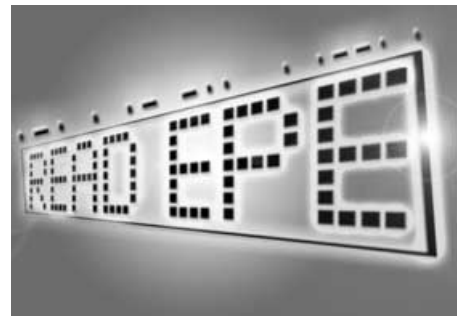
Constructional Project

EPE MORSE CODE READER



JOHN BECKER

"View" the meaning of Morse-coded tones on the air-waves, and enhance your skills at keying your own coded messages.

MORSE is not dead! With modern communications systems abounding, it may seem so to the uninitiated, but in fact it is "alive and keying".

Whilst a cursory scan through the wavebands on a modern "normal" domestic radio receiver may reveal little in the way of Morse code transmission, this communications technique is still very much in use. Tuning in via a "communications receiver" or an older domestic receiver on the short wave (SW) bands will reveal Morse activity.

Furthermore, if you have internet access and do a search through **www.google.com** on such words as "Morse", "Morse code" and "Morse transmission", you will find literally thousands of sites devoted to the subject and its continuation in the modern world. It is, after all, an historically well-proven communications system, depending purely on switching electrical, audio or visual signals on and off at regular intervals.

Several printed publications which encourage the continued use of Morse as an "art form" also exist, of which *Morsum Magnificat* is one such in the UK.

CHALLENGING

Although the author once could claim that he knew Morse code, having been taught it (and qualified!) in the Combined Cadet Force (CCF) at school, he too had let his knowledge decline and become one of the "uninitiated". Until, that is, Editor Mike showed him an American radio publication, *Worldradio*, in which there was an advert for a small handheld Morse Code Reader, undoubtedly microcontrolled.

"Can you design one?" asked Mike. "Of course", replied the author, keen to sustain the myth that he can do *anything* with PIC microcontrollers!

In fact, he has designed several Morse decoders before. The last one being published in *Everyday Electronics* in Jan '87 (long before the merger with *Practical Electronics* to become *EPE*).

At that time PICs were probably not even a twinkle in the eye of any semiconductor manufacturer. They were certainly not reality. Consequently, the *EE* design was based on a hardware mark-space ratio detector which fed separate Morse dots, dashes and spaces via individual data lines to a pre-PC computer (Commodore PET 32K). This compiled the incoming logic into a binary format, matched it against a lookup table and displayed the results on screen.

The design presented here is physically simpler, although the software is considerably more complex (that's not your problem, though, it was merely the author's! But where's the fun in designing if it's not a challenge?).

Details on obtaining the free software, and pre-programmed PICs are given at the end.

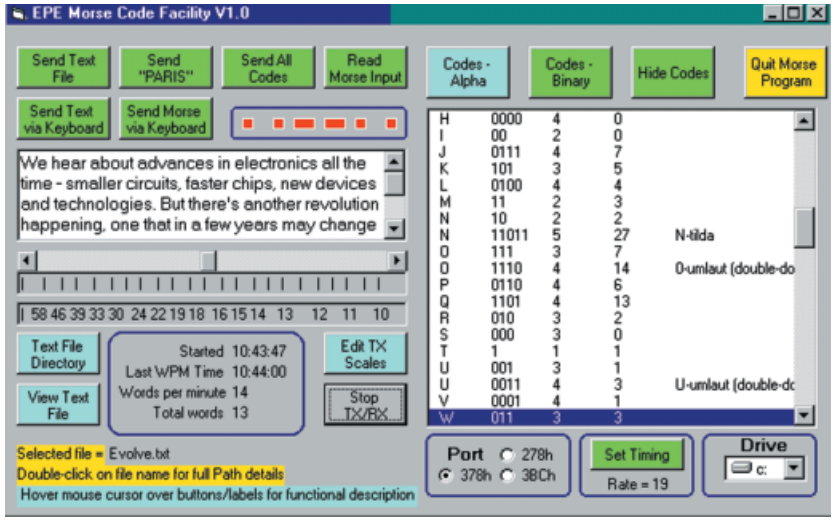
AWESOME MORSUM

There are three main aspects to this new design. It comprises:

- A handheld unit that can receive Morse code, via audio input (internal microphone) or direct signal connection, and translate it for display on an in-built liquid crystal (l.c.d.) alphanumeric screen. The received Morse signals are also available as pulses (0V/5V logic) for external use via a separate connection. Signals re-modulated at approximately 1kHz can be output to high-impedance headphones. With a suitably connected Morse key, signals can be input manually.

- Using a PC computer, Windows-based software can input the signal being repeated from the handheld unit, convert and display the code on the PC monitor, and store the translation to disk for future examination as a text file.

- The PC software can additionally be used to output Morse code to the handheld unit, for display on its screen, or monitoring as an audio signal. There are several modes of code output from the PC: translation of a text file to Morse; direct keying of alphanumeric characters for immediate translation to Morse; use of the keyboard as a Morse key with the duration of keypresses simulating Morse dots and dashes.



Typical example of the main PC screen for the EPE *Morse Code Reader.*

Several other features are also included in the PC software, as will be described later.

The handheld unit can be used on its own. It is not necessary to use it with a computer.

Various aspects of the PC software can be used on their own, too, without the need for the handheld unit. In principle, Morse code signals at normal logic levels (0V to 5V pulses) can be directly input to the computer from other sources.

The system can be used as a learning aid by those who wish to expand their understanding of Morse. It will also satisfy the curiosity of those who just want to "eavesdrop" on what radio operators are saying to each other.

TRANSLATION REQUIREMENTS

International Morse code uses the dot-dash combinations listed in Table 1. Conventionally, a dot is known as "DIT", and a dash as "DAH".

