Orton Mk14 computer manual

Index

Disclaimer

The information in this document is provided purely as a record of the construction of one of my computer projects. I provide no assurance or guarantee whatsoever as to the accuracy, safety or originality of the contents of this document. I provide no warranty whatsoever as to the suitability for any purpose of the contents of this document. I accept no responsibility whatsoever for any deaths, injuries or losses resulting from the use of the information contained in this document. I have no claims to most of the technology used in my projects or described in this document, indeed, it must be assumed that I have used much copyrighted and/or patented material. But since I do what I do for purely recreational, educational or personal instructional purposes, I do not believe that I am breaking the law. It is up to the individual using the information in this document to determine whether, and to ensure that, their use of the information I provide is both legal and safe. This is supposed to be fun.

Karen Orton 2018

Chapter 1 Unexpanded Mk14 PIC emulator

Engineers of my generation in the UK will remember the Science of Cambridge Mk14. Intended to be Sir Clive Sinclair's first home computer kit for the general public, it served to train a generation of electronics engineers in how to program microprocessors. The Mk14 was based on, and virtually identical to, the 'Introkit' that was available in America at around the same time. It had a 512 byte monitor ROM, 256 bytes of RAM, an eight or nine digit LED display with bubble magnifiers, and a notoriously bad twenty key keyboard.

My PIC emulator is a 'cycle perfect' emulation of an unexpanded Mk14. My emulator executes every SC/MP instruction in the same time that it would on a real Mk14. However, the emulation departs in a number of ways from true Mk14 behaviour:

- The memory map of my emulator is not identical to the Mk14 although it is broadly compatible insofar as the ROM, display and standard RAM are to be found in the originally specified Mk14 locations. The emulator's memory map is shown in an associated table which includes the Mk14 memory map alongside for comparison. Note that items not present in the unexpanded Mk14 are shown in brackets. Note also that, although the RAM features several times in the emulator memory map, they are just copies – there is only 256 bytes of RAM in total.

- Sense A interrupts are not implemented.

- SC/MP paging is not implemented. Both program counter increment and pointer arithmetic will carry normally into the top four bits of any address calculation. Mk14 behaviour follows from the fact that these four bits play no part in address decoding. Nonetheless, if a program examines these bits, they may be found to be different to what they would be on an Mk14.

- The display column latch is not updated by reads of the display.

- The emulation is of a 4MHz SC/MP. The Mk14 had a 4.43MHz crystal and clock. In practise this will just cause programs to run 11% slower than they would on an Mk14. Programs that synthesise tones on the flag outputs, for example, will still produce clean tones but they will be around 2 semitones lower in pitch.

- Illegal SC/MP instructions cause a reset. This is actually an advantage – in the Mk14 a program crash went completely undetected and often lead to total RAM obliteration.

- The emulation is cycle perfect only for programs executing in RAM. ROM execution runs slightly slower due to approximately an extra cycle overhead on program fetches from ROM (note however, that data reads from ROM by programs running in RAM execute at the correct speed). Despite ROM

execution being slower, the cassette routines supplied in the ROM have been tested and found to work okay.

One additional nice feature: the RAM, as programmed through the SCIOS monitor, is non-volatile. This means that programs are retained through power cycles. It also means that a program can be recovered if it overwrites itself.

Emulator memory map

Address	Mk14	PIC emulator
000	SCIOS	SCIOS
100		
200	SCIOS	SCIOS
300		
400	SCIOS	SCIOS
500		
600	SCIOS	SCIOS
700		
800	(RAMIO)	DISPLAY
900	DISPLAY	DISPLAY
A00	(RAMIO)	RAM
B00	(Expansion RAM)	RAM
C00	(RAMIO)	DISPLAY
D00	DISPLAY	DISPLAY
E00	(RAMIO)	RAM
F00	Standard RAM	RAM

Finally, my emulator brings the SC/MP flags out to a 9 way female 'D' connector for attachment of external peripherals (e.g. cassette interface):

Pin	Function
1	GND
2	SIN
3	SA
4	F2
5	F0

6	SOUT
7	SB
8	F3
9	F1

See Appendix 1 for the circuit of my emulator. The observant will notice a fourth flag (Flag F3). This is actually the SC/MP IE status bit, but since Sense A interrupts are not supported by the emulation, this bit has been made available as a fourth flag. This flag must not be used if backward compatibility is to be maintained. Note that the IEN and DINT instructions do not affect this flag output. There are two additional features of my emulator:

Loudspeaker and volume control	An LM386 amplifier IC was added to allow tones generated on Flag F0 to be heard.
SC/MP flag monitor LEDs	The states of the SC/MP flags are shown on a column of LEDs.

