```
 TITLE " K8 Keyer "

; LIST   P=12C509,R=D

 LIST P=16F84A,R=D

 ifdef __16F84A
  include "p16f84a.inc"
 else
  include "p12c509a.inc"
 endif

;
;*****************************************************************
;
; Copyright (C) 1998 Steven T. Elliott.  All rights reserved.
; Permission is granted to use, modify, or redistribute this software
; so long as it is not sold or exploited for profit.
;
; Modified for 16F84A 11/01/01 Ian M. Wilson K3IMW
; ** Please note: pin assignments are different from 12C509A **
;
; THIS SOFTWARE IS PROVIDED AS IS AND WITHOUT WARRANTY OF ANY KIND,
; EITHER EXPRESSED OR IMPLIED.
;
; I am trying to keep track of how many shareware K8s there are out
; there. Please send me an email when you get yours working.
; email: steve@k1el.com or k1el@aol.com
;
;*****************************************************************
; The following defines setup the basic configuration of the K8
; take a look at how they are used in the code to get an understanding
; of how they work. If you leave them as they are here you will get
; a working K8 without beacon mode, code protect off.
;
#define BEACON_TIME 4
#define DEBUG
;#define BEACON
;#define NO_DE
;#define SETFARNS
;*****************************************************************
; Assemble this code with the Microchip MPASM assembler. The target
; device is a PIC12C509 although it will work with the PIC12C509A as
; well. The configuration word is definde in the code but if you
; programmer is unable to understand it, program the part with Watchdog
; Disabled, Internal Reset and Internal RC Oscillator enabled.
;
;*****************************************************************
; Be sure to select decimal as the deafault radix in your assembler !
;*****************************************************************
;
; This code will produce a working part if you follow the above
; guidelines. Until you fully understand the code, only change
; the values in the K8 customization header. The 12C509 is nearly
; full, any modifications must be done with extreme care.
;
; Download the K8 Maunual from the K1EL website to learn how to
; build and operate a keyer using this chip.
;*****************************************************************
;
;    MORSE CODE KEYER CONTROLLER
;    PIC MODE = PIC12C509(A)/PIC16F84(A)   CLK=4.0MHZ
;
;       Program:          MORSE9.ASM
;       Initial Date:     8 MARCH, 1997
;
;  12.24.98  ste    public release
;  11.01.01  imw    16f84a version
;
;*****************************************************************
;
; Pinout and notes on F84 K8 implementation imw 12/1/01
;
; 16F84A, 4MHz P-DIP part
; ======================
; pin   name        connection
```

```
; 1 RA2     N/C
; 2 RA3     N/C
; 3 RA4     N/C
; 4 MCLR/     thru 10k to +5v
; 5 VSS     0v
; 6 RB0     KEY
; 7 RB1     N/C
; 8 RB2     ST
; 9 RB3     STK
; 10    RB4     DIT
; 11    RB5     DAH
; 12    RB6     N/C
; 13    RB7     PB
; 14    VDD     +5v !
; 15    O2      *
; 16    O1      *
; 17    RA0     N/C
; 18    RA1     N/C
;
; * O1, O2 connect to the outside pins of a 4MHz ceramic resonator.
; The center pin of the resonator goes to ground.
;
; ! A 0.01uF decoupling capacitor connects between pins 14 and 5.
;
; The unit is powered from 3, 1.5v cells in series. You may want to
; add an on/off switch (this is also the only way to reset the unit).
;
;-----------------------------------------
;       Morse Equates Table
;-----------------------------------------

M_END   EQU     0x00    ; Ending delimiter
M_SKIP  EQU     0x01    ; Skip this entry
M_SP    EQU     0x02    ; Word space
M_USER  EQU     0x03    ; Toggle to USER RAM
M_CALL  EQU     0x05    ; Toggle to USER Callsign
M_TO    EQU     0x80    ; Timeout return from GETCW
M_0     EQU     0xFC    ; 0 1111 1100 = DAH-DAH-DAH-DAH-DAH
M_1     EQU     0x7C    ; 1 0111 1100 = DI-DAH-DAH-DAH-DAH
