Programming Editor (c) Revolution Education Ltd 1996-2009

Filename: c:\windows\desktop\ray\fs5000pi\fs5000_v22.bas
Date: 24/02/10

```
;*****************************************************************
;                  FS5000 DCU (Digital Control Unit) V2.2
;                            27 October 2009
;                           Ray Robinson vk2no
;*****************************************************************
; Initialise LCD
; Write a character to the LCD
; Scan the keyboard
; Get the key legend from the buffer
; Do not repeat a key
; A = toggle receiver on/off
; B = toggle bandwidth 300/3000
; C = enter COMMON frequency (receive)
; D = enter DIFFERENT frequency (transmit)
; # = receive
; * = transmit
;-----------------------------------------------------------------
; DEVELOPMENT
; 1/. Changed all routines, so that all "portc" output lines are always HIGH,
; and that they go LOW only when doing something, then return HIGH.
; Used for scanning the keyboard, and loading the LCD display.
; 2/. Changed all routines, so that all "port" output lines are always LOW,
; and that they go HIGH only when doing something, then return LOW.
; Used for loading data into the CONTROL latch, FSET latch, and chip select.
; 3/. Unreliable LCD setup fixed. Was due to LCD enable line being made low
; in the INIT routine, and that first negative edge, clocked in unkown data.
; Making these lines always HIGH, removed this problem.
; 4/. Transmitter frequency load, turns radio OFF! Why?
; 5/. Added variables KEY and TEMP so that variable CHAR is only used for LCD display
; 6/. Made Printed Circuit Board. Rewrite all routines.
; 7/. New PICAXE 40X-1. Requires editor V5.2.7 (Dec 09)
; 8/. New PICAXE 40X-1 has no osc waveforms on external pins, all internal.
; 9/. LCD, keyboard works
; 10/. A key turns radio ON/OFF
; 11/. B key changes bandwidth N/W (Dec09)
; 12/. display status of synth lock line
; 13/. subtract offset from tx and rx frequency
; 14/. set transmitter LPF
; 15/. power on/off control (Jan10)
;
;-----------------------------------------------------------------
;
CONSTANTS:
        SYMBOL      TMP    = b0         ; Temp
        SYMBOL      TXM    = b1         ; Transmitter mode (1=TX, 0=RX)
        SYMBOL      FPTR   = b2         ; Frequency pointer
        SYMBOL      FCNT   = b3         ; Frequency digit count
        SYMBOL      KEY    = b4         ; Key
        SYMBOL      KYOLD  = b5         ; old key
        SYMBOL      ROW    = b6         ; Keyboard row
        SYMBOL      COL    = b7         ; Keyboard column
        SYMBOL      CNT    = b8         ; Keyboard counter
```

(continued...)

```
        SYMBOL      KPTR    = b9            ; Keyboard pointer
        SYMBOL      CHAR    = b10           ; Character store
        SYMBOL      STAT    = b11           ; Status
        SYMBOL      FSET    = b12           ; FREQUENCY port status
        SYMBOL      CTRL    = b13           ; CONTROL port status
        SYMBOL      FSETI = %00000000   ; initial FREQUENCY port status
        SYMBOL      CTRLI = %00000000   ; initial CONTROL status
        SYMBOL      CMD     = %00000000 ; select LCD COMMAND register (bit 4 low)
        SYMBOL      DAT     = %00010000 ; select LCD DATA register (bit 4 high)
        SYMBOL      NIBL    = %00001111 ; Mask for low nibble
        SYMBOL      NIBH    = %11110000 ; Mask for high nibble
        SYMBOL      LCDL1 = 128             ; Move LCD to start of line 1
        SYMBOL      LCDL2 = 192             ; Move LCD to start of line 2
        SYMBOL      LCDL3 = 207             ; Move LCD to last position of line 2
        SYMBOL      LCDL4 = 206             ; Move LCD to second last position of line 2
        SYMBOL      RAM1    = $50           ; RAM memory START (48 bytes)
        SYMBOL      RAM1E = $7F             ; RAM memory END
        SYMBOL      RAM2    = $C0           ; RAM memory START (64 bytes)
        SYMBOL      RAM2E = $FF             ; RAM memory END
        SYMBOL      FQB     = RAM1          ; Frequency input buffer
        SYMBOL      RXFQB  = FQB + 7        ; Frequency Receiver buffer
        SYMBOL      TXFQB  = RXFQB + 7  ; Frequency Transmitter buffer
        SYMBOL      KYB_BUF         = RAM2          ; Keyboard buffer start
        SYMBOL      KYB_BUFE        = RAM2E         ; Keyboard buffer end
        SYMBOL      LEN     = 15                    ; Title string length -1
        SYMBOL      KLR0    = LEN + 1               ; Keyboard legend row0
        SYMBOL      KLR1    = KLR0 + 12             ; Keyboard legend row1