Chapter 2 Cassette recorder

Science of Cambridge sold a cassette interface for the Mk14 which uses a simple amplitude modulation scheme for recording of binary data. In conjunction with routines supplied in the SCIOS ROM, it is possible to save and load programs or data at a rate of 4 bytes per second (32 baud). This is slow even by the standards of its day, but quite adequate for programs residing in the Mk14's 256 byte memory. The interface could actually be used at 110 baud, allowing use at teletype speed.

A reprint of the original documentation for this adapter is supplied in Appendix 4. I have built this circuit into a case containing a cassette mechanism and associated record/play electronics. The complete unit is able to interface to the required SC/MP flags through a 9 way female 'D' connector and suitable lead.

Note that my Mk14 PIC emulator includes a serial download facility to allow Intel hex files to be loaded into the Mk14 memory. I have not used this in my most recent Mk14 emulator since I planned on using the cassette interface for program loading and retrieving. But the feature is still present in the emulator source code and can be re-enabled by setting an assemble time switch and re-assembling as follows:

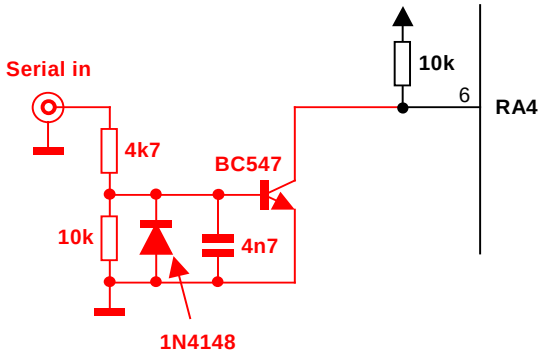
```
SERIAL  EQU     0       ; Intel hex serial download disabled.

                            or

SERIAL  EQU     1       ; Intel hex serial download enabled.
```

If enabled, the feature is entered automatically on power-up. If this feature is enabled but not required, then it can be exited by pressing the SC/MP reset key, in which event the RAM will be loaded from non-volatile memory. Intel hex files should be transferred at 9600 baud, 8 data bits, 1 stop bit and no parity. As the link is one way, no flow control or handshake should be enabled for the file transfer. The Mk14 emulator proper will be entered once the hex file download is complete.

The following additional components are required to add a serial port for Intel hex file download (additional components are shown in red):



The serial input expects standard RS232 levels.

Chapter 3 Oscilloscope adapter

Science of Cambridge sold a 'VDU' kit for the Mk14 which displayed low resolution graphics or text on an attached TV. This was quite an impressive design - it used direct memory access to read pixel/text data straight out of Mk14 memory. The problem with using a peripheral such as this is memory - firstly to store the image data and secondly to hold the code to drive it. With an unexpanded Mk14 it is not possible to realise the full capabilities of this VDU device.

As an alternative, an oscilloscope adapter has been constructed for the Mk14 which was inspired by the PDP-1 and its 'Type 30 precision CRT display'. This adapter allows multiple points to be plotted onto an attached oscilloscope. The number of points is limited only by available Mk14 memory and refresh rate considerations. The adapter uses a pair of integrators to hold point coordinates and is controlled by SC/MP flags as follows:

Flag F0	Pulsing this flag high resets the two integrators in preparation for new coordinates.
Flag F1	Pulsing this flag high defines the X coordinate, according to the pulse width.
Flag F2	Pulsing this flag high defines the Y coordinate, according to the pulse width.
SOUT	Holding this SC/MP output high forces the oscilloscope beam off-screen so that coordinates can be reprogrammed without showing trails.

The pulse width for X and Y coordinate programming should be in the range 47 microseconds (leftmost X / bottommost Y) and 173 microseconds (rightmost X / topmost Y). This range can be generated in 64 discrete steps using the SC/MP DLY instruction. The X and Y outputs from the adapter are approximately in the range +/- 3 Volts.

Chapter 4 New programs for the Mk14

Alarm clock program

A version of the ‘Alarm Clock’ example found in the Mk14 manual has been produced which is cycle trimmed for a 4MHz SC/MP.

