A SIMPLE IAMBIC KEYER

Andy Palm N1KSN, March 2009

This project is based on the keyer circuit of amateur radio operator EI9GQ as found in http://homepage.eircom.net/~ei9gq. What is unique and intriguing about this circuit is that the code speed is controlled by varying the frequency of an external RC clock circuit driving the microcontroller chip. Usually the MCU has a fixed frequency oscillator and speed changes are made in software. This approach restricts keyer functionality, but greatly simplifies the software.

The original circuit used the PIC16F84, but I decided to use the smaller PIC12F675, an 8-pin chip. An even less expensive model could be used, as the software does not make use of the chip's peripherals. Also, the program from the website above did not work very well, so I substituted my own main routine, keeping a modified version of the delay subroutine of EI9GQ. I wrote the program in assembly language to guarantee speed of execution and so I could experiment with the PIC power reduction command SLEEP.

The schematic diagram is in Figure 1 below. The 50K pot and 27K fixed resistor together with the 39 pf capacitor form the RC circuit for the oscillator. As the pot resistance is decreased, the oscillator frequency, given by freq = 1/RC , increases which in turn increases the code sending speed. The resistance and capacitance values were chosen to give a WPM range of approximately 13 to 35 WPM. Details of the timing calculations are in the delay subroutine comments in the code listing. Breadboard empirical adjustments to the component values were necessary, but the calculations got me close.

In an attempt to reduce power draw when the MCU is sleeping, I used internal pull-ups on the two paddle input pins. If you are concerned about high noise on the paddle lines, you can substitute external 10K pull-up resistors (and change the program code accordingly), but this may increase power draw. The output circuit is an NPN switching transistor.

I chose to power the unit with three AAA cells giving about 4.5 volts. The current draw of the circuit is very low, so they should last for a long time. I used a separate on/off switch rather than one attached to the speed control so I could keep the speed setting between operating sessions. Although the speed control isn't calibrated, it's pretty easy to find the right speed with a few dits.

