Appendix I – Firmware Source Code for Duodecimal Clock

The source code was written in PIC Basic Professional, available from microEngineering Labs, Inc., 2845 Ore Mill Road, STE 4, Colorado Springs CO 80904, telephone 719-520-5323, http://melabs.com.

```
'*****************************************************************
'*  Name    : e-key                                              *
'*  Author  : D. M. Gualtieri                                    *
'*  Notice  : Copyright (c) 2014 D.M. Gualtieri                  *
'*          : All Rights Reserved                                *
'*  Date    : 06/06/2014                                         *
'*  Version : 0.4                                                *
'*  Notes   : Firmware for PIC12F675                             *
'*  Notes   : electronic key w/o RFID                            *
'*****************************************************************

' PIC PIC12F675 Pin Functions
' GP0 = Output (1=Proper Key)
' GP1 = Timer Gate
' GP2 = Timer 0 Clock Input
' GP3 = Key Jumper 0
' GP4 = Key Jumper 1
' GP5 = Optional oscillator input

' The usual congiguration
'     __CONFIG  _CP_OFF & _WDT_OFF & _BOD_ON & _PWRTE_ON &
'_INTRC_OSC_NOCLKOUT & _MCLRE_OFF & _CPD_OFF

@ DEVICE MCLR_OFF 'Use GP3 as an input pin
@ DEVICE BOD_ON
@ DEVICE WDT_OFF

DEFINE OSCAL_1K 1
Define OSC 4  ' 4 MHz internal clock
'Define OSC 20  ' Use for 20 MHz external clock

F_Gate var GPIO.1
KEY_OUT var GPIO.0
K0 var GPIO.3
K1 var GPIO.4

freq var word
f_avg var word
f_key var word
```

```
OK_key var byte

I var word
ps_complement var byte

' Interrupts in assembly language

wsave   var     byte $20 system
ssave   var     byte bank0 system
psave   var     byte bank0 system

Goto main ' Jump around interrupt handler

' Interrupt handler here

define  INTHAND myint

' Assembly language interrupt handler

asm
; Save W, STATUS and PCLATH registers
myint   movwf   wsave
        swapf   STATUS, W
        clrf    STATUS
        movwf   ssave
        movf    PCLATH, W
        movwf   psave
ENDASM

; Interrupt code is called here

gosub get_frequency

'Set up for next interrupt

PIR1 = $00 ' clear interrupt flags
T1CON = %00000000  'bit 0 = TMR1 disable
'Freq = 25 Hz for 4 MHz clock
TMR1H = 236
TMR1L = 120
T1CON = %00110001  'bit 0 = TMR1 enable; bits 4-5 = prescaler (11 =
1:8)

ASM
; Save and restore registers
        bcf     INTCON, 1       ; Clear interrupt flag
```

```
; Restore saved registers
        movf    psave, W
        movwf   PCLATH
        swapf   ssave, W
        movwf   STATUS
        swapf   wsave, F
        swapf   wsave, W
        retfie                  ; Return from interrupt
endasm


main:
'let oscillator stabilize
Pause 500
'GPIO 0 set as output, GP1 originally set as output
'GPIO 2-GPIO 5 set as inputs
' 1 = Input and 0 = Output
TRISIO = %00111100

'Set Analog Control Registers
CMCON = %00000111 'All comparators off
ANSEL = %00000000 'All A/D off

'Key output initially zero
KEY_OUT = 0

'frequency counter input gated off
F_Gate = 0

'set timer 0 option register
'external clock with 1:256 prescaler
OPTION_REG=%11100111

'clear timer 0
TMR0 = 0

'key state initially false
KEY_OUT=0

'Initial values
I = 0
ps_complement = 0
freq = 0
OK_key = 0

'Interrupt parameters
```

```
T1CON = %00000000  'bit 0 = TMR1 disable
'Freq = 25 Hz for 4 MHz clock
TMR1H = 236
TMR1L = 120
T1CON = %00110001  'bit 0 = TMR1 enable; bits 4-5 = prescaler (11 =
1:8)

' Set interrupt control registers
PIE1 = %00000001
PIR1 = 0 ' clear interrupt flags
'Enable Global Interrupts and Interrupt on TMR1
INTCON = %11000000


Operate:
'Infinite loop
goto operate

key_set:
'check for presence of jumpers
'normal operation if no jumpers
IF ((K0==1)&&(K1==1)) then
return
endif
'key one register
IF ((K0==1)&&(K1==0)) then
WRITE 0, f_avg.highbyte
WRITE 1, f_avg.lowbyte
endif
'key two register
IF ((K0==0)&&(K1==1)) then
WRITE 2, f_avg.highbyte
WRITE 3, f_avg.lowbyte
endif
'key three register
IF ((K0==0)&&(K1==0)) then
WRITE 4, f_avg.highbyte
WRITE 5, f_avg.lowbyte
endif
'halt to power down and remove jumpers
STOP


key_check:
OK_key = 0
READ 0, f_key.highbyte
```

```
READ 1, f_key.lowbyte
'compare f_key and f_avg
IF ((f_avg>f_key)&&((f_avg-f_key)<500)) then
OK_key = OK_key + 1
endif
IF ((f_key>f_avg)&&((f_key-f_avg)<500)) then
OK_key = OK_key + 1
endif
READ 2, f_key.highbyte
READ 3, f_key.lowbyte
'compare f_key and f_avg
IF ((f_avg>f_key)&&((f_avg-f_key)<500)) then
OK_key = OK_key + 1
endif
IF ((f_key>f_avg)&&((f_key-f_avg)<500)) then
OK_key = OK_key + 1
endif
READ 4, f_key.highbyte
READ 5, f_key.lowbyte
'compare f_key and f_avg
IF ((f_avg>f_key)&&((f_avg-f_key)<500)) then
OK_key = OK_key + 1
endif
IF ((f_key>f_avg)&&((f_key-f_avg)<500)) then
OK_key = OK_key + 1
endif
IF (OK_key > 0) then
KEY_OUT = 1
else
KEY_OUT = 0
endif
return

get_frequency:
'Disable timer and interrupts
'Gate off frequency counter input (make GPIO.1 an output set to zero)
F_Gate=0
'1 = Input and 0 = Output
TRISIO = %00111100
'turn off timer and disable interrupt
T1CON = %00110000  'bit 0 = TMR1 enable; bits 4-5 = prescaler (11 =
1:8)
PIE1.0 = 0 'Disable timer interrupt
INTCON.7 = 0 'Disable GIE interrupt
INTCON.6 = 0 'Disable PEIE interrupt
```

```
'Get frequency high byte from timer 0 register
freq.highbyte = TMR0

'do increment routine to find prescaler value = freq.lowbyte
while (freq.highbyte == TMR0)
ps_complement = ps_complement + 1
F_Gate = 1
F_Gate = 0
wend

freq.lowbyte = 255-ps_complement

I = I+1

If (I==4) then
f_avg = 0
endif

If (I>3) then
f_avg = f_avg + (freq>>2)
endif

If (I>7) then
I=0
gosub key_set
gosub key_check
endif

'reset timer 0 register
TMR0 = 0

'Re-enable timer and interrupts
PIR1 = 0 ' clear timer 1 interrupt flag
PIE1.0 = 1 'Enable timer interrupt
INTCON.7 = 1 ' enable GIE interrupt
INTCON.6 = 1 ' enable PEIE interrupt
T1CON = %00110001  'bit 0 = TMR1 enable; bits 4-5 = prescaler (11 =
1:8)
'Gate on frequency counter input (make GPIO.1 and input)
'1 = Input and 0 = Output
TRISIO = %00111110

return

End
```