# Microchip Morse Decoder

*A nifty gadget to decode and print morse (via a Centronics printer) from Phil Green, G4PHL.*

Very many amateur shacks are now equipped with some sort of computer, and morse reading programs are readily available for most of the popular types. A disadvantage of using the home micro in this way is that the machine is fully occupied for long monitoring periods, preventing its use for other purposes. Few machines offer true multi-tasking, so the morse reader must be interrupted if any locator, bearing, or logging programs need to be run. By the time a station has been located and the beam set up accordingly, the contact may well have finished.

This circuit can be regarded as a simple morse-to-centronics converter which releases the main station computer for more interesting tasks, and gives a hard-copy printout for later perusal. The circuit hardware is based on an idea from the Zilog handbooks, this being the simplest Z80 implementation imaginable. Comprising of only four chips, it is a well tried and tested circuit with the benefit of having very little to go wrong. Constructors with little experience in microprocessors can approach this project with confidence, especially since all the chips used have been seen on sale at various rallies for a fraction of the normal retail price. Never pay more than 50p for a Z80!

## Operation

When the power is first applied, the power-on-reset operates and the title "Morse to Centronics converter" is displayed on the printer. If good morse is then applied to the port A input, using a logic low for tone, and logic high for no tone, the decoded text will be printed out. A pause in sending is displayed as a space, and any unrecognisable characters are displayed as an asterisk. Word wrapping is prevented by printing a CR/LF sequence if a space is decoded after the 70th character typed (the interrupt page register is used for this count!).

The software can track accurately between about 8 to 24 wpm, but as in any decoding program, does need reasonable quality morse. It can therefore be usefully employed as a training aid by attaching a practise key to the input. All alphabetic characters, numbers, and punctuation signs are catered for as well as the barred procedural characters.

## System Considerations

A glance at the memory-map will reveal that no read-write memory (RAM) is provided. This is unusual, but the Z80 itself contains sufficient register storage for our needs, provided that certain restrictions are understood. Firstly, there can be no PUSH or POP's to the stack. There can be no stack without RAM! Secondly, no subroutines may be used, since without a stack there is nowhere to store the return address. Similarly interrupts cannot be used, though this does release the I register for other purposes. In fact a trick method of calling subroutines is used, as follows:

```
LD IY,$+7   ; IY holds return
              address
JP SUB      ; call the subroutine
//          ; next instruction . . .
//
```

The subroutine itself would be arranged as follows:

```
//          ; code . . .
//          ;       . . . code
JP (IY)     ; Pseudo return
              instruction
```

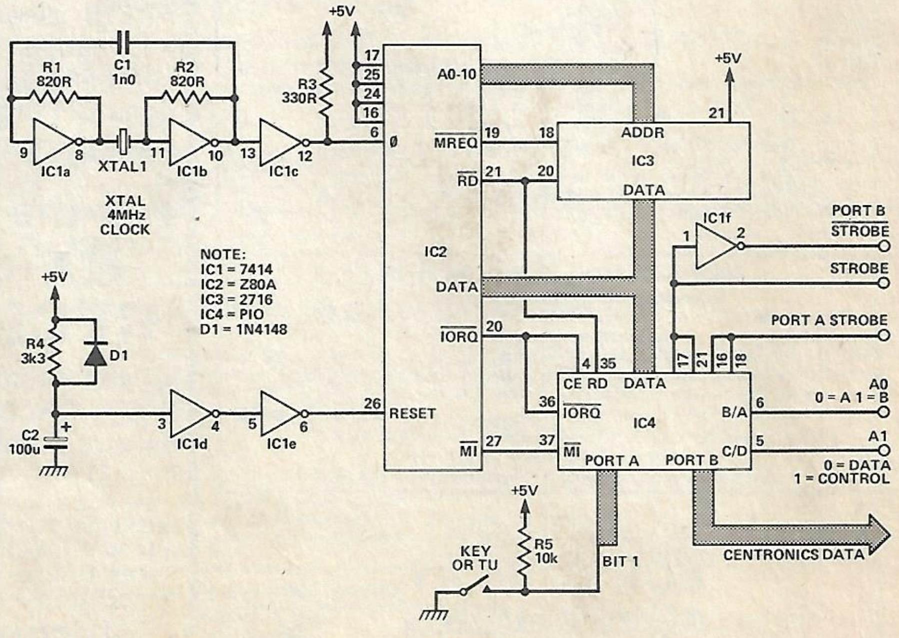



Fig. 1 Circuit diagram of the morse decoder.