Babbage difference engine program

Some would argue you can’t do anything in 256 bytes of RAM. To show just what is possible, I have written a Babbage difference engine emulation. I wrote this partly to demonstrate just what can be done with apparently meagre hardware, but also to provide a test of the emulator, in particular, the decimal addition algorithm. The Babbage difference engine was remarkable because it accomplished a lot with no more than simple addition. It used the principle of differences to generate high order polynomials.

To illustrate how this works, consider the simple situation where we keep adding a constant to a number. We know that the number will keep escalating linearly (e.g. 0, 1, 2, 3, 4…). Now, suppose the number we add is not a constant, but itself is escalating. The sequence we then produce is 0, 1, 3, 6, 10, 15. This sequence is actually a parabola – a second degree polynomial. By cascading stages of adders in this way, Babbage was able to synthesise polynomials of any order, typically tenth order.

The example Babbage.asm uses tenth order differences but only sixteen digits of precision. Babbage knew that resolution at high polynomial order was going to be a problem, and the real difference engines used thirty digits. The example has the engine pre-primed for generation of the function 4.arctan(x) over the interval 0.0..1.0, in fifty steps. If the run is successful, it should finish on the value 4.arctan(1) which is π. Even with just tenth order and sixteen digits, the example arrives at π to within about 11 parts per million.

Pong program

To demonstrate the oscilloscope adapter, I have written a simple, single player 'pong' game. The game uses three of the buttons on the Mk14 keypad as follows:

7	Ball serve
Term	Move bat to left
F	Move bat to right

Chapter 5 Photos

Chapter 6 Software

There are a number of software files associated with the Orton Mk14 computer:

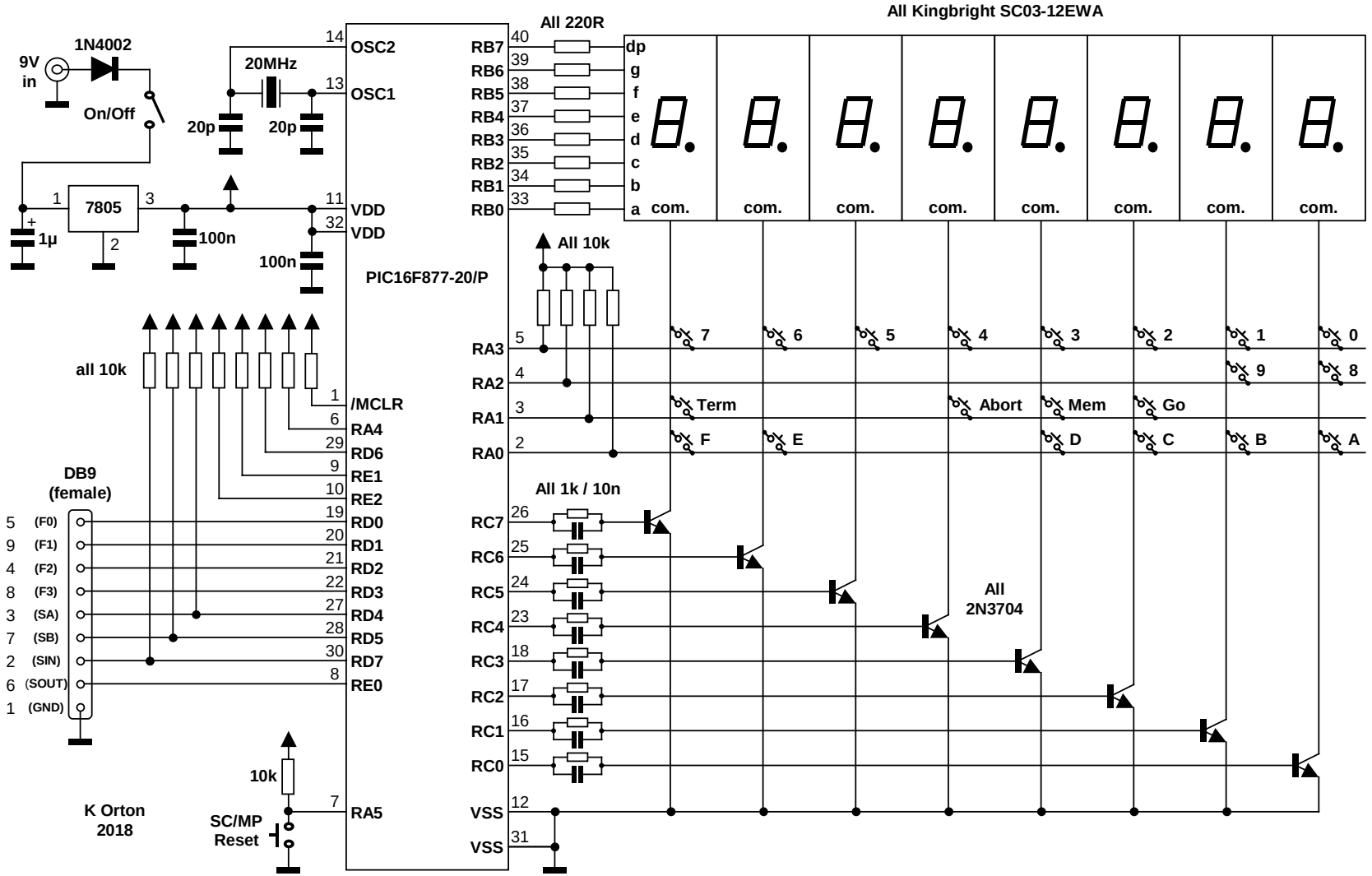
PIC14.asm	This is the Orton Mk14 computer emulator source file. The emulation runs on a PIC16F877-20/P microcontroller.
Clock.lst	This is the listing file* for the 'Alarm Clock' program from the Mk14 manual, cycle trimmed for a 4MHz SC/MP.
Babbage.lst	This is the listing file* for the Babbage difference engine program.
Pong.lst	This is the listing file* for the Pong program.