M_2     EQU     0x3C    ; 2 0011 1100 = DI-DI-DAH-DAH-DAH
M_3     EQU     0x1C    ; 3 0001 1100 = DI-DI-DI-DAH-DAH
M_4     EQU     0x0C    ; 4 0000 1100 = DI-DI-DI-DI-DAH
M_5     EQU     0x04    ; 5 0000 0100 = DI-DI-DI-DI-DIT
M_6     EQU     0x84    ; 6 1000 0100 = DAH-DI-DI-DI-DIT
M_7     EQU     0xC4    ; 7 1100 0100 = DAH-DAH-DI-DI-DIT
M_8     EQU     0xE4    ; 8 1110 0100 = DAH-DAH-DAH-DI-DIT
M_9     EQU     0xF4    ; 9 1111 0100 = DAH-DAH-DAH-DAH-DIT
M_AR    EQU     0x54    ;   0101 0100 = DI-DAH-DI-DAH-DIT
M_SK    EQU     0x16    ;   0001 0110 = DI-DI-DI-DAH-DI-DAH
M_PER   EQU     0x56    ;   0101 0110 = DI-DAH-DI-DAH-DI-DAH
M_COM   EQU     0xCE    ;   1100 1110 = DAH-DAH-DI-DI-DAH-DAH
M_BT    EQU     0x8C    ;   1000 1100 = DAH-DI-DI-DI-DAH
M_QUE   EQU     0x32    ; ? 0011 0010 = DI-DI-DAH-DAH-DI-DIT
M_DN    EQU     0x94    ;   1001 0100 = DAH-DI-DI-DAH-DIT
M_A     EQU     0x60    ; A 0110 0000 = DI-DAH
M_B     EQU     0x88    ; B 1000 1000 = DAH-DI-DI-DIT
M_C     EQU     0xA8    ; C 1010 1000 = DAH-DI-DAH-DIT
M_D     EQU     0x90    ; D 1001 0000 = DAH-DI-DIT
M_E     EQU     0x40    ; E 0100 0000 = DIT
M_F     EQU     0x28    ; F 0010 1000 = DI-DI-DAH-DIT
M_G     EQU     0xD0    ; G 1101 0000 = DAH-DAH-DIT
M_H     EQU     0x08    ; H 0000 1000 = DI-DI-DI-DIT
M_I     EQU     0x20    ; I 0010 0000 = DI-DIT
M_J     EQU     0x78    ; J 0111 1000 = DI-DAH-DAH-DAH
M_K     EQU     0xB0    ; K 1011 0000 = DAH-DI-DAH
M_L     EQU     0x48    ; L 0100 1000 = DI-DAH-DI-DIT
M_M     EQU     0xE0    ; M 1110 0000 = DAH-DAH
M_N     EQU     0xA0    ; N 1010 0000 = DAH-DIT
M_O     EQU     0xF0    ; O 1111 0000 = DAH-DAH-DAH
M_P     EQU     0x68    ; P 0110 1000 = DI-DAH-DAH-DIT
M_Q     EQU     0xD8    ; Q 1101 1000 = DAH-DAH-DI-DAH
M_R     EQU     0x50    ; R 0101 0000 = DI-DAH-DIT
M_S     EQU     0x10    ; S 0001 0000 = DI-DI-DIT
M_T     EQU     0xC0    ; T 1100 0000 = DAH
M_U     EQU     0x30    ; U 0011 0000 = DI-DI-DAH
M_V     EQU     0x18    ; V 0001 1000 = DI-DI-DI-DAH
```

```
M_W     EQU     0x70    ; W 0111 0000 = DI-DAH-DAH
M_X     EQU     0x98    ; X 1001 1000 = DAH-DI-DI-DAH
M_Y     EQU     0xB8    ; Y 1101 1000 = DAH-DAH-DI-DAH
M_Z     EQU     0xC8    ; Z 1100 1000 = DAH-DAH-DI-DIT
;
;-----------------------------------------
; Code Speed Index Defines
; Each index points to a delay value in the
; code speed index table. Use the call IDX2SPD
; to convert index to speed delay.
;
WPM_5   EQU     00D
WPM_6   EQU     01D
WPM_7   EQU     02D
WPM_8   EQU     03D
WPM_9   EQU     04D
WPM_10  EQU     05D
WPM_11  EQU     06D
WPM_12  EQU     07D
WPM_13  EQU     08D
WPM_14  EQU     09D
WPM_15  EQU     10D
WPM_16  EQU     11D
WPM_17  EQU     12D
WPM_18  EQU     13D
WPM_19  EQU     14D
WPM_20  EQU     15D
WPM_21  EQU     16D
WPM_22  EQU     17D
WPM_23  EQU     18D
WPM_24  EQU     19D
WPM_25  EQU     20D
WPM_26  EQU     21D
WPM_27  EQU     22D
WPM_28  EQU     23D
WPM_29  EQU     24D
WPM_30  EQU     25D
WPM_31  EQU     26D
WPM_32  EQU     27D
WPM_33  EQU     28D
WPM_34  EQU     29D
WPM_35  EQU     30D
WPM_36  EQU     31D
WPM_37  EQU     32D
WPM_38  EQU     33D
WPM_39  EQU     34D
WPM_40  EQU     35D
WPM_41  EQU     36D
WPM_42  EQU     37D
WPM_43  EQU     38D
WPM_44  EQU     39D
WPM_45  EQU     40D
WPM_46  EQU     41D
WPM_47  EQU     42D
WPM_48  EQU     43D
WPM_49  EQU     44D

MAXSPDIDX EQU   WPM_49

;********** K8 Customization Header **************
; This is where you customize the K8 for your
; own setup. If you don't know what you are
; doing, only modify the CALL, SPEED_DEFAULT,
; message selection, and whether you want AR on.
; DON'T CHANGE MODE_DEFAULT, PERIOD, or
; SPD_CONST unless you are thoroughly familiar
; with the code.
;
; This version has been configured with beacon off
; and sidetone on. I'll give you a hint, MODE_DEFAULT
; governs these settings.

