The PICAXE-08M Morse Keyer

Gary E. J. Bold, ZL1AN, and Warwick Simpson

Document version 1.4a, uploaded 18 April 2005

Introduction

The keyer described here implements an iambic type B keyer with autospace. It uses a locally available chip, but you'll have to program it yourself, using the source code supplied. However, *you* have complete control over the code it runs and can change or experiment with it if you don't like it. The complete keyer schematic is shown in figure 1.

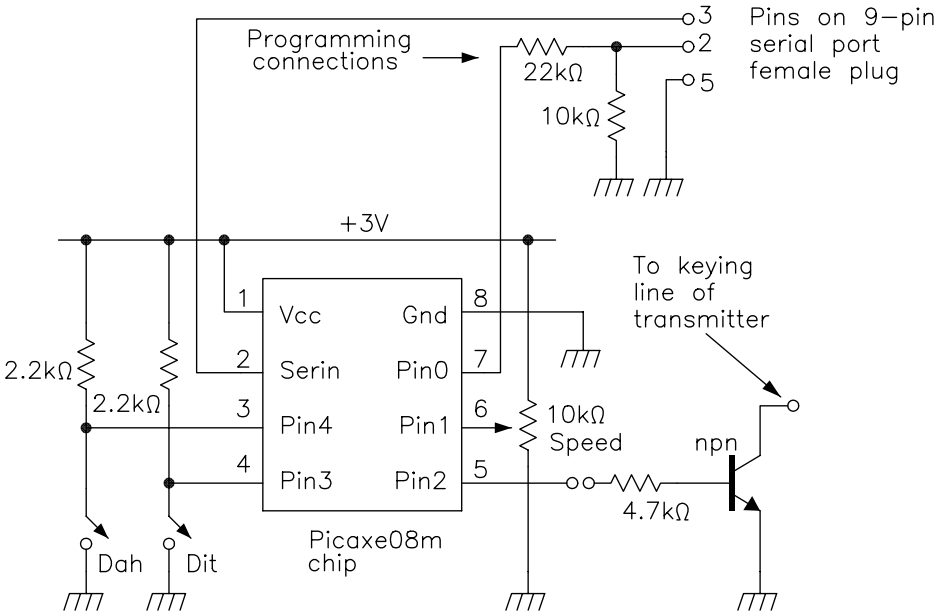

Figure 1: Schematic of PICAXE-08 keyer

1 Overview

Previous versions of this keyer used a PICAXE-08 chip. This one uses a PICAXE-08m, which has twice as much memory, and allows autospace - a feature which I insist on - to be included.

If you came here after reading my June/July 2004 *Morseman* column *Break-In* article, you'll see that this schematic of figure 1 contains two extra resistors, at top right, and the 3 connections used for programming the chip from a PC's 9-pin serial port. The top two pins had their numbers *transposed* in the document describing the earlier keyer. Connections to the serial download cable are shown in figure 2.

The *names* of the pins are shown *inside* the chip outline, while the *physical pin (leg) connections* are shown *outside*. Thus physical leg 3 is connected to the connection designated in the literature as "pin4". This is confusing. Use the *inside* numbers when writing programs! Use the *outside* numbers when wiring up the chip!

When loaded with the default program supplied in a separate file, you get an iambic, type *B* timing, autospace keyer with 16 speed steps in the range of about 13 to 33 wpm. Speed is controlled by the 10 kΩ potentiometer. Positive line logic level keying is implemented (standard on all modern transceivers) via a generic *npn* transistor, which can be any common type. I have used it now in many QSOs, and listeners are unable to tell whether I am using this, or my excellent CMOS Superkeyer. It

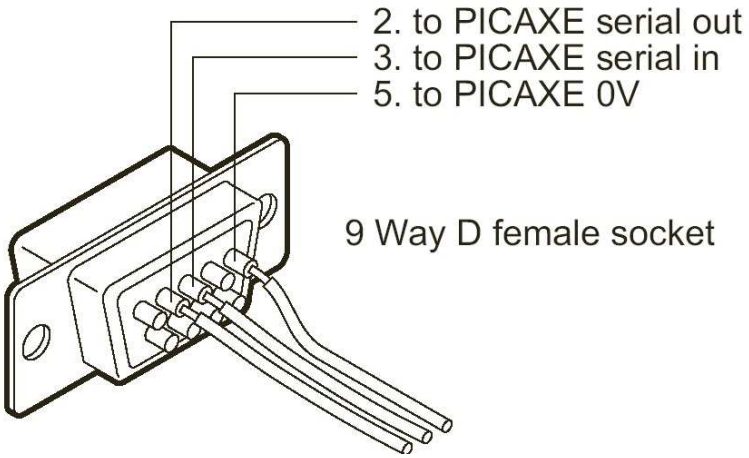

Figure 2: Serial plug connections

keys just the same, although the Superkeyer has many more features.