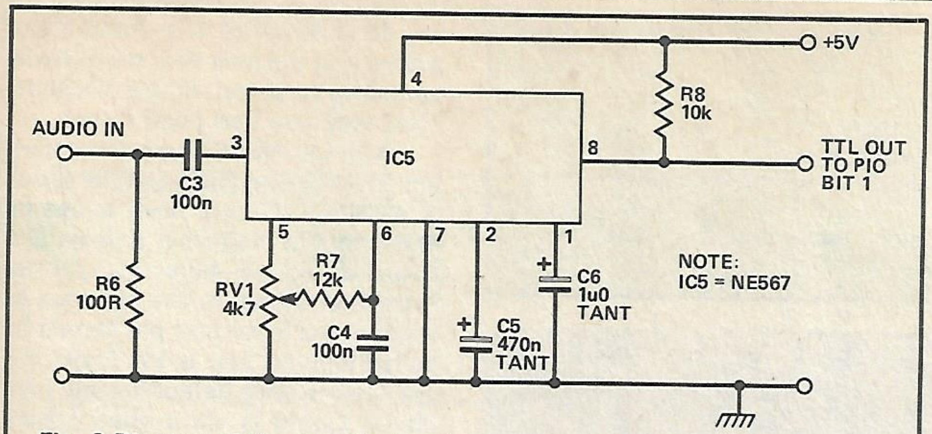



**Fig. 2 Diagram of the decoder interface.**

Provided the IY register is not used during the subroutine, this works fine. Remember we cannot PUSH IY to save it!

## Construction

The four IC's used are the Z80 itself, a parallel input/output chip, a 2716 EPROM containing the morse reading software, and a 7414 TTL chip which provides the 4MHz clock and power-on reset.