CALL_0  EQU  M_N    ;Put your call here, letter by letter
CALL_1  EQU  M_1    ; No more than 7 characters
CALL_2  EQU  M_N
CALL_3  EQU  M_P
```

```
CALL_4  EQU  M_END
CALL_5  EQU  M_END
CALL_6  EQU  M_END
CALL_7  EQU  M_END

SPEED_DEFAULT  EQU  WPM_15  ; can be WPM_5 to WPM_49

#define STANDARD    ; can be: STANDARD, QRP, CQTEST, or CONTEST
;#define AR_ON       ; comment this out if you don't want AR sent

MODE_DEFAULT  EQU  0B3H     ; IAM_A mode + ASP on
; MODE_DEFAULT  EQU  013H   ; original

PERIOD  EQU 120D
SPD_CONST  EQU 212D

 ifdef __16F84A
  ifdef BEACON
    ifdef DEBUG
    ;development mode, code protect off, watchdog on
    __CONFIG _CP_OFF & _WDT_ON & _XT_OSC
    else
    ;manufacture mode, code protect on, watchdog on
    __CONFIG _CP_ON & _WDT_ON & _XT_OSC
    endif
  else
    ifdef DEBUG
    ;manufacture mode, code protect off, watchdog off
    __CONFIG _CP_OFF & _WDT_OFF  & _XT_OSC
    else
    ;manufacture mode, code protect on, watchdog off
    __CONFIG _CP_ON & _WDT_OFF  & _XT_OSC
    endif
  endif
 else
  ifdef BEACON
    ifdef DEBUG
    __config 0000EH   ;development mode, code protect off, watchdog on
    else
    __config 00006H   ;manufacture mode, code protect on, watchdog on
    endif
  else
    ifdef DEBUG
    __config 0000AH   ;manufacture mode, code protect off, watchdog off
    else
    __config 00002H   ;manufacture mode, code protect on, watchdog off
    endif
  endif
 endif

;***************** End of Customization Header ******************

;-----------------------------------------
;       PIC Specific Equates
;-----------------------------------------

; standard register defines are taken from the include file
 ifdef __16F84A
PC9     EQU     Z           ; makes BSF STATUS, PC9 (etc) no-ops
PC  EQU PCL     ; for 8-bit PC adds, etc
 else
PC9     EQU     5       ; PC bit 9 in Status register
REGBANK EQU     5       ; Upper/Lower register bank select
GPWUF   EQU     7       ; GP input change wake up flag
 endif

;-----------------------------------------
;       File Register Assignments
;-----------------------------------------

 ifdef __16F84A
GPIO    EQU PORTB   ; I/O Port - see below for changed pin assignments

    CBLOCK 0x0C ; Registers

    PROCLAT         ; Process Latch
```

```
    DELAYHI         ; High delay counter register
    DELAYLO         ; Low delay counter register
    TIMEBASE        ; Morse Time Base
    CURMSG          ; Currently Selected Message
    MODEREG         ; Mode Bits
    WTEMP       ; for loading PCLATH
    SPEEDIDX        ; Current speed table index
    FARNS           ; Letterspacing
    BX              ;
    AL              ; General Purpose Registers
    AH              ;
    CL              ;
    CH              ;
    DL              ;
    DH              ;
    USRCALL: 8    ; User Callsign
    USERRAM: 16   ; User Message

    ENDC

 else
PC  EQU 02H ; Program counter
TIMER0  EQU 01H     ; Counter for delay

    CBLOCK 0x07 ; Registers

    PROCLAT         ; Process Latch
    DELAYHI         ; High delay counter register
    DELAYLO         ; Low delay counter register
    TIMEBASE        ; Morse Time Base
    CURMSG          ; Currently Selected Message
    MODEREG         ; Mode Bits
    NSTACK          ; Pseudo Stack
    SPEEDIDX        ; Current speed table index
    FARNS           ; Letterspacing
    BX              ;
    AL              ; General Purpose Registers
    AH              ;
    CL              ;
    CH              ;
    DL              ;
    DH              ;

    ENDC

USRCALL    EQU  17H     ; User Callsign
USERRAM    EQU  30H     ; User Message

 endif

;-----------------------------------------
;       MPU Bit Assignments
;    ** 16F84A uses port B **
;-----------------------------------------

; These bits also used in PROCLAT, differ from port assignments
DITPDL      EQU  00H     ; IN  DIT Paddle
DAHPDL      EQU  01H     ; IN  DAH Paddle
PB          EQU  03H     ; IN  Message/Config Push Button

 ifdef __16F84A
XDITPDL     EQU  04H     ; RB4, IN  DIT Paddle
XDAHPDL     EQU  05H     ; RB5, IN  DAH Paddle
KEY         EQU  00H     ; RB0, OUT Keyed Output
XPB         EQU  07H     ; RB7, IN  Message/Config Push Button
TONE        EQU  02H     ; RB2, OUT Sidetone Output
 ifdef BEACON
BCON        EQU  03H     ; RB3, OUT Beacon Request Output
 else
STK         EQU  03H     ; RB3, OUT Keyed Sidetone Output
 endif
 else
XDITPDL     EQU  00H     ; GP0, IN  DIT Paddle
XDAHPDL     EQU  01H     ; GP1, IN  DAH Paddle
KEY         EQU  02H     ; GP2, OUT Keyed Output
XPB         EQU  03H     ; GP3, IN  Message/Config Push Button
```

```
TONE        EQU  04H     ; GP4, OUT Sidetone Output
 ifdef BEACON
BCON        EQU  05H     ; GP5, OUT Beacon Request Output
 else
STK         EQU  05H     ; GP5, OUT Keyed Sidetone Output
 endif
 endif

;
;-----------------------------------------
;       Mode Bit Assignments
;
; Note: The STEN bit *must* be enabled in
; MODE_DEFAULT so the signon character will
; be audible, it gets setup properly at the
; beginning of SERVICE.
