PIC Iambic Keyer (PIK)

This article describes the design and implementation of a simple semi automatic electronic morse code keyer.

The keyer assists the formation of correctly times Morse signals by automating the timing of the duration of the dit and dah elements and the rest period between the dits and dahs, and optionally, between individual characters. The keyer also allows automatic generation of successive dits, successive dahs, and alternating dits and dahs or dahs and dits without keying each individual element.

Design criteria

Key design criteria were:

1. simple in operation;
2. basic functions;
3. simple in construction and adjustment;
4. low power requirements;
5. open collector output to key a modern solid state transceiver than requires the keyer to sink a current of the order of mA to earth, and a maximum open circuit keying voltage of less than 24V DC;
6. accommodate a hand key concurrently with the paddle;
7. integrated Tune switch;
8. operate from a nominal 12V DC, -ve ground supply.

The functions that would be supported were:

- automatic timing of duration of dits;
- automatic timing of duration of dahs;
- automatic timing of duration of rest period after a dit or dah;
- optionally, automatic timing of duration of inter-character rest period (referred to in this article as Automatic Spacing or Auto-space).

Functions that would not be supported included:

- grid block, or high voltage keying;
- side tone oscillator;
- bias adjustment;
- non standard Morse timing (weight, long dashes, long inter-character time);
- any kind of repertory memory feature.

This keyer will not suite grid block keying, any kind of valve cathode keying, any kind of high voltage keying without modification. It would be possible to use the keyer to key a suitable reed relay which in turn keyed one of these types of transmitter systems.

The implementation does not include a side tone oscillator, most (all) transceivers have a side tone function and it sounds better than a shabby implementation in a keyer, although I concede that a side tone oscillator in the keyer is handy for practice. You could add a small oscillator and speaker or phone jack yourself.

The right place to fix a bias problem if you have one is in the transmitter, not in the keyer, there shouldn't be a need for bias adjustment in the keyer.

Non standard Morse timing is not enhanced Morse, it is just a barrier to effective communication. If you want to "Swing", use a hand key.

I don't doubt that a repertory memory in a keyer may be handy, but nothing competes with a keyboard and screen if you want to have boilerplate stored and develop "macros", you have the power to implement an integrated decoder, and gee you could even create automated responses so you don't even need to be there!

Implementation

One could approach the implementation a number of ways, including:

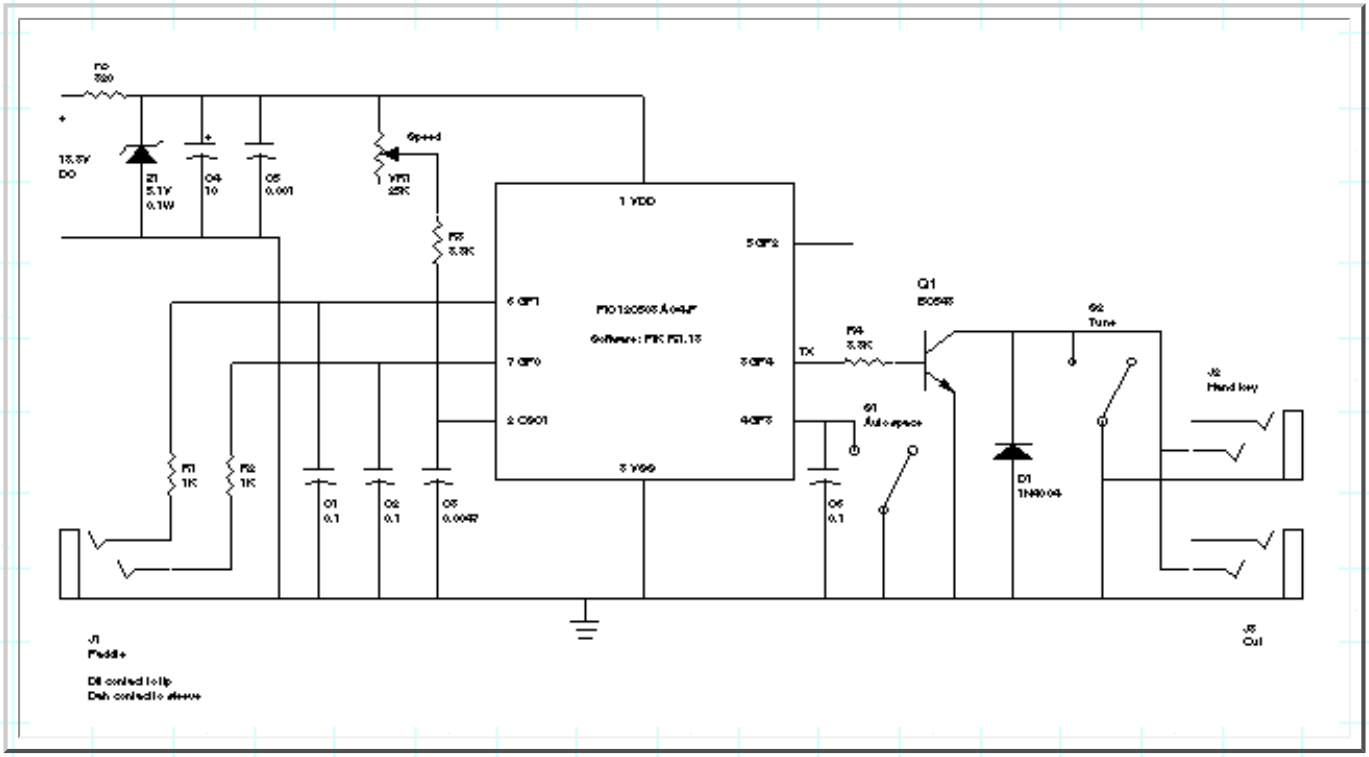
1. hardwired logic using gate and flip flop packages;
2. programmable logic array (PLA); and
3. microcontroller (MCU).

There is nothing novel about 1, it has been done before and there are so many designs in existence. The disadvantage is the number of chip packages and the amount of wiring favours a printed circuit board for implementation. This approach fails design criterion #1.

Implementation with a programmable logic array is relatively novel, but a gross overkill, expensive, and creates some construction difficulty with the package for the PLA. This approach fails design criterion #3.

An MCU implementation could again be an overkill, depending on the choice of MCU. The amount of software to implement a basic function set is quite small, the instruction path lengths are short, and only 3 or 4 I/O pins are required so a very modest MCU will be more than capable.

Figure 1: Schematic diagram



I have chosen a Microchip PIC12C508A, which is available in low cost plastic DIP8 package for through hole or socket mounting, has enough I/O pins, more than enough program memory and data memory, and plenty of CPU power for the job. The only downside of the 12C508A is that it is an OTP (one time programmable) which is one of the reasons why it is so cheap. An alternative would be a flash PIC chip like the 16F84, but since the software is so simple and should be stable, the added cost for the erase / reprogram capability is of little value once the development is done.

The design uses 1mA to 4mA at 5V, depending on speed and traffic. The zener regulator shown in the circuit will cause consumption of 10mA at 13.8V, it may be possible to reduce this by a few mA using a 78L05 regulator (although they can be a little hard to get). The VDD supply should not exceed 5V, but should run satisfactorily on 4.5V or 3V from series dry cells (although the speed range may change a little).

The chip provides two outputs, TX which goes to 5V for key down condition, and ~TX which is its inversion. These chip output pins are not open collector, do not short them to 0V for 5V (for example by putting a key or switch in parallel with the outputs). They are both provided for convenience for integration in a transmitter / transceiver.

The components are chosen to support a speed range of at least 10wpm to 40wpm. Based on a dit element duration of 122mS at 10wpm, the software is calibrated so that the oscillator runs at 10KHz. With the components shown, the prototype ranged from 6wpm to 50wpm (see Detailed derivation of Timing).

To maintain control of the software, the intention is to release programmed MCUs rather than distribute the software. To a proficient programmer, development of the software is a trivial task anyway - an interesting exercise. The programmed MCUs have the Code Protect feature enabled, and so should not be possible to copy the code. Don't ask for the source code or the hex code, it is not available at this time. (PS: hex code now available on line).

The schematic is shown in Figure 1. The circuit schematic is also available in Acrobat format here.

Software

The MCU operates in the External RC Clock mode, clock frequency is set by VR1, R6 and C3. The clock runs at 10KHz for 10 wpm nominal speed, and should produce a dit element duration of 122 mS, or a repeating dit frequency of 4.1 Hz. Clock speed is sensitive to VDD, and range calibration will change if you try to run the chip at 2V. Clock speed is a compromise between timing granularity and risk of RFI. With the choice of 10KHz at 10wpm , the basic cycle time of the MCU is 0.4mS, which provides timing granularity of 0.4mS for a dit interval of 122ms, which is a granularity (timing resolution accuracy) of under 0.5%.

Timing of all elements (dits, dahs, rests and inter-character space) is derived from the single oscillator, is fixed requiring no alignment or adjustment, and is quite accurate.

Algorithm

This section applies to V1.13. The algorithm in V1.12 is identical apart from the sleep.

For the technically minded, the following is a brief description of the algorithm:

Main loop

1. On startup, the software initialises;
2. It scans both paddles, latching closed contact conditions;

3. Based on the condition of the paddle latches, the type of the last element sent, and whether autospace is enabled, it will commence sending a dit, dah, execute an autospace wait, or go to sleep (if both paddles are closed within the same scan cycle, the keyer will process the dit first);

Sending a dit or dah:

1. Immediately it goes key down, it clears the paddle latches;
2. During the key down period and the subsequent rest period, it scans the opposite paddle type (every 2mS at 10WPM) and latches closed contact condition;
3. On completion of the rest period, it goes back to Step 2 of the Main loop.

Executing autospace wait:

1. The paddle latches are cleared;
2. A wait is executed (of duration twice the rest period);
3. On completion of the wait period, it goes back to Step 2 of the Main loop.

Sleeping:

1. The processor is shutdown. A closure or a paddle or autospace switch will cause the processor to wake-up and execute Step 1 of the Main loop.

Except for Iambic sequences, one must have the paddle closed at the time one wants an element to be initiated, and one must release the paddle anytime until the completion of the rest period to avoid sending another of that element type. This is true even when autospace forces an inter-character idle.

If while sending a dit (key down or rest period), one closes the dah paddle, a request for that dah be queued. One must release the dah paddle anytime until the completion of the rest period for the dah to avoid sending or queuing another dah. The converse also applies.

This is I believe referred to as Iambic Mode B operation, whereas Iambic Mode A would not queue the dah during the dit time, the dah paddle would have to be closed at the completion of the rest period following the dit to initiate the following dah. A Mode A keyer is short one of the features of a Mode B keyer. To my mind, an operator who exploits the capability of a Mode B keyer will notice the lack of iambic queuing in a Mode A keyer, whereas an operator accustomed to Mode A keyer will probably not notice any difference using a Mode B keyer.

If autospace is enabled, and neither paddle contact is closed at the completion of a rest period, the keyer will start a wait interval of a further two rest periods (during which it scans and latches the paddles), thereby forcing a standard inter character space.

I experimented with a range of schemes for the interval when the paddle latches were active, and when they were reset, monitoring error rates keying slow and fast, fresh and fatigued. This scheme above seemed most tolerant of my keying. On later checking, I think that the algorithm described above is identical to the operation of the once popular WB4VVF Accu-Keyer, though the timing is more accurate than the Accu-Keyer. Perhaps that is in part due to my training on an Accu-Keyer some twenty years ago!

Prototype

Fig 2: Prototype constructed on Veroboard

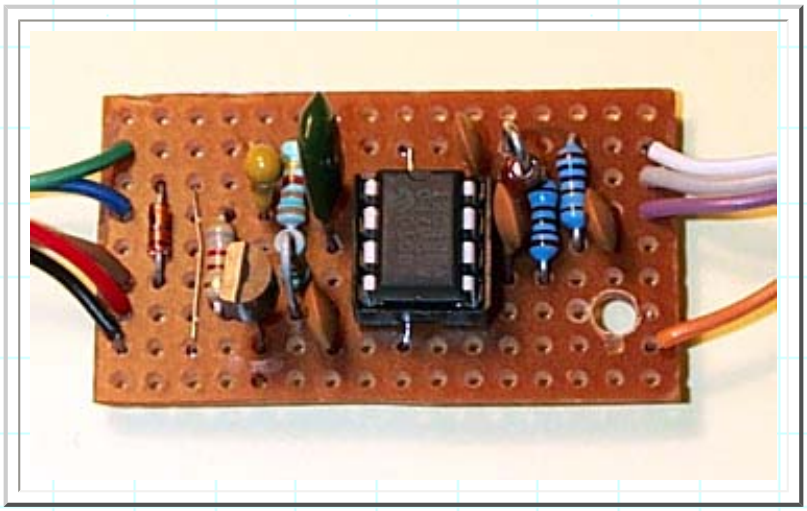


Fig 2 shows the prototype built on a piece of Veroboard 25mm x 42mm. The MCU is socketed so that the chip could be changed easily. All electronic components beside the pot are on the Veroboard, 15 parts in all 4 of which are for the regulated power supply.

Figure 3: Example veroboard track layout

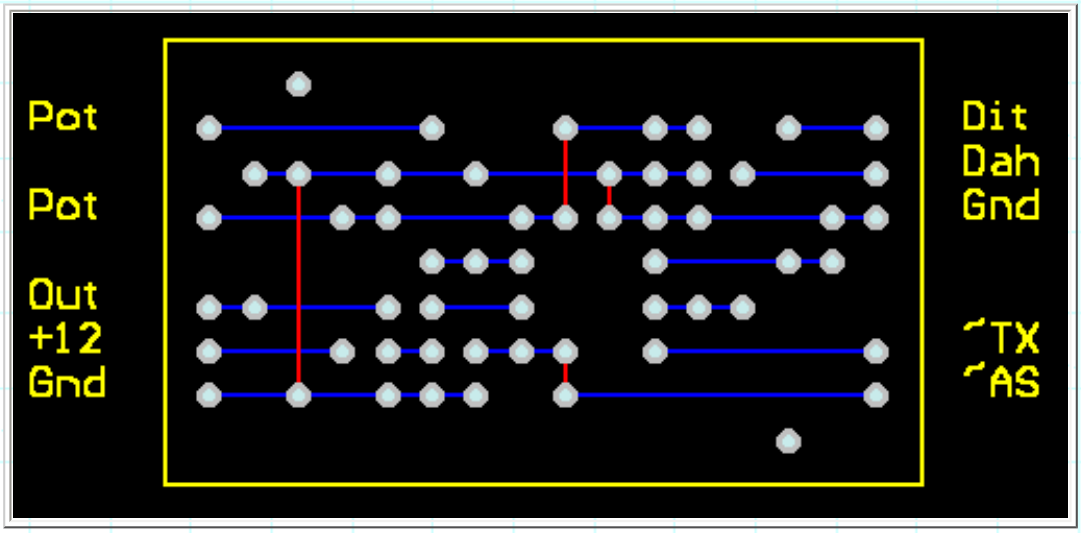


Figure 3 shows an example veroboard layout. The view is of the component side. The blue (horizontal) tracks are the veroboard tracks as used, the red tracks (vertical) are jumpers. Note that the tracks are cut between holes in some instances. Also note that the three jumpers are under the IC or its socket, they need to be installed first. This board layout does not include provision for mounting, you may need to provide additional space for the mounting system that you intend to apply, you could cut it overall to fit the slots of a jiffy box, or leave an extra 3 strips on one side and drill it to fit on threaded pillars. (Note that with subsequent revisions, the ~TX line is gone, and the ~AS pin is now AS). The Dit and Dah lines are actually ~Dit and ~DAH.

Figure 4: PIK output timing

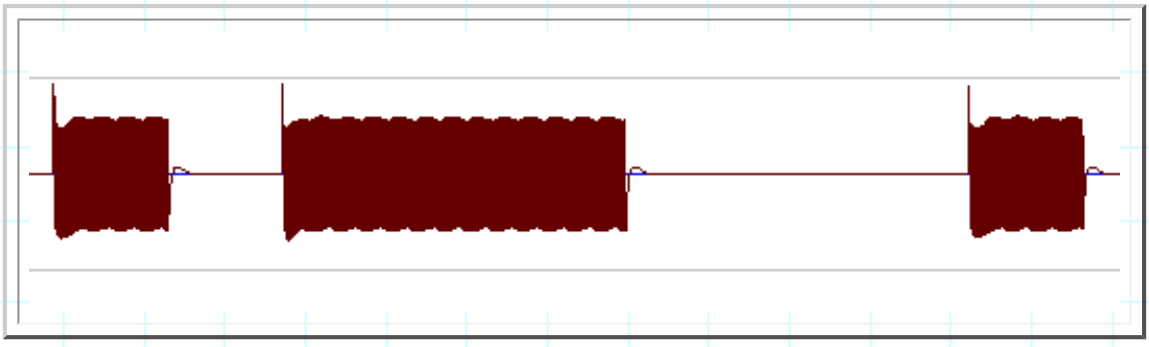


Figure 4 is a time domain display of a practice oscillator keyed by the PIK with the PIK oscillator

running at 10KHz, sending the characters AE with Auto-space enabled.

The oscillator output was recorded digitally and the duration of the tone bursts and intervals were measured with resolution better than 1mS. The duration of the tone bursts and silent intervals are 122mS, 122mS, 366mS, 366mS and 122mS. The last dit was delayed by the Auto-space, the 366mmS interval is a test of the Auto-space timing.

This keyer does not suffer any irregular timing of any elements as is evidenced in some other keyer designs, especially those that start the clock oscillator after an idle period and suffer irregular timing on the first cycle.

All significant timings and latencies are a function of the clock frequency which is adjusted for sending speed. The accuracy of the timing in relative terms should be equally good at all (practical) operating speeds. I looks just as good at 60wpm where the tone burst may be only 12 cycles of the side-tone oscillator.

Sourcing parts

All the parts should be easily obtained, or easily substituted. The 12C508A MCU is used widely as the basis for the so-called Playstation mod-chips and shouldn't be too hard to find. (I expect the firmware should work fine in a 12C508A, 12C509A, 12CE518 and 12CE519, but operation has been tested only in the 12C508A and 12CE519, and it may work on other 12Cxxx family chips.)

Programmed MCU chips may be available from the author for about A$10 including postage within Australia. Please contact me to ensure that stock is available.

Switches, pot, jacks and a case will comprise most of the other costs, the whole thing should be less than A$30 for switches, pot, jacks, case, MCU, and the other parts.

V 1.11 (20/12/01)

Version 1.11 firmware was modified to reduce idle current consumption for battery operation. It did that in two ways, one is entering a sleep mode where the processor shuts down when idle (between characters sometimes), and changing the sense of the Autospace switch (switch open is Auto ON).

V 1.12 (03/03/02)

The sleep feature introduced in version 1.11 firmware has a problem that occurs rarely and is difficult to replicate, and has been removed from V1.12. All V1.11 chips were replaced and have been destroyed.

V 1.13 (29/03/03)

It turns out that the problem with V1.11 was with ringing of the paddle inputs at the instant that the MCU was entering sleep. The fix is to increase the values of C1 and C2 to 0.1uF.

Version 1.13 firmware reduces idle current consumption for battery operation. It achieves that by entering a sleep mode where the processor shuts down when idle (between characters if Autospace is off). Because the MCU reinitialises during normal operation, the ~TX line has been removed due to the very short glitch on that line during MCU initialisation.

When idle (sleeping) and Autospace ON, the prototype keyer (exclusive of the regulator) draws 1.5uA on a 5V supply, and when keying, current rises to 1.5mA. It is practical to run this version on 3 x AA alkaline cells (~3000mAh), battery life on idle would be limited by internal battery

leakage.

Table 1: Estimated battery life

Battery	Capacity (mAh)	Battery life (hours)	
		Idle (1.5uA)	Keying (1.5mA)
3 x MN1500 (AA)	3000	2,000,000	2000
3 x MS76 (silver oxide button cell)	180	120,000	120

V 1.13 (17/04/04)

Code is identical to V1.13. Released under GPL. Download.

Last update: 16 October 2004 17:00

VK1OD on the 'net, your feedback is welcome.

Derivation of the relationship between clock speed, internal timers and WPM equivalent speed

The following is a table of frequency of occurrence of the alphabet in plain English text taken from a cryptanalysis text, and the length of Morse Code encoding of those characters.

Letter	Frequency (%)	Elements	Contribution
E	0.1300	4	0.520
T	0.0920	6	0.552
N	0.0790	8	0.632
R	0.0760	10	0.760
O	0.0750	14	1.050
A	0.0740	8	0.592
I	0.0740	6	0.444
S	0.0610	8	0.488
D	0.0420	10	0.420
L	0.0360	12	0.432
H	0.0340	10	0.340
C	0.0310	14	0.434
F	0.0280	12	0.336
P	0.0270	14	0.378
U	0.0260	10	0.260
M	0.0250	10	0.250
Y	0.0190	16	0.304
G	0.0160	12	0.192
W	0.0160	12	0.192
V	0.0150	12	0.180
B	0.0100	12	0.120

X	0.0050	14	0.070
Q	0.0030	16	0.048
K	0.0030	12	0.036
J	0.0020	16	0.032
Z	0.0010	14	0.014
Total	1.00000		9.076

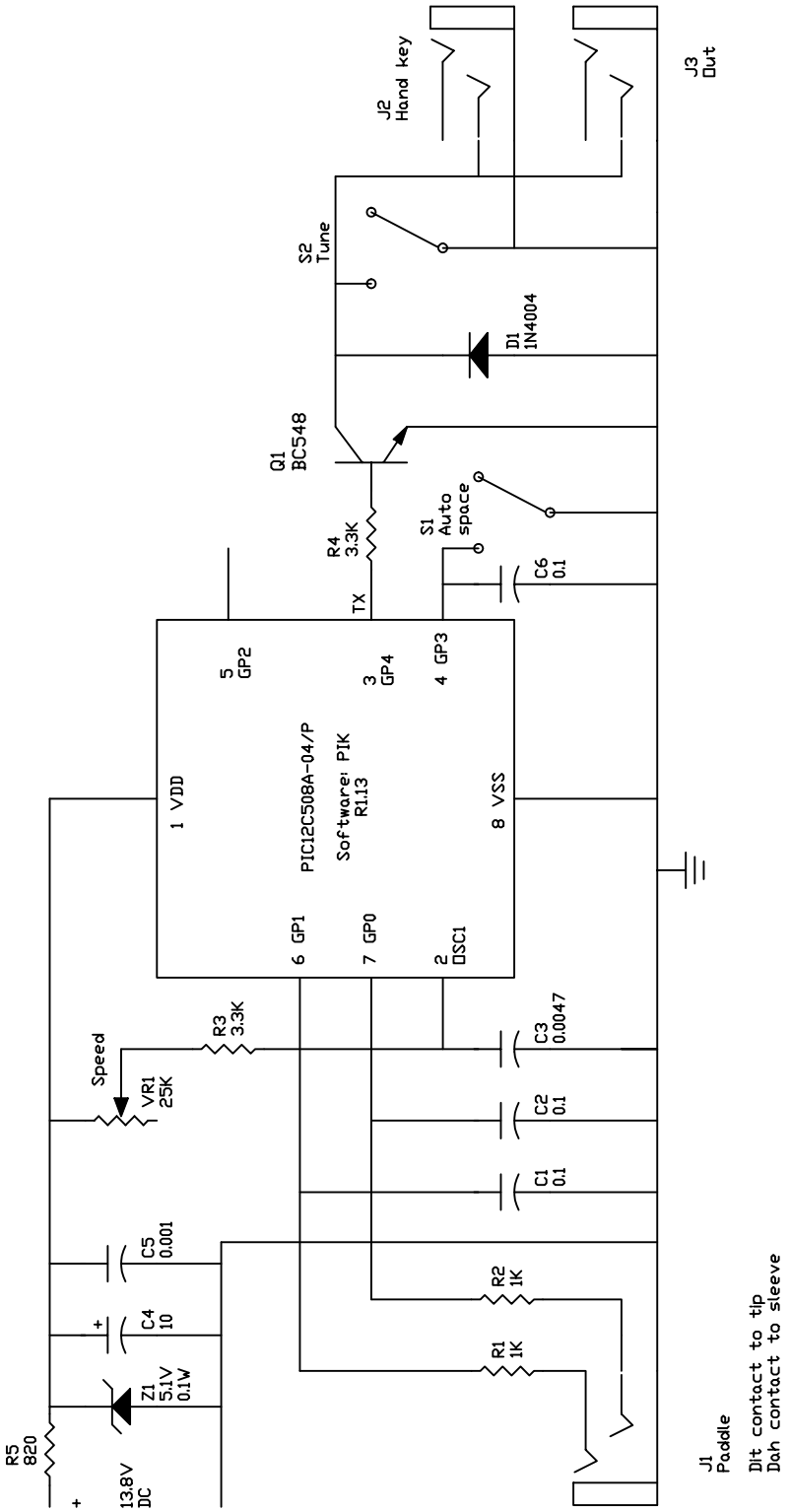
Average word duration will be 5 * 9.076 + 4 or 49.38

This represents 60/49.38/10 S (122mS rounded to the mS) per dit interval (Baud) at 10WPM. With that figure, the WPM speed is 2.44 times the dit frequency

I note that an older ARRL handbook suggests for electronic keyers that the WPM speed is 2.4 times the dit frequency, but it does not provide the derivation. This is about 1.6% lower than the above derivation.

VK1OD on the 'net, your feedback is welcome.

RevNo	Revision note	Date	Signature	Checked
1	V1.09 Pin reassignment, ~TX output	16/05/01	MD	MD
2	V1.13 C1, C2, C6 changed for power save mode	28/03/03	MD	MD



Itemref	Quantity	Title/Name, designation, material, dimension etc	Article No./Reference

Designed by	Checked by	Approved by - date	Filename	Date	Scale
Owen Duffy	OD	OD - 05/05/01	PIK001	05/05/01	1:1

Owen Duffy & Associates Pty Ltd

PIC Iambic Keyer

PIK-001	Edition 4	Sheet 1/1