Veroboard was used for all of the prototypes, however almost all the tracks are cut adjacent to each IC pin, interconnections being individually wired. The physical layout of the board is not at all critical, and could be made a little larger if required. Wiring should follow the table provided, veropins being used for the power connections. Where two connections are made to one pin, for example the data lines and address lines 0 and 1, it is easier to thread two wires through first before soldering. Lay all the wiring neatly between the IC's rather than taking the shortest route between two pins.

```
0000 3E 4F D3 02 3E 0F D3 03 AF ED 47 21 A8 A1 7E 23
0010 B7 28 0C D3 01 11 00 18 1B 7A B3 20 FB 18 EF 3E
0020 14 47 0E 01 21 00 00 FD 21 2E A0 C3 E5 A0 30 40
0030 24 7C CB 3F CB 3F CB 3F B8 38 0B FD 21 42 A0 C3
0040 E5 A0 30 DB 18 F5 FD 21 4D A0 C3 E5 A0 38 E1 2C
0050 78 CB 3F BD 38 10 FD 21 5D A0 C3 E5 A0 30 F0 7C
0060 85 67 2E 00 18 CA 7C CB 3F B8 CB 11 26 00 18 09
0070 2C 7D CB 3F CB 3F B8 30 20 FD 21 80 A0 C3 E5 A0
0080 30 EE 24 78 CB 3F BC 38 42 FD 21 90 A0 C3 E5 A0
0090 38 F0 7D 8C 6F 26 00 18 D7 79 FE 01 28 16 FD 21
00A0 A5 A0 C3 07 A1 11 00 0C 1B 7A B3 20 FB FD 21 B4
00B0 A0 C3 07 A1 2E 00 26 00 FD 21 BF A0 C3 E5 A0 30
00C0 F5 24 78 CB 3F BC 30 F0 C3 46 A0 78 CB 3F 80 BD
00D0 38 07 78 85 CB 3F 47 18 07 FD 21 E0 A0 C3 07 A1
00E0 2E 00 C3 46 A0 50 59 06 05 0E 00 DB 00 E6 02 81
00F0 4F D9 08 01 8E 00 0B 78 B1 20 FB 08 D9 0F 10 EB
0100 79 FE 06 42 4B FD E9 79 D9 01 32 00 21 40 A1 ED
0110 B1 01 31 00 09 7E D3 01 D9 ED 57 3C ED 47 79 FE
0120 01 20 19 ED 57 FE 46 38 13 3E 0D D3 01 11 00 0C
0130 1B 7A B3 20 FB 3E 0A D3 01 AF ED 47 0E 01 FD E9
0140 01 06 17 15 0B 03 1D 09 1F 07 18 0A 1B 04 05 08
0150 19 12 0D 0F 02 0E 1E 0C 16 14 13 30 38 3C 3E 3F
0160 2F 27 23 21 20 2E 6A 2D 4C 35 BA 7A 73 47 2A 52
0170 29 00 20 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D
0180 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 31 32 33
0190 34 35 36 37 38 39 30 3D 2E 2F 2C 40 2D 21 3F 3A
01A0 3E 29 3C 2A 00 00 00 00 47 34 50 48 4C 20 4D 6F
01B0 72 73 65 20 74 6F 20 43 65 6E 74 72 6F 6E 69 63
01C0 73 20 43 6F 6E 76 65 72 74 65 72 2E 20 20 38 2D
01D0 32 34 20 57 50 4D 2E 0D 0A 0A 00 FF FF FF FF FF
01E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

**Fig. 3 Hexadecimal listing of the EPROM contents.**

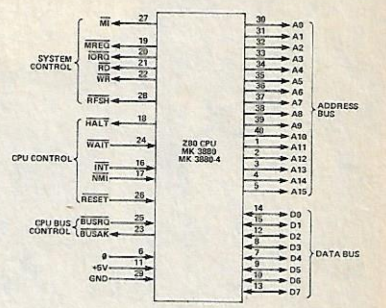
Insert the 7414 first, apply power, and listen for the 4MHz clock or a harmonic on your commuications receiver, to confirm operation. When inserting the larger chips, be carful not to bend any pins under the chip.

The input/output connectors were fashioned from 16 pin turned DIL sockets, by cutting them to form two eight-way strips. These can be conveniently mounted directly alongside the PIO. Suitable plugs can be made in a similar manner by cutting a 16 pin header.

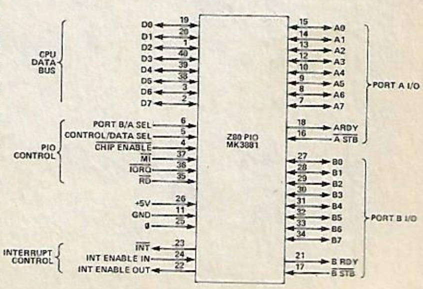
An alternative to Veroboard is the specially designed PCB, available through our Reader Services Department. Details of how to order it are available at the end of the article.

It is important the IC sockets are used for projects like this one since this encourages experimentation and allows for the occasional "blow-up" without too much disappointment. Junk box and surplus components can be tested on the board without risk of damaging anything expensive.

4MHz crystals were used in all the prototypes since these were to hand, however 3.579MHz colour burst crystals would probably suffice giving a reduced speed coverage of say 7 to 20 WPM without altering the software. If this is acceptable then by all means use them and save a little. 4.433MHz British colour crystals may work equally well for




**Z80 CPU pin-out diagram.**

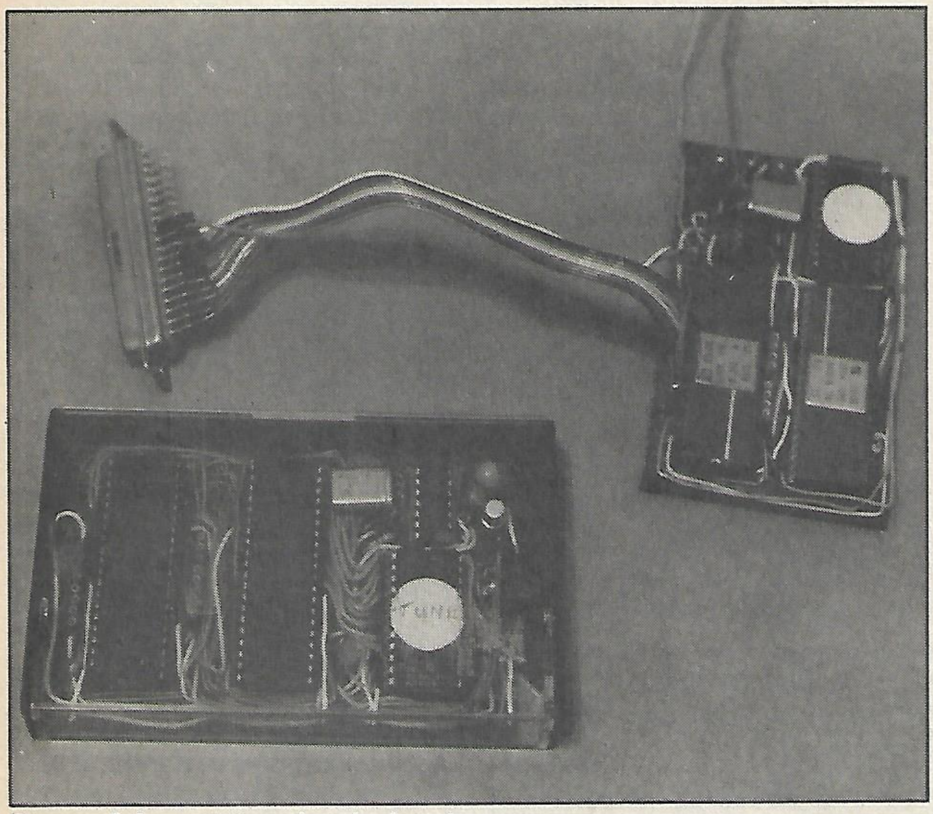



**Z80 PIO pin-out diagram.**

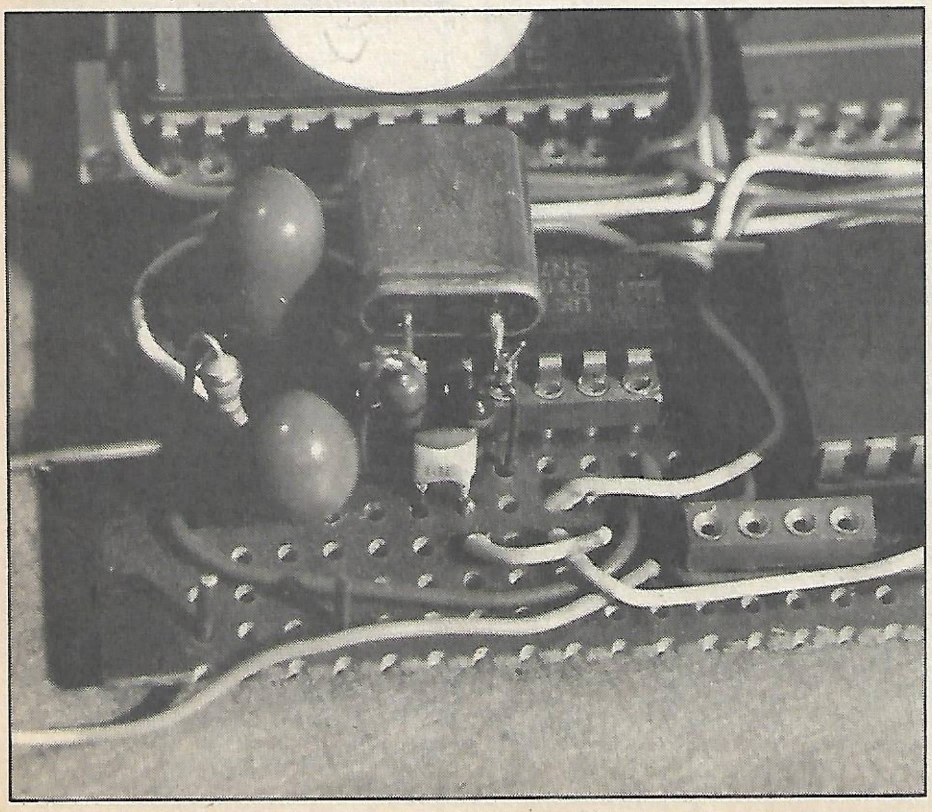
  

### Memory and I/O Map

| | | | |
|---|---|---|---|
| **EPROM:** | 0000 — 07FF | | reflected at 2k intervals |
| **I/O** | 00 — 03 | | reflected at 4 byte intervals |
| **PORT A:** | data — 00 | | software configured for key or TU input |
| | control — 02 | | |
| **PORT B:** | data — 01 | | software configured for Centronics output |
| | control — 03 | | |



**Photos of the prototype decoder boards.**



say 10 to 30 WPM. If in doubt about a crystal in the junk box, try it! Avoid spending a lot on simple projects. That way you can build more!

A simple NE567 type of terminal unit is sufficient, and a sample circuit is provided. Others have appeared elsewhere in magazine articles and there is scope here for further experimentation. In the absence of a signal the input port pin should be pulled high by, say, a 10k resistor if the TU does not default to this condition. Ouput is via a ribbon cable carrying 8 bits of data and a strobe signal. No busy signal is required from the printer due to the limited speed requirements.