;-----------------------------------------

STEN        EQU  00H     ; Sidetone enable
STRQ        EQU  01H     ; Sidetone request
 ifndef __16F84A
FSRMSB      EQU  02H     ; FSR MS Address bit
 endif
SWAP        EQU  03H     ; Swap paddles enable bit
I_MODE      EQU  04H     ; Full iambic mode enable (no auto DE message)
IAM_A       EQU  05H     ; Iambic Mode A flag (A=1, B=0)
SKM         EQU  06H     ; Straight Key Mode enable bit
ASP         EQU  07H     ; Autospace enable bit

STRQBIT     EQU  02H     ; Use this to XORF STRQ
SWAPBIT     EQU  08H     ; Use this to XORF SWAP
IAMBIT      EQU  10H     ; Use this to XORF I_MODE
IABBIT      EQU  20H     ; Use this to XORF IAM_A
SKMBIT      EQU  40H     ; Use this to XORF SKM
ASPBIT      EQU  80H     ; Use this to XORF ASP

;-----------------------------------------
;       PROCLAT  Equates
;-----------------------------------------

PDLMASK     EQU  03H     ; Mask: Paddle input bits
PROCMASK    EQU  0FCH    ; Mask: Process bits
INLAST      EQU  02H     ; Last input flag, 1 means Dit was just entered
USERON      EQU  03H     ; Use USERRAM as message source when set
CONVERSE    EQU  04H     ; Conversational mode flag (for DE message)
OUTLAST     EQU  05H     ; Last output flasg, 1 means Dit was sent last
BOTH_ON     EQU  06H     ; Both paddles were pressed, used in iambic mode B
TXSQ        EQU  07H     ; Transmit Key Squelch bit in PROCLAT

;-----------------------------------------
;       Miscellaneous Equates
;-----------------------------------------

SELDELAY_H EQU  120D     ; Delay constants for
SELDELAY_L EQU  240D     ;      SELDELAY routine
GC_TO_L     EQU  220D    ; GETCW timeout inside loop
GC_TO_M     EQU  170D    ; GETCW timeout middle loop
GC_TO_H     EQU   3D     ; GETCW timeout outside loop
SPEEDCNT    EQU   4D     ; Number of characters sent in SPEED loop

;*****************************************
;       MACROS
; 16F84A has address range and stack depth
; to implement calls directly w/o trickery
;*****************************************

 ifdef __16F84A
NCALL   MACRO   LABEL
    CALL    LABEL
    ENDM

NRETURN MACRO
    RETURN
    ENDM
 else
NCALL   MACRO   LABEL
```

```
    MOVF    PC, W   ; Get Current PC
    MOVWF   NSTACK  ; Save in fake stack
    GOTO    LABEL
    ENDM

NRETURN MACRO
    MOVLW   2         ; PC offset to return
    ADDWF   NSTACK, W ; Calculate return
    MOVWF   PC        ; Go to it
    ENDM
 endif

;*****************************************
;       RESET ENTRY
;*****************************************

    ORG     0

 ifndef __16F84A
    MOVWF   OSCCAL  ; Load Internal Oscillator  TRIM Value
 endif
    GOTO    INIT

 ifdef __16F84A
    ORG 4
INTVEC  GOTO    INTVEC
 endif

;
;=========================================
;       Subroutines
;
; Must reside in first
; 256 bytes of codespace
;=========================================

;=========================================
;       Sample and Latch Input State
;=========================================

SAMPLE
    NCALL   NSAMPLE
    RETLW   0

NSAMPLE
 ifdef BEACON
    CLRWDT          ; Reset watchdog timer
 endif
    BTFSC   GPIO, XDITPDL
    GOTO    RIGHT_TEST
    BTFSC   MODEREG, SWAP
    GOTO    NS_DAH1
LEFT_TEST
    BSF     PROCLAT, DITPDL
    BSF     PROCLAT, INLAST   ; Set on DIT
    GOTO    RIGHT_TEST
NS_DAH1
    BSF     PROCLAT, DAHPDL
    BCF     PROCLAT, INLAST   ; Clear on DAH
RIGHT_TEST
    BTFSC   GPIO, XDAHPDL
    GOTO    NS_DONE
    BTFSS   MODEREG, SWAP
    GOTO    NS_DAH2
    BSF     PROCLAT, DITPDL
    BSF     PROCLAT, INLAST   ; Clear on DAH
    GOTO    NS_DONE
NS_DAH2
    BSF     PROCLAT, DAHPDL
    BCF     PROCLAT, INLAST   ; Set on DAH
NS_DONE
    NRETURN

;=========================================
;       Send DIT
;=========================================
```

```
DITOUT
 ifndef BEACON
    BTFSC   MODEREG, STEN   ; Sidetone enabled ?
    BSF     GPIO, STK   ; Key Sidetone
 endif
    BTFSS   PROCLAT, TXSQ   ; Transmit Key Squelch ?
    BSF     GPIO, KEY

    NCALL   NDITDELAY
 ifndef BEACON
    BCF     GPIO, STK        ; Unkey Sidetone
 endif
    BCF     GPIO, KEY

    NCALL   NDITDLY_NT
    BCF     PROCLAT, DITPDL  ; Clear DIT bit in paddle register
    BSF     PROCLAT, OUTLAST ; Mark that Dit was sent
    RETLW   0

;=========================================
;       Send DAH
;=========================================

DAHOUT
 ifndef BEACON
    BTFSC   MODEREG, STEN    ; Sidetone enabled ?
    BSF     GPIO, STK        ; Key Sidetone
 endif
    BTFSS   PROCLAT, TXSQ    ; Transmit Key Squelch ?
