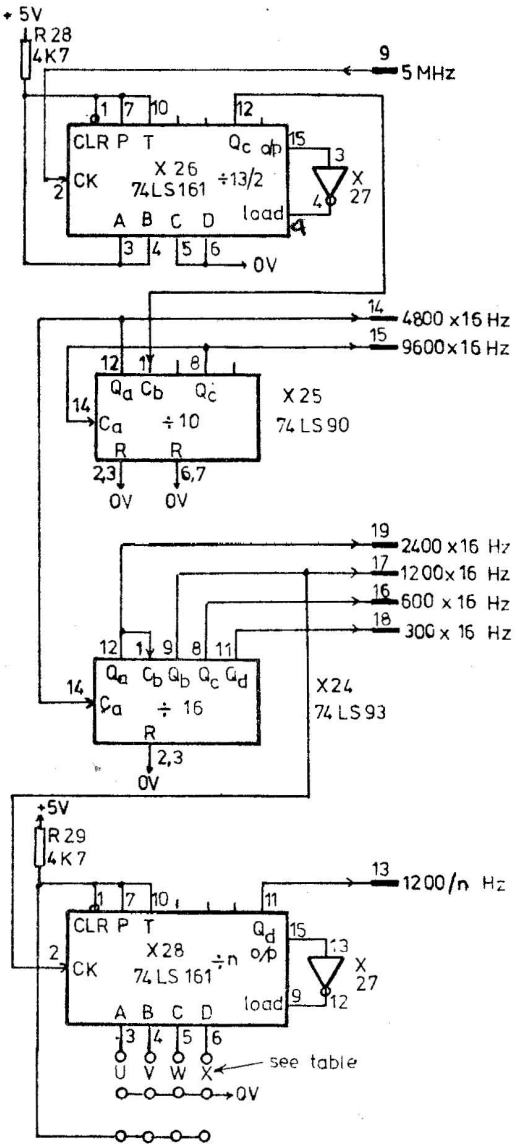
Repeat for other RX & TX CLK input buffers.


ADDRESS BUS
1024 x1 RAM
ADDRESS
X22
X21
X20
X19
X18
X17
X16
X15
CE
R/W
D0 BD0
D1 BD1
D2 BD2
D3 BD3
D4 BD4
D5 BD5
D6 BD6
D7 BD7
X5
6850
ACIA (a)
TX data
RX data
TX clk
RX clk
IRQ
RTS
CTS
DCD
ACIA (a)
BUFFERS
X6
6850
ACIA (b)
ACIA (b)
BUFFERS
+5V
R2 10K
R3 10K
X1
X4
32 x 8 ROM
X3
32 x 8 ROM
DATA OUT
74 LS 20
X14
X13
X12
X11
X10
X9
BOOT
R/W
BR/W
BE
+12V
-12V
+5V
0V
ACIA CLOCK DIVIDERS
5MHz
DATA BUS
7768 MON 1
FIGURE 1



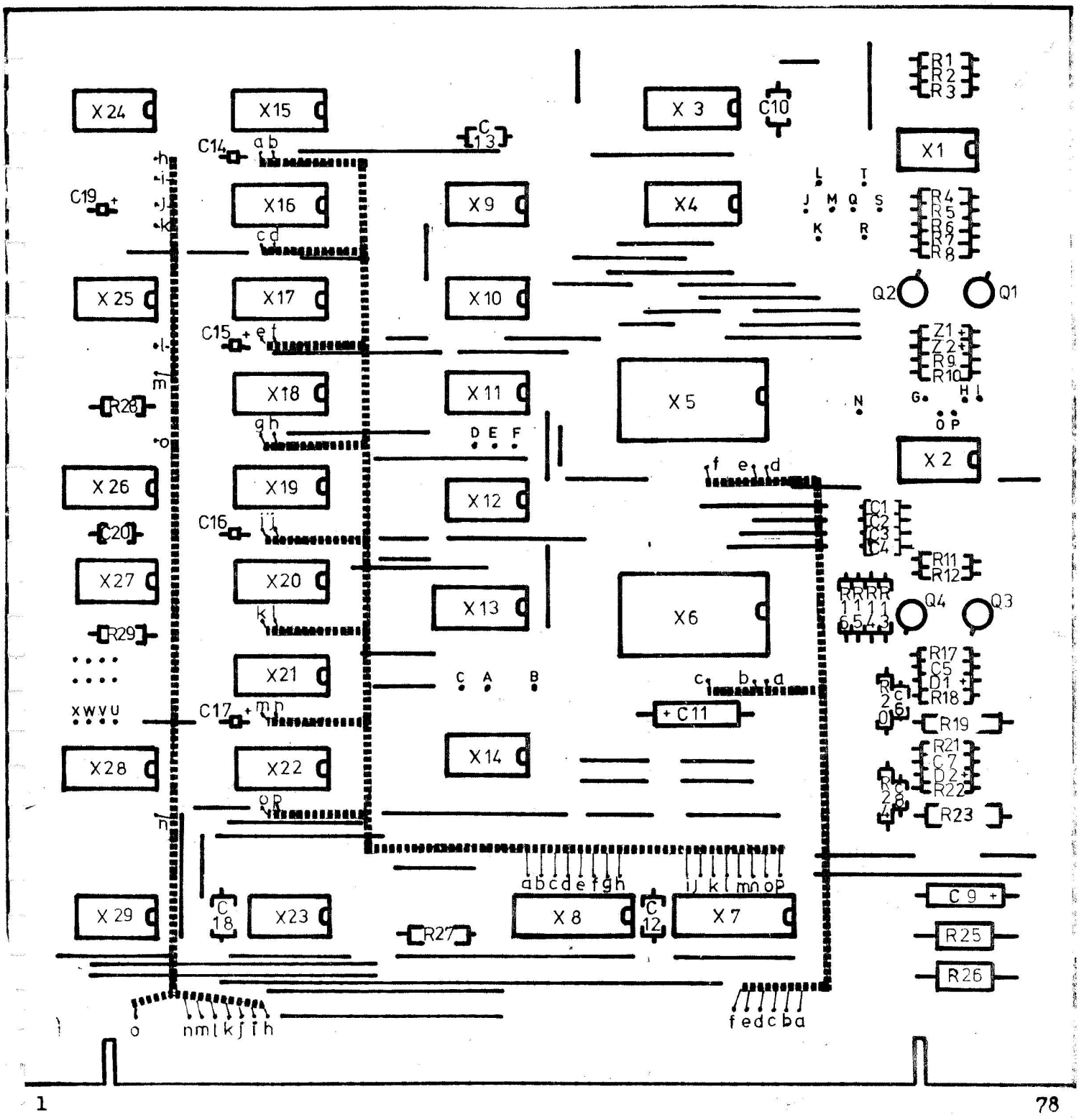
ACIA BUFFERS

* = 20mA only



ACIA CLOCK DIVIDERS

FIGURE 2



7768 MON 1 BOARD

Component side view

Notes;

——————— Strap on component side of board

yyyyyyyyyyyyyyyyyyyy Ribbon cable

See Fig 4 for connections to points labelled with capital letters (A,B,C .... X)

Fig 3

7768 MON 1 ; OPTIONAL STRAPS

Write Protection

Strap D-E to write protect 1k RAM, else strap D-F

CPU Board Memory Addressing

Strap A-B to make 256 word RAM on CPU board respond to low addresses (i.e. if system consists of CPU & MON 1 boards only). Else strap A-C

ACIA (a) Interfaces

For 20 mA Current Loop strap G-H, K-M

For RS232C strap G-I,J-K,L-M, & connect edge conn. pins 70 & 72.

For TTL omit R23,R24,C8 & R9,R25,Q1,Z2 , strap G-I,J-K,L-M, replace R6 by a strap & connect X1 pin 12 to edge conn pin 72.

ACIA (b) Interfaces

For 20 mA strap N-O,R-T

For RS232C strap N-P,R-Q,S-T & connect edge conn. pins 69,71

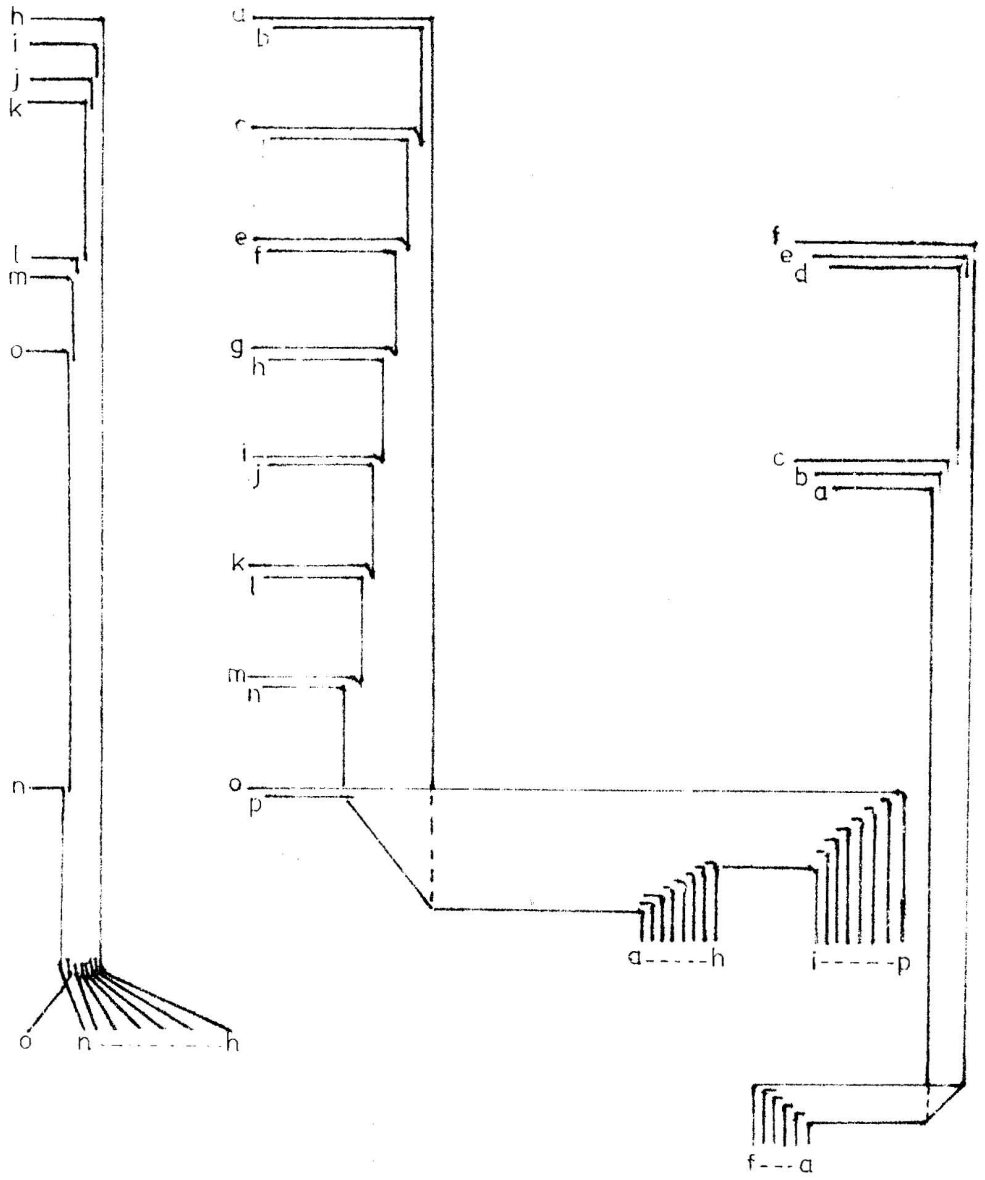
For TTL omit R19,R20,C6 & R10,R26,Q2,Z1, strap N-P,R-Q,S-T, replace R5 by a strap & connect X1 pin 8 to edge conn pin 71.

X28 i/p Strapping;

U	V	W	X	n	Nominal output frequency
0V	0V	0V	0V	16	75 x 16 Hz
R29	0V	0V	0V	15	80 x 16 Hz
0V	R29	0V	0V	14	86 x 16 Hz
R29	R29	0V	0V	13	92 x 16 Hz
0V	0V	R29	0V	12	100 x 16 Hz
R29	0V	R29	0V	11	110 x 16 Hz
0V	R29	R29	0V	10	120 x 16 Hz
R29	R29	R29	0V	9	133 x 16 Hz
0V	0V	0V	R29	8	150 x 16 Hz
R29	0V	0V	R29	7	171 x 16 Hz
0V	R29	0V	R29	6	200 x 16 Hz
R29	R29	0V	R29	5	240 x 16 Hz
0V	0V	R29	R29	4	300 x 16 Hz
R29	0V	R29	R29	3	400 x 16 Hz
0V	R29	R29	R29	2	600 x 16 Hz

Note; for 50 baud use 200 x 16 Hz as ACIA clocks, and program ACIA for ÷ 64 working.

Fig 4



7768 MON 1 BOARD

Arrangement of ribbon cables - not to scale.

Use solid cored type e.g. Doram's miniature cable No 357-491

Fig 5

7768 MON 1 BOARD

Components

Logic

X1	7406	; hex open collector inverter.	X13	74LS139	; dual 1/4 decoder
X2	74LS14	; hex schmitt inverter	X14	74LS20	; dual 4 i/p nand
X3	PROM 2	; see text	X15-22	2102	; 1024 x 1 RAM
X4	PROM 1	; " "		or 2102-1	; for 1.2uS CPU
X5	6850	; ACIA (a)	X23	74LS04	; hex inverter
X6	6850	; ACIA (b)	X24	74LS93	; divide by 16
X7,8	81LS97	; octal tri-state buffer	X25	74LS90	; divide by 10
X9,10	74LS00	; quad 2 i/p nand	X26	74LS161	; prog. counter
X11	74LS10	; triple 3 i/p nand	X27	74LS04	; hex inverter
X12	74LS00	; quad 2 i/p nand	X28	74LS161	; prog. counter
			X29	74LS04	; hex inverter

Transistors

Q1,2	2N2907	Q3,4	BC107

Diodes

D1,2	1N4148	Q3,4	36V 400mW zener

Resistors

All are miniature types except R19,23,25,26

R1	4k7	R19	470R ½W
R2,3	10k	R20	100R
R4,5,6,7,8	4k7	R21,22	4k7
R9,10	220R	R23	470R ½W
R11,12	4k7	R24	100R
R13,14,15,16	47R	R25,26	820R 1W
R17,18	4k7	R27,28,29	4k7

Capacitors

C1,2,3,4	10nF ceramic	C12,13	0.1uF
C5	1nF ceramic	C14	0.047uF small ceramic
C6	0.1uF	C15	6.8uF 10V tant. bead
C7	1nF ceramic	C16	0.047uF small ceramic
C8	0.1uF	C17	6.8uF 10V tant. bead
C9	33uF 10V tant. bead	C18	0.1uF
C10	0.1uF	C19	6.8uF 10V tant. bead
C11	33uF 10V tant. bead	C20	0.1uF

Misc.

16 pin DIL sockets (2 needed for X3,4 - a further 8 may be used for X15-22)

24 pin DIL sockets (for X5,6)

Ribbon cable - miniature solid cored type

7768 MON 1 printed circuit board (available from Newbear)

Edge connector - 76 way + 2 polarising positions (78 positions total), 0.1" pitch single sided.

Single pole on/off 'Boot' switch (not mounted on board)

Note;

Complete component list shown, but individual constructors may sub-equip depending on number of PROM's & ACIA's fitted and choice or TTL or 20mA or RS232 interface(s).

Fig 6

7768 MON 1 MEMORY MAP

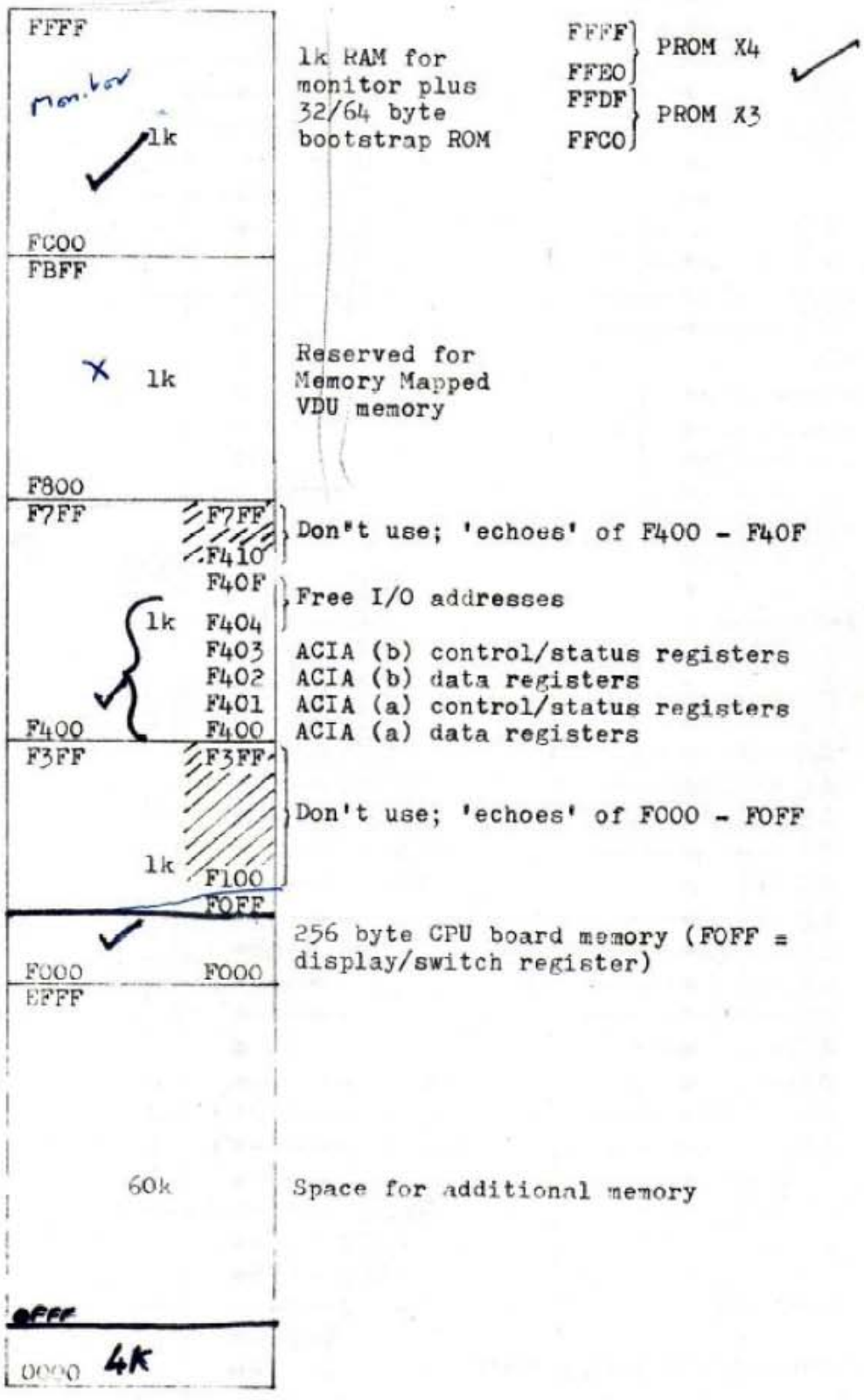
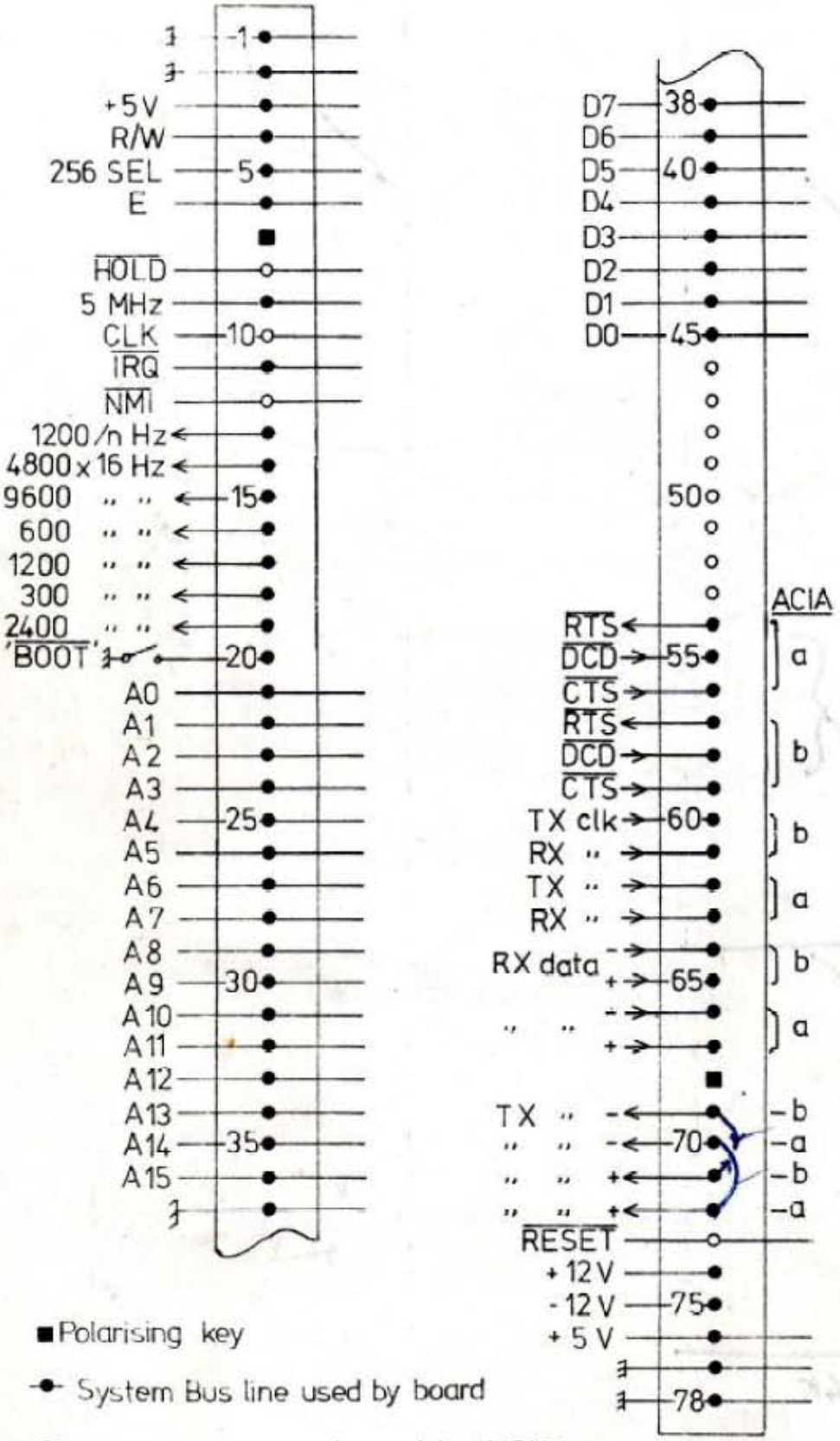


Fig 7

7768 MON 1 : CONNECTOR WIRING



■ Polarising key

● System Bus line used by board

○ " " " not used by MON 1

● Other connection to board

REAR VIEW

Fig 8

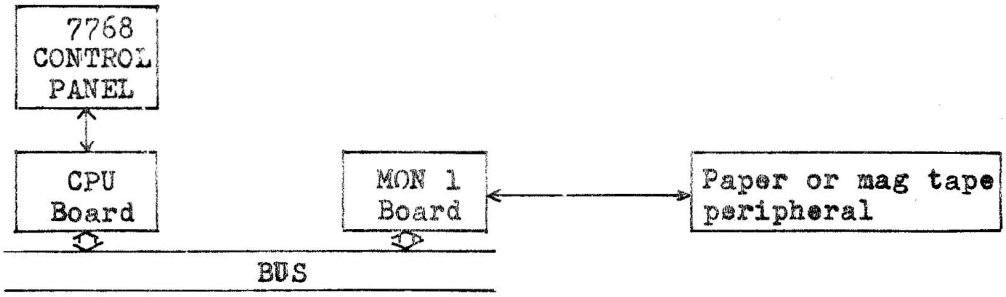
69-71

white 71 — white 9
grey 64 — black 10
brown 65 red 14
blue 64 black 12

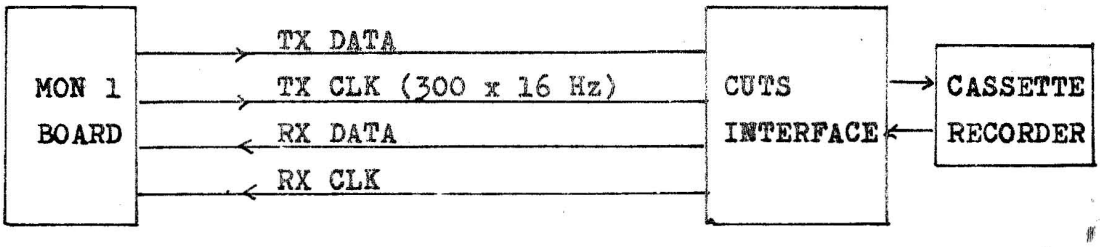
7768 MON 1 BOARD

Appendix 1 - Configuration Guide

The minimum system using the MON 1 board requires the CPU board with its control panel, the MON 1 board itself, and a paper or mag tape peripheral for back-up storage;



The connection between the MON 1 board and the peripheral must be serial 8 bit asynchronous. If a cassette tape recorder is fitted then it is recommended that a 'Kansas City' (CUTS) compatible interface is used , these generally have TTL or RS232 level ports, either of which can be handled by the MON 1 board.

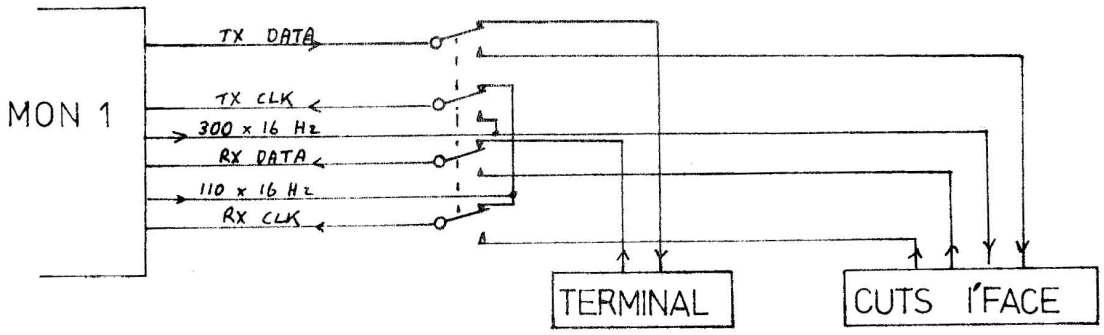


Parallel paper tape readers & punches will need parallel - serial conversion circuits added; a UART and a few TTL or CMOS chips will do the trick easily).

Although a VDU/Keyboard interface card will be introduced for the 7768, the user may wish to connect an external VDU or hard copy terminal. The monitor program given in Appendix 5 in fact requires the use of an external terminal, a separate monitor will be issued for use with the 'inboard' 7768 VDU/Keyboard interface card. Provided that the terminal has a serial interface then there should be no problem in connecting it to either ACIA. The back-up store (mag or paper tape) should always be connected to ACIA(a), however, for proper operation of the Bootstrap Loader.

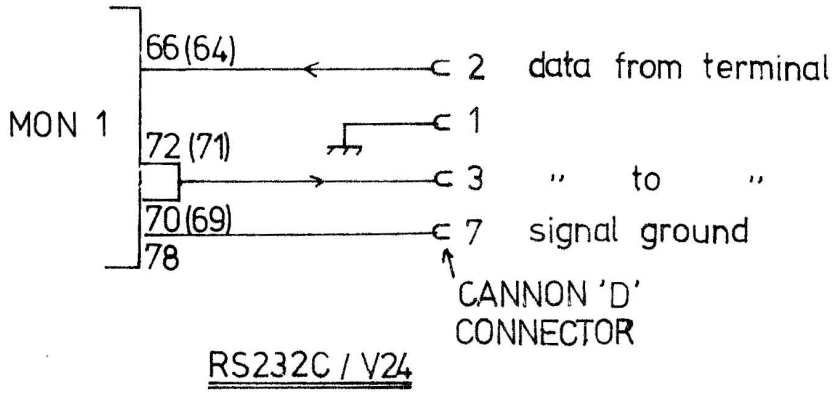
If a terminal with inbuilt paper (or magnetic) tape handling facilities is used (e.g. an ASR33) this can fulfil the functions of general purpose terminal as well as back-up store by connecting it to ACIA(a).

However, if your terminal doesn't have this facility, then it may still be connected to ACIA(a), but via a switch which allows the separate back up store to be connected for program loading (& dumping). This switch may also change between the clock frequencies required for the back up store and the terminal. Thus, for a CUTS interface and a 110 baud terminal;

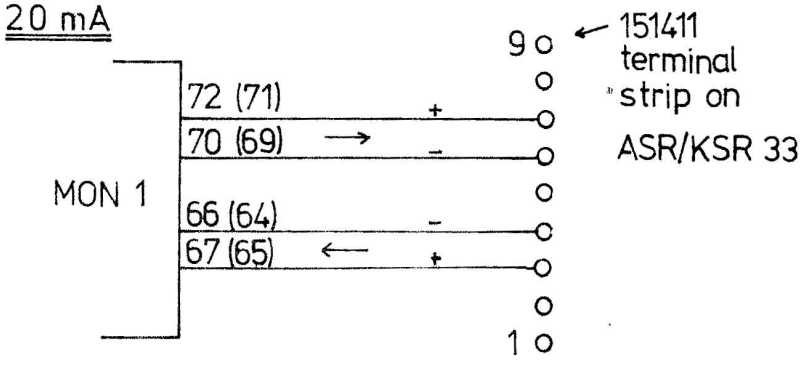


Alternatively, the terminal may be connected to ACIA(b),and the back up store to ACIA(a); or two back up stores may be connected - one to each ACIA - for editing & copying files which are too large to be stored entirely within the computer's own memory. The choice is up to the user, and will depend upon the peripheral equipment he has and the uses to which he intends to put the system.

Although not essential, it is recommended that the RS232C(V24) interface and its associated 25 way connectors are adopted as standard even if this means adding interface level converters to the peripherals. This will help experiments and quick trials of different equipments for exhibitions and demonstrations. The appropriate connections are;



However, for those lucky enough to have an ASR33 or KSR33 Teletype, and who wish to use a '20mA Current Loop' interface, then the following connections should be used;



7768 MON 1 BOARD

Appendix 2 - User Tips

$\overline{\text{CTS}}$, $\overline{\text{RTS}}$ & $\overline{\text{DCD}}$

These ACIA connections are intended for use with a data modem, but can be used to provide two auxilliary interrupting inputs ($\overline{\text{CTS}}$ & $\overline{\text{DCD}}$) and an output ($\overline{\text{RTS}}$) for, say, controlling a cassette interface.

If not used, then $\overline{\text{CTS}}$ & $\overline{\text{DCD}}$ inputs must be connected to OV, otherwise they are likely to pick up stray signals and cause much confusion.

+ & - 12V Supplies

Although strictly speaking these are required to generate 'within spec' RS232 or 20mA Current Loop interface levels, it is often possible to work successfully with a particular peripheral using only +5V and -12V or even +5V only; '+12V' being connected to the +5V rail and '-12V' to OV. Some experimentation is required, and the data buffer circuit resistor values will probably need altering.

7768 MON 1 BOARD

Appendix 3: ACIA Programming

Addressing

Each ACIA contains two pairs of user-accessible registers; one pair (Transmit Data Register & Control Register) being 'Write Only', the other pair (Receive Data Register & Status Register) being 'Read Only'. Thus only two addresses are used by each ACIA;

Address	MPU 'Read' Accesses	MPU 'Write' Accesses
KANSAS F400	ACIA(a) Rec Data Reg	ACIA(a) Trans Data Reg
F401	ACIA(a) Status Reg	ACIA(a) Control Reg
F402	KBD ACIA(b) Rec Data Reg	VDU ACIA(b) Trans Data Reg
VDU F403	ACIA(b) Status Reg	ACIA(b) Control Reg

Because the same address accesses two different registers, depending on whether a Read or a Write operation is being performed, then those instructions which operate on the contents of a memory location (e.g. ASL) must not be used with ACIA addresses as these instructions read information, modify it, then write it back to the same address.

Transmit Data Register

If the transmit half of the ACIA is idle (not transmitting a character) then the act of writing a character into the Transmit Data Register will start the ACIA transmitting that character in serial, asynchronous, form;

start bit — B0 - - - - - - - B7 — optional parity bit — one or two stop bits

The exact format (number of data & stop bits & parity) will depend upon the state set into the Control Register.

Since the ACIA is 'double buffered', a new character can be loaded into the Transmit Data Register while the previous character is still being transmitted. Examination of the contents of the Status Register will tell the program when the Transmit Data Register is empty and ready to accept another character.

Receive Data Register

Data received at the RX Data input of the ACIA is converted to parallel form, the start, stop & parity bits stripped, then the received character is transferred to the Receive Data Register and the Status Register contents changed to show that a new character is available.

When this received character is read from the Receive Data Register, the Status Register 'Received Data Register Full' indication is removed and the ACIA waits for the next character to be received.

As the receive half of the ACIA is also double buffered, a character can be read from the Receive Data Register while a new character is being received.

Control Register

Before the ACIA is used, it must be set up by writing suitable data into the Control Register.

First, the number 03 (Hex) should be loaded in order to reset the ACIA's

internal registers. Note that the reset kills any characters that the ACIA may have been transmitting at the time. When this has been done, a second byte should be written into the Control Register to set up the required mode. This byte should be composed as follows;

D0,1 ; Establish Rec & Trans data baud rate as a sub-multiple of the ACIA Clock input frequencies;

D0	D1	
0	0	Baud Rate = Clock Frequency
1	0	" " " " " divided by 16
0	1	" " " " " " " 64
1	1	Used for ACIA Reset

D2,3,4 ; Establishes number of data, stop bits & parity;

D2	D3	D4	No of Data bits	Parity	No of Stop bits
0	0	0	7	Even	2
1	0	0	7	Odd	2
0	1	0	7	Even	1
1	1	0	7	Odd	1
0	0	1	8	-	2
1	0	1	8	-	1
0	1	1	8	Even	1
1	1	1	8	Odd	1

D5,6 ; Establish state of $\overline{\text{RTS}}$ output & Transmit Interrupt Enable;

D5	D6	$\overline{\text{RTS}}$ output	Trans Interrupt	
0	0	Low	Disabled	
1	0	Low	Enabled	
0	1	High	Disabled	
1	1	Low	Disabled	Transmits 'Break'

D7 ; When set to '1' enables Receive Interrupt, and IRQ will be caused by any of;

- Receive Data Register Full.
- Receiver Overrun.
- $\overline{\text{DCD}}$ input going from low to high.

Status Register

This regiater provides the program with information as to the current status of the Transmit & Receive halves of the ACIA;

D0 ; '1' when Receive Data Reg full (character received).

D1 ; '1' when Transmit Data Reg empty (ready to accept a new character)

D2 ; '1' when $\overline{\text{DCD}}$ (Data Carrier Detect) ACIA input is high.

D3 ; '1' when $\overline{\text{CTS}}$ (Clear To Send) ACIA input is high.

D4 ; '1' when a framing error has occurred in the process of converting the received data from serial to parallel form, i.e. the 'stop' bit was missing.

D5 ; '1' when a Receiver Overrun condition has occurred due to a new character being received before the previous one had been read from the Receive Data Register.

D6 ; '1' when a Parity Error has been detected in the received data.

D7 1 '1' whenever the ACIA $\overline{\text{IRQ}}$ (Interrupt Request) output is low.

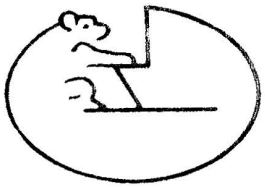
x x x x | x x x x
7 6 5 4 | 3 2 1 0

Appendix 4 - 32 byte Bootstrap & Dump Programs

A minimum bootstrap ROM program is given on the next page. This merely loads the first 1024 bytes of data to come in via ACIA(a) into the RAM without any form of checking, but this has proved sufficient in practice. To run, momentarily operate the Reset switch with the BOOT switch set to 'Bootstrap'. When 1024 bytes have been loaded, the system RUN lamp will go out. The BOOT switch is then set to the normal position and the system Reset switch momentarily operated to start running the monitor that has been loaded. Since the program doesn't look for a header, but simply stores whatever arrives at the receive port (a), care must be taken when loading from a paper tape reader to position the tape correctly; the system can't distinguish between an 'all 0's' byte and blank header tape.

The DUMP program given can be used to generate tapes containing a monitor program for subsequent loading (Bootstrapping). One trick worth noting is that as DUMP sends out bytes from memory starting at the location determined by the LDX instruction (normally location FC00), and carries on until it reaches FFFF, and as the bootstrap program just takes the first 1024 bytes received, one can use a monitor (such as BUG 1) to put a new monitor program into another area of memory, then by altering the LDX instruction in DUMP, create a tape containing the new monitor.

The 32 byte bootstrap program shown here is available already 'burnt into' a PROM from Newbear, however anyone who wishes to generate his own bootstrap program - perhaps to take advantage of the maximum 64 bytes available - can use virtually any of the common 32 x 8 Tri-state TTL PROM's available. Care should be taken, however, as the address inputs to the PROM's on this board are inverted and out of sequence relative to the connections assumed in the manufacturers data sheets and PROM programmers. Page 4 of this appendix shows, as an example, the 32 byte bootstrap program translated into the form required for PROM programming.



BEAR MICROCOMPUTER SYSTEMS

24 College Road, Maidenhead, Berks, SL6 6BN

HEXADECIMAL CODING FORM

PROGRAM BOOT	VERSION 1	AUTHOR MRL	DATE 3-2-78	PAGE 1 OF 1

ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
FFE0	8603	START	LDA A #3	RESET ACIA (a)
FFE2	B7F401		STA A F401	
FFE5	8611		LDA A #11	SET UP ACIA (a) CONTROL REG.
FFE7	B7F401		STA A F401	
FFEA	CEFC00		LDX #FC00	POINT TO START OF 1K RAM
FFED	35		TXS	POINT STACK SOMEWHERE !
FFEE	B6F401	LOOP	LDA A F401	ACIA (a) RX REG FULL ?
FFF1	8501		BIT A #1	IF NOT THEN
FFF3	27F9		BEQ LOOP	LOOK AGAIN
FFF5	B6F400		LDA A F400	GET DATA BYTE
FFF8	A700		STA A 0,X	STORE IT
FFFA	08		INX	POINT TO NEXT RAM LOC^N
FFFB	26F1		BNE LOOP	PASSED LOC FFFF ?
FFFD	3E		WAI	THEN KILL 'RUN' LAMP
FFFE	FFE0		START	RESET VECTOR
				THIS PROGRAM PUTS FIRST
				1024 BYTES TO BE RECEIVED
				BY ACIA (a) INTO TOP 1K
				OF MEMORY SPACE .
				TO START : HIT 'RESET'
				WITH SYSTEM SET TO 'BOOT'.
				WHEN FINISHED : 'RUN'
				LAMP GOES OUT.

BEAR MICROCOMPUTER SYSTEMS
24 College Road, Maidenhead, Berks, SL6 6BN

HEXADECIMAL CODING FORM

PROGRAM DUMP	VERSION 1	AUTHOR M.R.L	DATE 3-2-78	PAGE 1 OF 1

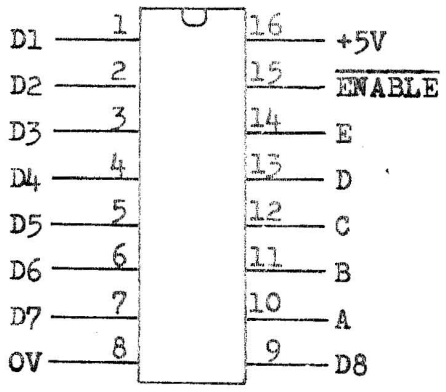
ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
0000	8603	START	LDA A #3	RESET ACIA(a)
0002	B7F401		STA A F401	
0005	8611		LDA A #11	SET UP ACIA(a) CONTROL REG
0007	B7F401		STA A F401	
000A	CEFC00		LDX #FC00	POINT INDEX REG AT START OF
				TOP 1K OF MEMORY SPACE
000D	B6F401	LOOP	LDA A F401	ACIA(a) TX BUSY?
0010	8502		BIT A #2	IF SO THEN
0012	27F9		BEQ LOOP	LOOK AGAIN
0014	A600		LDA A 0,X	GET BYTE FROM MEMORY
0016	B7F400		STA A F400	SEND IT
0019	08		INX	POINT INDEX REG AT NEXT
→	8CFFFF		CMP X #FFFF	BYTE OF MEMORY
001D	26F1 (EE)		BNE LOOP	FINISHED?
001F	3E		WAI	HALT CPU & KILL 'RUN' LAMP
				THIS PROGRAM TRANSMITS TOP
				1024 BYTES OF MEMORY VIA
				ACIA (a)
				START @ LOC 0000.
				ENDS WITH 'RUN' LAMP GOING
				OUT.
				STACK POINTER REGISTER
				MUST BE SET UP BEFORE
				RUNNING THIS PROGRAM, AS
				'WAI' INSTRUCTION WILL USE
				THE STACK.

ADDRESS						DATA (Binary)							
As seen by CPU (Hex)	As seen by PROM Programmer (binary)					D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1
	E	D	C	B	A								
FFFF	0	0	0	0	0	1	1	1	0	0	0	0	0
EF	0	0	0	0	1	1	1	1	1	0	1	0	0
F7	0	0	0	1	0	0	0	0	0	0	0	0	0
E7	0	0	0	1	1	1	0	1	1	0	1	1	1
FB	0	0	1	0	0	0	0	1	0	0	1	1	0
EB	0	0	1	0	1	1	1	1	1	1	1	0	0
F3	0	0	1	1	0	0	0	1	0	0	1	1	1
E3	0	0	1	1	1	1	1	1	1	0	1	0	0
FD	0	1	0	0	0	0	0	1	1	1	1	1	0
ED	0	1	0	0	1	0	0	1	1	0	1	0	1
F5	0	1	0	1	0	1	0	1	1	0	1	1	0
E5	0	1	0	1	1	1	0	0	0	0	1	1	0
F9	0	1	1	0	0	0	0	0	0	0	0	0	0
E9	0	1	1	0	1	0	0	0	0	0	0	0	1
F1	0	1	1	1	0	1	0	0	0	0	1	0	1
E1	0	1	1	1	1	0	0	0	0	0	0	1	1
FE	1	0	0	0	0	1	1	1	1	1	1	1	1
EE	1	0	0	0	1	1	0	1	1	0	1	1	0
F6	1	0	0	1	0	1	1	1	1	0	1	0	0
E6	1	0	0	1	1	0	0	0	1	0	0	0	1
FA	1	0	1	0	0	0	0	0	0	1	0	0	0
EA	1	0	1	0	1	1	1	0	0	1	1	1	0
F2	1	0	1	1	0	0	0	0	0	0	0	0	1
E2	1	0	1	1	1	1	0	1	1	0	1	1	1
FC	1	1	0	0	0	1	1	1	1	0	0	0	1
EC	1	1	0	0	1	0	0	0	0	0	0	0	0
F4	1	1	0	1	0	1	1	1	1	1	0	0	1
E4	1	1	0	1	1	0	0	0	0	0	0	0	1
F8	1	1	1	0	0	1	0	1	0	0	1	1	1
E8	1	1	1	0	1	1	1	1	1	0	1	0	0
F0	1	1	1	1	0	0	0	0	0	0	0	0	1
E0	1	1	1	1	1	1	0	0	0	0	1	1	0

Tri-state
32x8 PROM

'1'=high
'0'=low

Dn = outputs
A-E= address



7768 MON 1 BOARD

Appendix 5 ; 7768 BUG 1

Purpose

7768 BUG 1 is a minimal system monitor which allows the user to enter and run programs via a Teletype or similar asynchronous data terminal.

It occupies only the top 256 words of memory so that it may be initially entered via the system control panel switches. Once entered into memory it can be dumped onto paper tape or cassette tape for subsequent re-loading by the MON 1 board Bootstrap Load facility.

Hardware

An ASCII send/receive terminal must be connected to ACIA (a) port, or to ACIA (b) port if locations FF84,FF92 are changed from 01 to 03, and locations FF8A, FF99 are changed from 00 to 02.

Use

When 7768 BUG 1 is loaded, operation of the system Reset button will start BUG 1 running, and cause the terminal to print a '*' at the beginning of the next line. The system will then wait for the user to enter one of the following single letter commands;

A (Alter memory location)
E (Examine memory location)
G (Go to start of user program and run it)
R (Print contents of 6800 MPU registers resulting from run of user program)
C (Continue to run user program after SWI)

The user may enter a program (into an area of memory other than that used by BUG 1) using the A & E commands, then start it running with the G command. If the processor encounters a SWI (Software Interrupt) instruction in the user program, then it will save the 6800 MPU registers on the stack and go to BUG L, printing a '*'. The saved MPU registers may then be printed out (R), altered (A), then the user program Continued at the instruction following the SWI, or started at a different location.

RAM

7768 BUG 1 uses RAM locations F0F5 to F0FE for temporary storage, and also has its own stack extending below F0F4. The user program may either use the BUG 1 stack, or set up its own stack in a separate area of RAM.

E (Examine) Command

When BUG 1 is waiting for a command ('*' printed), type 'E'. The monitor will respond by printing a space, then the user should type the address (in hexadecimal) of the memory location to be examined.The monitor will print a space followed by the (hex) contents of the memory location specified. e.g.;

*E FF00 8E (user input underlined)

The user may then either;

- Type a CR (carr return) to terminate the Enter command.

or - Type a space, which will cause the monitor to print a space followed by the contents of the next memory location;

*E FF00 8E_F0_F3_86_03

A (Alter) Command

When 'A' is typed after the monitor's '*', the monitor will print a space, then the user should type the address (in hexadecimal) of the memory location who's contents are to be changed. The monitor will then print a further space, after which the user should type (in hex) the new contents of that location. e.g.;

*A 0000 12

The user may then either;

- Type a CR to terminate the Alter command.

or - Type two hex digits to be stored in the next memory location;

*A 0000 123456789ABC↵ (↵ = CR typed by user)
*E 0000 12_34_56_78_9A_BC↵
*

G (Go) Command

When the user has loaded a program (using A & E commands) then to start it running type G. The monitor will respond by printing a space. Then type the program starting address. The system will then run the user program until it encounters a SWI instruction or the Reset button is operated.e.g.;

*G 0000

C (Continue) Command

When a running user program encounters a SWI command it will put the contents of the CC,A,B,X registers and the current Program Counter (= address of location following the SWI instruction) onto the stack, then jump to the Monitor program.
If 'C' is typed when in Monitor mode then the old values of the MPU registers are retrieved from the stack,and the user program restarted at the address following the SWI instruction.eg;

*C

R (Register Print) Command

Typing 'R' when in command mode causes a print out of the contents of the MPU registers that were saved on the stack by the last SWI instruction encountered in the user program, and also the content of the stack pointer on entry to the Monitor following the SWI.e.g.;

```
*R F0 1E 55 BE00 0010 0079
*  |  |  |  |    |    |
   CC B  A  X    P    S
```

In the above example 'P' is the address of the location in the user program immediately after the SWI instruction (and is therefore the address at which the program will be restarted by a 'C' command).

The values of CC,B,A,X & P are obtained from the stack immediately above location 0079 (current position of S; stack pointer), thus;

*E 007A F0_1E_55_BE00_0100↵

If altered;

*A 007C FF↵

*R F0 1E FF BE00 0010 0079

Then the altered values will be loaded into the MPU when the next 'C' or 'G' command is given.

Example of BUG 1 use

To load & run the following program;

```
Loc  Instr
0000 0F       START  SEI            ; set interrupt mask
 01  8E 00 80        LDS     0080   ; define user prog stack
 04  86 AA    LOOP   LDA A   AA     ; display '1010 1010'
 06  B7 F0 FF        STA A  DISPLAY
 09  3F               SWI            ; return to monitor
 0A  86 55            LDA A   55     ; display '0101 0101'
 0C  B7 F0 FF        STA A  DISPLAY
 0F  3F               SWI            ; return to monitor
 10  20 F2           BRA    LOOP
```

*A 0000 0F8E008086AAB7F0FF3F8655B7F0FF3F20F2↵

*E 0000 0F_8E_00_80_86_AA_B7_F0_FF_3F_86_55_B7_F0_FF_3F_20_F2↵

*G 0000 display should change to '1010 1010'

*R F8 1E AA BE00 000A 0079

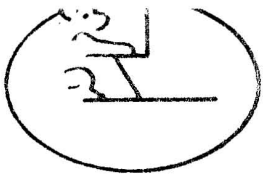
*C display should change to '0101 0101'

*R F0 1E 55 BE00 0010 0079

*C display should change to '1010 1010'

*R F8 1E AA BE00 000A 0079

etc.



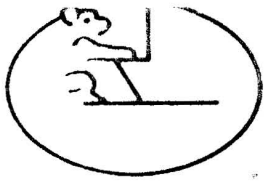
BEAR MICROCOMPUTER SYSTEMS

24 College Road, Maidenhead, Berks, SL6 6BN

HEXADECIMAL CODING FORM

PROGRAM 7768 BUG 1	VERSION 01	AUTHOR M R L	DATE 3-2-78	PAGE 1 OF 4

ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
		CTRL.A	EQU $F401	ACIA (a) CONTROL/STATUS REG ADDRESS
		CTRL.B	EQU $F403	" (b) " " " "
		DATA.A	EQU $F400	ACIA (a) DATA REG ADDRESS
		DATA.B	EQU $F402	" (b) " " "
		TOPSTK	EQU $F0F3 (FDED)	TOP OF STACK USED BY 7768 BUG 1
		JMPNMI	EQU $F0F5	SPACE FOR JUMP TO NMI HANDLER
		JMPIRQ	EQU $F0F8	" " " " IRQ "
		TMPXHI	EQU $F0FB	TEMP STORAGE FOR 7768 BUG 1 USE
		TMPXLO	EQU $F0FC	" " " " " "
		TMPSTK	EQU $F0FD	" " " " " "
F F 0 0	8 E F 0 F 3 (E D)	RESET	LDS #TOPSTK	SET UP BUG STACK
0 3	8 6 0 3		LDA A #$3	
0 5	B 7 F 4 0 1		STA A CTRL.A	RESET ACIA's
0 8	B 7 F 4 0 3		STA A CTRL.B	
0 B	B F F 0 F D (E D)	SWINT	STS TMPSTK	SAVE PROG's STACK POINTER
0 E	8 E F 0 F 3 (E D)	RSTART	LDS #TOPSTK	& SET UP BUG's STACK
1 1	8 6 1 1		LDA A #$11	
1 3	B 7 F 4 0 1		STA A CTRL.A	SET UP ACIA's
1 6	B 7 F 4 0 3		STA A CTRL.B	
1 9	8 D 5 B		BSR PCRLF*	
1 B	8 D 7 1		BSR GETCH	GET COMMAND
1 D	8 1 4 1		CMP A #'A	ALTER ?
1 F	2 7 3 6		BEQ ALTER	
2 1	8 1 4 3		CMP A #'C	CONTINUE ?
2 3	2 7 2 E		BEQ CONT	
2 5	8 1 4 7		CMP A #'G	GO ?
2 7	2 7 1 5		BEQ GOGOGO	
2 9	8 1 5 2		CMP A #'R	REGISTER PRINT ?
2 B	2 7 3 3		BEQ REGPNT	
2 D	8 1 4 5		CMP A #'E	EXAMINE ?
2 F	2 6 D D		BNE RSTART	IT WAS RUBBISH - RESTART
3 1	8 D 7 2		BSR GETADR	GET ADDRESS TO EXAMINE
3 3	8 D 6 A	EXAMLP	BSR P2HXS	PRINT CONTENTS
3 5	8 D 4 B		BSR GETCH	WAIT FOR NEXT KEYBOARD INPUT
3 7	8 1 2 0		CMP A #$20	SPACE ?

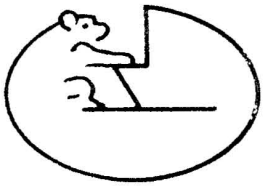


BEAR MICROCOMPUTER SYSTEMS
24 College Road, Maidenhead, Berks, SL6 6BN

HEXADECIMAL CODING FORM

PROGRAM 7768 BUG 1	VERSION 01	AUTHOR MRL	DATE 3-2-78	PAGE 2 OF 4

ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
F F 3 9	2 6 D 3		BNE RSTART	
3 B	0 8		INX	POINT @ NEXT ADDRESS
3 C	2 0 F 5		BRA EXAMLP	GO BACK & PRINT CONTENTS
3 E	8 D 6 5	GOGOGO	BSR GETADR	GET PROGRAM START ADDRESS
4 0	F E F 0 F D		LDX TMPSTK	POINT X @ PROGRAM COUNTER
4 3	0 8		INX	POSITION IN STACK
4 4	0 8		INX	(DON'T YOU WISH YOU HAD
4 5	0 8		INX	A PDP-11)
4 6	0 8		INX	
4 7	0 8		INX	
4 8	0 8		INX	
4 9	B 6 F 0 F B		LDA A TMPXHI	LOAD IT WITH DESIRED
4 C	A 7 0 0		STA A X	STARTING ADDRESS
4 E	B 6 F 0 F C		LDA A TMPXLO	(OR EVEN A 6809 ?)
5 1	A 7 0 1		STA A X,1	
5 3	B E F 0 F D	CONT	LDS TMPSTK	RESTORE STACK POINTER
5 6	3 B		RTI	& OVER TO USER PROGRAM
5 7	8 D 4 C	ALTER	BSR GETADR	GET ADDRESS TO BE CHANGED
5 9	8 D 5 B	ALTLP	BSR HEX2IN	(BACK TO RSTART IF NOT HEX)
5 B	A 7 0 0		STA A X	STORE NEW DATA
5 D	0 8		INX	POINT @ NEXT LOCATION
5 E	2 0 F 9		BRA ALTLP	GO BACK TO GET NEW DATA
6 0	8 D 3 F	REGPNT	BSR PNTSP	
6 2	F E F 0 F D		LDX TMPSTK	POINT X @ USER PROG STACK
6 5	8 D 3 7		BSR IP2HXS	PRINT CC REG FROM STACK
6 7	8 D 3 5		BSR IP2HXS	PRINT B REG " "
6 9	8 D 3 3		BSR IP2HXS	PRINT A REG " "
6 B	8 D 2 E		BSR IP4HXS	PRINT X REG " "
6 D	8 D 2 C		BSR IP4HXS	PRINT PROGRAM'S P.C. "
6 F	C E F 0 F D		LDX #TMPSTK	
7 2	8 D 2 8		BSR P4HXS	PRINT USER PROG'S S.P.
7 4	2 0 9 8	START2	BRA RSTART	& BACK FOR NEXT COMMAND
7 6	8 6 0 D	PCRLF*	LDA A #$0D	(CARR. RETURN)
7 8	8 D 1 6		BSR PNTCH	
7 A	8 6 0 A		LDA A #$0A	(LINE FEED)



BEAR MICROCOMPUTER SYSTEMS
24 College Road, Maidenhead, Berks, SL6 6BN

HEXADECIMAL CODING FORM

PROGRAM 7768 BUG 1	VERSION 01	AUTHOR M R L	DATE 3-2-78	PAGE 3 OF 4

ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
F F 7 C	8 D 1 2		BSR PNTCH	
7 E	8 6 2 A		LDA A #'*	
8 0	2 0 0 E		BRA PNTCH	(BR & RTS)
8 2	B 6 F 4 0 1	GETCH	LDA A CTRL·A	WAIT FOR A CHARACTER TO
8 5	4 7		ASR A	BE RECEIVED INTO ACIA(a)
8 6	2 4 F A		BCC GETCH	
8 8	B 6 F 4 0 0		LDA A DATA·A	PUT IT INTO A REG.
8 B	8 4 7 F		AND A #$7F	LIFE IS SIMPLER WITHOUT PARITY
8 D	3 9		RTS	
8 E	8 D F 2	G4PCH	BSR GETCH	GET A CHARACTER INTO A REG
9 0	F 6 F 4 0 1	PNTCH	LDA B CTRL·A	WAIT UNTIL ACIA(a) TRANSMIT
9 3	C 5 0 2		BIT B #2	BUFFER IS EMPTY
9 5	2 7 F 9		BEQ PNTCH	
9 7	B 7 F 4 0 0		STA A DATA·A	THEN TRANSMIT A REG
9 A	3 9		RTS	
9 B	0 8	IP4H&S	INX	
9 C	8 D 2 6	P4H&S	BSR P2HEX	
9 E	0 8	IP2H&S	INX	
9 F	8 D 2 3	P2H&S	BSR P2HEX	
A 1	8 6 2 0	PNTSP	LDA A #$20	(SPACE)
A 3	2 0 E B		BRA PNTCH	(BR & RTS)
A 5	8 D F A	GETADR	BSR PNTSP	
A 7	8 D 0 D		BSR HEX2IN	GET FIRST BYTE OF ADDRESS
A 9	B 7 F 0 F B		STA A TMPXHI	& STORE IT
A C	8 D 0 8		BSR HEX2IN	GET SECOND BYTE
A E	B 7 F 0 F C		STA A TMPXLO	& STORE THAT
B 1	F E F 0 F B		LDX TMPXHI	TRANSFER ADDRESS TO X REG
B 4	2 0 E B		BRA PNTSP	(BR & RTS)
B 6	8 D 2 4	HEX2IN	BSR GETHEX	GET ONE HEX DIGIT
B 8	4 8		ASL A	MOVE IT LEFT 4 BITS
B 9	4 8		ASL A	
B A	4 8		ASL A	
B B	4 8		ASL A	
B C	3 6		PSH A	& SAVE ON STACK
B D	8 D 1 D		BSR GETHEX	GET NEXT HEX DIGIT

BEAR MICROCOMPUTER SYSTEMS
24 College Road, Maidenhead, Berks, SL6 6BN

HEXADECIMAL CODING FORM

PROGRAM 7768 BUG 1	VERSION 01	AUTHOR MRL	DATE 3-2-78	PAGE 4 OF 4

ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
FFBF	840F		AND A #0F	THIS IS RIGHT 4 BITS
C1	33		PUL B	RETRIEVE LEFT 4 BITS
C2	1B		ABA	& MERGE THEM
C3	39		RTS	
C4	A600	P2HEX	LDA A X	GET BYTE TO BE PRINTED
C6	8D04		BSR PNTLH	PRINT HEX CHAR = LEFT 4 BITS
C8	A600		LDA A X	RELOAD A REG
CA	2004		BRA PNTRH	(BR & RTS), PRINT RIGHT HEX
CC	44	PNTLH	LSR A	SHIFT A REG RIGHT BY 4 BITS
CD	44		LSR A	
CE	44		LSR A	
CF	44		LSR A	
D0	840F	PNTRH	AND A #$0F	BLANK OUT LEFT 4 BITS
D2	8B30		ADD A #$30	CONVERSION, HEX TO ASCII
D4	8139		CMP A #$39	
D6	2302		BLS PNTCH	(BR & RTS)
D8	8B07		ADD A #$7	
DA	20B4		BRA PNTCH	(BR & RTS)
DC	8DB0	GETHEX	BSR G&PCH	GET ASCII CHARACTER
DE	8130		CMP A #$30	CONVERT TO HEX,
E0	2B92		BMI START2	GO TO START2 IF CHARACTER
E2	8139		CMP A #$39	IS NOT 0-9, A-F
E4	2F0A		BLE RTS	
E6	8141		CMP A #$41	
E8	2B8A		BMI START2	
EA	8146		CMP A #$46	
EC	2E86		BGT START2	
EE	8007		SUB A #$7	
F0	39		RTS	
				FFF1 - FFF7 SPARE !
FFF8	F0F8		FDB JMPIRQ	IRQ VECTOR
FFFA	FF0B		FDB SWINT	SWI "
FFFC	F0F5		FDB JMPNMI	NMI "
FFFE	FF00		FDB RESET	RESET "