## Other Uses

The board will adapt very easily to other uses simply by changing the EPROM, and has been used, among projects, as an RS232 to Centronics converter (and vice versa), a code converter, a musical box, an EPROM blower interface and a musical doorbell. The author can supply details of the software that is available if you send him an SAE at the address given below. Don't be frightened of microprocessors and associated chips, or regard them as useful only within a computer. They are after all only components, and cheap and readily available components at that!

EPROMs may be programmed from the hex listing shown, or preprogrammed EPROMs may be purchased from the author for £3 each at: 6 Yews Close, Worrall, Sheffield S30 3BB.

*The author would like to thank Geoff Taylor, G4KPU, for building the second prototype, and Stewart Ward, G6BCM, for photographing both prototypes.*

### Special PCB

The PCB for this project is available from our readers services department, price £9.40 including VAT and P&P. Orders should be sent to: Argus Specialist Publications Ltd, Readers' Services, 9 Hall Road, Hemel Hempstead, Herts HP2 7BH. Credit card orders can be placed on 0442 211882 (office hours only, for the present). Please remember to quote the board reference number, which is R8701-1, or you're liable to get the wrong PCB! Please allow up to 28 days for delivery.

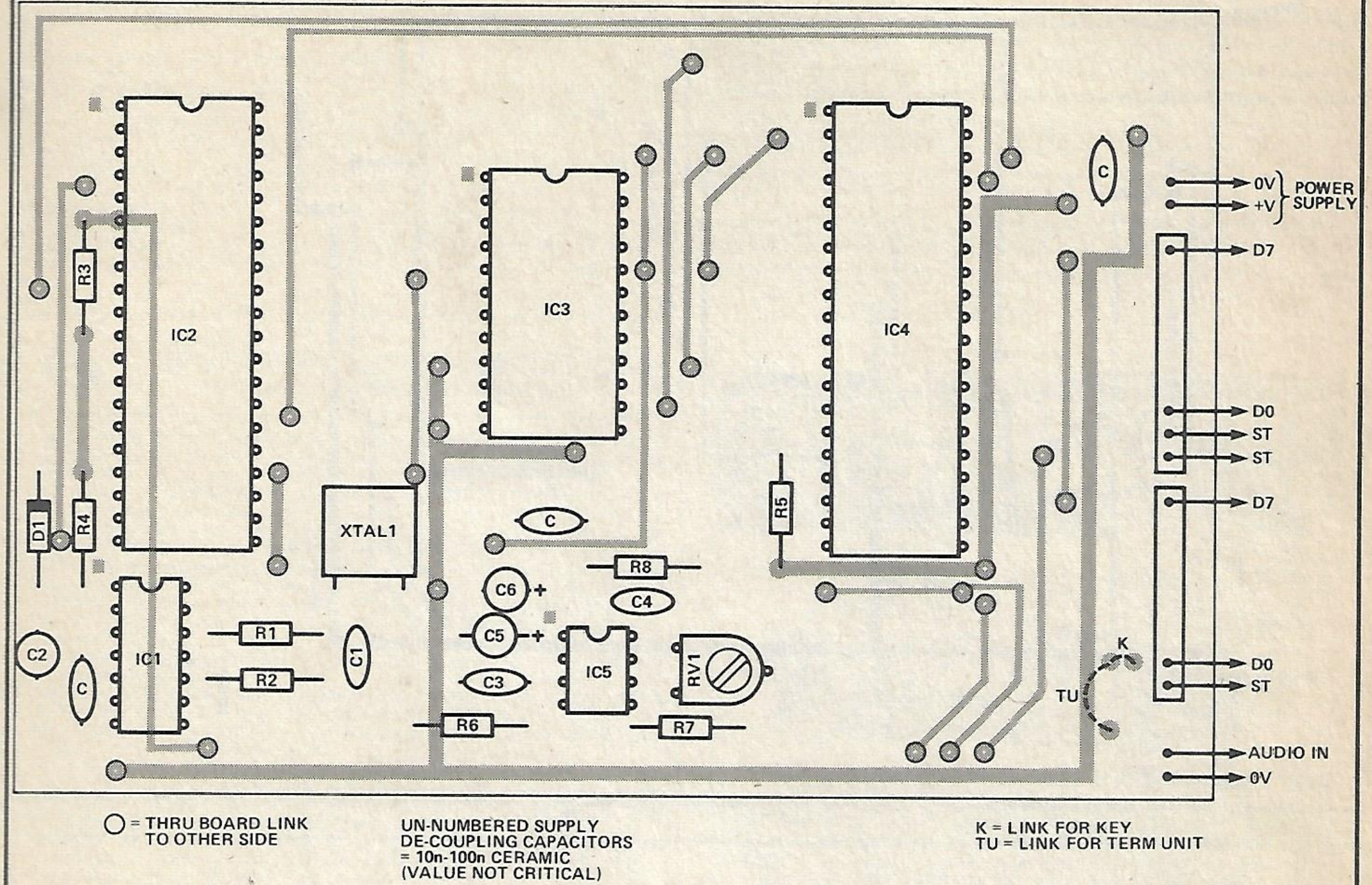



**Fig. 4 Component placement for the decoder**

## Components List

**RESISTORS**

| | |
|---|---|
| R1,2 | 820R |
| R3 | 330R |
| R4 | 3k3 |
| R5,R8★ | 10k |
| R6★ | 100R |
| R7★ | 12k |
| RV1★ | 4k7 preset potentiometer |

**CAPACITORS**

| | |
|---|---|
| C1 | 1n0 |
| C2 | 100u 12V electrolytic |
| C3★,C4★ | 100n |
| C5 | 470n tantalum |
| C6 | 1u0 tantalum |

**SEMICONDUCTORS**

| | |
|---|---|
| IC1 | 7414 |
| IC2 | Z80A CPU |
| IC3 | 2716 programmed EPROM |
| IC5★ | NE567 |
| IC4 | 280PIO |
| D1 | 1N4148 |

**MISCELLANEOUS**

XI 4MHz (or close) crystal
Centronics printer plug; circuit board (PCB or stripboard); wire, etc.

★ Denotes an item used in the terminal unit — may be omitted if not required.

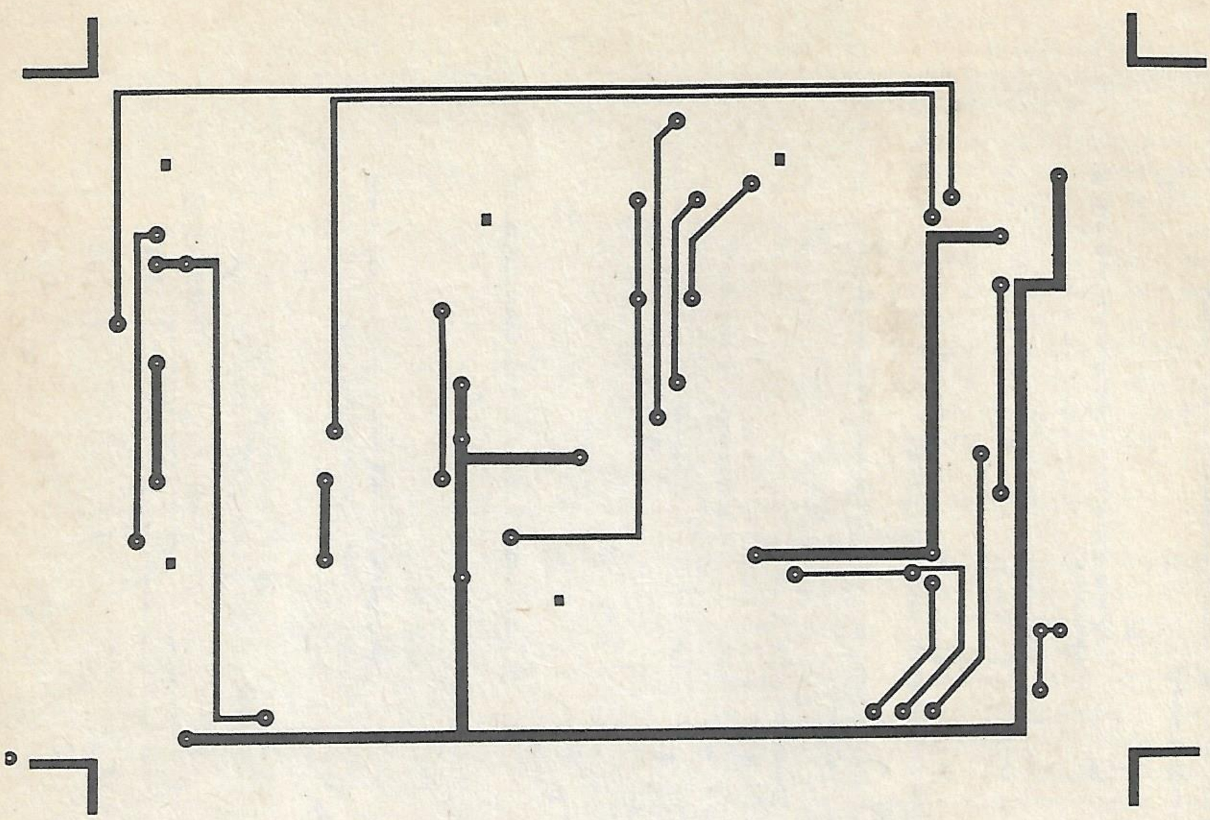