        SYMBOL      KLR2    = KLR1 + 12             ; Keyboard legend row2
        SYMBOL      KLR3    = KLR2 + 12             ; Keyboard legend row3
        SYMBOL      KLR4    = KLR3 + 12             ; Keyboard legend row4
        SYMBOL      RFQI    = KLR4 + 12             ; Frequency for rx init
        SYMBOL      TFQI  = RFQI + 7                ; Frequency for tx init
        SYMBOL      FROCW = TFQI + 7                ; Frequency offset for rx cw
        SYMBOL      FTOCW = FROCW + 7           ; Frequency offset for tx cw
        SYMBOL      DEND    = FTOCW + 7             ; Data end
        DATA        0,("FS5000 DCU  V2.2"); Title string
        DATA        KLR0,   (".....123A")   ; row 0 keyboard legend
        DATA        KLR1,   (".....456B")   ; row 1 keyboard legend
        DATA        KLR2,   (".....789C")   ; row 2 keyboard legend
        DATA        KLR3,   (".....*0#D")   ; row 3 keyboard legend
        DATA        KLR4,   (".....")       ; row 4 keyboard legend
        DATA        RFQI, ("0070300")               ; frequency to initialise receiver
        DATA        TFQI, ("0360000")               ; frequency to initialise transmitter
        DATA        FROCW,("0000100")               ; Frequency offset for receiver in CW mode
        DATA        FTOCW,("0020000")               ; Frequency offset for transmitter in CW mode
;
;-----------------------------------------------------------------
; uses no variables
; calls INIT, LCD_INIT, TITLE, KYBD_SCAN, KYBD_CHK, ACT
;
START:                                  ; here on reset and power on
        gosub INIT                              ; initialise ports and variables
```

(continued...)

```
        gosub LCD_INIT                  ; initialise the LCD display
        gosub TITLE                     ; write the title
        gosub BW                        ; set the bandwidth
        gosub RX                        ; set receive mode
LOOP1:
        gosub KYBD_SCAN                 ; scan the keyboard
        gosub  KYBD_CHK                 ; identify the key
        gosub ACT                       ; action the key
        gosub  SLK                      ; check SYNTH LOCK line
        goto  LOOP1                     ; loop
;
;*********************************************************************
;ok              Action the key
;*********************************************************************
; uses KYOLD, KEY
; calls LCD_CHAR,BW,RX
; on entry with KEY = $FF, nothing happens
; on entry with KEY = a valid ASCII character, it is actioned
; on entry with KEY = KYOLD, nothing happens (as this is a repeat)
; when returning from a subroutine, make sure KEY, KYOLD have not changed
;
ACT:
        if KEY = $FF then ACTX                  ; exit if no key found
        if KYOLD = KEY then ACTX1       ; jump if this is a repeat
        KYOLD = KEY                     ; save key (prevent a repeat)
;       CHAR = KEY                              ; diagnostic, write the character to LCD
;       gosub LCD_CHAR                          ; diagnostic, write the character to LCD
;       gosub DIAG1                             ; diagnostic, toggle pin 40, endless loop
        if KEY = "0" then GFQ           ; goto get frequency digit
        if KEY = "1" then GFQ           ; goto get frequency digit
        if KEY = "2" then GFQ           ; goto get frequency digit
        if KEY = "3" then GFQ           ; goto get frequency digit
        if KEY = "4" then GFQ           ; goto get frequency digit
        if KEY = "5" then GFQ           ; goto get frequency digit
        if KEY = "6" then GFQ           ; goto get frequency digit
        if KEY = "7" then GFQ           ; goto get frequency digit
        if KEY = "8" then GFQ           ; goto get frequency digit
        if KEY = "9" then GFQ           ; goto get frequency digit
        if KEY = "A" then AX            ; goto turn radio on routine (will be ATU tune)
        if KEY = "B" then BW            ; goto set BANDWIDTH routine
        if KEY = "C" then RXFQ          ; goto set RECEIVER frequency routine
        if KEY = "D" then TXFQ          ; goto set TRANSMITTER frequency routine
        if KEY = "#" then RX            ; goto RECEIVE routine
        if KEY = "*" then TX            ; goto TRANSMIT routine
;
        goto ACTX1                              ; exit
ACTX:
        KYOLD = 0                               ; clear for new button press
ACTX1:
        return                          ; finished
;
;*********************************************************************
```

(continued...)

```
;ok             set Receiver mode
;*********************************************************************
; entry, nothing required
; uses STAT,CTRL,RXFQB,FPTR
; calls SFRQ
;
RX:     TXM = 0                         ; set rx mode
        STAT = CTRL & %11100011         ; clear Tx, RF, Tdc bits to 0 (set rx mode)
        CTRL = STAT                     ; save control port status
        pins = CTRL                     ; load frequency port
        high portc 6                ; latch control data
        low portc  6                ; latch control data
;
        gosub SFRQ                      ; set the frequency
;
        pins = %00000000                ; set lines low
        return                      ; finished
;
;*********************************************************************
;ok             set Transmit mode
;*********************************************************************
; entry, nothing required
; uses STAT,CTRL,TXFQB,FPTR
; calls SFRQ,SFIL
;
TX:     TXM = 1                         ; set tx mode
;       STAT = CTRL & %00000100         ; mask for Tdc bit
;       if STAT > 0 then TXX        ; jump to exit if bit high
;
        STAT = CTRL & %11100011         ; clear Tdc, Tx, RF bits to 0
        CTRL = STAT | %00000100         ; set Tdc bit to 1
;       CTRL = STAT                     ; save control port status
        pins = CTRL                     ; load frequency port
        high portc 6                ; latch frequency data
        low portc 6                 ; latch frequency data
;
        gosub SFRQ                      ; set the frequency
;
        gosub SFIL                      ; set the transmitter LPF
;
;       STAT = CTRL | %00001000         ; set RF bit to 1
;       CTRL = STAT                     ; save control port status
;       pins = CTRL                     ; load frequency port
;       high 6                      ; latch frequency data
;       low  6                      ; latch frequency data
;
TXX:    pins = %00000000                ; set lines low
        return                      ; finished
;
;*********************************************************************
;ok             Set the transmitter LPF
;*********************************************************************
```

```
; entry, nothing required
; uses PTR,CHAR,STAT,FSET,TXFQB
; calls nothing
;
SFIL:   PTR = TXFQB                     ; point to the transmit frequency buffer
        peek PTR,CHAR           ; get tens mhz digit
        STAT = CHAR - 48                ; convert from ASCII to decimal
        FSET = STAT * 10                ; multiply by 10
        PTR = TXFQB + 1                 ; point to the transmit frequency buffer
        peek PTR,CHAR           ; get units mhz digit
        STAT = CHAR - 48                ; convert from ASCII to decimal
        PTR = FSET + STAT               ; freq in mhz (0-30)
;
        if PTR < 3  then SFIL3          ; set LPF filter to 3 mhz
        if PTR < 4  then SFIL4          ; set LPF filter to 4 mhz
        if PTR < 6  then SFIL6          ; set LPF filter to 6 mhz
        if PTR < 8  then SFIL8          ; set LPF filter to 8 mhz
        if PTR < 12 then SFIL12         ; set LPF filter to 12 mhz
        if PTR < 16 then SFIL16         ; set LPF filter to 16 mhz
        if PTR < 20 then SFIL20         ; set LPF filter to 20 mhz
          STAT = 0                      ; set LPF filter to 30 mhz
          goto SFILS
SFIL3:  STAT = 1
          goto SFILS
SFIL4:  STAT = 2
          goto SFILS
SFIL6:  STAT = 3
          goto SFILS
SFIL8:  STAT = 4
          goto SFILS
SFIL12: STAT = 5
          goto SFILS
SFIL16: STAT = 6
          goto SFILS
SFIL20: STAT = 7
;
SFILS: FSET = STAT & %00000111  ; set LPF
         pins = FSET            ; load LPF port
         high portc 5           ; latch LPF data
         low portc  5           ; latch LPF data
;
        pins = %00000000                ; clear lines
        return
;
;*****************************************************************
;ok              Diagnostic
;*****************************************************************
; it should never get here
; catch routine, if there is a code error
;
DIAG1: high 7  ; make PICAXE pin 40 high
               low 7            ; make PICAXE pin 40 low
```

```
                goto DIAG1      ; repeat
;
;*********************************************************************
;ok             Set the frequency
;*********************************************************************
; uses FPTR, FSET, FCNT, STAT, CHAR, TMP, FTOCW, FROCW, CNT
; gets each rx or tx frequency digit
; subtracts the rx or tx offset
; loads each digit into the FS5000
; calls nothing
;
SFRQ:   STAT = FSET | %00100000         ; set FMAN bit to 1 (initiates set FREQ)
        pins = STAT                     ; load FREQ SET port
        high portc 5                    ; latch FREQ port
        low portc 5                     ; latch FREQ port
        pause 10                        ; wait 10ms
;
        CNT = 0                                 ; set the borrow flag to zero
        for FCNT = 6 to 0       step -1 ; do the 7 frequency digits
;
        if TXM = 0 then SFR1            ; jump if rx mode
        FPTR = TXFQB + FCNT                     ; point to the tx frequency
        goto SFR2                               ; jump
SFR1:   FPTR = RXFQB + FCNT                     ; point to the rx frequency
;
;       STAT = FPTR + FCNT          ; point to the frequency buffer digit
SFR2:   peek FPTR,CHAR                  ; get freq digit
        STAT = CHAR - 48                ; convert from ASCII to decimal
        CHAR = STAT & %00001111         ; clear top bits (freq digit is in CHAR)
;
;---------------------------------
; this section subtracts the offset
;
        if TXM = 0 then SFR3            ; jump if rx mode
        FPTR = FTOCW + FCNT                     ; point to the tx offset frequency
        goto SFR4                               ; jump
SFR3:   FPTR = FROCW + FCNT                     ; point to the rx offset frequency
;
SFR4:   read FPTR,TMP                   ; get offset digit
        STAT = TMP - 48                         ; convert from ASCII to decimal
        TMP = STAT + CNT                        ; add borrow flag
        STAT = TMP & %00001111                  ; clear top bits
        TMP = STAT                              ; move it to TMP (offset digit is in TMP)
;
        if CHAR >= TMP then SFR5        ; jump if no borrow
        STAT = CHAR + 10                        ; add decimal 10
        CHAR = STAT                             ; move it to CHAR
        CNT = 1                                 ; set borrow flag to 1
        goto SFR6                               ; jump
;
SFR5: CNT = 0                           ; set borrow flag to 0
SFR6: STAT = CHAR - TMP                         ; subtract offset from this digit
```

```
        CHAR = STAT & %00001111                 ; clear top bits
;
;-------------------------
;
        STAT = CHAR | %00100000         ; set FMAN bit to 1
        CHAR = FSET & %11000000         ; clear bits 0 to 5
        FSET = CHAR | STAT              ; combine bits
        pins  = FSET                    ; load digit into FREQ SET port
        high portc 5                    ; load FREQ digit
        low  portc 5                    ; load FREQ digit
        pause 10                                ; wait 10ms
;
        STAT  = FSET | %00010000        ; set FC bit to 1 (set FREQ digit)
        pins  = STAT                    ; load FREQ SET port
        high portc 5                    ; latch FREQ digit
        low  portc 5                    ; latch FREQ digit
        pause 10                                ; wait 10ms
;
        pins  = FSET                    ; clear FC bit to 0, load FREQ SET port
        high portc 5                    ; latch FREQ port
        low  portc 5                    ; latch FREQ port
        pause 10                                ; wait 10ms
;
        next FCNT                               ; continue
;
        CHAR = FSET
        FSET = CHAR & %11000000         ; clear bits 0 to 5
        pins  = FSET                    ; load FREQ SET port(finished setting FREQ)
        high portc 5                    ; latch FREQ port
        low portc 5                             ; latch FREQ port
        pause 10                                ; wait 10ms
;
        pins = %00000000                        ; clear lines
        return                          ; finished
;
;*****************************************************************
;ok             Get the Receiver Frequency
;*****************************************************************
; uses TXM, FCNT
; calls LCD_RFQ
;
RXFQ: TXM = 0                                   ; set receiver mode
        FCNT = 0                                        ; reset digit count
;
        gosub LCD_RFQ                           ; clear the LCD rx display
        return
;
;*****************************************************************
;ok             Get the Transmitter Frequency
;*****************************************************************
; uses TXM, FCNT
; calls LCD_TFQ
```

```
;
TXFQ: TXM = 1                           ; set transmitter mode
      FCNT = 0                                  ; reset digit count
;
      gosub LCD_TFQ                             ; clear the LCD tx display
      return                            ; finished
;
;*********************************************************************
;ok              Get the Frequency
;*********************************************************************
; future.....ADD A LIMIT (500khz - 30mhz)
; get 7 digits from the keyboard to make the frequency
; uses FPTR, FCNT, FQB, KEY, CHAR, TXM, TXFQB, RXFQB
; enters with an ASCII character 0 to 7 in KEY
; loads 7 digits into the input buffer
; displays the difits
; moves the frequency into the rx or tx buffer
; calls LCD_FQ, LCD_CMD, LCD_CHAR
;
GFQ:  FPTR = FQB + FCNT                         ; point to the input frequency buffer
      if FCNT > 6 then GFQX             ; exit if more than 7 digits
      poke FPTR,KEY                     ; save digit
;
      CHAR = KEY                                ; move to CHAR
      gosub LCD_CHAR                            ; display the character
      if FCNT != 1 then GFQ1            ; test for decimal point position in display
      CHAR = "."                                ; load a decimal point
      gosub LCD_CHAR                            ; send a character
;
GFQ1: FCNT = FCNT + 1                           ; increment digit count
      if FCNT < 7 then GFQX             ; exit if less than 7 digits
;
      if TXM = 0 then GFQR              ; jump if rx mode
;
      for FCNT = 0 to 6                         ; transfer the frequency to the tx buffer
      FPTR = FQB + FCNT                         ; point to the frequency buffer
      peek FPTR,CHAR                            ; get digit
      FPTR = TXFQB + FCNT               ; point to the tx frequency buffer
      poke FPTR,CHAR                            ; save digit
      next FCNT                                 ; continue
      goto GFQX                                 ; exit
;
GFQR: for FCNT = 0 to 6                         ; transfer the frequency to the rx buffer
      FPTR = FQB + FCNT                         ; point to the frequency buffer
      peek FPTR,CHAR                            ; get digit
      FPTR = RXFQB + FCNT               ; point to the receive frequency buffer
      poke FPTR,CHAR                            ; save digit
      next FCNT                                 ; continue
;
GFQX: return                            ; finished
;
;*********************************************************************
```

(continued...)

```
;ok              Turn  radio ON and OFF
;*********************************************************************
; uses STAT, CTRL, TXM
; calls nothing
; toggles CONTROL bit 0 (DB25 pin 8) low = RX off, high = RX on
; toggles CONTROL bit 5 low = power off, high = power on
;
AX:     TXM = 0
;
        STAT = CTRL & %00000001         ; mask for RX on bit
        if STAT > 0 then AXOF           ; jump if bit high
;
        STAT= CTRL | %00100001          ; set bits 0 and 5
        CTRL = STAT                     ; save status
        pins = CTRL                     ; turn radio ON (bit 0) latch power ON (bit 5)
        high portc 6                    ; latch control data
        low portc 6                     ; latch control data
        pins = %00000000                ; set lines low
        return                          ; finished
;
AXOF:   STAT = CTRL & %11011110         ; clear bits 0 and 5
        CTRL = STAT                     ; save status
        pins = CTRL                     ; turn radio OFF (bit 0) turn power OFF (bit 5)
        high portc 6                    ; latch control data
        low portc 6                     ; latch control data
        end                             ; halt CPU
;
;*********************************************************************
;ok              Set the Bandwidth
;*********************************************************************
; Toggles CONTROL bit 1 (DB25 pin 1) low=300hz BW, high=3Khz BW
; uses STAT, CTRL, TEMP
; enters with nothing, leaves with nothing
; calls LCD_CMD, LCD_CHR
;
BW:     STAT = CTRL & %00000010         ; mask for BW bit
        if STAT > 0 then BWL            ; jump if bit high
;
        STAT = CTRL | %00000010         ; set BW bit to 1 (wide)
        CTRL = STAT                     ; save status
;
        CHAR = LCDL3                    ; set LCD to the end of line 2
        gosub LCD_CMD                   ; send command
        CHAR = "W"                      ; write W to LCD display
        gosub LCD_CHAR                  ; send a character
        goto BWX
;
BWL:    STAT = CTRL & %00111101         ; set BW to zero (narrow)
        CTRL = STAT                     ; save status
;
        CHAR = LCDL3                    ; set LCD to the end of line 2
        gosub LCD_CMD                   ; send command
```

```
        CHAR = "N"                      ; write N to LCD display
        gosub LCD_CHAR                  ; send a character
;
BWX:    pins = CTRL                     ; load control port
        high portc 6                ; latch control data
        low portc 6                     ; latch control data
        pins = %00000000                ; set lines low
        return                      ; finished
;
;*****************************************************************
;ok     check synth lock line
;*****************************************************************
; read SYNTH LOCK line and display "*" if high
; uses STAT, FCNT, CHAR
; calls LCD_CMD, LCD_CHAR
;
SLK:    if FCNT < 7 then SLKX       ; exit if currently inputing the frequency
;
        readadc 6,STAT                  ; read the SYNTH LOCK line
;
        if STAT > 120 then SLKH         ; jump if higher than 120
        if STAT <  70 then SLKL         ; jump if lower than 70
        goto SLKX                       ; exit if between 70 and 120
;
SLKH:   CHAR = LCDL4                ; set LCD to the "end but one" of line 2
        gosub LCD_CMD                   ; send command
        CHAR = "*"                      ; write * to LCD display(means transmitter on line)
        gosub LCD_CHAR                  ; send a character
        goto SLKX
;
SLKL:   CHAR = LCDL4                ; set LCD to the "end but one" of line 2
        gosub LCD_CMD                   ; send command
        CHAR = " "                      ; write a blank to LCD display (means not on line)
        gosub LCD_CHAR                  ; send a character
;
SLKX:   return                      ; finished
;
;*****************************************************************
;#################################################################
;ok             Check the keyboard
;#################################################################
;*****************************************************************
; uses ROW, COL, KEY, KPTR, KLR0
; calls nothing
; if entry with ROW = $FF, it exits with KEY = $FF
; if entry with ROW and COL values, it exits with KEY = an ASCII character
;
KYBD_CHK:                               ; CHECK FOR KEYS
        if ROW = $FF then KYBD_CHKX     ; jump if no key found
        KPTR = ROW*12 + KLR0 + COL      ; point to character (multiply first)
        read KPTR,KEY               ; get the character
        goto KYBD_CHKX1                 ; jump to exit
```

(continued...)

```
KYBD_CHKX:
        KEY = $FF                                   ; set no character found
KYBD_CHKX1:
        return                                  ; finished
;
;*********************************************************************
;ok              Scan the keyboard
;*********************************************************************
; uses ROW,COL,CNT
; calls nothing
; if no key found, it exits with ROW = $FF
; if key found, it exits with ROW = 0 to 4, and COL = 0 to 11
;
KYBD_SCAN:                                      ; SCAN THE KEYBOARD
        ROW = $FF                               ; set ROW to hexFF (it means no key)
        for CNT = 0 to 4                        ; scan 5 rows
        low portc CNT                   ; set row low
        if pin0 = 0 then KC0            ; Jump if key DOWN in column 0
        if pin1 = 0 then KC1            ; Jump if key DOWN in column 1
        if pin2 = 0 then KC2            ; Jump if key DOWN in column 2
        if pin3 = 0 then KC3            ; Jump if key DOWN in column 3
        if pin4 = 0 then KC4            ; Jump if key DOWN in column 4
        if pin5 = 0 then KC5            ; Jump if key DOWN in column 5
        if pin6 = 0 then KC6            ; Jump if key DOWN in column 6
        if pin7 = 0 then KC7            ; Jump if key DOWN in column 7
        if portA pin0 = 0 then KC8      ; Jump if key DOWN in column 8
        if portA pin1 = 0 then KC9      ; Jump if key DOWN in column 9
        if portA pin2 = 0 then KC10     ; Jump if key DOWN in column 10
        if portA pin3 = 0 then KC11     ; Jump if key DOWN in column 11
KYS8:   high portc CNT                  ; set row high
        next CNT                                ; do next row
        return                          ; finished
;
KC0:    COL = 0                                 ; save COL
        ROW = CNT                               ; save ROW
        goto KYS8                               ; do next key
;
KC1:    COL = 1                                 ; save COL
        ROW = CNT                               ; save ROW
        goto KYS8                               ; do next key
;
KC2:    COL = 2                                 ; save COL
        ROW = CNT                               ; save ROW
        goto KYS8                               ; do next key
;
KC3:    COL = 3                                 ; save COL
        ROW = CNT                               ; save ROW
        goto KYS8                               ; do next key
;
KC4:    COL = 4                                 ; save COL
        ROW = CNT                               ; save ROW
        goto KYS8                               ; do next key
```

(continued...)

```
;
KC5:    COL = 5                         ; save COL
        ROW = CNT                       ; save ROW
        goto KYS8                       ; do next key
;
KC6:    COL = 6                         ; save COL
        ROW = CNT                       ; save ROW
        goto KYS8                       ; do next key
;
KC7:    COL = 7                         ; save COL
        ROW = CNT                       ; save ROW
        goto KYS8                       ; do next key
;
KC8:    COL = 8                         ; save COL
        ROW = CNT                       ; save ROW
        goto KYS8                       ; do next key
;
KC9:    COL = 9                         ; save COL
        ROW = CNT                       ; save ROW
        goto KYS8                       ; do next key
;
KC10:   COL = 10                        ; save COL
        ROW = CNT                       ; save ROW
        goto KYS8                       ; do next key
;
KC11:   COL = 11                        ; save COL
        ROW = CNT                       ; save ROW
        goto KYS8                       ; do next key
;
;*****************************************************************
;ok              Initialise variables
;*****************************************************************
; uses no variables
; calls nothing
; port b output is always low, changes with valid data, then back to all low
; port c bits 0,1,2,3,4 always high, low pulses when scanning keyboard
; port c bits 5 and 6 always low, high pulse to latch data in FREQ and CONTROL latches
; port c bits 7 always high, low pulse to latch data in LCD
;
INIT:                                   ; INITIALISE
        TXM = 0                         ; set to receive mode
;
        pinsc = %10011111               ; set port c data bits all high
        dirsc = %11111111               ; set port c to 8 outputs
;
        pins  = FSETI               ; initialise FREQ port
        high portc 5                ; latch FREQ port
        low portc  5                ; latch FREQ port
;
        pins  = CTRLI               ; initialise CONTROL port
        high portc 6                ; latch CONTROL port
        low portc  6                ; latch CONTROL port
```

(continued...)

```
;
        for FCNT = 0 to 6                       ; load the default frequency to the tx buffer
        FPTR = TFQI + FCNT              ; point to the frequency buffer
        read FPTR,CHAR                          ; get digit
        FPTR = TXFQB + FCNT             ; point to the tx frequency buffer
        poke FPTR,CHAR                          ; save digit
        next FCNT                               ; continue
;
        for FCNT = 0 to 6                       ; load the default frequency to the rx buffer
        FPTR = RFQI + FCNT              ; point to the frequency buffer
        read FPTR,CHAR                          ; get digit
        FPTR = RXFQB + FCNT             ; point to the receive frequency buffer
        poke FPTR,CHAR                          ; save digit
        next FCNT                               ; continue
;
        FCNT = 0                                ; set digit count to zero
        pins = %00000000                        ; set lines low
        return                          ; finished
;
;*****************************************************************
;ok              Initialise the LCD display
;*****************************************************************
; uses no variables
; calls nothing
;
LCD_INIT:                                       ; INITIALISE THE LCD
        pause 100                               ; wait 100ms for LCD to reset
;
        pins = $03                              ; send $03 three times
        low portc 7                             ; latch the data
        high portc 7                    ; raise the enable line
        pause 10                                ; wait 10ms
        low portc 7                             ; latch the data
        high portc 7                    ; raise the enable line
        pause 1                         ; wait 1ms
        low portc 7                             ; latch the data
        high portc 7                    ; raise the enable line
        pause 1                         ; wait 1ms
;
        pins = $02                              ; set to 4 bit data mode
        low portc 7                             ; latch the data
        high portc 7                    ; raise the enable line
        pause 1                         ; wait 1ms
;
        pins = $02                              ; high nibble
        low portc 7                             ; latch the data
        high portc 7                    ; raise the enable line
        pause 1                         ; wait 1ms
        pins = $08                              ; set 2 character lines, 5x7 pixels
        low portc 7                             ; latch the data
        high portc 7                    ; raise the enable line
        pause 1                         ; wait 1ms
```

(continued...)

```
;
        pins = $00                      ; high nibble
        low portc 7                     ; latch the data
        high portc 7                ; raise the enable line
        pause 1                     ; wait 1ms
        pins = $0C                      ; display ON, cursor OFF, cursor STEADY
        low portc 7                     ; latch the data
        high portc 7                ; raise the enable line
        pause 1                     ; wait 1ms
;
        pins = $00                      ; high nibble
        low portc 7                     ; latch the data
        high portc 7                ; raise the enable line
        pause 1                     ; wait 1ms
        pins = $06                      ; cursor to INCREMENT, cursor to MOVE
        low portc 7                     ; latch the data
        high portc 7                ; raise the enable line
        pause 1                     ; wait 1ms
;
LCD_CLEAR:
        pins = $00                      ; high nibble
        low portc 7                     ; latch the data
        high portc 7                ; raise the enable line
        pause 1                     ; wait 1ms
        pins = $01                      ; send CLEAR command
        low portc 7                     ; latch the data
        high portc 7                ; raise the enable line
        pause 10                        ; wait 10ms
;
        pins = %00000000                ; set lines low
        return                      ; finished
;
;*****************************************************************
;ok              Write a character to the LCD display
;*****************************************************************
; enters with CHAR = character to be written to the LCD display
; uses CHAR, NIBH, NIBL, DAT
; calls nothing
;
LCD_CHAR:                               ; SEND A CHARACTER (CHAR)
        pins = CHAR & NIBH / 16 | DAT   ; output the high nibble
        low portc 7                     ; latch the data
        high portc 7                ; raise the enable line
        pause 10                        ; wait 8ms for LCD
;
        pins = CHAR & NIBL  | DAT   ; output the low nibble
        low portc 7                     ; latch the data
        high portc 7                ; raise the enable line
        pause 10                        ; wait 8ms for LCD
;
        pins = %00000000                ; set lines low
        return                      ; finished
```

(continued...)

```
;
;*****************************************************************
;ok              Write the Title
;*****************************************************************
; uses KPTR, LEN, CHAR
; calls LCD_CHAR
;
TITLE:                                  ; WRITE A STRING TO THE LCD DISPLAY
        for KPTR = 0 to LEN             ; loop for the string length
        read KPTR,CHAR                          ; get the character
        gosub LCD_CHAR                          ; write the character
        next KPTR                               ; inc the pointer and loop
        return                          ; finished
;
;*****************************************************************
;ok              Write a command to the LCD display
;*****************************************************************
; enters with CHAR = command to be written to the LCD display
; uses CHAR, NIBH, NIBL, CMD
; calls nothing
;
LCD_CMD:                                        ; SEND A CHARACTER (CHAR)
        pins = CHAR & NIBH / 16 | CMD           ; output the high nibble
        low portc 7                             ; latch the data
        high portc 7                    ; raise the enable line
;
        pins = CHAR & NIBL  | CMD       ; output the low nibble
        low portc 7                             ; latch the data
        high portc 7                    ; raise the enable line
        pause 8                         ; wait 8ms for LCD
;
        pins = %00000000                        ; set lines low
        return                          ; finished
;
;*****************************************************************
;ok              Clear the LCD RX frequency display
;*****************************************************************
; uses CHAR
; calls LCD_FQ, LCD_CMD, LCD_CHAR
;
LCD_RFQ:
        CHAR = LCDL1                            ; set LCD to line 1
        gosub LCD_FQ                            ; clear
        CHAR = LCDL1                            ; set LCD to line 1
        gosub LCD_CMD                                   ; send command
        CHAR = "R"                                      ; Write R to LCD display
        gosub LCD_CHAR                                  ; send a character
        CHAR = " "                                      ; load a blank
        gosub LCD_CHAR                                  ; send a character
        return                                  ; finished
;
;*****************************************************************
```

(continued...)

```
;ok              Clear the LCD TX frequency display
;*****************************************************************
; uses CHAR
; calls LCD_FQ, LCD_CMD, LCD_CHAR
;
LCD_TFQ:
        CHAR = LCDL2                    ; set LCD to line 2
        gosub LCD_FQ                    ; clear
        CHAR = LCDL2                    ; set LCD to line 2
        gosub LCD_CMD                           ; send command
        CHAR = "T"                              ; Write T to LCD display
        gosub LCD_CHAR                          ; send a character
        CHAR = " "                              ; load a blank
        gosub LCD_CHAR                          ; send a character
        return
;
;*****************************************************************
;ok              Clear the LCD frequency display
;*****************************************************************
; uses CHAR, CNT
; calls LCD_CMD, LCD_CHAR
; at entry, CHAR contains a pointer to line 1 or line 2
;
LCD_FQ:
        gosub LCD_CMD                           ; send command
        CHAR = " "                              ; load a blank
        for CNT = 0 to 3                        ; Clear the start of the LCD display
        gosub LCD_CHAR                          ; send a character
        next CNT                                ; continue
;
        CHAR = "."                              ; load a decimal point
        gosub LCD_CHAR                          ; send a character
;
        CHAR = " "                              ; load a blank
        for CNT = 4 to 8                        ; Clear the rest of the LCD display
        gosub LCD_CHAR                          ; send a character
        next CNT                                ; continue
;
        CHAR = "0"                              ; Write 0 to LCD display
        gosub LCD_CHAR                          ; send a character
        return
;
;*****************************************************************
;ok              DIAGNOSTIC: toggle pin 8 on the DB25, every second
;*****************************************************************
; uses no variables
; calls AX
;
;DIAG2:                                 ; TEST ROUTINE 2
;       gosub AX                        ; radio ON
;       pause 1000                      ; wait 1000 ms
;       gosub AX                        ; radio OFF
```

(continued...)

```
;       pause 1000                      ; wait 1000 ms
;       goto DIAG2
;       return                  ; finished
;
;*****************************************************************
;ok              DIAGNOSTIC: toggle PICAXE port C pin 5,6,7, every 100ms
;*****************************************************************
; uses no variables
; calls nothing
;
;DIAG3:                                 ; TEST ROUTINE 1
;       low portc 5                     ; make pin 5 low
;       high portc 5            ; make pin 5 high
;       pause 100                       ; wait 100 ms
;
;       low portc 6                     ; make pin 6 low
;       high portc 6            ; make pin 6 high
;       pause 100                       ; wait 100 ms
;       low portc 6                     ; make pin 6 low
;       high portc 6            ; make pin 6 high
;       pause 100                       ; wait 100 ms
;
;       low portc 7                     ; make pin 7 low
;       high portc 7            ; make pin 7 high
;       pause 100                       ; wait 100 ms
;       low portc 7                     ; make pin 7 low
;       high portc 7            ; make pin 7 high
;       pause 100                       ; wait 100 ms
;       low portc 7                     ; make pin 7 low
;       high portc 7            ; make pin 7 high
;       pause 100                       ; wait 100 ms
;
;       pause 100                       ; wait 100 ms
;       return                  ; finished
;
;*****************************************************************
```

(continued...)