Whilst the rate of code transmission is up to the Morse operator, the relative duration of the DITs, DAHs and associated spaces has been established by international agreement:

- The DIT is the basic unit of length
- The DAH is equal in length to three DITs
- The space between the DITs and DAHs within a character (letter) is equal to one DIT
- The space between characters in a word is equal to three DITs
- The space between words is equal to seven DITs

These are the basic requirements that any human operator or translation software must observe.

The sending of Morse signals can take many forms, ranging from audio and radio transmission, modulation of light (e.g. Aldis lamps and torches), varying electrical pulse levels (e.g. sending to a computer), to bashing the water pipe if the sender is incarcerated at "Her Majesty's Pleasure"!

In audio and radio transmission, the technique is to turn the modulation of a carrier frequency (CW – continuous wave) on and off at the required rate. In audio work, the received signal is already within the audio range of the listener. Radio signals must, of course, be demodulated to become an equivalently pulsed audio signal.

There are no set rules regarding the audio frequency of Morse signals, but they must, naturally, lie in the range most likely to be heard clearly, at about 1kHz, for example.

Automatic decoding equipment therefore, must be able to accept Morse signals as a pulse-modulated frequency. It must also be able to recognise unmodulated pulse levels originating from a voltage simply being switched on and off.

The equipment must be capable of differentiating DITs from DAHs, and letter spaces from word spaces, irrespective of the rate at which the Morse signals are being received. Ideally, it should detect if the transmission rate changes and then readjust its DIT-DAH criteria.

The author's last Morse decoder had to be manually tuned so that the software correctly recognised DIT-DAH ratios. The unit described now makes its own adjustment, typically within about eight to 16 keypresses (DITs and DAHs) being received.

Thus, all you need to do is place the unit near the loudspeaker of a radio receiver, or directly plug it into the coded signal source, and observe the unit displaying the received code as an intelligible text message.

The term "intelligible" is used loosely, of course. The unit won't translate from Swahili into English, for instance! It will simply show the letters being received. The advert mentioned earlier did actually state that its unit "instantly displays CW in English!" – a very clever device indeed if it really does that for the quoted $79.95 US!

BINARY FORMAT

If you examine Morse codes as though DITs are logic 0 and DAHs are logic 1, a binary coded pattern will be seen. Converting from binary to decimal reveals a snag, however. There are some Morse codes that have one or more "leading" DITs, i.e. leading zeros. For example, take the letters E, I, S and H, which are Morse coded as DIT, DIT-DIT, DIT-DIT-DIT and DIT-DIT-DIT-DIT (the phrase *Elephants In Straw Hats Ten Miles Off* was that taught to the author to remember these four and their three DAH counterparts T, M, O – DAH, DAH-DAH, DAH-DAH-DAH!).

With each DIT as logic 0, the binary value of each of the first four letters converts to zero decimal. Not a helpful fact if regarding Morse codes as being true binary symbols.

The answer is to also take note of the number of keypresses (DITs or DAHs - call them binary bits) in a coded letter. Now each code can be allocated two decimal numbers, its length as well as its binary value. Separate lookup tables can now be used, each dedicated to a particular code length, and then to the binary value. Table 2 illustrates the idea.

Table 1. Morse codes and their reference formats used in the PC and PIC programs.

Symbol	Code	"Binary"	Count	Number	
!	·–·–·–	001101	6	13	
"	·–··–·	010010	6	18	
'	·––––·	011110	6	30	
(	–·––·	10110	5	22	
)	–·––·–	101101	6	45	
+	·–·–·	01010	5	10	
,	––··––	110011	6	51	
–	–····–	100001	6	33	
.	·–·–·–	010101	6	21	
/	–··–·	10010	5	18	
0	–––––	11111	5	31	
1	·––––	01111	5	15	
2	··–––	00111	5	7	
3	···––	00011	5	3	
4	····–	00001	5	1	
5	·····	00000	5	0	
6	–····	10000	5	16	
7	––···	11000	5	24	
8	–––··	11100	5	28	
9	––––·	11110	5	30	
;	–·–·–·	111000	6	56	
=	–···–	10001	5	17	
?	··––··	001100	6	12	
A	·–	01	2	1	
B	–···	1000	4	8	
C	–·–·	1010	4	10	
D	–··	100	3	4	
	–·–·–	10101	5	21	"starting"
	·–···	01000	5	8	"wait"
	···–·	00010	5	2	"understood"
	···–·–	000101	6	5	"End of work"
	········	00000000	8	0	"error"

These five codes not recognised by the PIC

E	·	0	1	0
F	··–·	0010	4	2
G	––·	110	3	6
H	····	0000	4	0
I	··	00	2	0
J	·–––	0111	4	7
K	–·–	101	3	5
L	·–··	0100	4	4
M	––	11	2	3
N	–·	10	2	2
O	–––	111	3	7
P	·––·	0110	4	6
Q	––·–	1101	4	13
R	·–·	010	3	2
S	···	000	3	0
T	–	1	1	1
U	··–	001	3	1
V	···–	0001	4	1
W	·––	011	3	3
X	–··–	1001	4	9
Y	–·––	1011	4	11
Z	––··	1100	4	12
Á	·––––·	01110	5	13
Ä	·–·–	0101	4	5
Ch	––––	1111	4	15
É	··–··	00100	5	4
Ñ	––·––	11011	5	27
Ö	––– ·	1110	4	14
Ü	··––	0011	4	3

Table 2

4-bits			3-bits			2-bits			1-bit		
Bin	Dec	Char	Bin	Dec	Char	Bin	Dec	Char	Bin	Dec	Char
0000	0	H	000	0	S	00	0	I	0	0	E
0001	1	V	001	1	U	01	1	A	1	1	T
0010	2	F	010	2	R	10	2	N	none such		

Table 1 shows the full range of allocated codes and equivalent conversion values used in the PIC software and the PC program.

It will be seen that some letters appear to be repeated but having different Morse codes, A, O and U, for example. This is because some languages (e.g. German) have letters that look similar to our "English" ones but have a double-dot above them (*umlaut*), e.g. Ä, Ö, Ü. Some letters also have "acute" and "tilde" signs as well, e.g. É and Ñ.

In this unit the codes for accented letters are recognised, but the translation is to the "standard" letter form.

Some Morse codes can have meanings that are specific phrases. For instance, DIT-DAH-DIT-DIT-DIT (01000) means "wait" and DAH-DIT-DAH-DAH-DIT (10110) means "starting". This unit's software ignores such expansions, although the optional PC interface software recognises some.

RECEPTION RATE

It will be obvious that the software must have a "base-timing" value against which it assesses DIT, DAH and space lengths. Such lengths depend on the sending operator's keying speed, which can vary considerably between operators. A novice might send at, say, only five words per minute (WPM). An experienced operator could even be sending at 50 WPM (about 25 WPM is a more typical rate).

The software assesses the sending rate by looking for the shorter pulses (the DITs). Initially, a temporary reference value is set to a high timing number, greater than the expected incoming pulse lengths. For a cycle covering the next 16 keypress pulses, each pulse timing length is compared with this reference. If it is less, the reference is set to the same value as the pulse.

The comparison is repeated for all 16 keypresses. It is then assumed that the reference value is that representing a DIT. The DAH and space values referred to earlier are then set in respect to this value. Again the reference value is set higher than the expected incoming pulse lengths and the cycle repeats.

Simultaneously with the reference value comparisons, each incoming keypress is compared against the current DIT, DAH and space lengths, and each code sequence compiled as an equivalent binary value and in relation to its bit count. During the letter spaces the equivalent character is found from the respective lookup table and displayed on screen. If a word space is found, a space character is also sent to the screen.

DIT length comparison, of course, is not fool-proof and noise or sporadic changes of operator keying rate may cause temporary misinterpretation of incoming codes, probably signified by a sequence of the letter T being seen. Usually, a recovery from such instances is made within 16 keypresses.

It was also found that when feeding the unit with computer-generated codes, slippage could still occasionally occur. This is due to the PC monitoring other aspects of its system even though it is also running the Morse program.

One PC in particular was excessively prone to this. It periodically decides that it wants to check all sorts of things on the hard drive and the floppies, thoroughly disrupting Visual Basic (and QuickBasic) timings. The reason cannot be found (the machine came to the author second-hand).

Visual Basic does not allow internal "interrupts" to be stopped. They can be stopped if a machine code program is being run, as the author used to do when using QB with an m/c sub-routine, but he has not yet found a way to integrate m/c (8086 assembly dialect) with VB. (Advice from anyone who does know would be appreciated!)

Using a PC as the Morse source, translation rates in excess of 50 WPM were achieved with the PIC unit.

CIRCUIT DIAGRAM

The complete circuit diagram for the *EPE* Morse Code Reader is shown in Fig.1. Not much to it! Basically, Morse signals are input, amplified, translated by a PIC16F84 microcontroller and displayed on the l.c.d. screen.

Microphone MIC1 is a miniature electret type which receives its power via resistor R1 and allows the unit to be placed near the speaker of a radio receiver to pick up Morse signals without any physical connection to it.

Socket SK1 enables direct connection to, say, a radio receiver's audio output socket, low level or line-level. The microphone is automatically disconnected in this instance.

Signals from the selected source are a.c. coupled to level control VR1 and fed to the amplification stage around IC1a. The gain is set at about 100 by resistors R2 and R5. The values of capacitors C2 and C4 respectively give a bit of bass and treble cut to the audio frequency being received, helping to reduce (although not totally eliminate) false triggering by any out-of-band noise on the signal.

From IC1a, the signal is a.c. coupled to the second amplification stage, around IC1b. Here the gain is set at around 10 by resistors R6 and R7. Resistors R3 and R4 provide a midway bias level to both stages.

The next stage extracts the Morse pulse "envelope" from the audio carrier signal. In the presence of pulses (DITs and DAHs), capacitor C6 is held charged via diode D1. When each pulse ceases, C6 discharges through preset VR2.

For as long as the voltage on the wiper of VR2 is above about 0·6V, transistor TR1 is turned on into full saturation, i.e. its collector voltage is at 0V. When each pulse ceases, the collector voltage returns high, to 5V (the power rail voltage).

Socket SK2 allows external (unmodulated) Morse pulses to be input in place of the audio signal (from a Morse key or computer, for example). Their amplitude should swing between 0V and greater than about 0·6V. The software automatically compensates for the signal inversion by TR1.

PIC PROCESSING

The output from TR1 is coupled to the Schmitt trigger input, RA4, of PIC microcontroller IC2. The software monitors the status of the input, from which information Morse pulse lengths are assessed.

The status of pin RA4 is copied by software (suitably re-inverted) to pin RA3. This allows demodulated pulses to be sent, via socket SK4, to other equipment, such as a PC which itself can decode signals into characters and display them on its screen. While developing the software, the author actually coupled two PCs to the PIC, one transmitting to it, the other receiving from it.

The pulsed signal from RA3 also drives an l.e.d., D3, via ballast resistor R13. This serves as an additional Morse code monitor. At lower transmission rates, the relative DIT and DAH lengths can be observed,

PANEL 1. ORIGINS

Samuel Finley Breese Morse was born in Charleston, Massachusetts in 1791. Studying to be a painter in the US and Europe, in 1832 he became intrigued by the telegraph, a system first built in 1774. At that time, telegraph machines required 26 separate wires, one for each letter of the alphabet (presumably numerals had to be spelt out, and punctuation ignored).

In 1833 a German 5-wire system was introduced, but Morse recognised that a 1-wire signalling system was possible, in which a series of long and short electrical pulses could be sent in a coded order relating not only to alphabet characters but to numerals and other symbols as well.

Now known as the "American" Morse Code, the original code additionally used embedded spaces as part of the coded characters. Thus dot-space-dot represented letter "O". There were even codes that used extra-long dashes, e.g. letter "L" and numeral "0".

As the code's popularity spread, it evolved to suit the needs of international users. All embedded letter spaces were eliminated and the standardised use of dots and dashes became the code now in use, the "International" (or "Continental") Morse Code. Letter "O", for instance, has now become dash-dash-dash. Several web sites quote both code formats.

PANEL 2. COMMERCIAL MORSE DEMISE

The power and sophistication of modern communications systems, especially those via satellite, eventually eclipsed the need to use Morse code commercially and many radio stations worldwide have ceased Morse transmission and reception.

On January 1st 1999, the UK's 500kHz Coast Radio Station Service, for instance, ceased to maintain a distress watch and British Telecom MF Morse radio stations ceased all commercial Morse services. Other similar services also closed on the same day, some whose history goes back around a hundred years.

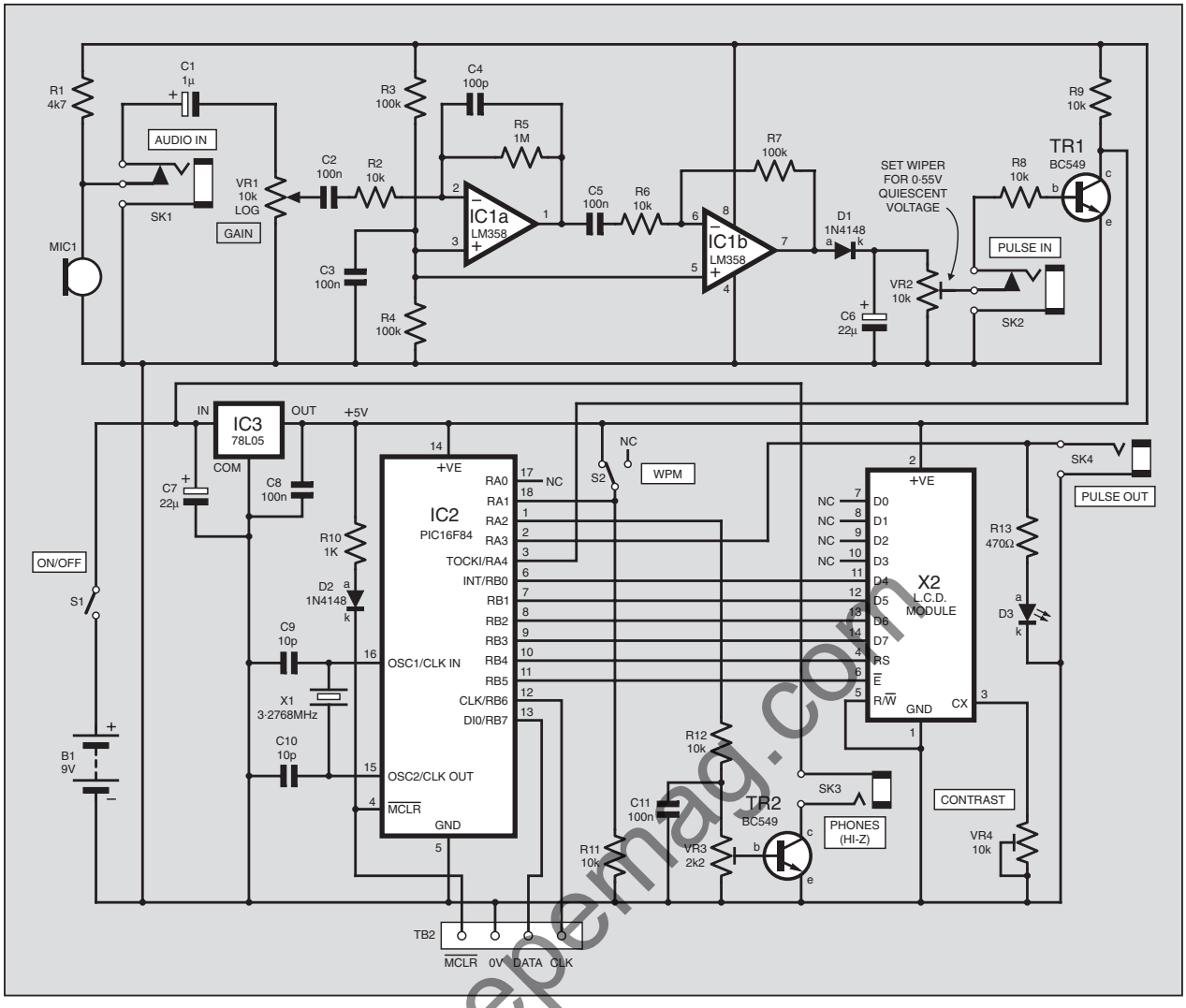


Fig.1. Complete circuit diagram for the EPE *Morse Code Reader.*

and even used as a visual source of Morse code.

Provision has been made for received Morse signals to be repeated as re-modulated audio tones (at roughly 1·3kHz) via high impedance headphones, connected via socket SK3. This facility is included as part of the unit's "learning" aspect, so that code transmissions simulated by the computer can be listened to and so test the listener's ability to mentally decode them.

The tones are output from pin RA2, sent via level control VR3 to transistor TR2, to which the headphones can be connected via socket SK3. It is stressed that high-impedance headphones (e.g. at least 40 ohms) *must* be used. The use of low impedance 'phones or a loudspeaker will kill the transistor (which has a rating of about 100mA).

Capacitor C11 smooths some of the harshness of the audio square wave – no attempt has been made to provide a "musical" tone!

In fact, using the software to generate the tone while carrying out other activities does not allow a precise audio frequency. Whilst the dominant frequency is about 1·3kHz, other underlying tones are just noticeable.

If you would prefer to listen to "cleaner" tones and you have an existing audio oscillator that can be keyed by voltage level changes, it could be driven via the pulse output at SK4.

MESSAGE DISPLAY

Visual display of the decoded Morse signals is via the 2-line 16-character (per line) alphanumeric l.c.d., X2. This is operated in standard 4-bit mode, with contrast setting performed by preset VR4.

Switch S2 causes the l.c.d. to show either two lines of message, or one line plus WPM data on the other.

In 2-line mode, the message is compiled on the lower line, the characters being placed consecutively from left to right across 16 "cells". On each 16th character the whole lower line is transferred to the top, the lower one cleared and message compilation starts again from the left. The message is not stored after being lost from the display.

In WPM mode, the WPM count is assessed every 60 seconds and output to the left of the upper line. The lower line shows the message as it progresses, with it being cleared after each 16 characters.

At the right of the upper line is displayed a sub-count of the words received since the last one-minute display occurred. It is updated after each batch of 16 characters has been received.

Typical l.c.d. screen in 2-line code display mode.

Typical l.c.d. screen showing WPM count on the top line.

OTHER ASPECTS

The PIC is run at 3·2768MHz, as set by crystal X1, and powered at +5V from regulator IC3. The unit itself may be powered, via IC3, at any d.c. voltage between about 7V and, say, 12V. Current consumption depends on the use of the headphones and their loudness. In quiescent mode consumption is about 7mA.

As usual with the author's PIC designs, provision for programming the PIC *in situ* has been made via connector TB2, whose connections, and those to the l.c.d., are in his "standard" order. *PIC Toolkit* Mk2 or Mk3 are suited to programming PICs *in situ* on board designs such as that used for this unit.

COMPONENTS

Resistors

R1	4k7
R2, R6, R8, R9, R11, R12	10k (6 off)
R3, R4, R7	100k (3 off)
R5	1M
R10	1k
R13	470Ω

All 0·25W 5% carbon film.

See SHOP TALK page

Potentiometers

VR1	10k rotary, log
VR2, VR4	10k min. preset, round (2 off)
VR3	2k2 min. preset, round

Capacitors

C1	1μ radial elect. 16V
C2, C3, C5, C8, C11	100n ceramic, 5mm pitch (5 off)
C4	100p ceramic, 5mm pitch
C6, C7	22μ radial elect. 16V (2 off)
C9, C10	10p ceramic, 5mm pitch (2 off)

Semiconductors

D1, D2	1N4148 signal diode
D3	red l.e.d.
TR1, TR2	BC549 *npn* general purpose small-signal transistor
IC1	LM358 dual op.amp
IC2	PIC16F84-4 microcontroller, preprogrammed, see text
IC3	78L05 100mA +5V voltage regulator

Miscellaneous

MIC1	min. electret microphone insert
S1, S2	min. s.p.d.t. (or s.p.s.t.) toggle switch (2 off)
SK1 to SK4	3·5mm *plastic* jack socket (4 off)
X1	3·2768MHz crystal
X2	2-line, 16-character (per line) alphanumeric l.c.d. module

Printed circuit board, available from the *EPE PCB Service*, code 368; plastic case, 150mm x 80mm x 50mm; 8-pin d.i.l. socket; 18-pin d.i.l. socket; panel-mounting l.e.d. clip; knob for VR1; PP3 battery and clip; pin header strips and sockets for TB1 and TB2 (see text); p.c.b. mounting supports (4 off); connecting wire; solder, etc.

Approx. Cost Guidance Only **£35** ***excluding battery***

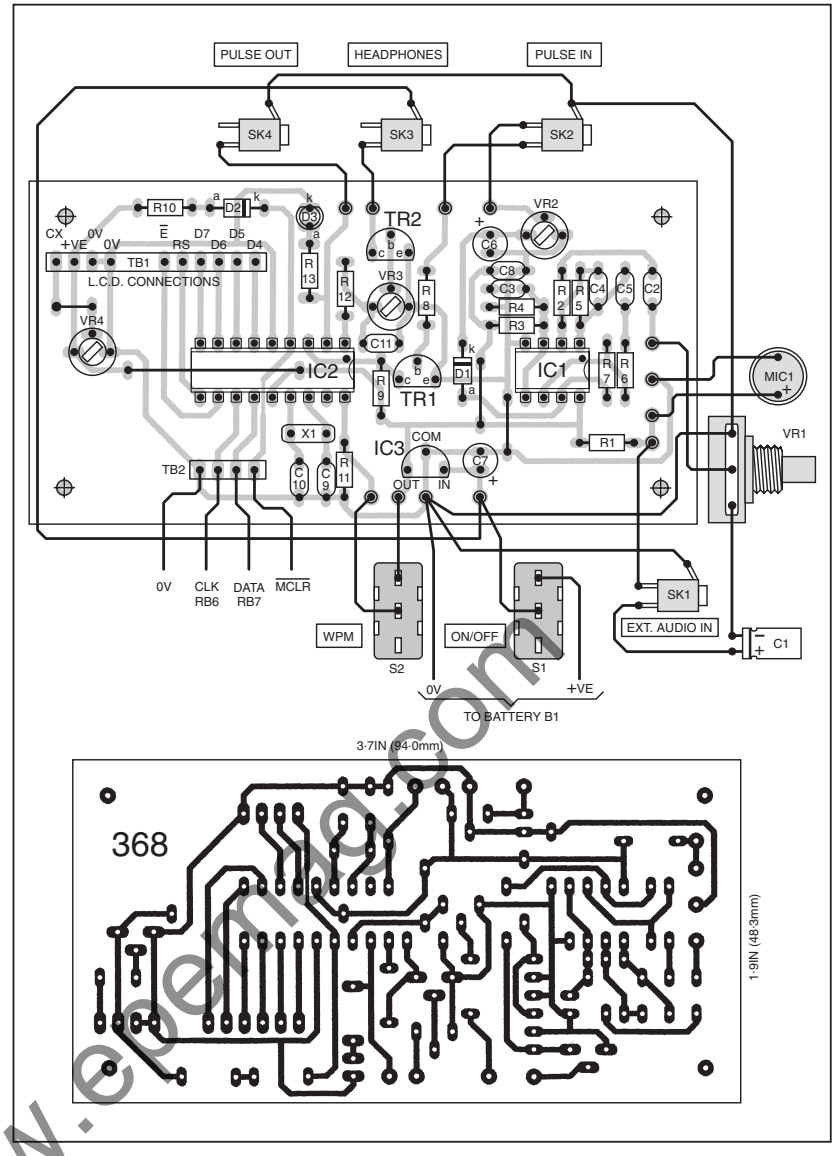


Fig.2. Component and full-size master track pattern layouts for the EPE *Morse Code Reader.*

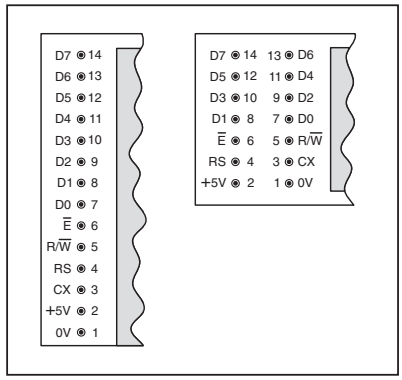


Fig.3. The two "standard" l.c.d. module pinout arrangements.

CONSTRUCTION

Printed circuit board layout details for the EPE Morse Code Reader are shown in Fig.2, This board is available from the *EPE PCB Service*, code 368.

Use sockets for IC1 and IC2. Assemble in order of component size, preferably link wires first (noting that one lies under the socket for IC2).

Pin header strips were used in the prototype for connections to the TB1 and TB2 pins. Alternatively, 1mm terminal pins could be used.

Before inserting IC1, IC2 or the l.c.d., check the correctness of your assembly and that +5V is present where indicated in the circuit diagram of Fig.1.

The unit was mounted in a plastic case, having cut a viewing slot to suit the l.c.d. screen, and holes drilled for switches, connectors, or direct external input/output wiring, etc.

A hole should also be drilled for the l.e.d. D3 (although this was not done with the prototype).

Sockets SK1 to SK4 should be *plastic* types (note that SK3 has its "common" terminal connected to the principal power line).

Typical pinout details for the l.c.d. module are shown in Fig.3.

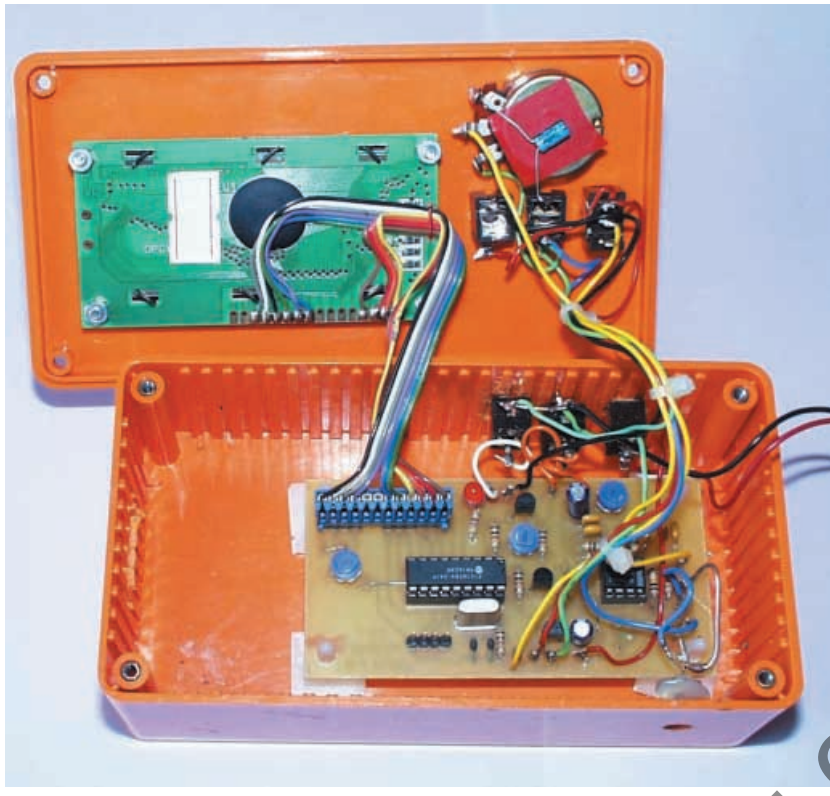
COMPUTER INTERFACE

For learning purposes, the use of the PC-based Windows software for this design is ideal. Written in Visual Basic 6 (VB6) the software is completely standalone and does not require VB6 to be resident on your PC (but see later).

For output mode, the PC uses parallel port pin D0, and for input it uses pin ACK (see Fig.4).

Five Morse code output modes are available from the PC software:

- send text file in Morse
- send "Paris" as test word (reason given presently)
- send all characters for which a Morse code is known, either in alphanumeric or "binary" order



- Send characters directly keyed-in via keyboard
- Send Morse pulses in respect of duration of any keypress

The word *Paris* is that generally used to determine the words per minute rate (WPM) at which Morse code is transmitted. It supposedly represents the average DIT-DAH-letterspace-wordspace ratio encountered in a typical message transmission.

Clicking on the Send Paris button causes the software to repeatedly output this single word followed by a wordspace. Clicking on Stop TX/RX ends this mode (as it does for any transmission or reception mode). During any transmission, the message being processed is displayed in the left hand panel.

WPM RATES

A slider below the message panel sets the WPM rate at which any transmission is sent. The actual rate for a given setting is likely to vary between different PCs (the scale values shown below the slider are those used by the author). A facility has been provided to "tune" the scales to the rate actually produced by your PC for a given slider setting. Transmission of "Paris" is the mode to use for this.

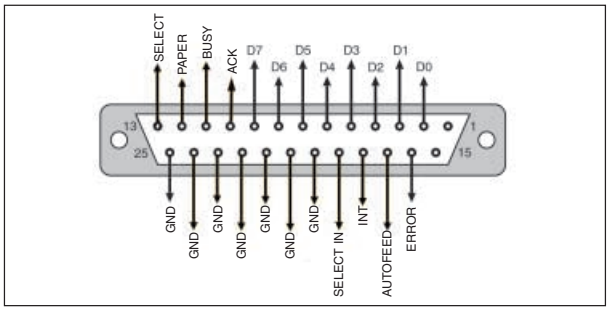


Fig.4. 25-way male D-type connector.

displaying timed counts in the box below the scales. The assessment is in relation to the number of DITs in the message multiplied by 1·2 (a factor quoted to the author by a Morse expert).

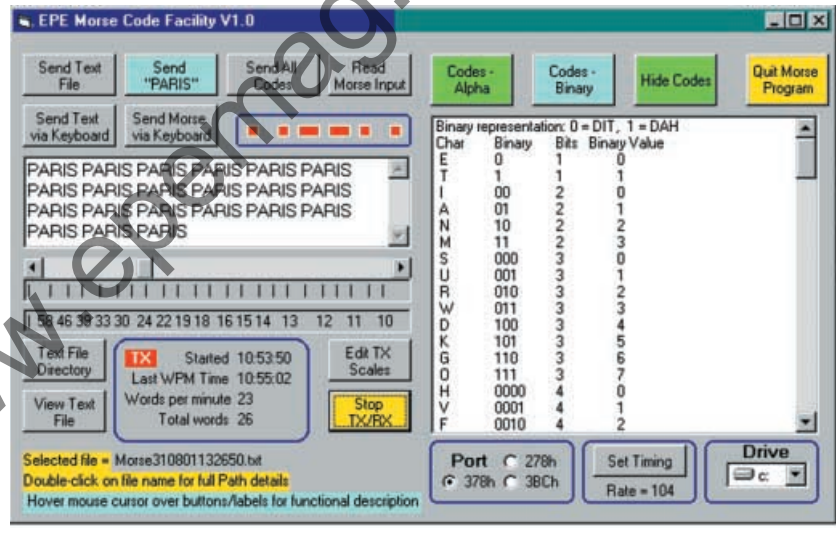
CODE SIMULATIONS

While browsing the web for Morse code info, it became apparent that learners of Morse can become tired of repeatedly listening to pre-recorded Morse test message tapes. The option to send *any message you like* has thus been included here.

It is intended that any Morse output from the PC is routed to the EPE Morse Code Reader unit. How it is then used is up to you. The most obvious use is to output an unfamiliar text file, listen to the Morse tones generated by the unit, and practice your decoding skills. The software is not linked to any audio interface installed on the PC.

To send a text file, click on Text File Directory to enter a typical Windows-style folder, path and file selector, through which any listed text file can be loaded for output as Morse code. Double-click on the file name chosen to select it and return to the main screen. The selected file name is confirmed at the bottom left of the screen. Only files having a **.TXT** extension can be accessed.

Then click on Send Test File to start transmission. Click Stop TX/RX to end it.

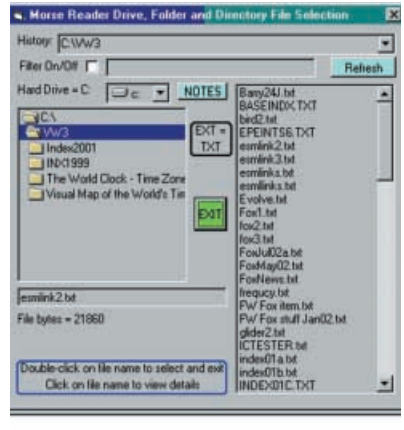


Above: Example of the main screen showing the Paris transmitted test message in the left hand text box and codes in "binary" order in the right hand panel.

Below: Typical example of the directory screen

Clicking on the button Edit TX Scales allows you to change scale legends via the keyboard. A Save TX Scales button is displayed while in this facility. Having changed the scales to your satisfaction, click this button to save the information to disk. It will be recalled next time you run the program.

Irrespective of the slider setting, the software makes a separate assessment of the WPM rate actually occurring,



The chosen file can also be viewed via the PC's Notepad text editor by clicking on View Text File.

The Directory screen is a cut down version of that used in the author's *Toolkit TK3* PIC programming software (Oct-Nov '01). More details about using this option can be viewed via the Directory screen's NOTES button (there are several sub-options discussed).

ALL CODES MODE

Clicking Send All Codes outputs standard Morse codes in alphanumeric or binary order, depending on whether the Codes-Alpha or Codes-Binary button has been clicked (confirmed by it showing blue instead of green). The full code set for each selection is displayed in the right hand sub-window (it can be scrolled up or down for viewing).

The codes displayed can be hidden should you wish to by clicking Hide Codes.

KEYED TRANSMISSION

Clicking either Send Text via Keyboard or Send Morse via Keyboard, appropriately activates one of those two modes. In the keyed Text mode you should not allow your keying rate to exceed the WPM rate set by the slider (although the normal Windows keyboard buffer will take up the "slack" for a while).

Use almost any key you like to send Morse highs and lows as DITs and DAHs (but Num Lock will probably need to be On in order to use the right hand numeric keypad). The rate of keying is up to you and is not affected by the WPM slider.

The author is much too way out of Morse keying practice (by many decades!) to know how successful the keyboard DIT-DAHing might be, but it seemed an option that was worth adding in case it might be of any use.

FIRST USE

When first loaded, the PC software detects whether or not the program has been run before on the machine in use. If it has not, a set of internal routines are initiated which, among other things (relating to clearing any personal directory and other records accidentally left in by the author!), establishes a nominal base value used during code transmission.

Such values will vary from PC to PC. The value can also be set via the Set Timing button should you wish, the resulting value being displayed below this button. Note that it is simply a "looping" value used by the software and not a timing value in terms of specific units of time.

The assessment takes a couple of seconds, and it is normal for the value to differ slightly each time the assessment is made (due to the PC's own interrupts as mentioned earlier).

PORT SETTING

The software does not assess for itself the printer port register that your computer has been set to use. This is typically register 378h (hex), but could be 278h or 3BCh. At the bottom right of the screen is a PORT box with radio buttons which select the register to be used.

To check which one is correct, first leave the setting at the default of 378h and start sending "Paris" to the Morse Reader unit and observe its l.c.d. screen. If nothing appears on the l.c.d. within a few seconds, click on 278h. Again wait a while. If still nothing appears, try 3BCh.

If there is still no success, re-check the unit and its connections to the PC.

The selected port register value is automatically stored to disk for recall next time the program is run.

RELOCATION

It is believed that the PC software is totally relocatable in terms of which drive or folder it is run from. Normally, it is likely that you will wish to run it from your C-drive. Alternative drive letters (including a partition) may be selected via the Drive option at the bottom right of the screen. This selection is also stored to disk for future recall.

Do not use a floppy or CD-drive from which to run the software. It is preferable to remain with the standard C-drive if possible.

ERROR TRAPPING

The PC software has been subjected to extensive "error-mode" checking and includes various error-trapping routines. If something unexpected occurs in an error-trapped routine, you will be advised so via a separate Error Message screen.

However, if something occurs for which the author has not provided an answer or interception, let him know via the Editorial office (not via the *Chat Zone* as he does not necessarily access this on a regular basis and your message might be missed).

PROGRAM EXIT

Finally, to exit the PC Morse software click on the Quit Morse Program button, which will fully close it. If you use the top-right Windows X button, the program may only become "hidden", remaining active in the Desktop screen's lower toolbar.

It is worth exploring your Morse screen with the mouse cursor. There are various notes that appear when it hovers briefly over buttons and labels.

MORSE CODE DATA

Should you wish to add a new Morse code item to those known to the PC software (author!), you can do so by accessing file **MorseCode.txt** held in the main Morse folder. Double-click on the file to open it through Notepad. It can now be amended, and resaved as the same name and file type.

Do not edit the file via a wordprocessing program since this might add format codes which would affect the correct use of the file by the Morse software.

SOFTWARE

Software for the PIC unit and PC interface is available on 3·5-inch disk from the Editorial office (a small handling charge applies) or downloaded free from our ftp site. The latter is accessible via the top of the title page of the main *EPE* web site at **www.epemag.wimborne.co.uk**. Click on "FTP Site (downloads)", then in turn on PUB and PICS, in which page the files are in the folder named MORSE.

More details of both options are given on this month's *Shoptalk* page, plus information on obtaining pre-programmed PICs.

The PIC program (ASM) was written in TASM, although the run-time assembly is supplied both as a TASM OBJ file and an MPASM HEX file (the latter has configuration values embedded in it). Users of the TASM OBJ file should configure their PIC for crystal XT, WDT off, POR on.

Regarding the PC interface, if you have Visual Basic 6 already installed on your machine you only need to use files **Morse.exe**, **INPOUT.DLL** and **Morse Code.txt**. Copy them into a new folder named MORSE (or any other of your choosing).

If you do not have VB6, you need three other files, **comdlg32.ocx**, **Mscomctl.ocx** and **Msvbvm60.dll**, held on our 3·5-inch disk named Interface Disk 1, and in the Interface folder on the ftp site (they are also included with the *Toolkit TK3* software). These files must be copied into the same folder as the other three files.

MORSE WEB SITES

There are too many web sites devoted to Morse code for them to be listed here. However, do as the author did (and mentioned earlier), search via **www.google.com** (an excellent search engine).

Those who wish to know more about becoming an amateur radio operator in the UK should contact the Radio Society of Great Britain (RSGB). They will also advise details of their Morse test transmissions, courses and exam requirements. RSGB, Lambda House, Cranborne Road, Potters Bar, Herts EN6 3JE. Web: www.rsgb.org.

Books by our *New Technology* author, Ian Poole, also provide information about amateur radio. Browse **lineone.net/~ian_poole/books**.

Morsum Magnificat is a bi-monthly magazine that has been around since 1983 and "is for all Morse enthusiasts, amateur or professional, active or retired. It brings together material which would otherwise be lost to posterity, providing an invaluable source of interest, reference and record relating to the traditions and practice of Morse".

Information about *MM* can be obtained through *Morsum Magnificat*, The Poplars, Wistanswick, Market Drayton, Shropshire TF9 2BA. Tel: 01630 638306. Fax: 01630 638051. E-mail: zyg@MorseMag.com. Web: www.MorseMag.com. (It can also be found at www.morsemag.com.) □