    BSF     GPIO, KEY

    MOVLW   3
    MOVWF   BX
NDLP
    NCALL   NDITDELAY
    DECFSZ  BX, F
    GOTO    NDLP

 ifndef BEACON
    BCF     GPIO, STK        ; Unkey Sidetone
 endif
    BCF     GPIO, KEY

    NCALL   NDITDLY_NT

; if (I_MODE is off &&
;   we are in conversational mode &&
;     both paddles are pressed)
;       then GOTO TRAILER

    BCF     PROCLAT, DAHPDL    ; Clear DAH bit in paddle register
    BCF     PROCLAT, OUTLAST   ; Mark that Dah was sent

    BTFSS   MODEREG, I_MODE    ; Is DE mode allowed ?
    BTFSC   PROCLAT, CONVERSE  ; Not in converse or iambic modes
    RETLW   0                  ; No, return
    BTFSS   GPIO, XDAHPDL
    BTFSC   GPIO, XDITPDL       ; if both paddles pressed goto TRAILER
    RETLW   0                  ; else return

;=========================================
;       Send Trailer Message
;=========================================

TRAILER
;       BCF     PROCLAT, CONVERSE  ; End conversational mode
 ifdef NO_DE
    CALL    DITOUT             ; Send rest of 'K'
    CALL    DAHOUT
 else
    CALL    DITOUT             ; Send rest of 'D'
    CALL    DITOUT
 endif
    CALL    LETTERSPACE
TRAIL0
```

```
    BTFSC   GPIO, XDITPDL    ; Wait till both paddles
    BTFSS   GPIO, XDAHPDL    ; are released
    GOTO    TRAIL0
    MOVLW   MSG7-MSGBASE    ; Point to 'DE'+1 message
    GOTO    L_SENDMSG       ; We will return to caller from there

;=========================================
; Wait sidetone delay with transmitter
; keyed. Used by TUNE & KEYDIRECT
; TUNE and KEYDIRECT call NDITDELAY thru
; STCLK, which sets an outside loop count
; of 1 and a reduced sidetone "low" time.
; This gives us a better match between
; KEYER and TUNE/KEYDIRECT sidetone.
;=========================================

STCLK
    MOVLW   10D             ; Ten passes
    GOTO    DITDEL

;=========================================
;       Dit Delay
;
; Set delay period to be 1.15mSec to give a
; sidetone frequency of 800Hz.
;=========================================

NDITDELAY
    MOVF    TIMEBASE, W     ; Get interval
DITDEL
    MOVWF   DELAYHI
    MOVLW   PERIOD-4    ; Compensate for NSAMPLE
    GOTO    DITDEL15
DITDEL0
    MOVLW   PERIOD          ; 1/2 Period High, 575 us
DITDEL15
    MOVWF   DELAYLO
DITDEL1
    GOTO    DTL0
DTL0
    DECFSZ  DELAYLO, F      ; Inner Loop 1 Test
    GOTO    DITDEL1

    BTFSC   MODEREG, STEN   ; Sidetone enabled ?
    BSF     GPIO, TONE      ; Key Sidetone
    MOVLW   PERIOD          ; 1/2 Period Low, 575 us
    MOVWF   DELAYLO
DITDEL2
    GOTO    DTL1
DTL1
    DECFSZ  DELAYLO,F       ; Inner Loop 2 Test
    GOTO    DITDEL2
    BCF     GPIO, TONE      ; Unkey Tone

    DECFSZ  DELAYHI,F       ; Outer Loop Test
    GOTO    DITDEL0
    GOTO    NSAMPLE         ; Latch early input
                ; NRETURN from there
;
;=========================================
;       Dit Delay without Sidetone
;=========================================

NDITDLY_NT
    MOVF    TIMEBASE, W     ; Get interval
    MOVWF   DELAYHI
DD_NT0
    MOVLW   PERIOD          ; 1/2 Period
    MOVWF   DELAYLO
DD_NT1
    GOTO    DNT0
DNT0
    GOTO    DNT1
DNT1
    GOTO    DNT2
DNT2
```

```
    NOP

    DECFSZ  DELAYLO, F      ; Inner Loop  Test
    GOTO    DD_NT1

    DECFSZ  DELAYHI,F       ; Outer Loop Test
    GOTO    DD_NT0
    GOTO    NSAMPLE         ; Latch early input
                ; NRETURN from there
;
;=========================================
;       Autospace Handler
;
; The idea is to keep the morse "pipe" full,
; that means that there should always be
; something in PROCLAT to send after the
; current dit or dah is sent. If not it is
; interpreted as an intercharacter space
; and a pause is inserted to fill out the
; remainder of a letterspace period.
; Any paddle events will be recorded in
; PROCLAT and issued in the order received
; thanks to the INLAST bit.
;=========================================

AUTOSP
    CALL    SAMPLE          ; Refresh PROCLAT
    BTFSS   MODEREG, ASP    ; Leave if autospace disabled
    GOTO    SERVLOOP
    BTFSS   PROCLAT, DITPDL ; Check if any in latch
    BTFSC   PROCLAT, DAHPDL
    GOTO    SERVLOOP        ; Yes: go send
    CALL    LETTERSPACE     ; No:  wait full letterspace
    GOTO    SERVLOOP        ; before allowing more sending
;
;=========================================
;       Word and Letter Spacing
;
; Remember that all characters have one
; dit delay tacked on automatically so
; only two are needed to provide a three
; bit letterspace. Six are needed for a
; seven bit word space. Extra letterspace
; is added via the value FARNS.