You will need to program the chip yourself at least once (initially) but then it can be disconnected from the PC.

2 The PICAXE Chips

These are PIC microcontrollers, pre-programmed with a simple `BASIC` interpreter developed by *Revolution Education*. This keyer uses the enhanced version of the simplest and cheapest of their range, the 8-pin PICAXE-08M. This chip runs from an in-built 4 MHz clock, so no external crystal is needed. Since you write code in `BASIC` instead of assembler language, it is *much* simpler to write and debug programs. Furthermore, *all* of the required software programming tools are downloadable, with documentation, free, from the *Revolution Education* website at

`http://www.rev-ed.co.uk/picaxe/`

You'll need to visit this website to download the software. The disadvantages of programming in `BASIC` instead of assembler are

- the code is *interpreted* statement by statement at run-time, instead of being compiled into machine language. Thus it runs slower, but nevertheless, fast enough to implement a keyer.
- The available program memory is limited to 256 bytes. However, this is sufficient for about 80 lines of `BASIC`, more than enough for this keyer.

The chip contains a 4-bit ADC (analogue to digital convertor) which we use to sense the keying speed.

3 Construction

You can build the keyer on a small section of veroboard, spotboard, or the evaluation pcb which can be obtained as a kitset. The Auckland agent is *Surplustronics*. They have a website.

4 The Keyer Source Code

```
 ; Keyer1b is an iambic Morse keyer
 ; using PicAxe08m (8 pin) chip.
 ; modified from Keyer1a, using 08 (smaller memory) chip
 ; implements autospace finction.
 ; connect dit paddle to ground and leg 4,
 ; dah paddle to ground and leg 3.
 ; connect legs 3 and4 to Vcc though separate 2.2 k resistors.
 ; leg 5 is output, high = on, suitable for driving transistor base
 ; for positive logic keying (pull down positive line)
 ; Speed pot slider connects to leg 6. Pot ends to Vcc and ground.
 ; runs fine from 2 AA cells.
 ; Keyer logic by Gary, ZL1AN
 ; Coded by Warwick Simpson
 ; uses 135 of 256 memory bytes
 ; This software version: Gary ZL1AN, 31 October 2004.

 symbol ditmemory = b1
 symbol dahmemory = b5
 symbol ditcontact = pin4        ; zero when closed
 symbol dahcontact = pin3        ; zero when closed
 symbol keyout = 2               ; high is on

 input 3          ; pin 3 is dah contact
 input 4          ;pin 4 is dit contact
 output keyout    ; set pin 2 as output low
 keyout           ; and set it low (off)

loop:
        if ditcontact = 0 then dit
        if dahcontact = 0 then dah
        goto loop

dit:                    ; send a dit plus space
        dahmemory = 0
        high keyout     ; turn output on

        gosub ditwait   ; for a ditspace

        low keyout      ; turn output off
        gosub ditwait   ; for a ditspace

        if dahmemory = 1 then dah
        goto autospace

dah:                          ; send a dah plus space
        high keyout
        ditmemory = 0

        b0=30
        gosub dahwait

        low keyout

        b0=10
        gosub dahwait
```

```
        if ditmemory = 1 then dit
        goto autospace

checkspeed:                     ; read the speed pot
        readadc 1,b3
        b3=b3/3
        b3=b3+20
        b4=b3/10
        goto loop

ditwait:                        ; send ditspace
        b0=10                   ; checking for dah set
ditwaitloop:
        pause b4
        if dahcontact = 1 then dontsetb5
        dahmemory = 1           ; next element is dah
dontsetb5:
        b0=b0-1
        if b0>0 then ditwaitloop
        return

dahwait:                        ; send dahspace
        pause b4                ; checking for dit set
        if ditcontact = 1 then dontsetb1
        ditmemory = 1           ;next element is dit
dontsetb1:
        b0=b0-1
        if b0>0 then dahwait
        return

autospace:
      if ditcontact=0 then dit
      if dahcontact=0 then dah
      gosub ditwait
      gosub ditwait
      goto checkspeed
```

This code is included separately as the ASCII text file `Keyer1a.bas`. This is the source code needed to compile and load the keyer chip.

5 Obtaining the Software

To download the software, go to the site

`http://www.rev-ed.co.uk/picaxe/`

Move the mouse over the menu item `PICAXE information`, select `Programming editor` from the sub-menu that appears. Go to the heading

Programming Editor Full Download (new user - not yet registered)

and click on the link below, labelled something like

Programming Editor v4.1.5 (full version, approx. 23MB)

The version number and size slowly increase as Revolution Education update their product.

As a new user, you will sent a password immediately by email, which enables you to unlock the zip file and install the software.

A start menu entry will be created under the heading "`Revolution Education`", entitled "`programming editor`". It's convenient to copy a short-cut to this onto the desktop, or into some other suitable folder.

Run the editor in the normal manner. When it starts, an "options" screen appears. Select `PICAXE-08M`. (or whichever chip you want to use. With pin number changes, *any* of the PICAXE chips can run this code). If you have only one serial port, this will be automatically selected, otherwise, go to the `serial port` option and select whichever port you want to use.

Select "`file/open`". The standard "open" dialog box appears. I created another folder called "`PicKeyer`" in this box, and copied the downloaded source code file "`Keyer1b.bas`" - or whatever is the latest version I've released - into that. Open this file. Once you've done this, the screen should look something like figure 3. Here the screen has been re-sized to fit the editor window(click the square button in the top toolbar)

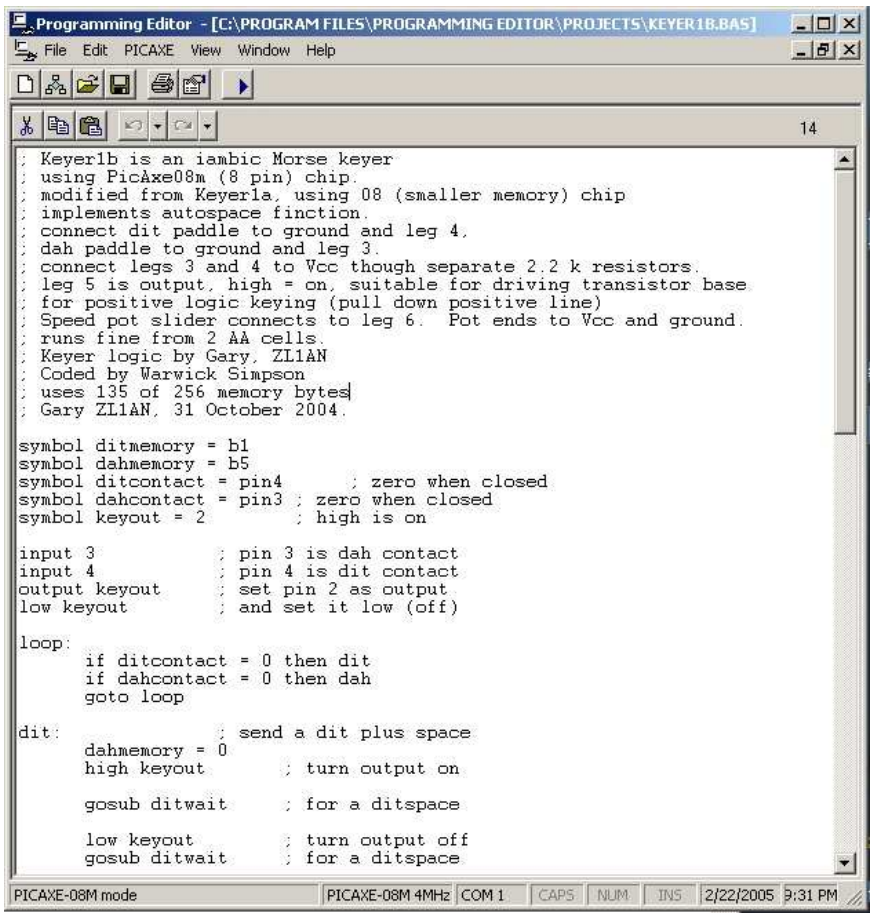

Figure 3: Screen of Editor

If you want to try a code of your own, select "`new`" instead, and a blank screen will appear. Many simple sample programs are included in the software distribution.

6 Programming the Chip

The useful operations you'll now need are in the sub-menu accessed by pressing the "`PICAXE`" button on the top menu bar. If this button doesn't appear, press the button "`view`", select "`options`" and make sure that the Picaxe-08M option is selected.

Check that the source code has loaded into the editor without corruption by selecting "`PICAXE/Check Syntax`". A box should appear telling you that it's OK.

Connect the chip to the serial port of the computer, using the pin numbers shown in figures 1 and 2. Apply power. Anything between 2 and 5 volts is fine. I use 2 AA cells.

Select "`PICAXE/run`". If the connections are correct and the chip has powered up correctly, a progress bar will appear, and after a second or so you'll be told that the operation was successful. If something was wrong, a box will appear reporting "`serial port error`". Remove power to the chip, and check the connections, the problem is probably there. However, if the program loads correctly, the keyer is ready to go and should now operate.

7 Introduction: The Alternatives:

To obtain a *pre-programmed*, versatile keyer chip, go to Steve, K1EL's website at

`http://k1el.tripod.com/`

where you can order either his K9 or K10 models, or his excellent K20 keyboard chip. Several other chips or kits are offered over the web, but I use Steve's, as they implement some features that I suggested, and so I like them. But you have to order them from Steve in the USA.

For the source code and instructions for making a keyer using an 8-pin PIC12C508A, programmed in PIC assembler language, see Owen Duffy's document at

`http://www.vk1OD.net/pik/pik.htm`

Owen also sells pre-programmed chips.