A TINY IAMBIC KEYER REVISITED

Andy Palm N1KSN, April 2009

This project is a software re-work of a small iambic keyer that I built and installed in a small homebrew crystal-controlled QRPp transceiver for 3.560 MHz. The original software was written in C and included three recorded messages “hardwired” into the code. A single pushbutton was used to play the messages or change code speed in conjunction with the paddles. A PIC12F675 was used.

After successfully building a simple keyer programmed in PIC assembly language that felt very smooth in operation, I decided to re-work the old keyer’s software. To keep things simple, I eliminated the recorded messages but retained the sending speed adjustment feature with the pushbutton. I also kept the straight key feature. If a mono plug is in the key jack (or if the dah paddle contact is closed) on power-up, then the keyer simply passes the straight key (or dit paddle contact) closures through to the rig.

The basic iambic keying code was taken from the simple keyer project mentioned above and the straight key and code speed change functions from the older keyer were rewritten in assembly language. Since this keyer has to run at a fixed oscillator speed (unlike the simple keyer), the dit delay routine was also rewritten to use a variable dit delay time in milliseconds and sending speed is changed in software.

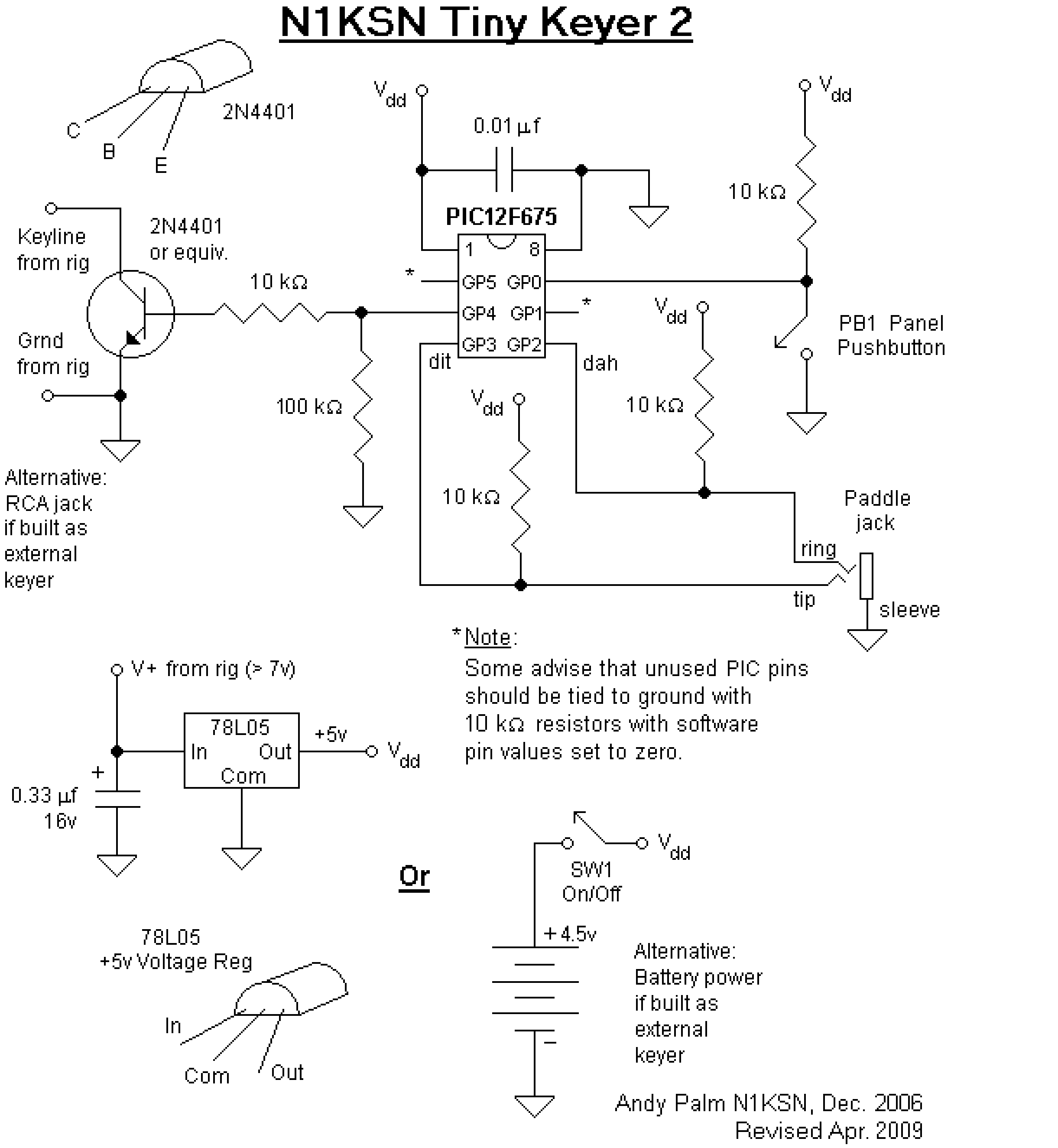
When the pushbutton is held down, the dit paddle increases the Words per Minute (WPM) sending speed and the dah paddle decreases the speed. Once the new speed is selected, the corresponding dit delay time is looked up in a table. The table entries are derived from the formula

$$\text{Dit delay time (ms)} = 1200 / \text{WPM}$$

The table entries are approximate due to the need to round them to whole numbers.

Since this project was a pin-to-pin replacement for the older keyer, the software was set up to use the pre-existing external pull-ups on the paddle and button input pins. However, the sleep features of the simple keyer were retained to help conserve some (perhaps very little) power. The schematic diagram is in Figure 1 below. This circuit could also be used for a very compact stand-alone keyer.

The new keyer works quite smoothly and I don’t miss the message feature which I hardly used anyway.



*Note:
Some advise that unused PIC pins should be tied to ground with 10 kΩ resistors with software pin values set to zero.

Andy Palm N1KSN, Dec. 2006
Revised Apr. 2009

Figure 1. Schematic diagram of Tiny Keyer 2.

The next three pages contain the program listing. Items of note are the method of doing a simple table lookup for the dit delay time (in milliseconds) for the current WPM sending speed and the dit delay loop routine. Also note the code for using the PIC12F675 factory calibration value for the internal RC clock. If internal weak pull-ups are used instead of the external pull-ups shown in Fig. 1, two lines in the initialization section should be “uncommented.”

```
  title  "asmTinyKeyer2 - Tiny keyer with single button speed control"
;======================================================================
;  This keyer is primarily for installation in a transceiver.  It uses
;  a pushbutton together with the paddles to made allow speed
;  adjustments.  When the button is held down the dit paddle increases
;  sending speed, the dah paddle decreases speed.
;
;  If the ring (dah) contact is grounded on power-up, the program
;  functions in straight key mode, the tip (dit) contact being the
;  straight key input.
;
;  This version uses external pullups since its target application
;  is an existing installation.  If weak pullups are used for the
;  inputs there may be better power savings under the sleep command.
;
;  Hardware Notes:
;
;    PIC12F675 running on 4 MHz internal RC oscillator.
;
;    Keyed output is GP4, Pin 3.
;    Dit paddle input is GP3, Pin 4, with external pullup.
;    Dah paddle input is GP2, Pin 5, with external pullup.
;    Speed control button input is GP0, Pin 7, with external pullup.
;
;  Andrew Palm
;  2009.04.03
;======================================================================
;------------ Defines, Includes, and Configuration Word ---------------

  #define OUTPUT GPIO, 4    ; Keyed output to transmitter
  #define BUTTON GPIO, 0    ; Speed control button, = 1 if pressed
  #define DIT_IN GPIO, 3    ; Dit paddle input
  #define DAH_IN GPIO, 2    ; Dah paddle input

  #define DIT_BUFFER Buffers, 0 ; = 1 if dit paddle pressed
  #define DAH_BUFFER Buffers, 1 ; = 1 if dah paddle pressed

; These values must be consistent with Dit_ms_Table below
  #define DIT_MS_INIT 60    ; Default dit high loop count (ms)
  #define WPM_INIT 20       ; Default words per minute

  LIST R=DEC
  INCLUDE "p12f675.inc"
  ERRORLEVEL -302, -305

  __CONFIG  _CP_OFF & _CPD_OFF & _BODEN_OFF & _MCLRE_OFF & _WDT_OFF & _PWRTE_ON & _INTRC_OSC_NOCLKOUT

;--------------------- Variables --------------------------------------
  CBLOCK 0x20
    HCount, LCount      ; Counters for delay loops
    Dit_ms              ; Upper delay loop value for dit delay
    Wpm                 ; Words per minute sending speed
    Buffers             ; Buffers for paddle inputs
  ENDC

;----------------------- Main -----------------------------------------
  ORG     0x00
  nop                       ; For ICD Debug

; Calibrate internal clock
  call    0x3FF           ; Retrieve factory calibration value
  bsf     STATUS, RP0     ; Set file register bank to 1
  movwf   OSCCAL          ; Update register with factory cal value
  bcf     STATUS, RP0     ; Set file register bank to 0

; Initialize
  clrf      GPIO            ; Initialize I/O bits to off

  movlw     7               ; Turn off comparators
  movwf     CMCON
  bsf       STATUS, RP0     ; Bank 1
  clrf      ANSEL           ; All bits are digital
  movlw     b'101111'       ; Only GP4 an output
  movwf     TRISIO
  movlw     b'001101'       ; Interrupt on GPIO input change
  movwf     IOC             ; for both paddle and button inputs
  ; Uncomment two statements below if weak pullups are used
  ; movwf     WPU             ; Weak pullups on inputs
  ; bcf       OPTION_REG, 7   ; Enable weak pullups
  movlw     b'00001000'     ; Enable peripheral interrupts (GPIE)
  movwf     INTCON          ; but NOT overall interrupt (GIE)
  bcf       STATUS, RP0     ; Bank 0

  clrf      Buffers         ; Clear paddle input buffers
  movlw     DIT_MS_INIT     ; Initialize dit delay high loop count
  movwf     Dit_ms
  movlw     WPM_INIT        ; Initialize Words per Minute sending speed
  movwf     Wpm

  call      StraightKey     ; Check for closed dah contact on power-up
                            ; for straight key mode

Loop:                       ; Main loop
  sleep                     ; Sleep, awake on paddle input
  btfss     DIT_IN          ; Is dit paddle pressed (=0)?
  bsf       DIT_BUFFER      ; Yes, set dit buffer
  btfss     DAH_IN          ; Is dah paddle pressed (=0)?
  bsf       DAH_BUFFER      ; Yes, set dah buffer

  btfss     DIT_BUFFER      ; Send dit if dit buffer = 1
  goto      Loop2
  btfss     BUTTON          ; Check for button press
  call      Incr_Speed      ; If pressed, increment sending speed
  bcf       DIT_BUFFER      ; Clear dit buffer
  bsf       OUTPUT          ; Key output
  call      Delay_dit       ; Wait for length of dit
  bcf       OUTPUT          ; Unkey output
  btfss     DAH_IN          ; Is dah paddle pressed (=0)?
  bsf       DAH_BUFFER      ; Yes, set dah buffer
  call      Delay_dit       ; Wait for length of dit

Loop2:
  sleep                     ; Sleep, awake on paddle input
  btfss     DIT_IN          ; Is dit paddle pressed (=0)?
  bsf       DIT_BUFFER      ; Yes, set dit buffer
  btfss     DAH_IN          ; Is dah paddle pressed (=0)?
  bsf       DAH_BUFFER      ; Yes, set dah buffer

  btfss     DAH_BUFFER      ; Send dah if dah buffer = 1
  goto      Loop
  btfss     BUTTON          ; Check for button press
  call      Decr_Speed      ; If pressed, decrement sending speed
  bcf       DAH_BUFFER      ; Clear dah buffer
  bsf       OUTPUT          ; Key output
  call      Delay_dit       ; Wait for length of dah
  call      Delay_dit
  call      Delay_dit
  bcf       OUTPUT          ; Unkey output
  btfss     DIT_IN          ; Is dit paddle pressed (=0)?
  bsf       DIT_BUFFER      ; Yes, set dit buffer
  call      Delay_dit       ; Wait for length of dit
  goto      Loop

;------------------- Subroutines -------------------------------------
; Straightkey detection and operation
;
StraightKey:
  btfsc     DAH_IN          ; Is dah contact open on power-up?
  return                    ; Yes, return to main routine
SK_Loop:                    ; No, loop forever in straight key mode
  sleep
  btfsc     DIT_IN          ; Straight key (tip connection) closed?
  goto      Unkey_Rig       ; No
  bsf       OUTPUT          ; Yes, key transmitter
  goto      SK_Loop
Unkey_Rig:
  bcf       OUTPUT          ; Unkey transmitter
  goto      SK_Loop
```

```
;----------------------------------------------------------------------
; Change sending speed between max and min limits and retrieve
; corresponding dit delay time (ms) from table
;
  #define WPM_MAX 30
  #define WPM_MIN 12
Incr_Speed:
  movlw     WPM_MAX         ; Is Wpm < WPM_MAX?
  subwf     Wpm, w
  btfsc     STATUS, C
  return                    ; No, return
  incf      Wpm             ; Yes, increment WPM
  goto      Get_Dit_ms

Decr_Speed:
  movfw     Wpm             ; Is Wpm > WPM_MIN?
  sublw     WPM_MIN
  btfsc     STATUS, C
  return                    ; No, return
  decf      Wpm             ; Yes, decrement WPM

Get_Dit_ms:                 ; Get new dit delay time
  movlw     WPM_MIN         ; Calculate table offset
  subwf     Wpm, w          ; w = Wpm - WPM_MIN
  call      Dit_ms_Table    ; Look up new dit delay time in table
  movwf     Dit_ms          ; Store new dit delay time
  return

Dit_ms_Table:   ; Dit delay time = 1200 / WPM
  addwf     PCL, f          ; Add offset to program counter
  dt    100 ; 12 WPM
  dt     92 ; 13 WPM
  dt     86 ; 14 WPM
  dt     80 ; 15 WPM
  dt     75 ; 16 WPM
  dt     71 ; 17 WPM
  dt     67 ; 18 WPM
  dt     63 ; 19 WPM
  dt     60 ; 20 WPM
  dt     57 ; 21 WPM
  dt     55 ; 22 WPM
  dt     52 ; 23 WPM
  dt     50 ; 24 WPM
  dt     48 ; 25 WPM
  dt     46 ; 26 WPM
  dt     44 ; 27 WPM
  dt     43 ; 28 WPM
  dt     41 ; 29 WPM
  dt     40 ; 30 WPM

;----------------------------------------------------------------------
; Loops for dit delay time given by
;   Delay = HCount * (5 us * (LOW_COUNT + 1))
; based on 4 MHz clock = 1 us per basic operation.  Delays for Dit_ms
; milliseconds.
;
  #define LOW_COUNT 199     ; Inner loop count for Delay = HCount * 1 ms
Delay_dit:
  movfw     Dit_ms
  movwf     HCount          ; Counter for outer (high) loop
  movlw     LOW_COUNT
  movwf     LCount          ; Counter for inner (low) loop
  nop                       ; 1 us to give 5 us inner loop
  nop                       ; 1 us
  decfsz    LCount          ; 1 us
  goto      $ - 3           ; 2 us
  decfsz    HCount
  goto      $ - 7
  return

  END
```