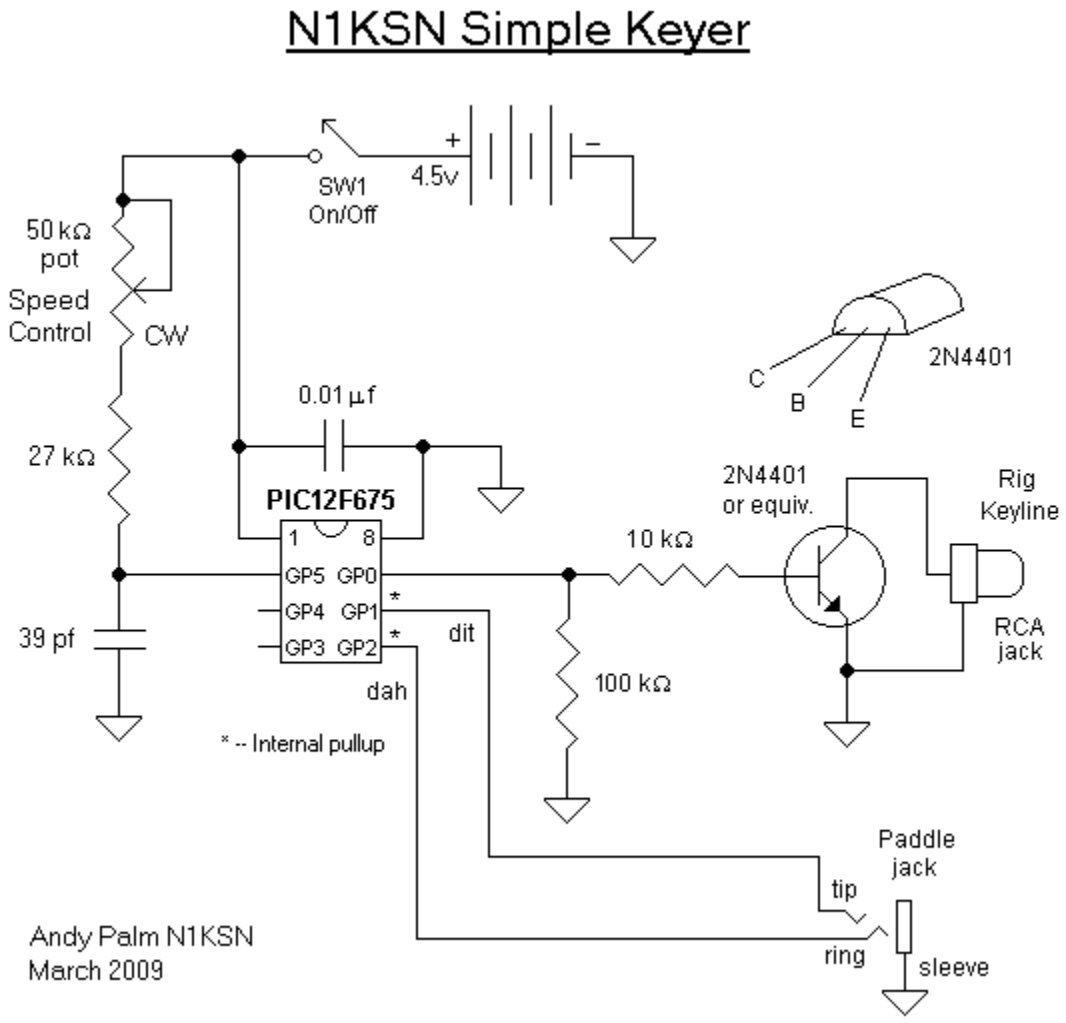

Figure 1. Schematic diagram of Simple Keyer.

The next two pages contain the program listing. Some simple #define statements for particular register bits are used to make the code a bit more readable. The variable Buffer contains two flags which indicate if a closure has been detected on the dit or dah paddle.

To wake up from the SLEEP command when a paddle is pressed, one must set up the input pin interrupt-on-change feature. This requires setting bits in the IOC register and enabling peripheral interrupts with GPIE set in INTCON. However, the overall interrupt enable bit GIE must not be set to avoid jumping to the interrupt address upon waking.

Note the additional (opposite) paddle read while a dit or dah is being sent. This is necessary to get smooth operation of the keyer.

```
  title  "asmKeyer - Very simple iambic keyer"
;=====================================================================
;  This project is based on the keyer circuit of amateur radio
;  operator EI9GQ as found in http://homepage.eircom.net/~ei9gq.
;  The code speed is controlled by varying the resistance in an
;  external RC oscillator attached to GP5.  However, only modified
;  delay subroutine code from EI9GQ is used, the rest being different.
;  Also, the PIC12F675 is used instead of the PIC16F84.
;
;  Hardware Notes:
;
;    PIC12F675 running on external variable RC oscillator attached
;    to GP5.  External oscillator component values are C = 39 pf,
;    R fixed = 27K, R variable 50K.  These values, plus the
;    loop count values in the delay subroutine give a useable range
;    of words per minute (WPM).  See comments in delay routine below.
;
;    Keyed output is GP0, Pin 7, pulled down by 100K resistor, then
;    to base of keying NPN transistor through 10K resistor.  Output
;    to transmitter is collector of transistor.
;
;    Dit paddle input is GP1, Pin 6, with internal pullup.
;    Dah paddle input is GP2, Pin 5, with internal pullup.
;
;  Andrew Palm
;  2009.03.31
;=====================================================================
;------------ Defines, Includes, and Configuration Word ---------------

  #define OUTPUT GPIO, 0    ; Keyed output to transmitter
  #define DIT_IN GPIO, 1    ; 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

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

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

;---------------------- Variables -----------------------------------
  CBLOCK 0x20
    HCount, LCount      ; Counters for dit delay
    Buffers             ; Buffers for paddle inputs
  ENDC

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

; 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'111110'       ; Only GP0 an output
  movwf     TRISIO
  movlw     b'000110'
  movwf     WPU             ; Weak pullups on inputs
  bcf       OPTION_REG, 7   ; Enable weak pullups
  movwf     IOC             ; Interrupt on GPIO input change
  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

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
  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
  bcf       DAH_BUFFER      ; Clear dah buffer
  bsf       OUTPUT          ; Key output
  call      Delay_dah       ; Wait for length of dah
  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 -------------------------------------
; Delay loop for dahs and dits
; Delay clock ticks for a dit is approximately given by:
;    12 clock ticks per inner loop x 200 interations inner loop
;      x 10 interations outer loop = 24,000 clock ticks per dit
; The clock frequency should then be:
;    freq = ticks per sec = (24000 ticks per dit) /
;                                 [(1.2 / WPM) sec per dit]
;         = 20000 x WPM
; Thus:
;     With R = 77 Kohm, C = 39 pf, f = 1/RC = 333 KHz, WPM = 16.7.
;     With R = 27 Kohm, C = 39 pf, f = 1/RC = 950 KHz, WPM = 47.5.
;
; These are approximate starting values for empirical determination.
; When used, range of code speed was actually more like 10 to 40.
;
  #define DAH_HCOUNT 0x1E   ; Outer loop count for dah = 30
  #define DIT_HCOUNT 0x0A   ; Outer loop count for dit = 10
  #define LOW_COUNT 0xC8    ; Inner loop count = 200
Delay_dah:
  movlw     DAH_HCOUNT
  goto      $ + 2
Delay_dit:
  movlw     DIT_HCOUNT
  movwf     HCount          ; Counter for outer (high) loop
  movlw     LOW_COUNT
  movwf     LCount          ; Counter for inner (low) loop
  decfsz    LCount          ; Inner loop 3 ops = 12 clock ticks
  goto      $ - 1
  decfsz    HCount
  goto      $ - 5
  return

  END
```