Fig. 5 Topside PCB track layout (full size).

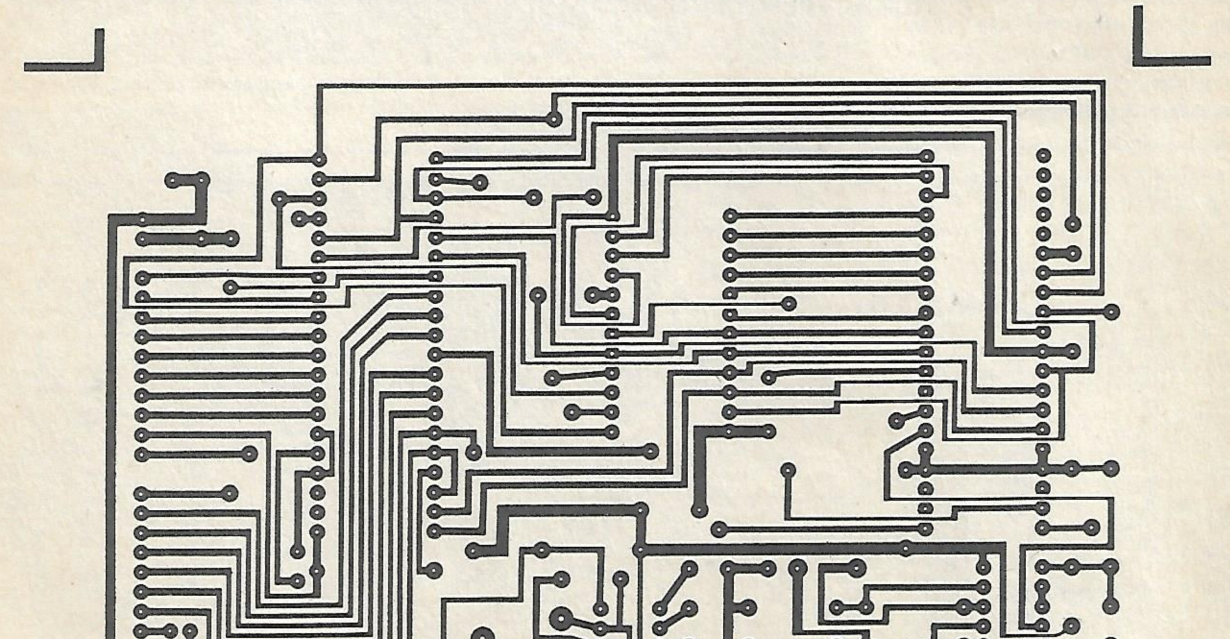


Fig. 6 Underside PCB layout (full size).

# UPDATE · MICROCHIP MORSE DECODER

By adding a plug-in selector switch and a changing of EPROM program, the HRT Z80 decoder board (described in the January 1987 edition) can be made to decode several additional modes other than Morse code.

The Morse software is very compact at well under half a kilo byte, and there is ample room in the 2716 EPROM for an additional RTTY and ASCII receive option. Selection of the required mode is done by reading port A, which is configured by the software to give 8 inputs. A simple switch to ground on each input and a pull-up resistor of about 1 to 10k are all that are required. This can be built conveniently on a small piece of stripboard, using an eight-way DIL switch (which can be socket mounted if preferred), an in-line resistor pack, and a plug made from a 16 pin DIL header.

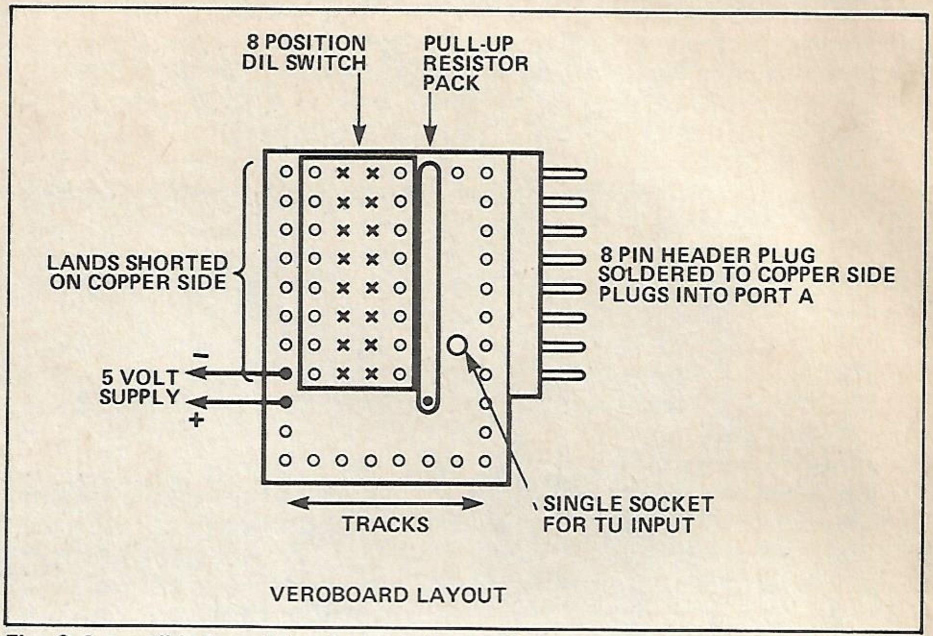



**Fig. 2 A possible layout for the switches and resistors.**

***Phil Green, G4PHL, adds a few ASCII bells and RTTY whistles to the Morse to Centronics decoder.***

A change of mode requires a reset or power off-on cycle, since port A is not continuously monitored by the program. After a reset, the new mode, clock rate and baud rate will be printed for reference. Note that since the TU input is to bit one (not zero!), no switch is needed on this pin.

The allocation of port A bits is shown in the Table. One of the switches allows the timing dependant routines to be adjusted to suit either a 3.579 or 4.000MHz clock oscillator crystal, allowing the most convenient crystal to be used. Bit zero (SW1) is unused, as is mode selection 4, and I hope that anyone with an interest in programming who produces a program for the board, will write to HRT so that we can produce a pool of useful software to be shared with other readers.

## The Terminal Unit

Much time has been spent trying to get reliable operation out of the built-in NE567 terminal unit on RTTY and ASCII. Unfortunately, it has proved barely adequate, as it was intended as a cost effective way to interface Morse only. However, with a little experimentation it is possible to get something out of it, although the author would recommend use of the ST5 TU with TTL output for consistent results.