;=========================================

WORDSPACE
    MOVLW   6
    GOTO    WLSPAC0
LETTERSPACE
    MOVLW   2
WLSPAC0
    BTFSS   PROCLAT, CONVERSE ; no extra space in converse mode
    ADDWF   FARNS, W
    MOVWF   BX
WLSPAC1
    NCALL   NDITDLY_NT
    DECFSZ  BX, F
    GOTO    WLSPAC1
    RETLW   0

;=========================================
;       Output Single Morse Character
; Uses AL
;=========================================

OSCHAR
    MOVWF   AL              ; Copy character
OSLOOP
    MOVF    AL, W           ; Get coded morse
    ADDWF   AL, F           ; AL*2, if (Z==0) DONE else C=DIT/DAH
    BTFSS   STATUS, Z       ; Skip if zero
    GOTO    OSCONT
    CALL    LETTERSPACE     ; Inter-letter space
    RETLW   0               ; All done, return
OSCONT
```

```
    BTFSC   STATUS, C       ; (C==1) then DAH else DIT
    GOTO    OSDAH
    CALL    DITOUT
    GOTO    OSLOOP
OSDAH
    CALL    DAHOUT
    GOTO    OSLOOP

;=========================================
; Decimal to Morse Conversion Table
;=========================================

DEC2CW
 ifdef __16F84A
    MOVWF   WTEMP
    MOVLW   HIGH($)
    MOVWF   PCLATH
    MOVF    WTEMP,W
 endif
    ADDWF   PC, F   ; Jump through table
DECTBL
    RETLW   M_0
    RETLW   M_1
    RETLW   M_2
    RETLW   M_3
    RETLW   M_4
    RETLW   M_5
    RETLW   M_6
    RETLW   M_7
    RETLW   M_8
    RETLW   M_9

;=========================================
;       Cross Page Jumps
;
; Note 1: Routines in high page assume they
; called from low page and clear PC9 before
; they return. It is not possible to call
; a high page routine from the high page.
;
; Note 2: Subroutines above the 256 byte
; boundary are called through this interface.
; A call is converted to a jump, the called
; subroutine in high page ends with a return
; which pops the full 10 bit return address
; of the low page caller.
;=========================================

L_SENDMSG
    BSF     STATUS, PC9     ; Set PAGE bit for long GOTO
    GOTO    SENDMSG

L_CW2IDX
    BSF     STATUS, PC9     ; Set PAGE bit for long GOTO
    GOTO    CW2IDX

L_IDX2SPD
    BSF     STATUS, PC9     ; Set PAGE bit for long GOTO
    GOTO    IDX2SPD

L_GETCW
    BSF     STATUS, PC9     ; Set PAGE bit for long GOTO
    GOTO    GETCW

L_CONV_HI
    BSF     STATUS, PC9     ; Set PAGE bit for long GOTO
    GOTO    CONV_HI

L_CONV_LO
    BSF     STATUS, PC9     ; Set PAGE bit for long GOTO
    GOTO    CONV_LO

;=========================================
;       Long delay for user prompts
; Uses: CL
; Returns:
```

```
;       CL MSB=1 if paddle pressed, 0 if not
;=========================================

SELDELAY
    MOVLW   SELDELAY_H      ; Set high select delay
    MOVWF   DELAYHI
SELD0
    MOVLW   SELDELAY_L      ; Set low select delay
    MOVWF   DELAYLO
SELD1
    CALL    SAMPLE          ; Latch paddle press
    GOTO    SLD0
SLD0
    DECFSZ  DELAYLO, F      ; Inner Loop 1 Test
    GOTO    SELD1

    DECFSZ  DELAYHI, F      ; Inner Loop 1 Test
    GOTO    SELD0

    BTFSS   PROCLAT, DAHPDL ; DAH closed ?
    BTFSC   PROCLAT, DITPDL ; DIT closed ?
    GOTO    SDYES
SDNO
    BCF     CL, 0           ; Timed out w/no paddle pressed
    GOTO    SDRET
SDYES
    BSF     CL, 0           ; A paddle was pressed
SDRET
    RETLW   0

;=========================================
;       Push Button Switch Handler
; Converse mode is defined as the phase
; where the user is actually entering some
; morse letters. We want letterspace off
; to improve response, and we want DE mode
; off to prevent undesired output.
;=========================================

CONFIG
; ------- Disable Letterspace Here -----------
    BSF     PROCLAT, CONVERSE ; Start converse mode
    BSF     PROCLAT, TXSQ   ; Disable Key
    BSF     MODEREG, STEN   ; Force sidetone
    MOVLW   M_R             ; Signal user to enter code
    CALL    OSCHAR
CONFWT
    CLRWDT
    BTFSS   GPIO, XPB        ; Still closed ?
    GOTO    CONFWT          ;  wait till released
    CALL    L_GETCW         ; Get user response
    CALL    L_CW2IDX        ; Translate CW to table index
    BCF     STATUS, PC9     ; Clear page bit
 ifdef __16F84A
    MOVWF   WTEMP
    MOVLW   HIGH($)
    MOVWF   PCLATH
    MOVF    WTEMP,W
 endif
    ADDWF   PC, F           ; Jump through table
    GOTO    SIDETONE        ; A=Sidetone enable
    GOTO    LOADCALL        ; C=Load new callsign
    GOTO    FARNSWORTH      ; F=Farnsworth Adjust
    GOTO    IAMBIC          ; I=Iambic mode toggle
    GOTO    KEYMODE         ; K=Keyer mode
    GOTO    LOADUSER        ; L=Load user RAM
    GOTO    SPEED           ; S=Speed
    GOTO    TUNE            ; T=Tune
    GOTO    ASP_TOGGLE      ; U=Autospace Toggle
    GOTO    SWAP_TOGGLE     ; X=Swap Paddles Toggle
    GOTO    DE_TOGGLE       ; Z=DE Mode Toggle
; ------- Allow Letterspace for below ---------
    GOTO    DUMPUSER        ; D=Play user RAM
    GOTO    MSGSELECT       ; M=Message select
    GOTO    PRACTICE        ; P=Practice mode
    GOTO    OUTWPM          ; W=Report Morse speed
```

```
    GOTO    QUERET          ; Unknown entry !?!?
                ; Use common return

;****************************************
;   Wake Up Handler
; BX is used as a beacon timer, it is
; initialized at the end of QVRET.
; The OPTION register is setup so that WDT
; will cause a reset every 2 seconds. From
; INIT we will get vectored here. Where
; BX will be checked to see if it is zero.
; If so pin 2 will be configured as an output
; and asserted low true. If BX is not zero
; it is is decremented. Note that a pin
; change will be handled in the same manner
; as a WDT timeout but this is moot since
; the beacon enable switch will be off in
; non-beacon mode. BX will normally rest at
; a value of zero courtesy of the DAHOUT and
; LETTERSPACE routines.
;****************************************

 ifdef BEACON
WAKEUP
    MOVF    BX, W        ; Test beacon timer
    BTFSC   STATUS, Z    ; If zero: Beacon ON
    GOTO    BEAC_ON
    DECF    BX, F        ; Else: Decrement beacon timer
    GOTO    SERVICE      ; Continue in case it was
                ;   a pin change wakeup
BEAC_ON
        BCF GPIO, BCON   ; Insure that GP5 reg is zero

 ifdef __16F84A
    MOVLW   B'11110000'  ; Make RB3 an output
    BSF STATUS, RP0
    MOVWF   TRISB ^ 0x80 ; to assert beacon request
    BCF STATUS, RP0
 else
    MOVLW   B'00001011'  ; Make Pin 2 (GP5) an output
    TRIS    GPIO         ; to assert beacon request
                ; Note: If switch happens to be open
                ; then we will sleep but beacon rqst
                ; stays asserted so that when switch
                ; is closed we will start beaconing.
 endif
 endif

;****************************************
;*      MAIN GPIO Service Routine          *
;****************************************

SERVICE
    BCF     MODEREG, STEN   ; Restore user's
    BTFSC   MODEREG, STRQ   ;   sidetone
    BSF     MODEREG, STEN   ;     preference.
    CLRF    PROCLAT         ; Clear TXSQ and any latched inputs
SERVLOOP
    CALL    SAMPLE
    BTFSS   MODEREG, SKM    ; Straight key mode ?
    GOTO    KEYER           ;  No: Go keyer
    BTFSC   PROCLAT, DAHPDL ; DAH paddle pressed ?
    GOTO    KEYDIRECT       ;  Yes: Go straight key
    GOTO    PBTEST          ;  No:  Test pushbutton
;
; Iambic operation: If iambic mode is enabled the following state
; machine outputs alternating dits and dahs when both paddles are
; pressed at the same time. DIT_PADDLE = GPIO, DIT_PADDLE_L = PROC_LAT
;
;           if ((DIT PADDLE && DAH PADDLE) && (I_MODE == TRUE)
;           {
;               set BOTH_ON         "ref. Mode B"
;               if (DIT LAST) {
;                   send DAH
;                   clear DIT LAST
;               }
```

```
;               else {
;                   (this also handles odd case where both paddles
;                    hit simultaneously)
;                   send DIT
;                   set DIT LAST
;               }
;           }
;           else if (BOTH_ON == TRUE) {  "ref: Mode B, BOTH_ON is
;               clear BOTH_ON                   never set in MODE A"
;               if (DIT LAST)
;                   send DAH
;               else
;                   send DIT
;           }
;           else {
;               if (L_DIT_PADDLE && L_DAH_PADDLE) {
;                   if (INLAST) {
;                       send DAH
;                       clear DIT LAST
;                   }
;                   else {
;                       send DIT
;                       set DIT LAST
;                   }
;               }
;               if (DIT PADDLE) {
;                   send DIT
;                   set DIT LAST
;               }
;               else if (DAH PADDLE) {
;                   send DAH
;                   clear DIT LAST
;               }
;               else {
;                  (if neither paddle is set, do nothing)
;               }
;           }
;       }
KEYER
    BTFSS   MODEREG, I_MODE  ; DE mode ?
    GOTO    CHK_SINGLE       ; Yes: Skip
;
; Iambic mode A and B operation depend on PROCLAT, BOTH_ON and current
; state of the paddles. In mode B PROCLAT is used to check if the
; both paddles were pressed at the end of toggle mode to see if an
; extra dit or dah are sent.
; In mode A the PROCLAT is cleared when leaving toggle mode to
; prevent any additional dits or dahs from being sent.
;
; We check GPIO here since we need to monitor real time paddle
; status, the case where both are set in PROCLAT must be
; handled specially since order must be observed.

    BTFSS   GPIO, XDITPDL      ; DIT paddle *and*
    BTFSC   GPIO, XDAHPDL      ;     DAH paddle pressed ?
    GOTO    CHK_BOTH_ON       ;  No: Go check if both is pending

; Both Paddles are pressed

    BSF     PROCLAT, BOTH_ON  ; Set both paddles on flag
TOGGLE
    BTFSC   PROCLAT, OUTLAST  ; If LAT==1 (dit was last) then DAH
    GOTO    LPDAH             ;
    GOTO    LPDIT             ; else, always DIT. This addresses
                  ; the case where both paddles hit at
                  ; the same time.

; Both paddles aren't pressed, check if they just were
; and if in MODE B, issue alternate. Otherwise just
; check for single dit or dah request.

CHK_BOTH_ON
    BTFSS   PROCLAT, BOTH_ON
    GOTO    CHK_SINGLE
    BCF     PROCLAT, BOTH_ON
    BTFSS   MODEREG, IAM_A
```

```
    GOTO    TOGGLE            ; Mode B, send one more
    CLRF    PROCLAT           ; Mode A, kill any in latch
    GOTO    AUTOSP

; First check to see if both bits are set, if they are we
; need to figure out which one to send first. INLAST tells
; the last paddle input received, so the opposite is sent.

CHK_SINGLE
    BTFSC   PROCLAT, DITPDL
    BTFSS   PROCLAT, DAHPDL
    GOTO    CHK_S1
    BTFSC   PROCLAT, INLAST   ; INLAST = 1 if DIT was last
    GOTO    LPDAH
                ; Fall thru, and a DIT gets sent
CHK_S1
    BTFSC   PROCLAT, DITPDL   ; DIT Paddle closed when = 1
    GOTO    LPDIT
    BTFSS   PROCLAT, DAHPDL   ; DAH Paddle closed when = 1
    GOTO    PBTEST
LPDAH
    CALL    DAHOUT
    GOTO    AUTOSP
LPDIT
    CALL    DITOUT
    GOTO    AUTOSP
PBTEST
    BTFSS   GPIO, XPB           ; Push Button closed when = 0
    GOTO    PBHANDLE        ; changed logic of skip (imw)
;
;=========================================
;       Go into sleep mode until one
;       of the switches are hit or
;       Watchdog timer times out.
;=========================================
;

 ifdef __16F84A
    SLEEP           ; RB change will wake us up. Since GIE not enabled, arrive
    NOP         ; here.

    BCF INTCON, 0   ; clear RBIF
    GOTO    SERVICE     ; resume
 else
    SLEEP           ;
 endif


PBHANDLE            ; jump logic changed for jump around sleep, etc
 ifdef BEACON
 ifdef __16F84A
    MOVLW   B'11111000'      ; Restore Pin 2 as an input
    BSF STATUS, RP0          ; to cancel beacon request
    MOVWF   TRISB ^ 0x80
    BCF STATUS, RP0
 else
    MOVLW   B'00101011'      ; Restore Pin 2 as an input
    TRIS    GPIO             ; to cancel beacon request
 endif
 endif
    CALL    SELDELAY
    BTFSC   GPIO, XPB         ; Is Push Button closed ? (==0)
    GOTO    PBMSG        ;  No: output message
    CALL    SELDELAY     ;  Yes: wait again
    BTFSS   GPIO, XPB         ; Is Push Button closed ? (==0)
    GOTO    CONFIG       ;  Yes: do config routine
PBMSG                ;  No: output message
    MOVF    CURMSG, W        ; Point to 'current' message
    GOTO    QVRET

;=========================================
;       Key transmitter for tuning
;=========================================

TUNE
    BTFSS   MODEREG, STRQ   ; Restore sidetone
```

```
    BCF     MODEREG, STEN   ;   preference.
 ifndef BEACON
    BTFSC   MODEREG, STEN   ; Sidetone enabled ?
    BSF     GPIO, STK       ; Key Sidetone
 endif
    BSF     GPIO, KEY       ; Key transmitter
TUNELP
    CALL    STCLK           ; Key XMTR and delay w/sidetone
    BTFSC   PROCLAT, DAHPDL ; DAH paddle ends tune
    GOTO    WAIT4OFF
    BTFSS   PROCLAT, DITPDL ; DIT paddle ends tune
    GOTO    TUNELP
                ; Fall through

;=========================================
;       Wait for paddle release
;=========================================

WAIT4OFF
    MOVLW   25D             ; debounce count
    MOVWF   AL
WAIT4LP
 ifdef BEACON
    CLRWDT          ; Reset watchdog timer
 endif
    BTFSC   GPIO, XDITPDL    ; Make sure paddles
    BTFSS   GPIO, XDAHPDL    ; remain unpressed
    GOTO    WAIT4OFF
    DECFSZ  AL, F
    GOTO    WAIT4LP
 ifndef BEACON
    BCF     GPIO, STK       ; Unkey Sidetone
 endif
    BCF     GPIO, KEY       ; Unkey transmitter
    GOTO    SERVICE

;=========================================
;       Key XMTR for straight key mode
;
; We only look at one paddle for straight
; keying but the user can select which one
; with the swap command.
;=========================================

KEYDIRECT
    BTFSS   MODEREG, STRQ   ; Restore sidetone
    BCF     MODEREG, STEN   ;   preference.
 ifndef BEACON
    BTFSC   MODEREG, STEN   ; Sidetone enabled ?
    BSF     GPIO, STK       ; Key Sidetone
 endif
    BSF     GPIO, KEY       ; Key transmitter
KEYDLP
    BCF PROCLAT, DAHPDL  ; Clear latched DAH
    CALL    STCLK            ; