77 - 68

THE CONSTRUCTION OF A

- SIMPLE MICROCOMPUTER -

USING A 6800 MICROPROCESSOR

© BEAR MICROCOMPUTER SYSTEMS 1977

24 College Rd, Maidenhead, Berks. SL6 6BN

First edition June 1977

Preface

77-68 was designed to be a low cost system with which the constructor could learn about microcomputing by direct experience. The board can then be expanded, without restriction, to the limit of the 6800 microprocessor's potential. Throughout the following months, additional items will be made available e.g. memory cards, Kansas City interface, RS 232C interface, Monitoring ROM's. The User Group will hold all the latest information and it is expected to play a major role in the development of this system.

If difficulty is experienced in obtaining any parts or in commissioning the microcomputer, do not hesitate to contact the "BEAR" or the User Group.

The Instruction set is reproduced by kind permission of Motorola Ltd.

BEAR MICROCOMPUTER SYSTEMS

JUNE 1977

2nd. Reprint.

Included in this edition are the details of the two avaiable extentions to 77-68;- the 4Kbytes RAM p.c.b. and the Soft Monitor V24/R232C interface p.c.b. Within the next few weeks a ROM monitor p.c.b. and a V.D.U./Keyboard p.c.b. will become available, plus several more during 1978. You are recommended to the User Group Newsletters No. 1,2 and 3 which have already been published.

NEWBEAR COMPUTING STORE

May 1978.

L I S T O F C O N T E N T S

L I S T O F F I G U R E S

Chapter 1. Introduction

The 77-68 was designed as a cheap and simple way to build a working microcomputer which could be expanded at a later date, when the constructor's pocket and time permit.

Although the basic machine described here is limited in terms of memory (256 8 bit words) and input/Output devices (only binary switches and lamps are provided), it will give the user valuable experience of the hardware and software techniques associated with microprocessors, without committing him to a large initial outlay.

Even in its simplest form, the 77-68 can be used to execute real programs. Mathematical routines can be run, simple computer games played, and the machine can be a controller/sequencer for household devices, simple production tools, or even a model train layout.

The basic 77-68 can be built for about £50, or less if the constructor has a reasonably deep junk box, and the only additional equipment needed is a source of +5V DC power. By using toggle switches and LED's as rudimentary input/output devices, the 77-68 user does not need to invest in a teleprinter or VDU before he can get started - although these and other enhancements can be added to the system later.

Care has been taken in the design to ensure that the machine can be expanded easily to have a capability close to that of any 8 bit microprocessor currently available. For example, the 6800 MPU chip has been chosen as the heart of the 77-68, rather than one of the simpler, slower and slightly cheaper types which would limit the power of an enhanced machine. Also, provision is made for fully buffered TTL data and address busses for flexibility in adding other units to build up a large system.

Finally, a word of warning. Although the 77-68 is relatively straightforward, it should not be attempted until the would-be constructor is familiar with the construction of TTL based logic circuits, and has some knowledge of microcomputer hardware and software. The following books are highly recommended as suitable

background reading;

'An Introduction To Microcomputers' Vols I & II by Adam Osborne. Published by Adam Osborne and Associates Inc., 2950 Seventh St., Berkekyl, California 94710, and distributed in Europe by SYBEX, Publications Dept, 313 Rue Lecourbe, 75015 Paris, FRANCE.

or Bear Microcomputer Systems

'M6800 Microcomputer System Designs Data', (or at least the M6800 data sheets) from Motorola.

or Bear Microcomputer Systems

Chapter 2 77-68 Characteristics

1. CPU

The heart of the 77-68 is the '6800' eight bit microprocessor, readily available from Motorola, AMI, and others. This features;

* 72 instructions.
* 7 Addressing modes (Direct, Relative, Immediate, Indexed, Extended, Implied and Accumulator).
* 6 internal registers (two accumulators, index register, program counter, stack pointer and condition code register).
* 8 bit parallel processing.
* Single (5V) power supply.

2. Memory

The basic 77-68 card has 255 words of 8 bit static Random Access MOS Memory. However, all 16 of the 6800 address lines are buffered and brought out to the card connector, allowing the constructor to expand to the full 64K word capability of the 6800 CPU chip.

3. CPU Cycle Time

A quartz crystal controlled clock gives a basic CPU cycle time of 1.6uS. This may be extended in increments of 0.2uS to a maximum of 5uS to cater for slow external memory should the need arise.

The time taken to execute an instruction depends upon the particular operation and the addressing mode used e.g.;

Add Accumulators;	2 cycles	3.2uS
Load Accumulator;	2-5 cycles	3.2 - 8 uS
Branch;	4 cycles	6.4uS

4. Input/Output

Peripheral data and control registers are treated as memory locations, and can therefore be handled using the full 6800 instruction set.

The basic 77-68 uses eight toggle switches as an input register for loading the user's program and for entering data during program execution, and eight LED's as a data display. Although this might be considered a rather primitive approach, it does give the constructor a

minimum 'first cost' system and emphasises the essentially binary nature of microprocessors. More sophisticated peripherals, such as a keyboard, display & printer, can easily be added to the basic system.

As well as the eight 'data' switches, eight 'address' switches allow the user to examine the contents of any of the 255 memory locations when the processor is halted.

5. Construction

The 77-68 is designed around the standard 8.0" square printed circuit board with a 0.1" single sided edge connector. Suitable 'prototyping' boards are readily available, alternatively the constructor may use a specially designed board supplied by BMS.

The basic machine comprises;

- a single board containing the 6800 microprocessor chip, 255 words of memory, data input and output registers and miscellaneous control circuitry.
- a simple control panel.
- a source of 5VDC at about 1A.

6. Expandability

Enhancements being designed for the 77-68 include;

- Memory extension
- Tape cassette interface ('Kansas City' standard)
- VDU and keyboard interfaces

These will allow the constructor to build up a system capable of running much of the vast amount of software that is readily available to 6800 based system users.

Chapter 3 Hardware Description

The Basic Machine

Fig 1 shows a block diagram of the basic 77-68. It comprises;

- The 6800 Microprocessor Unit itself. This performs the actual machine language instructions stored in the RAM, using data read from the RAM or the Data Switch Register.
- A 256 word eight bit Random Access Memory used to store data and instructions. The particular word to be read from or written into is selected by the eight address lines A0 - A7. The 6800 MPU actually has 16 address lines, providing an addressing range of 2^{16}, or approximately 65000 words, however the high 8 lines A8 to A15 are not used by the basic 77-68. Also, the address 255 (Hexadecimal FF) accesses the data switches and data display register as described below, thus the RAM is disabled when this address is selected.
- A 8 line, 2 way, data selector which allows the address bus lines A0 to A7 to be driven either from the 6800 MPU or from the address switches.
- An eight bit data input switch register. When the MPU selects address FF, a read operation will take information from these switches, rather than from the RAM.
- An eight bit output register, driving an eight bit 'data' display. An MPU write operation into location FF will load this register.
- Miscellaneous clock and control logic.

The HALT switch disables the MPU (after allowing it to complete the current instruction) so that its TRI-STATE outputs go to the high impedance state. The address selector is then set so that lines A0 - A7 are controlled by the address switches rather than by the MPU. At the same time, the data register input gates are opened so that the data display shows whatever information is on the data bus, and a read condition applied to the RAM. Thus the contents of any of the RAM locations can be examined by setting the appropriate address on the switches.

When the LOAD switch is operated (while the machine is HALTed) the RAM

is set to the write condition, and the contents of the data switch register gated onto the data bus to be written into memory at the location selected by the address switches.

Thus, without involving the MPU, we can examine the contents of memory, or load new information into any location.

When the RESET switch is operated, the MPU goes into an initialisation routine, then it reads the contents of location FF (the data switch register) as the location at which to start program execution when the HALT condition is removed. So, having loaded a program into memory, we can run it by simply setting the data switch register to the program starting address, momentarily operating the RESET switch, then removing the HALT condition. If the HALT condition is removed without the RESET switch having been operated, the MPU will resume operation from the point at which it had been halted.

The RUN lamp is lit whenever the 6800 MPU has control of the busses. Although the 6800 MPU does not have an explicit HALT instruction, WAI (Wait for Interrupt) has roughly the same effect and, when operated, will extinguish the RUN lamp.

Detailed Circuit Description

In describing the hardware, the following conventions are used;

- '1' is high (any voltage between about +2 and +5V)
- '0' is low (0 to 0.5V)
- A bar ($\overline{\quad}$) over a signal name means that it is asserted low. For example, the line $\overline{\text{SWSEL}}$ goes low to select the switch inputs. Similarly, a circle (o) on a logic element input means that input is asserted low, e.g. X13 pin 40 is pulled low (to 0) to reset the device.
- Gate functions can, in general, be drawn in two ways;

and

The version which best represents the logic function being performed is used in the schematic.

All inputs and outputs of the MOS devices (X13, 17 & 18) are protected against static electricity by being connected to some other device; a TTL device input or output will effectively clamp spikes greater than about +8V, or more negative than about -1V. Thus, although address lines A8 - A15 are not used by the basic 77-68, it is wise to include X11, 12 gates from the beginning to protect the valuable 6800.

X13, the 6800, is the heart of the machine. Its low order eight address lines A0 - A7 are routed to the data selectors X8 & 9, which take either the 6800 outputs (ADSEL = 0) or the pattern set on the address switches (ADSEL = 1) and apply the result to the RAM and to the connector for system expansion. As well as selecting the appropriate address source, X8 & 9 also buffer the rather low power outputs of the 6800 to full TTL drive levels. The eight high order address outputs A8 - A15 of the 6800 are buffered by the OR gates X12 & X11, which also force A8 - A15 at the connector to 1's when ADSEL is high.

The data switch information is transferred onto the data bus when required by the Tri-State buffers X22, X23. The inputs of these buffers are high impedance when $\overline{\text{SWSEL}}$ is '1', otherwise they are at 1 or 0 depending upon the setting of the data switches. (Note; for both the data and address switches, open circuit = 1, closed = 0). 74125's have been used for X22 & X23 as these are cheap and readily available.

The data register (X20, 21) takes information from the 6800 data bus when RSEL is at 1, and latches this information when RSEL falls to 0. The Low Speed Schottky version (74LS75) is used as a normal 7475 would load the 6800 data bus too much, while a 74L75 would need additional buffering to drive the LED's. The $\overline{Q}$ outputs of X20, 21 are fully loaded driving the LED's, so the Q outputs are also taken to the card connector to drive any peripheral devices which may be added.

DM81LS97 Octal Tri-State buffers X15, 16 may be added to buffer the data bus to full TTL drive capability when it is desired to expand the system, they are not required for the basic 77-68. 74125's cannot be used here as their inputs would load the 6800 data bus outputs too much.

The memory uses two 256 x 4 RAM, the widely available 2112 devices being chosen. These memories are enabled when $\overline{MCE}$ is low, the contents of the selected location are read out onto the data bus when MRW is 1, when MRW is 0 the information present on the data bus will be written into the selected location.

The 6800 MPU interrupt inputs $\overline{NMI}$ and $\overline{IRQ}$ are not used by the basic 77-68, but are brought to the card connector for future use, and held high by R13, 14 (these are 'active low' inputs).

Timing for the machine is derived from a 5 MHz crystal controlled oscillator. A cheaper RC oscillator using, say, a 7413 could have been used instead, but an accurate timing source is sufficiently valuable in some applications (e.g. a software routine to drive a serial teleprinter input) to make the slight extra expense worthwhile.

The 5 MHz is divided by 8 in X2, giving a normal MPU cycle time of 1.6uS. This is a bit slower than the maximum operating speed of the M6800, but allows the use of cheap (low speed) memories, and also eases various timing problems that occur if one tries to squeeze the last ounce of performance out of the MPU. The slight reduction in speed does not significantly reduce the system capability.

The MPU clock waveforms Ø1 and Ø2 and various other timing signals are derived as shown in Fig 2. Points worthy of note are;

- DBE is high for most of the time. As this line gates the 6800 outputs onto the bus during a Write operation, it allows the use of memories with a relatively long write time, and those which require that the data is present after their $\overline{CE}$ input goes high at the end of a write cycle.
- The general enable line (E) goes to 0 after the end of Ø2, ensuring that the 6800 input data hold time requirements are met even when using a fast memory.
- A derived clock is used to synchronise the HALT and LOAD inputs (by X10) to ensure that they only change state at the correct time in the MPU cycle.
- Ø2 can be extended by holding the $\overline{HOLD}$ input low by external logic. This allows external slow core (or cheap EPROM) memory

to be added to the system. The external logic must ensure that the Ø2 '1' time is not extended beyond the 4.5uS limit given in the M6800 specification.

The RESET input from the switch is 'de-bounced' by the bistable made from two of X24 gates.

When the HALT switch is thrown, X10 pin 5 goes low, applying a HALT signal to the 6800. After completing the current instruction, the 6800 will raise the BA (Bus Available) line to 1. Note that the BA line will also go to 1 after execution of the WAI instruction.

BA going to 1 turns off the RUN LED and, if the HALT switch had been set, puts ADSEL to 1 which puts A0 - A7 under control of the address switches and allows RSEL to go high (when the clock signal on X5 pin 4 is high) so the data register continually monitors the state of the data bus.

The E line carries a general purpose memory enable signal that goes high when the MPU clock Ø1 is low and a valid address is present on the address bus (X13 VMA output high or the HALT switch thrown).

The '256SEL' card input is not used in the basic 77-68, but if pulled low de-selects the on-card 256 word RAM, allowing the use of an external memory. In the basic 77-68 this card input is left open, pulled high by the 430 ohm resistor.

The $\overline{\text{MCE}}$ line enables the 256 word RAM when low.

X7 detects the 'all ones' condition on the address lines A0 - A7 which is the data switch/register address. It then inhibits the RAM and enables selection of the switch register (MRW high) or data register (MRW low) via X3, X19 etc.

The MRW line is normally high, and goes low for a write operation (into memory or the data register), including a LOAD.

Chapter 4. Construction & Testing

The component and connector numbering shown is that for the 77-68 CPU printed circuit board which is available as detailed elsewhere. The layout of the major components on this board is shown in Fig. 4. This board is 8.0" square with a gold plated 0.1" single sided edge connector. To cut costs a single sided board has been used thus some additional wiring is needed to complete the circuit. Nevertheless it forms the basis for a soundly constructed machine. The board has provision for fitting the optional data bus buffers X15,X16.

Compatible 'prototyping' boards and edge connectors are available from VERO as;

VB/10725/1 plug-in single sided 0.1" matrix board.

12681 SRBP)
12682 EPOXY) DIG plug-in boards

13845/1 single sided plug in DIY board (undrilled, copper clad with edge connector contacts already etched)

10859/4 solder-lug) single-sided 0.1" edge connector socket (77
13597/4 mini-wrap) ways plus polarising key position)

Also, the IMHOF-BEDCO MCV/5CX/100 'IMCARD' may be used, although this card is slightly smaller (7.9" wide x 7.5" long) and has only 75 edge contacts (plus polarising slot) corresponding to positions 2 - 77 on the VERO boards.

Should the constructor decide to 'wire his own', layout is not critical, except that the Ø1 & Ø2 drive lines from X5 to X13 (via R17, 18) should be kept short, and power lines, especially Ov (Ground) should be thick and laid out in the form of a mesh to minimise the impedance (and hence noise) between any two points in the circuit. Decoupling capacitors C1 - C11 should be distributed evenly across the board, with one of the electrolytics being positioned close to X13, another near X20, 21.

Whichever method is chosen, it is recommended that sockets be used for the MOS devices (X13, 17 & 18). The 40 way socket used by X13 should be of reasonable quality, not requiring excessive insertion force.

The wiring between the edge connector and the power supply and control panel are shown in Fig. 5. This figure also shows the basic 'Bus' connections from the CPU board to other boards (such as Memory, Cassette Interface, VDU control) which may be added later.

Testing

Most of the circuitry can be checked without the expensive X13, 17 or 18.

When the unit is fully assembled, but before plugging in X13, 17 & 18, check for any possible short-circuits between 0 and +5V inputs, and then apply power. None of the IC (except for X13, 17 & 18) consume much power, so any which feel more than slightly warm to the touch after a few minutes should be suspect.

Testing the oscillator and Ø1, Ø2 generator is easy with a 'scope of suitable bandwidth, but for those without, a moving coil voltmeter (greater than 10K / Volt resistance) should show;

X13 pin 3 (Ø1) approx. 1.8V
" " 37 (Ø2) " 2.5V
" " 36 (DBE) about 2.5 - 3.5V, and the measured voltage should rise by about 0.5V or fall to 0V (depending upon the state of the divider X2) when the oscillator is stopped by shorting X1 pin 6 to 0V (pin 7).

Edge connector pins 9 (5MHz) and 10 (CLK) should be about 1.5V, as should X10 pins 3 & 11 and X4 pins 4 & 5. These points should go to either '1' (about 3V) or '0' (about 0.1V) if the oscillator is stopped.

Check the $\overline{\text{HOLD}}$ input (pin 8) by earthing it and noting that X13 pin 37 (Ø2) goes to +5V.

X13 pin 40 (RESET) should be at '1', falling to '0' when the RESET switch is thrown.

The RUN LED should be out, but should light when pin 7 (BA) of X13 is temporarily connected to 0V.

X13 pin 2 (HALT) should be '1', falling to '0' when the HALT switch is set.

With X13, 17 & 18 still missing, set all address and data switches to '0' and switch to HALT. All data LED's should be on. Turn the address switches in turn to '1'. Only when they are all at '1' should the data LED go out (because the address FF of the switch register has

been selected). Leaving the address set to FF check that each data LED can be turned on and off by the corresponding data switch.

With the HALT switch in the RUN position, check that, regardless of the address switch settings, operation of the LOAD switch transfers the setting of the data switches to the LED's. With the LOAD switch unoperated the display should remain unaffected by any alteration of the data switch settings.

Now, plug in one of the 2112 memories (X17) (the right way round!), turn the power back on and switch to HALT.

You should now be able to store any pattern of bits 4 - 7 in any address (00 - FE) by;

- setting the data pattern and address on the switches and then operating the LOAD switch.

The stored pattern is read by setting the appropriate address on the switches (with the switch set to HALT).

For example;

set data 00, address 00, press LOAD
" " 10 " 01 " "
" " 20 " 02 " "
" " 30 " 03 " "

Then check that;

setting address to 00 displays 00
" " " 02 " 20

etc.

If this seems to be working, plug in the other memory (X18) and check that you can write into and read out of the low four bits as well.

Before proceeding, it is worthwhile spending half an hour or so practicing writing into and reading different memory locations and converting between hexadecimal notation and the switch settings. (Note that address FF is the switch/register, and the memory is inhibited at this address).

Next, carefully test the voltages on X13 socket pins (don't accidentally

short two pins together when doing this!);

pins 1, 21 & 39 are at 0V
pins 4, 6 & 8 are at +5V
other pins will be at various voltages but none should be higher than +4V.

REMOVE ALL POWER FROM THE BOARD.

PLUG IN X13 VERY CAREFULLY - exerting an even pressure on all 40 pins, making sure that it goes into the socket straight, watching for signs of any leads bending under the IC body. The ceramic package version in particular is rather prone to cracking if an uneven stress is applied.

CHECK THAT IT IS THE RIGHT WAY ROUND.

If the machine has been hand wired, rather than built on a PC board, check that plugging in X13 has not disturbed any wires causing possible short circuits.

Make sure the supply is the correct voltage and polarity, then reconnect it.

With the switch in the HALT position, check that you can still load and read memory as tested previously.

Then check that pressing the RESET button lights the RUN LED.

Now for the first program. Load the following;

	mem loc	data	instruction
START:	00	7C	INC FFFF (switch/register)
	01	FF	
	02	FF	
	03	20	BR START
	04	FB	

This program reads the contents of location FFFF (the switch register), adds one, then stores the result in location FFFF (the data display register). It then branches back to the beginning and repeats for ever.

Having loaded and checked the program, set the data switches to the start address (00), momentarily depress the RESET button, then switch to RUN. The RUN LED should light and the data display show 0L (switch register + 1). Check that whatever the setting of the switch register the display is always one greater. Also check that operation of the HALT switch turns off the RUN LED, and that the machine starts again when the HALT condition is removed.

By changing the instruction stored at location 00, the program can perform other functions;

loc 00	instr	display
7F	CLR	00
73	COM	One's complement (inverse) of sw reg
70	NEG	Two's complement (negative) of sw reg
7A	DEC	Sw reg minus one
79	ROL	Sw reg rotated left one position (B0 will be set according to what was in the C bit)
76	ROR	Sw reg rotated right one position (B7 will be set according to what was in the C bit)
78	ASL	Sw reg shifted left one position (B0 set to '0')
77	ASR	Sw reg shifted right one position, except for B7 which remains the same
74	LSR	Sw reg shifted right one position, B7 set to '0'
7D	TST	Sw reg unchanged

Notes;

Each time the ROL instruction is performed during the program, the C bit will be set according to B7 of the switch reg, and in the next operation this will be set into B0 of the display. B0 of the display will therefore equal B7 of the Switch reg.

Similarly, when using this program with the ROR instruction, the C bit, and hence B7 of the display, will equal B0 of the Sw reg.

The TST instruction as implemented on the 6800 actually reads from the addressed location, then stores the data, unaltered, back in the same

location. Thus in this program it transfers the contents of the switch reg into the display reg.

Further test programs are given in the following section, but take heart, having got this far the machine is now basically working.

Chapter 5. Programming

For a detailed knowledge of the 77-68 instruction set, the constructor should refer to a publication such as the Motorola M6800 Microprocessor Programming Manual or Desig Note 4, however this chapter gives a summary of the more important features, illustrated by programs which will run on the 77-68.

Programming Model

From the programmer's point of view, the 77-68 consists of;

- Two eight bit accumulators; A & B
- An eight bit 'Condition Code' register
- A 16 bit Program Counter
- A 16 bit Index Register
- 255 (decimal) 8 bit words of random access memory (addresses 00 to FE)
- An 8 bit write-only display register at address FF
- An 8 bit read-only input switch register at address FF

Notation

The programs in this book are written in Motorola 6800 Assembly Language. Mnemonic codes are used to represent the instructions, and the actual machine code is also given, in hexadecimal format (see Appendix 1).

To save space, each line contains a complete instruction, whether it consists of one, two or three bytes. Thus in the '77-68 FLASHER Version 1' example, address 06 contains machine code 20, and machine code F8 is stored in address 07.

Addressing Modes

The 6800 (and hence the 77-68) has seven addressing modes;

Accumulator Addressing;

One byte, single operand instructions which operate on either of the accumulators e.g. INC A DEC B

Inherent Addressing;

One byte instructions which imply the use of one or more of the 6800 registers e.g. INX ABA

Immediate Addressing;

Two or three byte instructions in which the second (and third) byte of the instruction is data. In writing Assembly Language, a # sign is written before the data to distinguish it from an address e.g.

```
ADD A     # 02 adds the value 2 to accumulator A
LDX       # ABCD loads the (hex) value ABCD into the index reg.
```

Direct Addressing;

These are two byte instructions in which the second byte contains the address of the operand (which must lie in the range 0 - 255 (decimal) 00 - FF hex), and are written in the form;

```
OP      N       where N is a number or symbol
```

```
e.g.  ADD A   27    adds the contents of location 27 to acc. A
      ADD A   ITEM  adds the contents of location ITEM to acc. A
```

Extended Addressing;

These are three byte instructions which contain full 16 bit address (in the range 0 to FFFF hex) in the second and third bytes. The second byte of the instruction contains the most significant 8 bits of the address and is ignored by the basic 77-68 (which only examines the eight least significant address bits)

Relative Addressing;

Branch instructions are all two bytes long, where the second byte contains an offset value which specifies the branch destination address according to the formula

$$D = PC + 2 + R$$

where D = address of destination
PC = address of first byte of branch instruction
R = 8 bit, two's complement, binary number stored in the second byte of the instruction.

When writing branch instructions in assembly code, the address is given as a label which points to the required destination. Thus, in the 'Flasher, Version 1' program;

Address	Machine Code	Assembly Language Mnemonics
00		START - -
-	----	- -
-	- - -	- -
06	20 F8	BRA START

Indexed Addressing;

These are two byte instructions in which the address of the operand is obtained by adding the (8 bit) value in the second byte of the instruction to the (16 bit) contents of the index register. Note that, unlike the branch instructions, the second byte of the instruction is treated as a positive unsigned integer in the range 0 to 255 (decimal). This addressing mode is identified in assembly language by the addition of ',X' at the end of the instruction, e.g. ADD A ITEM,X

77-68 Flashers

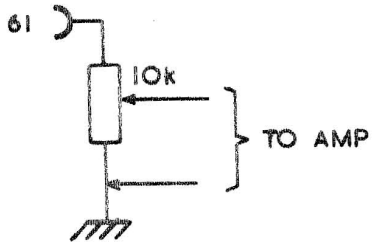
These three programs generate a changing pattern on the display, useful to amuse the children, impress the neighbours, or as a quick check that the system is working.

The main problem is to slow things down so that the pattern changes are visible. Versions 1 & 2 use the index register as a 16 bit counter; the program only gets beyond the 'BNE START' instruction when the index register steps to zero - every 65000 loops. Version 3 uses three nested loops; the inner loop increments accumulator A until it overflows and passes through zero, the second (middle) loop increments accumulator B once every 2.4mS until B equals the setting on the switch register. (Don't forget that when starting this program the switch register should be set to zero while the RESET switch is operated). The third, outermost, loop increments the index register every so often and displays the least significant 8 bits.

Tone Generator

This program generates an audio tone at a frequency determined by the setting of the switch register. To use it, connect bit 7 output of the

display register (connector pin 61) to an audio amplifier via a volume control;



The basic frequency is determined by the first three instructions, and the (square) waveform generated by incrementing the contents of accumulator B (hence changing the state of bit 7) after each timing period.

Taking the audio signal from other outputs of the display register will give a tone one or more octaves lower - for a given setting of the display register.

Tuner

Having made 77-68 generate tones, the next step must be to get it to play a tune! To do this we need to be able to store in memory details of the notes, the order and durations for which they are played (and the durations of the spaces between the notes).

In the program Tuner, the tune to be played is stored as a string of pairs of (8 bit) bytes in sequential locations, starting at location 'TUNE' (store address 40 hex). Each data byte is interpreted as an unsigned binary integer in the range 0 to 255 (decimal).

The first byte of the string, and subsequent bytes stored in even numbered locations, defines the length of the note or silence. A zero value indicates the end of the tune.

The second byte, and all those stored at odd numbered addresses, define the frequency according to the following table;

Value of byte (Hex)	Note	Value of byte (Hex)	Note
ØØ	Silence	64	C
27	E	6A	B
2C	D	77	A
31	C	86	G
34	B	97	F
3B	A	AØ	E
42	G	B4	D
4B	F	CA	C
51	E	D6	B
59	D	FØ	A

The first instruction of the program loads the index register with the address of the store location two bytes before the beginning of the 'tune' string, the two subsequent INX instructions increment the index register so that at instruction 'LDA A Ø,X', the index register is pointing at location 'TUNE'. (This may seem a roundabout way to do things, but it simplifies later parts of the program).

LDA A Ø,X loads the A accumulator with the value stored at the address 'Index Reg Contents + Ø' i.e. the location TUNE. This is the duration of the first note (or silence), and if it has zero value, the program branches back to START to begin the tune again. The duration value is then stored in the temporary storage location TIME, and accumulator A cleared.

LDA B 1,X loads the B accumulator with the value stored at the address 'Index Reg Contents + 1' i.e. the location TUNE + 1. This is the frequency of the note (branch to routine QUIET if the value is zero). The sequence INS, STS FE changes the state of bit 7 of the display register (beginning of one half-cycle of output).

The routine CYCLE is then entered. This decrements two counters;

- a 16 bit counter using accumulator A as the least significant eight bits and temporary location TIME as the eight most significant. When this counter reaches zero the program branches to NUNOTE where the index register is incremented by two to point to the next pair of bytes in the 'tune' list, and the next note (or period of silence) started.

- an eight bit counter using accumulator B which defines the length of one half-cycle of output. When accumulator B reaches zero the program branches to NUCYCL to reload accumulator B and change the state of bit 7 of the display register, thus starting the next half-cycle of output.

Routine QUIET merely decrements the 16 bit counter (accumulator A and location TIME) until the end of the period, without changing the state of bit 7 of the display register. The NOP instructions are fillers to make the routine cycle time similar to that of the routine CYCLE.

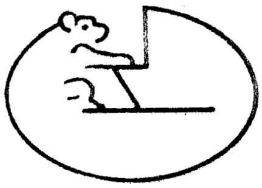
Constructors who have got this far might like to try the tune listed below;

Location	Values		Location	Values	
	MSB	LSB		MSB	LSB
40	20	42	70	20	4B
42	20	51	72	20	59
44	60	00	74	60	00
46	20	51	76	20	3B
48	20	4B	78	20	34
4A	20	42	7A	20	3B
4C	20	27	7C	20	31
4E	20	00	7E	20	42
50	20	27	80	60	00
52	20	00	82	20	42
54	58	31	84	20	4B
56	08	00	86	20	51
58	20	42	88	20	59
5A	20	51	8A	20	3B
5C	60	00	8C	20	00
5E	20	51	8E	20	64
60	20	4B	90	20	6A
62	20	51	92	20	42
64	20	42	94	20	00
66	20	00	96	20	42
68	20	42	98	FF	64
6A	20	00	9A	FF	00
6C	58	4B	9C	00	00
6E	08	00			

Notes; data & addresses given in hexadecimal.

MSB = data stored at even numbered location e.g. 40

LSB = data stored at odd numbered location e.g. 41



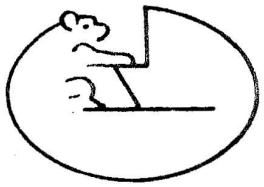
BEAR MICROCOMPUTER SYSTEMS

24 College Road, Maidenhead, Berks, SL6 6BN

HEXADECIMAL CODING FORM

PROGRAM 77-68 FLASHERS	VERSION 1	AUTHOR ML	DATE 18-5-77	PAGE 1 OF 1

ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
VERSION 1				DISPLAY COUNTS UP IN BINARY
00	08	START	INX	APPROX 1 SECOND
01	26 FD		BNE START	DELAY
03	4C		INC A	THEN INCREMENT ACC A
04	97 FF		STA A FF	& DISPLAY RESULT
06	20 F8		BRA START	AND AGAIN (AND AGAIN--)
				NOTE: TRY DEC A, ROL A ETC,
VERSION 2			SIMPLE 'RANDOM' SEQUENCE ON DISPLAY	
00	08	START	INX	
01	26 FD		BNE START	
03	4C		INC A	TRY COMBINATIONS OF DEC A,
04	49		ROL A	ROR, COM, DAA
05	97 FF		STA A FF	
07	20 F7		BRA START	
VERSION 3			VARIABLE SPEED COUNT - CONTROLLED BY SW REG.	
00	5F	START	CLR B	
01	4C	LOOP	INC A	2.4 mS DELAY (SLOWS IT DOWN
02	26 FD		BNE LOOP	SO WE CAN SEE DISPLAY CHANGE)
04	5C		INC B	
05	D1 FF		CMP B FF	B = SWITCH REG ?
07	26 F8		BNE LOOP	IF NOT WAIT ANOTHER 2.4 mS
09	08		INX	DISPLAY (ADDRESS FF) GETS FED
0A	DF FE		STX FE	WITH LOW 8 BITS OF INDEX REG
0C	20 F2		BRA START	



BEAR MICROCOMPUTER SYSTEMS

24 College Road, Maidenhead, Berks, SL6 6BN

HEXADECIMAL CODING FORM

PROGRAM TONE GENERATOR	VERSION 1	AUTHOR ML	DATE 22-5-77	PAGE 1 OF 1

ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
00	4C	START	INC A	WAIT FOR PERIOD DETERMINED
01	91 FF		CMP A FF	BY SW REG SETTING (FF)
03	26 FB		BNE START	
05	5C		INC B	CHANGE STATE OF DATA REG
06	D7 FF		STA B FF	BIT 7
07	4F		CLR A	AND ROUND AGAIN !
09	20 F5		BRA START	

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

HEXADECIMAL CODING FORM

PROGRAM TUNER	VERSION 1	AUTHOR ML	DATE 25-5-77	PAGE 1 OF 1

ADDRESS	MACHINE CODE	LABEL	OPERATOR & OPERAND	COMMENTS
00	CE 00 3E	START	LDX # TUNE-2	
03	08	NUNOTE	INX	POINT INDEX REG AT NEXT
04	08		INX	'NOTE' PAIR OF BYTES.
05	A6 00		LDA A 0,X	GET NOTE DURATION.
07	27 F7		BEQ START	END OF TUNE ?
09	97 31		STA A, TIME	STORE DURATION IN 'TIME'
0B	4F		CLR A	
0C	E6 01	NUCYCL	LDA B 1,X	GET FREQUENCY
0E	27 10		BEQ QUIET	QUIET IF FREQ = 0
10	31		INS	TOGGLE BIT 7 OF DATA
11	9F FE		STS FE	O/P REGISTER
13	4A	CYCLE	DEC A	
14	26 05		BNE DECB	
16	7A 00 31		DEC TIME	NEXT NOTE IF TIME
19	27 E8		BEQ NUNOTE	HAS RUN OUT
1B	5A	DECB	DEC B	
1C	26 F5		BNE CYCLE	END OF ½ CYCLE ?
1E	20 EC		BR NUCYCL	
20	02	QUIET	NOP	
21	02		NOP	
22	02		NOP	FILLERS - MAKE 'QUIET'
23	02		NOP	TIMING APPROX EQUAL
24	02	QCYCL	NOP	TO TONE 'CYCLE'
25	02		NOP	
26	02		NOP	
27	4A		DEC A	
28	26 FA		BNE QCYCL	
2A	7A 00 31		DEC TIME	
2D	27 D4		BEQ NUNOTE	END OF SILENCE ?
2F	20 F3		BRA QCYCL	
31		TIME		TEMP STORAGE FOR TIME BYTE
40		TUNE		'TUNE' LIST STARTS HERE

Chapter 6 Extending the system

Enhancements planned by BMS for the basic 77-68 system include;

- additional memory.
- tape cassette interface.
- VDU and keyboard interface.

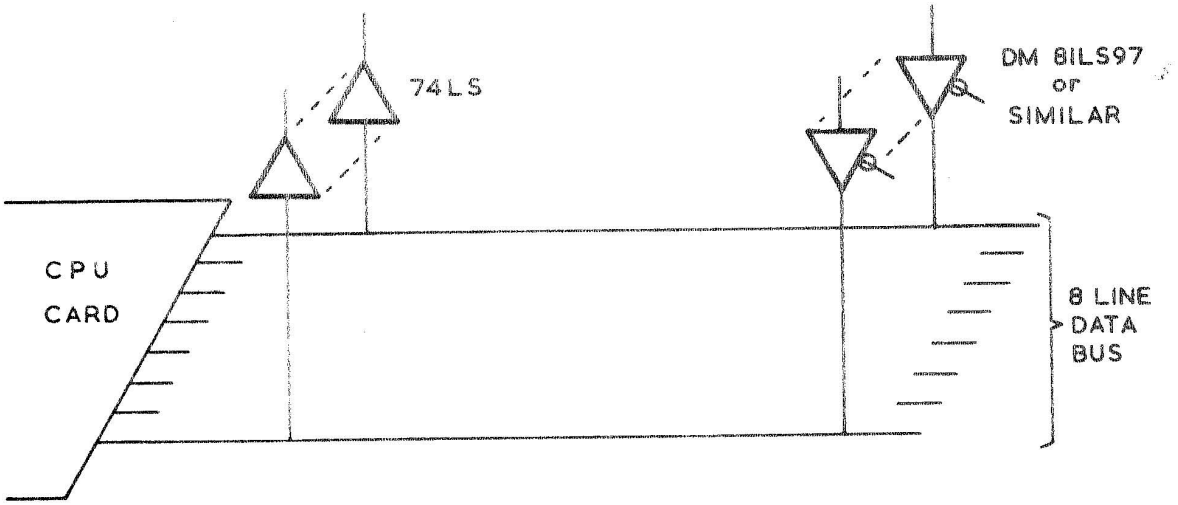
However, the following notes are offered as guidance to the experimenter who wishes to design his own circuitry, or interface the 77-68 CPU board to other computer parts.

77-68 Bus

This is in three main parts;

1. A bidirectional data bus (pins 38-45) which carries data and instructions between the CPU board and memory or peripheral devices. The optional buffers X15 and X16 convert the low power outputs of the 6800 to normal three-state TTL levels.

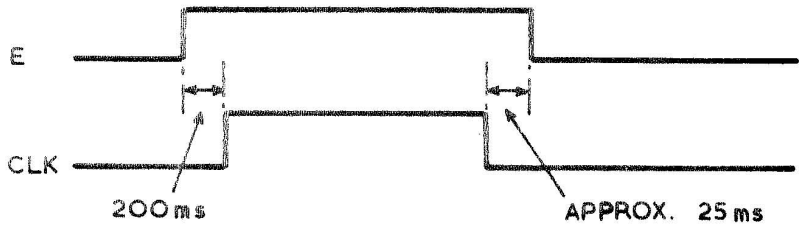
A maximum of 10 standard TTL inputs can be driven by the bus, or 44 low power Schottky TTL inputs. Data from external memory or peripherals should be put on to the bus via a three-state buffer (DM81LS97 or similar), suitably controlled so that no more than one buffer is trying to drive the bus at any time!



2. A unidirectional 16 line address bus (pins 21 - 26) with a maximum drive capability of 9 standard TTL gate inputs (or 40 low power schottky).

3. Miscellaneous timing and bus control signals (pins 4 - 12);

- R/W is a signal from the CPU card which controls the direction of transmission on the data bus. When it is at logical 1 (read) information should flow to the 6800 from memory or peripherals. When at logical 0 (write), data is being sent from the CPU and therefore no other part of the system should be trying to put information onto the bus.
- When extra memory (or peripherals) are added to the basic system, suitable address decoding circuitry should be included to discriminate between those addresses which refer to the external memory (or peripheral) and those which refer to the memory and data register/display on the CPU board. This external address decoding circuitry should pull the '256SEL' line to 0 (low) with a three-state or open collector gate output whenever external memory is selected by the address bus.
- The 'E' output from the CPU board goes to logical 1 when there is a valid address on the address bus lines.
- The $\overline{\text{HOLD}}$ input to the CPU card should normally be high as when pulled low (by the output of a three-state or open collector gate) it will hold the 6800 clock driver in the 'Ø2 high' state. This allows the use of slow external memory, however care should be taken to ensure that the 'Ø2 high' state is not extended beyond the 4.5uS specification limit for the 6800.
- '5MHz' and 'CLK' outputs are provided to drive external circuitry (for example that which may be incorporated to extend the Ø2 time). Note that 'CLK' is equivalent to Ø2 and thus goes to '1' about 200nS later than E. It also changes from '1' to '0' slightly earlier than E and is therefore useful for gating information from the data bus into external memory or peripheral control registers.



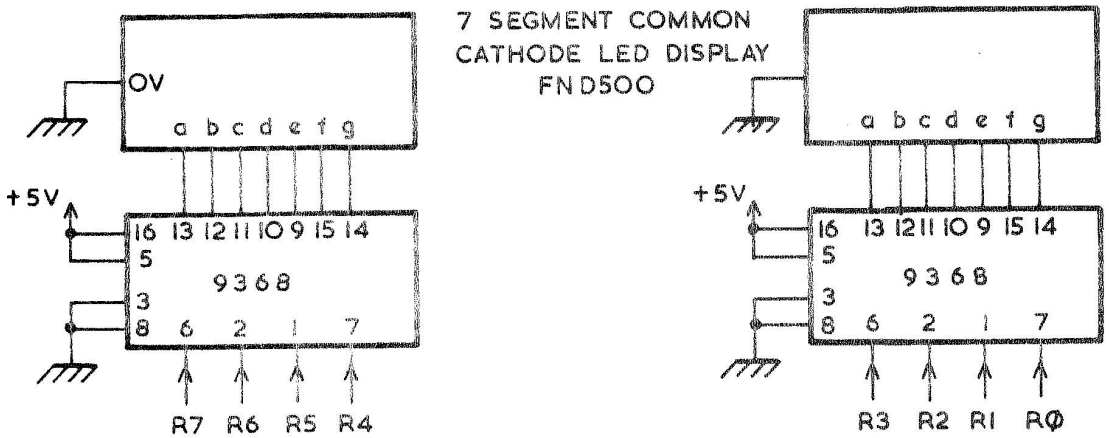
- '$\overline{IRQ}$' and '$\overline{NMI}$' are the 6800 interrupt inputs and are normally at +5V. For details of their use, the constructor should refer to the 6800 data sheets.

Critical bus timing which should be examined when designing external memory or peripheral circuits are given in Fig. 3. The times shown are the worst case values, thus for a normal (i.e. not extended) cycle;

- data read from memory should be valid not later than 800nS after the 0 to 1 transition of E, and not later than 1100nS after the R/W line has settled into the '1' (read) state.
- during a write cycle, the address lines are stable for at least 1300nS before the '1' to '0' transition of E, and the data from the CPU board is valid for at least 1000nS before the '1' to '0' transition of E, and for at least 150nS afterwards.

Spare Data Register Outputs R0 - R7

These are brought out to the board edge connector for use as the constructor sees fit. For example, a two digit hexadecimal display using 7 segment LED displays may be added as shown below;



Alternatively, these outputs could be used to drive relays (model train control), a loudspeaker ('music' generation), a digital to analogue converter (waveform synthesis) or in whatever way the user wishes.

Chapter 7. User Group

77-68 is a "live" project. It is a small start to a large system (by microcomputer standards). Designs for other boards will be made. Some by the "Bear" and some no doubt by users. News of all the latest developments and available software will be held by the User group and circulated by the quarterly newsletter. Contained in the newsletter will be a register of all the names and addresses of users, this is to enable a free discussion between constructors as to the merits and deficiences of the system.

The purchase of this book plus registration gives the individual a year's free subscription to the user group.

It is hoped that this "back-up" support will encourage those individuals with a non-technical background to take the plunge .....

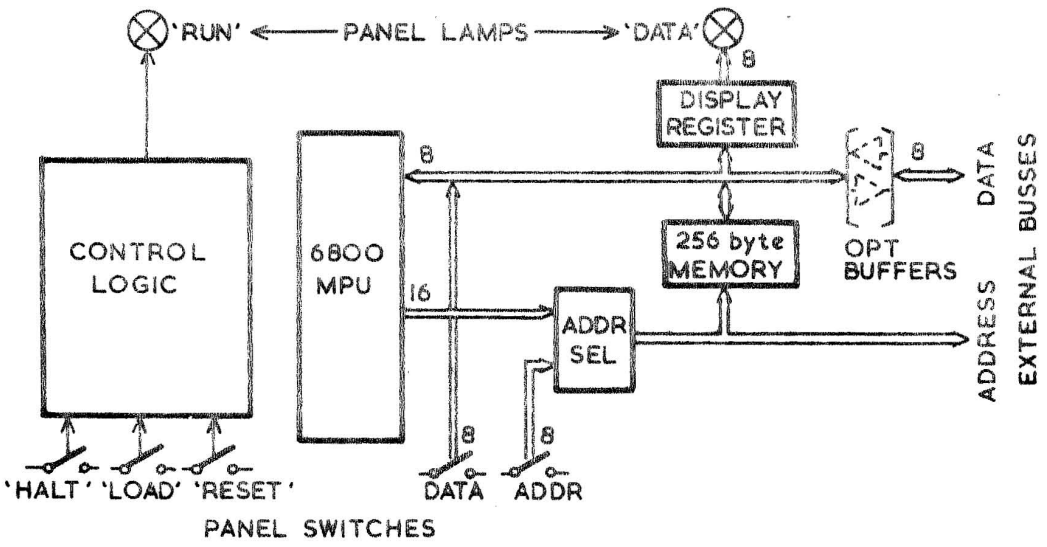


FIG.1 77 68 BLOCK DIAGRAM

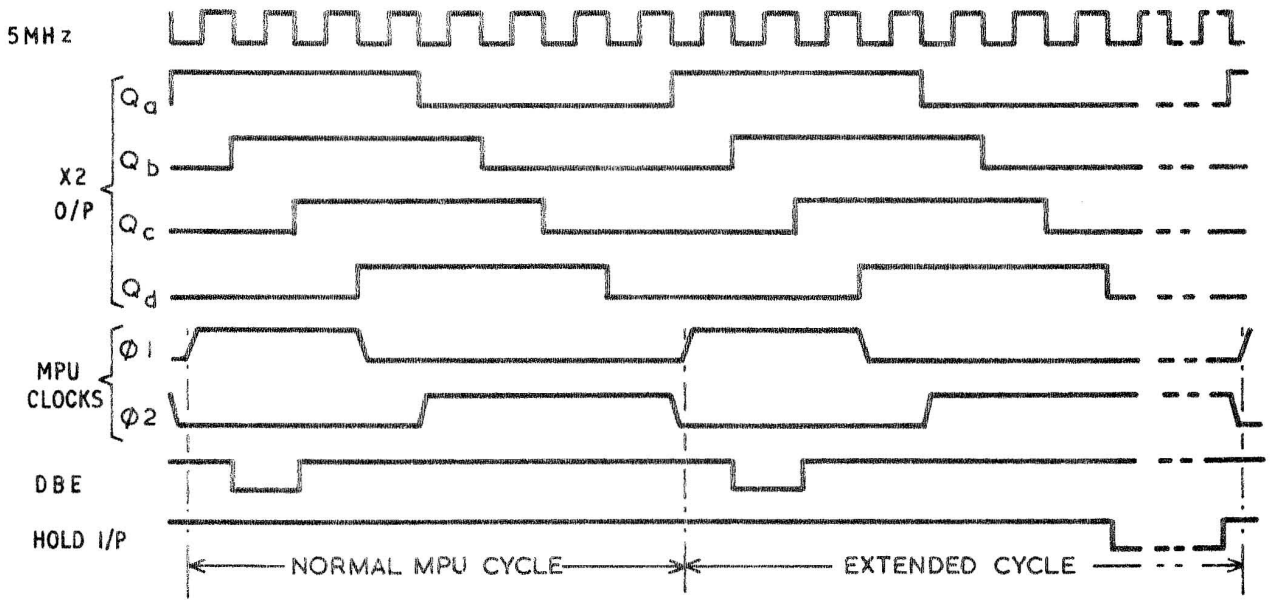


FIG.2 MPU CLOCK WAVEFORMS

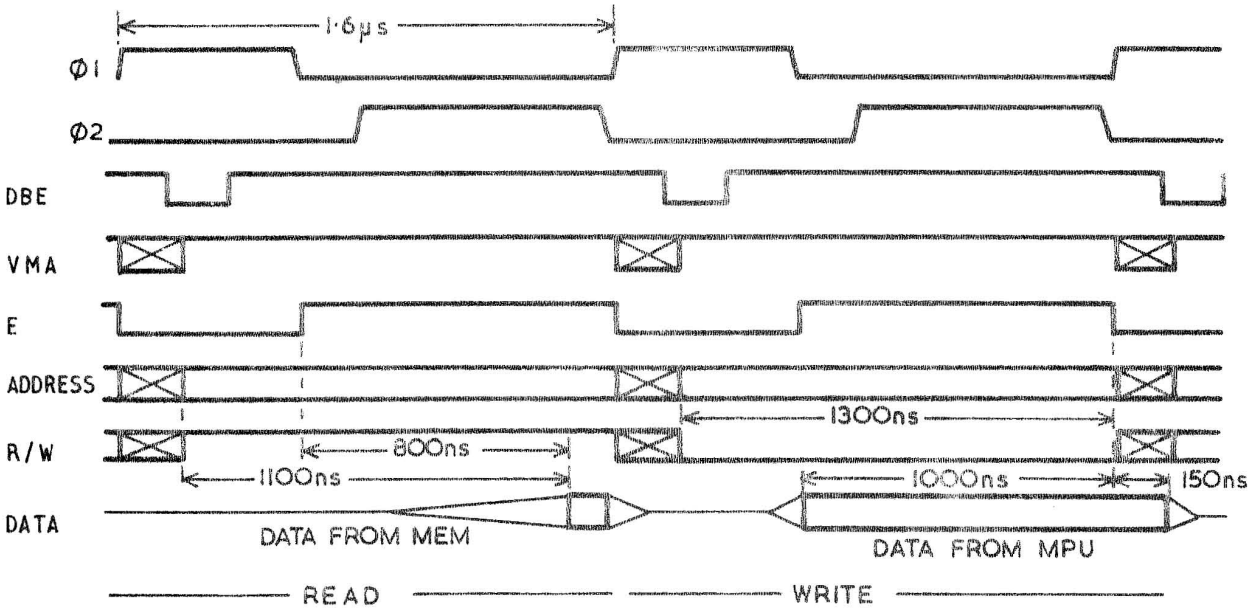
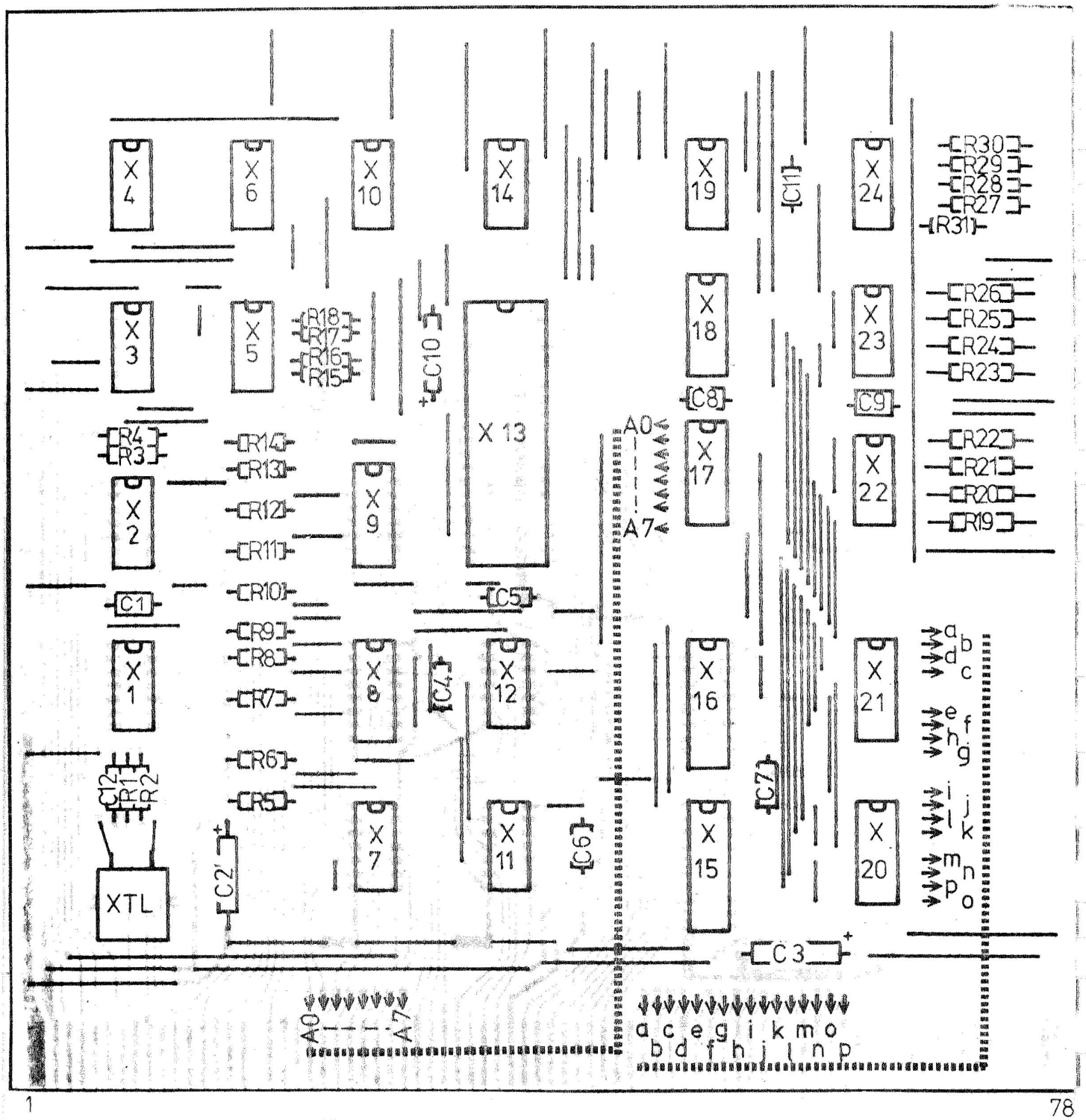


FIG. 3 IMPORTANT BUS TIMINGS FOR EXTERNAL MEMORY



77-68 CPU BOARD - VIEWED FROM COMPONENT SIDE

Showing positions of components and straps.
Join A0 - - A7 to A0 - - A7, and a,b,c etc. to a,b,c (8 way ribbon cable makes a neat job; use solid cored type such as Doram 10 way miniature cable No 357-491)

Note: all IC's are the same way round, all straps are parallel to board edges.

FIG. 4 CPU BOARD - LAYOUT OF MAJOR COMPONENTS

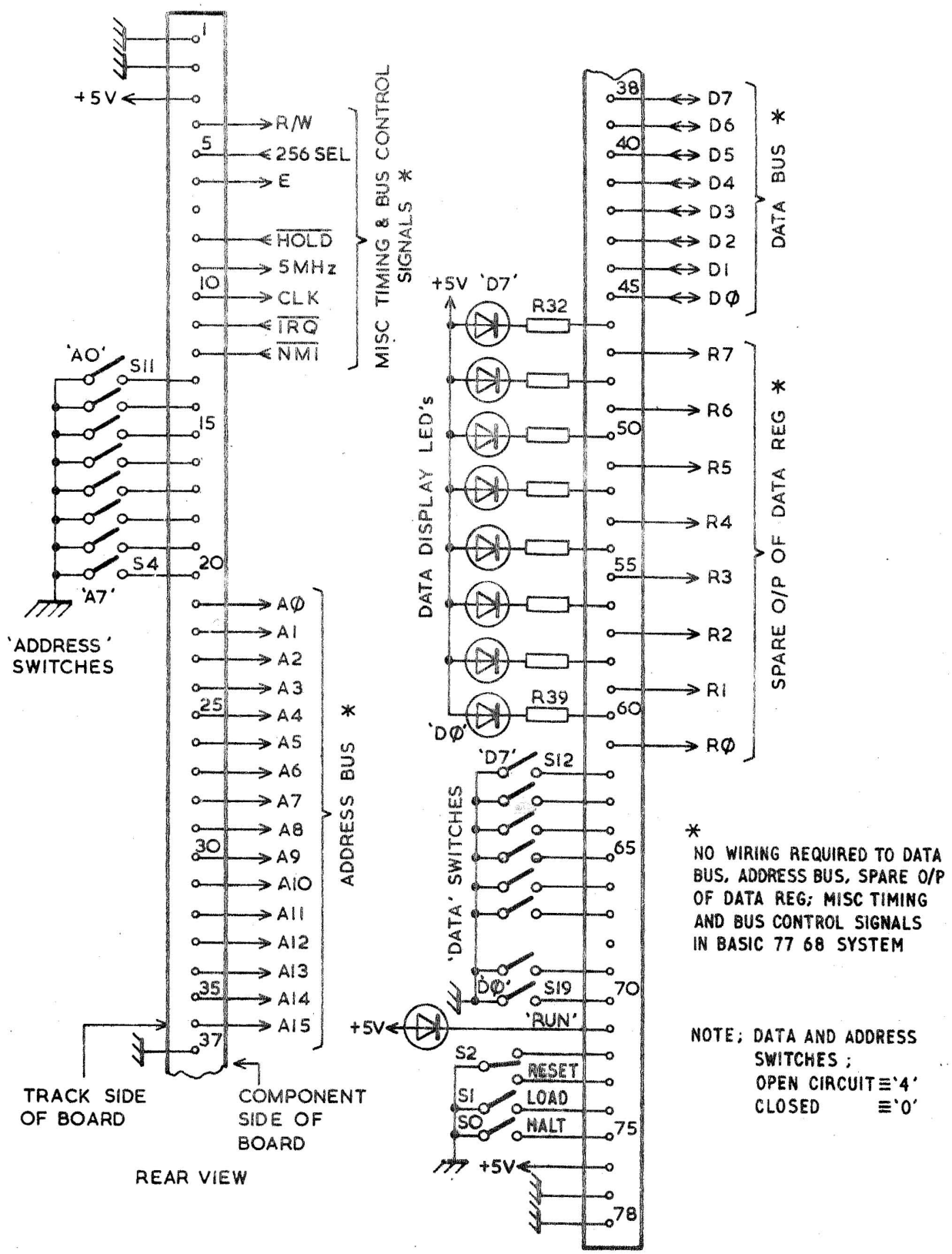


FIG.5 77-68 CPU BOARD CONNECTOR WIRING

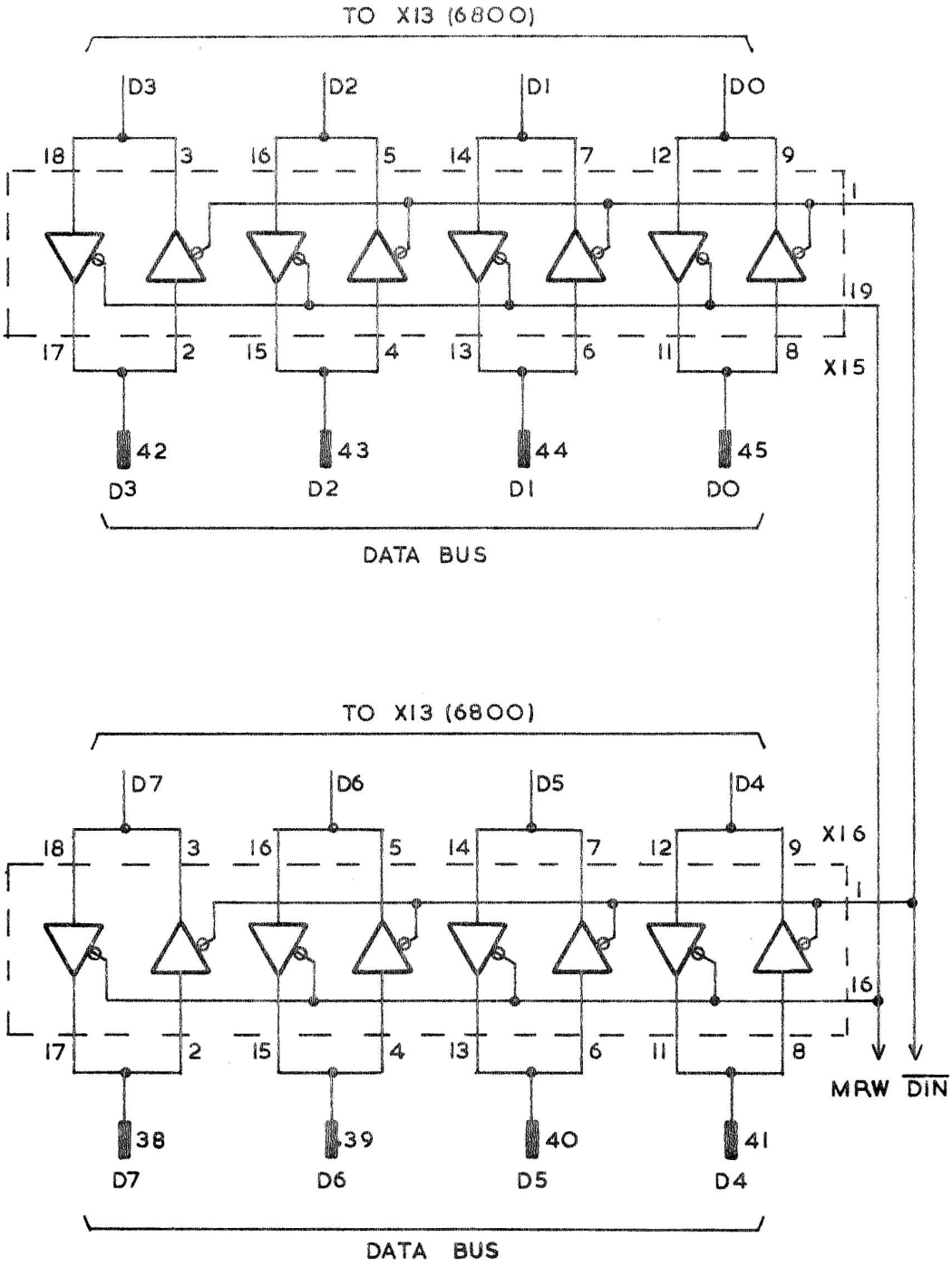
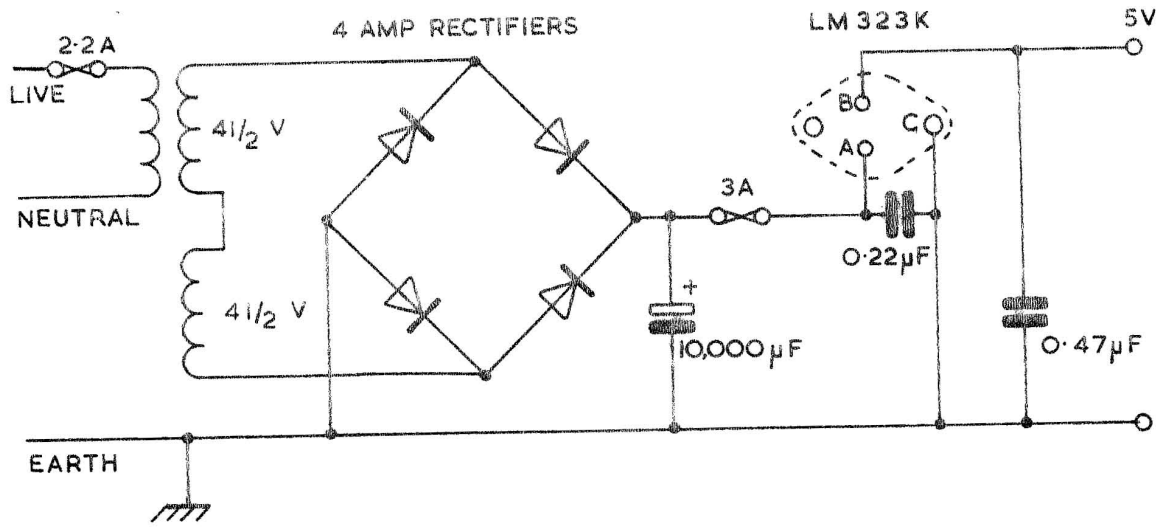


FIG.6 OPTIONAL BUFFERS DM81LS97



OR

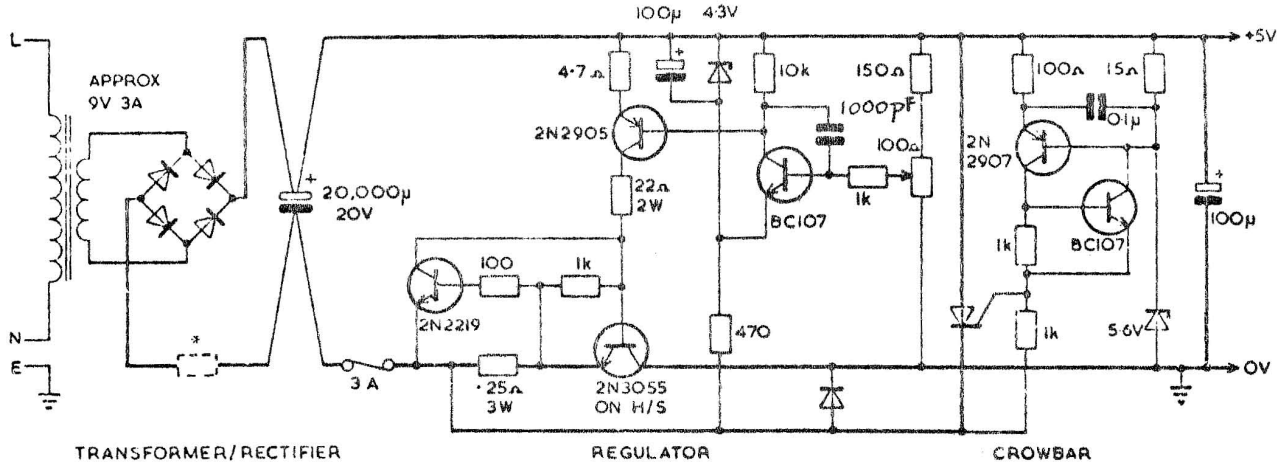
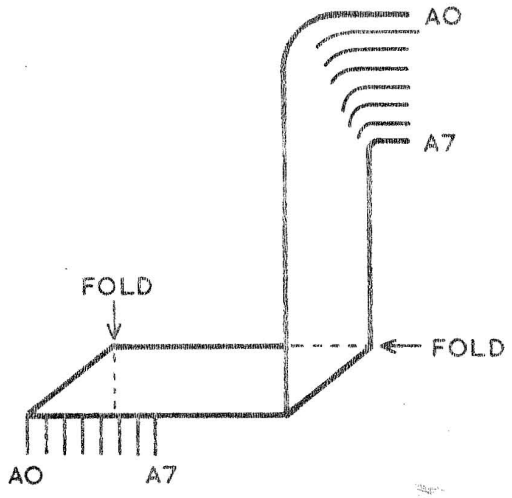
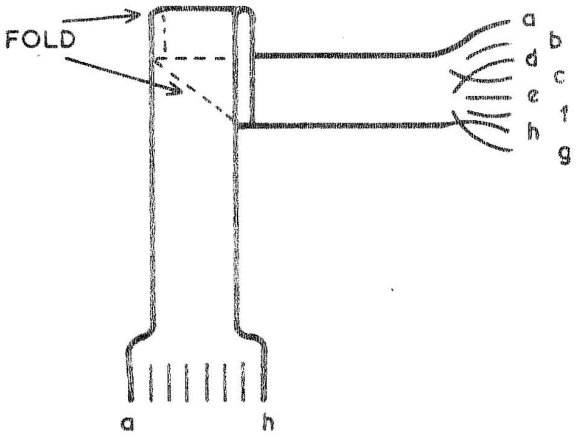
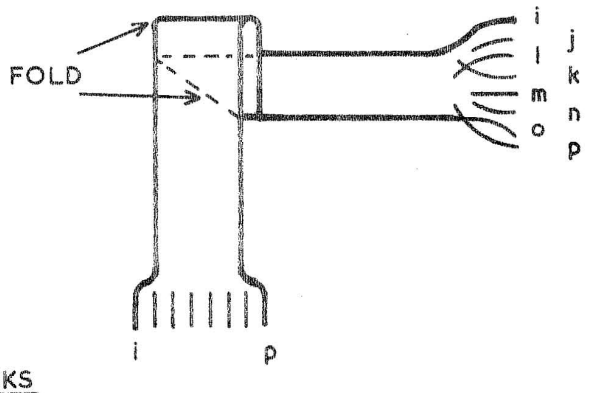


FIG.7 POWER SUPPLY







FITTING WIRE LINKS

The easiest way is to measure sleeving to length, slide on to length of 22 S.W.G. wire leaving $1/2$" bare wire either end, bend and insert.

FIG. 8 NOTES ON THE P.C.B. WIRING 8 WAY RIBBON CABLE

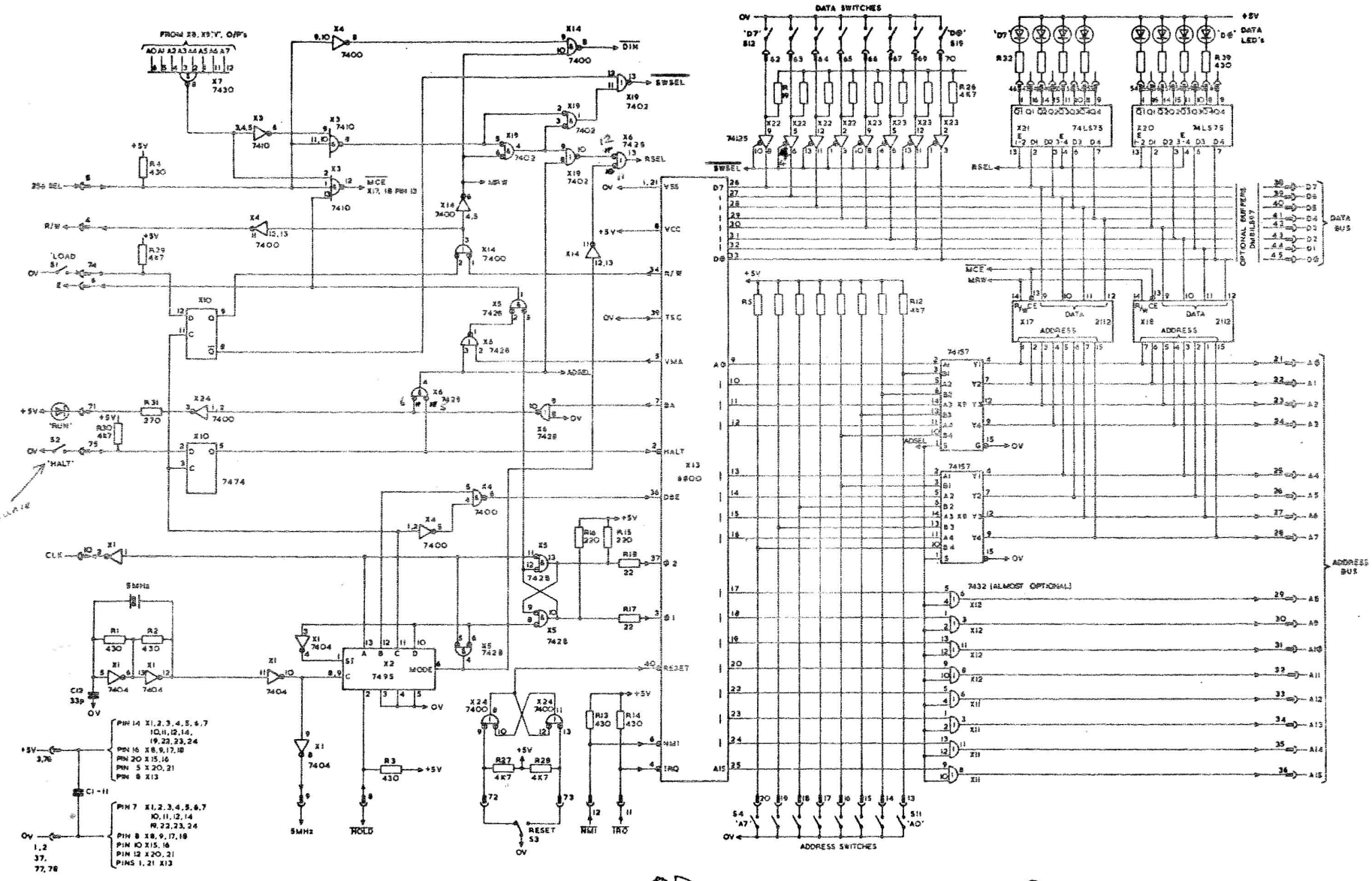


FIG. 9

© BEAR MICROCOMPUTER SYSTEMS 1977

Table 1

77-68 Component Distribution

Logic

X1	7404	X13	6800
X2	7495	X14	7400
X3	7410	X15	81LS97
X4	7400	X16	81LS97
X5	7428 or 7402*	X17	2112) any type
X6	7428 or 7402*	X18	2112) & speed
X7	7430	X19	7402
X8	74157	X20	74LS75
X9	74157	X21	74LS75
X10	7474	X22	74125
X11	7432	X23	74125
X12	7432	X24	7400

Resistors (all ½ Watt)

R1	430	R20	4.7K	
R2	430	R21	4.7K	
R3	430	R22	4.7K	
R4	430	R23	4.7K	
R5	4.7K	R24	4.7K	
R6	4.7K	R25	4.7K	
R7	4.7K	R26	4.7K	
R8	4.7K	R27	4.7K	
R9	4.7K	R28	4.7K	
R10	4.7K	R29	4.7K	
R11	4.7K	R30	4.7K	
R12	4.7K	R31	270	
R13	430	R32	430)	
R14	430	R33	430)	
R15	220 or 1K*	R34	430)	
R16	220 or 1K*	R35	430)	not mounted on
R17	22	R36	430)	p.c.b.
R18	22	R37	430)	
R19	4.7K	R38	430)	
		R39	430)	

Capacitors

C1	0.1uF	C8	0.1uF
C2	33uF+	C9	0.1uF
C3	33uF+	C10	33uF+
C4	0.1uF	C11	0.1uF
C5	0.1uF	C12	33pF
C6	0.1uF		
C7	0.1uF	+ or similar small electrolytics	

LED's 9 off

Crystal 5 MHz

Switches

S1	Single pole n.o. push	S2,S4-S11	Single pole on/off toggle
S3	Single pole c.o. push		

Power Supply

+5V at 1 Amp. See separate diagram.

* option

Table 2

77-68 Components List

Logic

3 off	7400	X4, X14, 24	1 off	7495	X2
1 off	7402	X19	2 off	74125	X22, X23
1 off	7404	X1	2 off	74157	X8, X9
1 off	7410	X3	*2 off	81LS97	X15, X16
2 off	7428	X5, X6	2 off	2112	X17, X18
1 off	7430	X7	1 off	6800	X13
2 off	7432	X11, X12			
1 off	7474	X10			
2 off	74LS75	X20, X21			

* optional buffers

Resistors

20 off	4.7k	½w
*2 off	1k	½w
14 off	430	½w
1 off	270	½w
2 off	220	½w
2 off	22	½w

Capacitors

3 off	33uF	6.3V tant
8 off	0.1uF	ceramic
1 off	33pF	

L.E.D.'s 9 off

Crystal 5 MHz

Switches

17 subminiature toggle switches
single pole 2 way S3-19
1 push single pole n.o. S1
1 push single pole c.o. S2

* option

Low profile sockets 2 off 16 pin

1 off 40 pin

3 feet 22 s.w.g. wire and sleeving

1 foot 8 way ribbon cable

Printed Circuit Board

Edge Connector, 77 way + polarising key 0.1" single sided.

ACCUMULATOR AND MEMORY INSTRUCTIONS

14 AND A ← B

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED =	DIRECT OP	DIRECT ~	DIRECT =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	3B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑤	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑤	↕
	ROLB													59	2	1	B — C b7 b0	•	•	↕	↕	⑤	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑤	↕
	RORA													46	2	1	A	•	•	↕	↕	⑤	↕
	RORB													56	2	1	B — C b7 b0	•	•	↕	↕	⑤	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑤	↕
	ASLA													48	2	1	A — 0 C b7 b0	•	•	↕	↕	⑤	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑤	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B — b7 b0 C	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A — 0 b7 b0 C	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Acmltr	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

LEGEND:

OP	Operation Code (Hexadecimal);	+	Boolean Inclusive OR;
~	Number of MPU Cycles;	⊕	Boolean Exclusive OR;
=	Number of Program Bytes;	M̄	Complement of M;
+	Arithmetic Plus;	→	Transfer Into;
−	Arithmetic Minus;	0	Bit = Zero;
·	Boolean AND;	00	Byte = Zero;
M_{SP}	Contents of memory location pointed to be Stack Pointer;		

Note Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS:

H	Half-carry from bit 3;
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry from bit 7
R	Reset Always
S	Set Always
↕	Test and set if true, cleared otherwise
•	Not Affected

Table 3 6800 Instruction Set (i)

INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	⑦	↕	⑦	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BRANCH TEST	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										02	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	⑪	•	•	•	•

CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

OPERATIONS	MNEMONIC	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	$0 \rightarrow C$	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	$0 \rightarrow I$	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	$0 \rightarrow V$	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	$1 \rightarrow C$	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	$1 \rightarrow I$	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	$1 \rightarrow V$	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	$A \rightarrow CCR$	⑫					
CCR → Acmltr A	TPA	07	2	1	$CCR \rightarrow A$	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES:

(Bit set if test is true and cleared otherwise)

1 (Bit V) Test: Result = 10000000?
2 (Bit C) Test: Result = 00000000?
3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4 (Bit V) Test: Operand = 10000000 prior to execution?
5 (Bit V) Test: Operand = 01111111 prior to execution?
6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.
7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?
8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?
9 (Bit N) Test: Result less than zero? (Bit 15 = 1)
10 (All) Load Condition Code Register from Stack. (See Special Operations)
11 (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
12 (All) Set according to the contents of Accumulator A.

Table 3 6800 Instruction Set (ii)

1	0.V		40	D5
2	0.V		41	D4
3	+5V		42	D3
4	R/W		43	D2
5	SEL		44	D1
6	E		45	DØ
7	blank		46	$\overline{D7}$
8	$\overline{HOLD}$		47	D7
9	5MHZ		48	$\overline{D6}$
10	$\overline{CLK}$		49	D6
11	$\overline{IRQ}$		50	$\overline{D5}$
12	$\overline{NMI}$		51	D5
13	AØ		52	$\overline{D4}$
14	A1		53	D4
15	A2		54	$\overline{D3}$
16	A3		55	D3
17	A4	Address switches	56	$\overline{D2}$
18	A5		57	D2
19	A6		58	$\overline{D1}$
20	A7		59	D1
21	AØ		60	$\overline{DØ}$
22	A1		61	DØ
23	A2		62	D7
24	A3		63	D6
25	A4		64	D5
26	A5		65	D4
27	A6		66	D3
28	A7	Address Bus	67	D2
29	A8		68	Blank
30	A9		69	D1
31	A1Ø		70	DØ
32	A11		71	'RUN' LED
33	A12		72	RESET
34	A13		73	
35	A14		74	LOAD
36	A15		75	HALT
37	0.V		76	+5V
38	D7		77	0V
39	D6		78	0V

Data Bus

Data LED's & Spare Data Reg Outputs

Data Switches

To Control Switches

Table 4 77-68 pin assignment - edge connector

Appendix 1

Binary, Decimal and Hexadecimal Notations

Although the 77-68, like most other computers, actually handles information in binary form, humans have great difficulty in remembering long strings of '1's and '0's. However, by taking groups of 4 binary bits and expressing each group in 'hexadecimal' (base 16) format, we can write binary patterns in a more readable form.

Binary MSB LSB	Hexadecimal representation
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F

Notes; MSB = most significant of 4 binary bits
LSB = least significant

Thus each 8 bit word can be expressed as two hexadecimal characters;

Binary (e.g. inf as stored in computer or set on data switch reg or displayed) : 1 0 1 0 0 0 1 1

Hexadecimal (as written for human use) : A 3

And a 16 bit address as four hex characters;

0 1 1 0 0 0 0 1 1 1 1 1 1 0 1 0

6 1 F A

Generally the information contained in an eight bit word is considered to be a positive (unsigned) integer in the range 0 to 255 (decimal). (In practice the information may instead be an ASCII coded character, or two BCD digits, or whatever else the user decides - but ignore these possibilities for the moment). Then each binary bit in the word has a particular value; B0 (least significant) = 1, B1 = 2, B2 = 4 - - - B7 (most significant) = 128. Thus for example;

1 0 0 0 0 0 1 1 = 128 + 2 + 1 = 131 (decimal)

However, this binary word would be written in hexadecimal as '83'.

Conversion between hexadecimal representations and their decimal equivalents may be done by reference to the following table;

Hex Character	Equivalent decimal	
	(Hex char on left)	(Hex char on right)
0	0	0
1	16	1
2	32	2
3	48	3
4	64	4
5	80	5
6	96	6
7	112	7
8	128	8
9	144	9
A	160	10
B	176	11
C	192	12
D	208	13
E	224	14
F	240	15

Thus Hex '83' = decimal 128 + 3 = 131

Note again that this conversion table applies only to unsigned positive integers stored in natural binary form.