* Listing files have been supplied because they include hexadecimal machine code for typing directly into an Mk14.

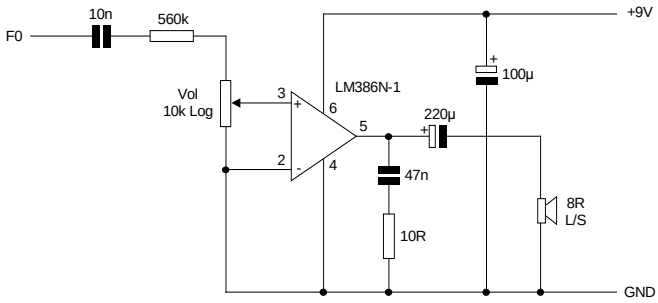
Acknowledgement

The Mk14 program listing files were generated using the very excellent AS macro assembler (V1.42) which was developed by Alfred Arnold, Stefan Hilse, Stephan Kanthak, Oliver Sellke and Vittorio De Tomasi.

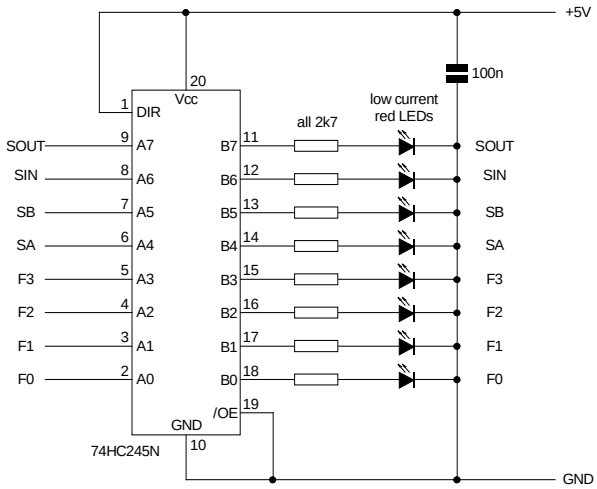
Appendix 1 Emulator circuit



Audio amplifier addition



Monitor LED addition



Port A assignments

RA0	Active low keyboard row sense (as per D4 data bus assignment on Mk14)
RA1	Active low keyboard row sense (as per D5 data bus assignment on Mk14)
RA2	Active low keyboard row sense (as per D6 data bus assignment on Mk14)
RA3	Active low keyboard row sense (as per D7 data bus assignment on Mk14)
RA4	(not used – tie high through 10k)
RA5	Active low SC/MP reset

Port B assignments

RB0	Active high segment a drive
RB1	Active high segment b drive
RB2	Active high segment c drive
RB3	Active high segment d drive
RB4	Active high segment e drive
RB5	Active high segment f drive
RB6	Active high segment g drive
RB7	Active high segment dp drive

Port C assignments

RC0	Active high column 0 drive (rightmost)
RC1	Active high column 1 drive
RC2	Active high column 2 drive
RC3	Active high column 3 drive
RC4	Active high column 4 drive
RC5	Active high column 5 drive
RC6	Active high column 6 drive
RC7	Active high column 7 drive (leftmost)

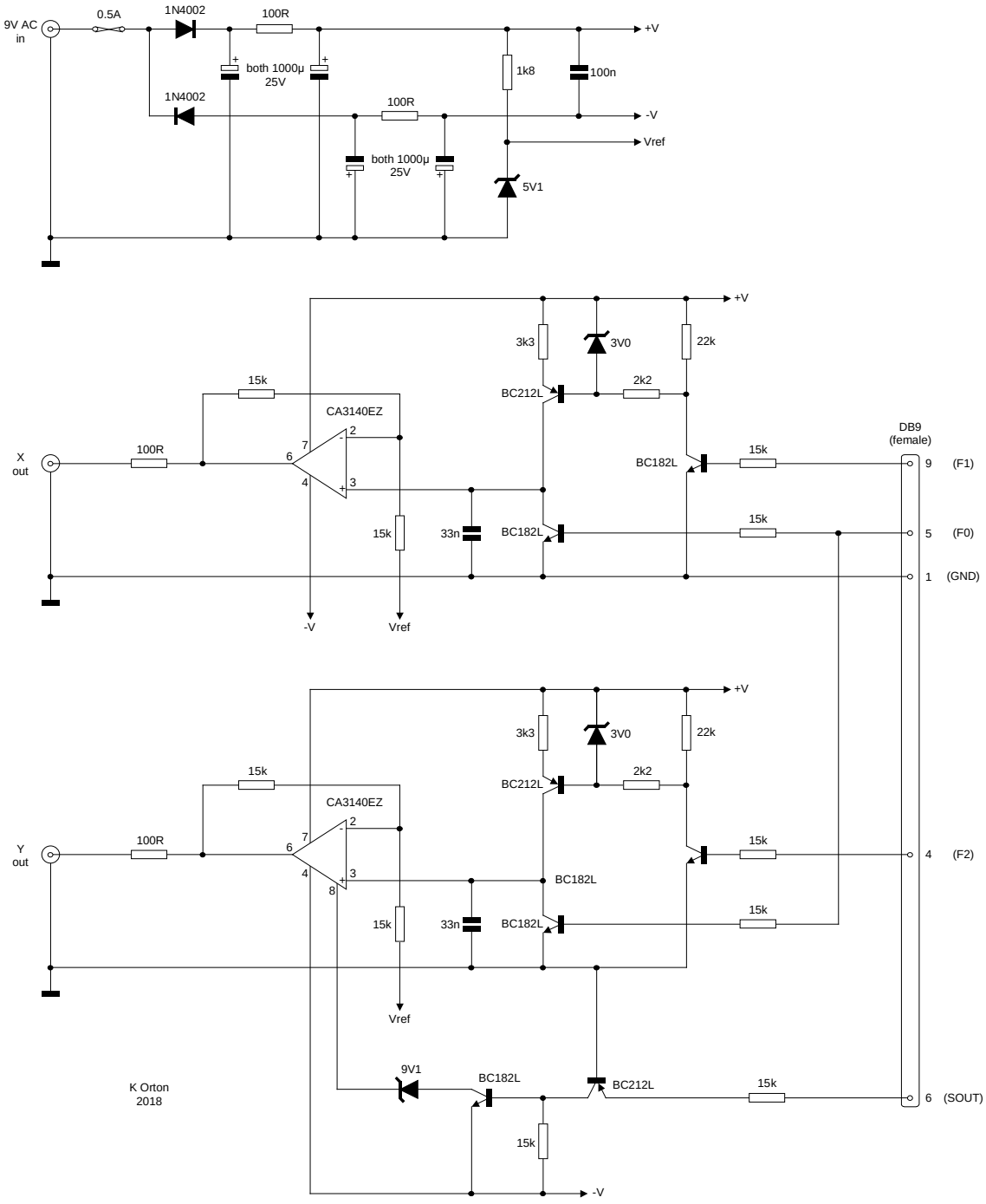
Port D assignments

RD0	Flag F0
RD1	Flag F1
RD2	Flag F2
RD3	Flag F3 Actually SC/MP IE status bit, but available for use as a flag.
RD4	Sense A
RD5	Sense B
RD6	(not used – tie high through 10k)
RD7	SIN

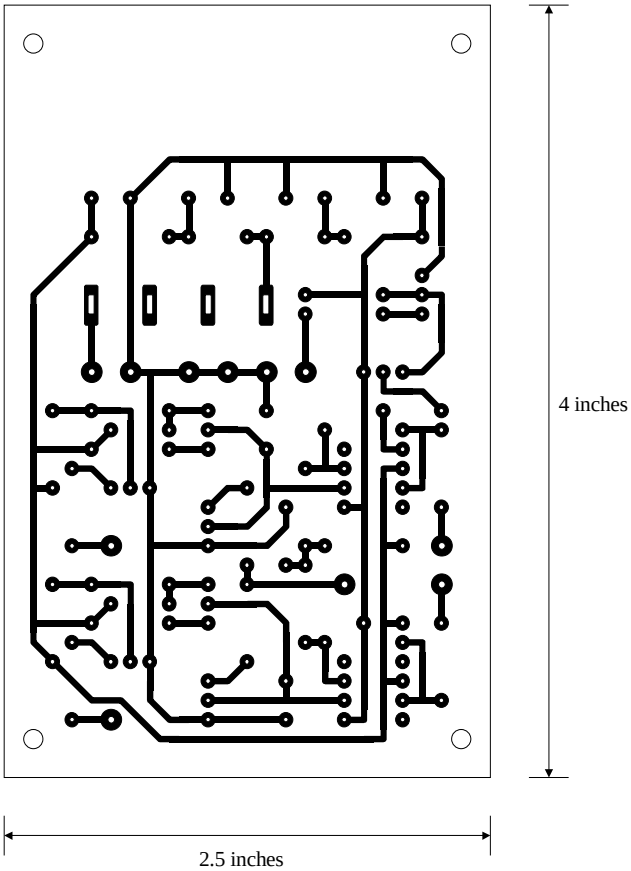
Port E assignments

RE0	SOUT
RE1	(not used – tie high through 10k)
RE2	(not used – tie high through 10k)

Appendix 2 Oscilloscope adapter circuit



Appendix 3 Oscilloscope adapter PCB artwork



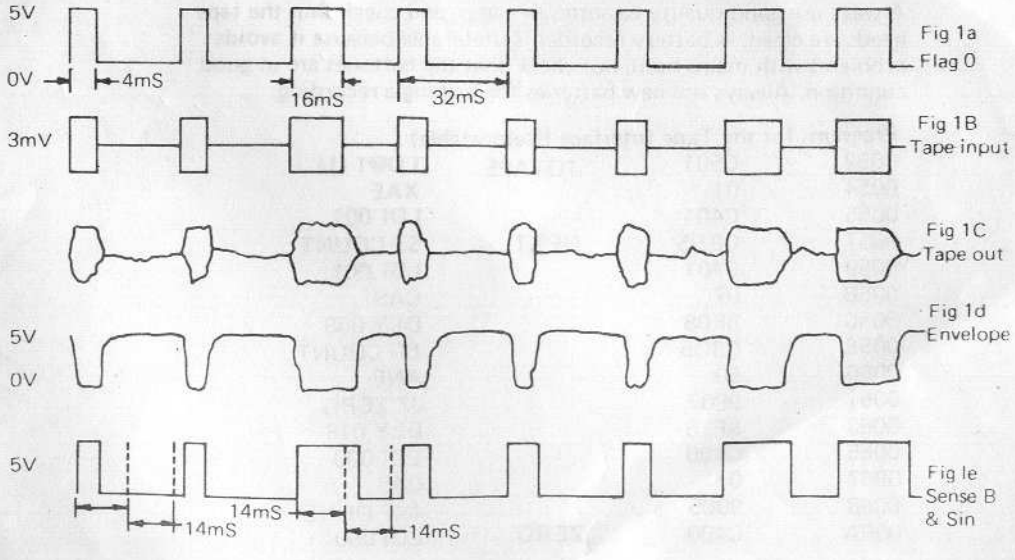
Appendix 4 Science of Cambridge cassette interface

A Simple Tape Interface for the MK14

The hardware and software described below form a simple system for storing programs or data on a tape recorder and then reloading them anywhere in RAM memory. Programs can be relocated if they use suitable addressing modes.

Writing to tape The software reads the contents of the next address pointed to by the pointer register P1 and scans the bits from right to left pulsing flag 0 high for 4mS if the bit is a zero and for 16mS if it is a one. The leading edges of these pulses are separated by 32mS. If the byte was C4 (binary 11000100) and we connected an oscilloscope to flag 0, we would observe the waveform shown in Fig. 1a. The hardware contains a gated 1KHz oscillator which is controlled by flag 0. The e WRITE LED is also controlled by flag 0. The output of the oscillator is shown in Fig. 1b and this is connected to the input of the tape recorder. Notice how the flag 0 pulses are mixed with the output of the oscillator to give a zero-centred waveform. This reduces recording distortion and allows an automatic volume control recorder to adopt a higher input sensitivity. When all the bits of the byte have been scanned, the software increments P1 and repeats the process with the next byte. The system stores 4 bytes per second. When all the bytes have been scanned, control is returned to the monitor by the instruction XPPC3.

Reading from tape Typical output (C4) from the tape recorder is shown in Fig. 1c. This is amplified and demodulated by the hardware to give a waveform like than shown in Fig. 1d. This is then squared off by a CMOS gate to give the signal shown in Fig. 1e which is connected to SENSE B and SIN. The software cycles in a tight loop until SENSE B goes high, waits for 14mS, and then reads what is present on SIN into the extension register (from the right). The program waits a further 14mS before returning to the previous tight loop. When eight bits have been read into the extension register it will contain the original byte (C4). The software stores this at the next location pointed to by P1, increments P1, and returns to the tight loop to await the first bit of the next byte. Since the software has no way of knowing then the playback has finished, it is up to the user to press the RESET button, when the READ LED ceases to flash.



Operation of the tape interface

Writing to tape:

1. Put the number of bytes to be stored in location OFF5 (in OFF8 if you are using the new monitor). This will be a hex number not exceeding 256_{10}. If one wanted to store seventy bytes one would enter 46.
   (NB: to store the full 256 bytes one should enter 00).
2. Put the starting address of the program to be stored in OFF9 and OFFA (P1H and P1L).
3. Enter the starting address of the TOTAPE routine (0052 in new monitor).
4. If desired make a short voice recording indicating the nature of the programe to be stored. Connect the tape interface to the recorder.
5. Start recording and after about five seconds press GO (GO-GO-TERM for old monitor).
6. When the WRITE LED has ceased flashing turn off the tape recorder.

Reloding from tape:

1. Put the starting address of where the program on tape is to be reloded in OFF9 and OFFA (P1H and P1L).
2. Enter the starting address of the FRTAPE routine (007C in new monitor).
3. Start the recorder playing back. Wait until after the switch on click which follows the voice header before pressing GO. It is essential to wait until after this click as it might get read as a zero bit and cause an error.
4. When the READ LED has stopped flashing press RESET and turn off the tape recorder.

General Comments:

The system is intended for use with the high impedance (monitor) output of the tape recorder. If there is only a low impedance (loudspeaker) output, things should still work although the volume control setting may be critical.
Always use good quality cassettes or tapes, and check that the tape heads are clean. A battery recorder is preferable because it avoids problems with mains hum, but check that the batteries are in good condition. Always use new batteries for making a recording.

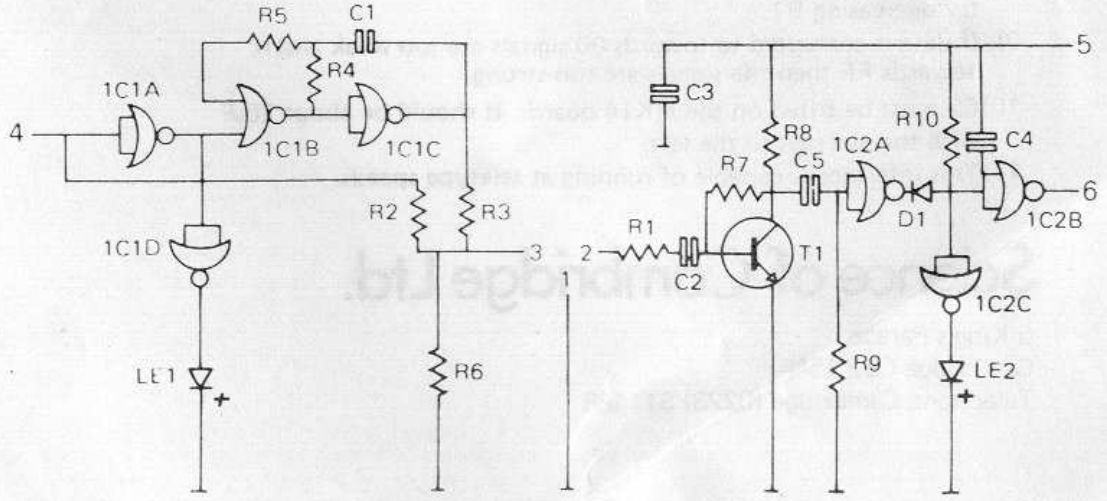
Programs for the Tape Interface (Relocatable)

```
0052  C501          TOTAPE:  LD@1 (1)
0054  01                     XAE
0055  C401                   LDI 001
0057  CBD5          NEXT:    ST COUNT
0059  C401                   LDI 001
005B  07                     CAS
005C  8F08                   DLY 008
005E  C3D5                   LD COUNT
0060  50                     ANE
0061  9807                   JZ ZERO
0063  8F18                   DLY 018
0065  C400                   LDI 000
0067  07                     CAS
0068  9005                   JMP DONE
006A  C400          ZERO:    LDI 000
```

```
006C    07          CAS
006D    8F18        DLY 018
006F    8F20  DONE:  DLY 020
0071    C3D5        LD COUNT
0073    F3D5        ADD COUNT
0075    9CE0        JNZ NEXT
0077    BBD6        DLD LEN
0079    9CD7        JNZ TOTAPE
007B    3F          XPPC 3
007C    C408  FRTAPE: LDI 008
007E    CBD5        ST COUNT
0080    06    LOOP: CSA
0081    D420        ANI 020
0083    98FB        JZ LOOP
0085    8F1C        DLY 01C
0087    19          SIO
0088    8F1C        DLY 01C
008A    BBD5        DLD COUNT
008C    9CF2        JNZ LOOP
008E    40          LDE
008F    CD01        ST@1 (1)
0091    90E9        JMP FRTAPE

0FF4    COUNT = D5 (3)      (OFF7 in new monitor)
0FF5    LEN=D6 (3)          (OFF8 in new monitor)
0FF9,0FFA  STARTING ADDRESS
```

R1	27K	C1	10N
R2	IM	C2	10N
R3	470K	C3	100N (optional)
R4	27K	C4	10N
R5	150K	C5	2N2
R6	150	D1	
R7	4M7	LE1	NSL5057
R8	27K	LE2	NSL5057
R9	4M7	T1	2N2926G
R10	150K	1C1	4001
		1C2	4001



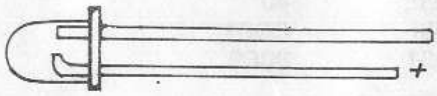
Edge connector details

1. earth
2. tape output
3. tape input
4. flag zero (30 on MK14 edge connector)
5. + 5V
6. SENSE B and SIN (27 and 28 on MK14 edge connector)

Extra Notes on Construction

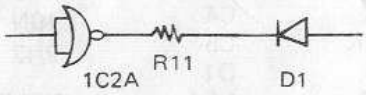
1) The white band on D1 is the positive end.
2) On some boards two positions are marked for C4. The position in the centre of the board is for C5.
3) The connections for the transistor are as shown below.
4) In some boards the +ve sign for the LEDS is the wrong way round. It should be on the left.
5) The polarity of the LEDS is as shown below.

2N2926
Transistor



LED

6) On later boards there is an extra 27K resistor in series with D1. This improves the noise immunity.



7) Cheap foreign recorders may require an input of about 100mV. To get this larger signal increase R6 to 2K7.
8) If the output level from your tape recorder is less than 200mV try decreasing R1.
9) If data is corrupted to towards 00 signals are too weak and if towards FF then the signals are too strong.
10) C2 must be fitted on the MK14 board. It should be about 20μF with the +ve sign to the left.
11) This interface is capable of running at teletype speeds.

Science of Cambridge Ltd.

6 King's Parade
Cambridge CB2 1SN
Telephone Cambridge (0223) 311488