An external TU is certainly necessary for ASCII as this uses CUTS tones (1200/2400Hz) at 300 baud. Standard tone frequencies used for RTTY are 1445Hz for mark (the standing condition when no information is being transmitted) and 1275Hz for a space. The on-board terminal unit can detect only one of the two RTTY tones, assuming that if one tone is absent, the other must be present. It cannot perform as well as, say, the ST5 or similar, but works pretty well on CW. Noisy HF signals are bound to cause trouble, unless some filtering is added.

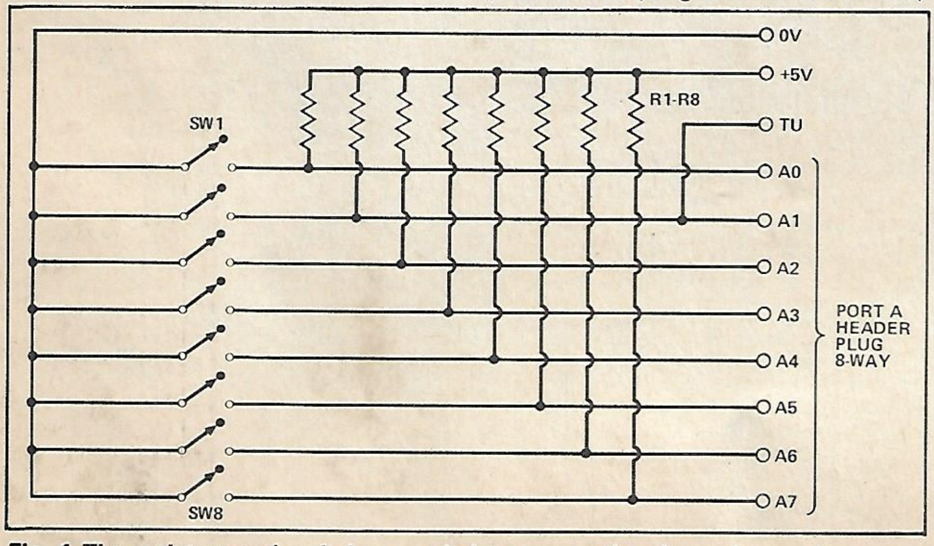



**Fig. 1 The resistors and switches needed to access the alternative modes.**

The bandwidth of the NE567 is set by C5 on pin 2, and some experimentation here may give an improvement. Generally the higher the capacitance, the tighter the bandwidth. RTTY frequencies can be easily achieved by replacing resistor R7 (12k) with one of 4k7. In this case, the NE567 should be tuned to the lower pitched RTTY tone, 1275Hz. During a mark condition therefore the NE567 output will be high (out of lock, no tone detected), and during a space condition the NE567 will lock and drive its output low.

RV1 should be carefully set to tune the TU to 1275Hz, using either a counter on pin 5, or alternatively by beating pin 5 audibly with a known reference tone. Test tones can readily be generated with reasonable accuracy by most personal computers, using the SOUND or PLAY commands. The NE567 output, one for mark, and a zero for space, is then applied to the PIO input bit one as in the Morse code.

Another unknown when tuning an RTTY signal is its baud rate. On the HF amateur bands, 45.5 is the most common, but 50 is the norm for VHF. Most commercial news stations use 50 baud, but there are several at 75. Usually, experience will enable the user to estimate the baud rate by ear. Given the inherant tolerance of the start/stop teleprinter code, however, selection is not as difficult as it sounds!

## New Facilities

Baud rates of 45, 50, 56, and 75 are provided on RTTY, and 100 or 300 in ASCII mode (but note that some printers may not be able to cope with continuous data at 300 baud, there being no 'busy' lead. Automatic unshift-on-space is provided during RTTY reception, to prevent prolonged errors should a figure shift be induced by noise. The Morse software is unchanged from the original January article. Incidentally, the 300 baud to Centronics mode allows computers with only a serial port access to a parallel printer, but using the Microchip Morse Decoder to do this really is taking overkill to the extreme!

Although the baudot to centronics mode will be used mostly for receiving radio-teleprinter signals, it is of course equally happy printing a 'land-line' telex, via a suitable level shifter (an opto-isolator is ideal). The prototypes proved invaluable in this mode, which, although limited to receive only, allows an inconspicuous printer to be located on the office desk for incoming telex calls. Signals are +/− 80 volts at 50 baud from the exchange (mark is −80) and a suitable interface circuit is shown in Fig. 3 since the module and printer are effectively in parallel with the main office teleprinter two paper copies are available.

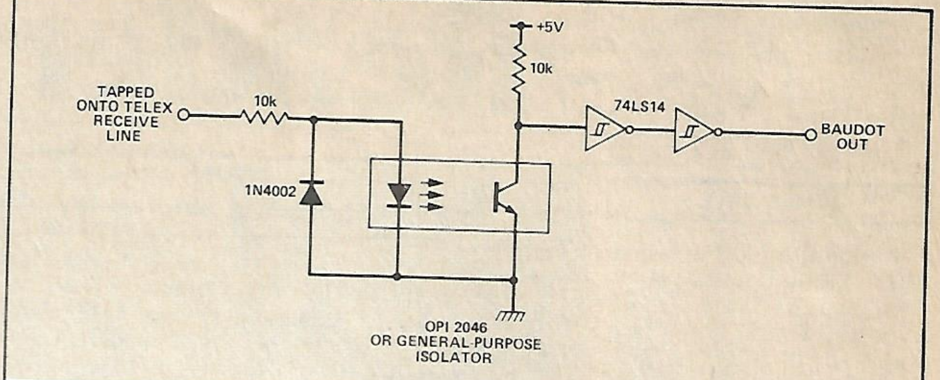



**Fig. 3 Suggested level shifter for telex to TTL conversion.**

As the reader will appreciate, some considerable work has gone into programming this little unit, and so the new software will be slightly more expensive at £5. For the same reason, no dump of the EPROM contents will be published. Anyone who has the 'Morse only' software can on return of the EPROM, have it updated to multi-code for a charge of £3. These are available from the author at 6 Yews Close, Worrall, Sheffield, S30 3BB.

## Gremlins Corner

Three gremlins crept into the original article, which were:

- The 'earthy' end of RV1 shouldn't be earthed at all. Cut the track between the NE567 pin 7 and RV1. The TU circuit diagram needs amending too.
- The TU input goes to bit ONE of the PIO port A, as in the diagram, and not to bit ZERO as on early PCB's. Move the link if necessary.
- The clock input to the PIO has been omitted from the circuit diagram, but the PCB is OK. PIO pin 25 should go to Z80A pin 6. The PIO should be a Z80A PIO, the 4MHz version.

| | SW8 | SW7 | SW6 | SW5 | SW4 | SW3 | SW2 | SW1 |
|---|---|---|---|---|---|---|---|---|
| SWITCH FUNCTIONS | | | ACSII BAUD RATE | CLOCK FREQ 0 = 3.579 1 = 4.000 | | | OPEN TU INPUT | SPARE |
| CLOSED = '0' OPEN = '1' SW2 ALWAYS OPEN | MODE 00 = MORSE 01 = RTTY 10 = ASCII 11 = SPARE | | 0 = 110 1 = 300 | | RTTY BAUD RATE 00 = 45 01 = 50 10 = 56 11 = 75 | | | |

**Table 1 The switch functions.**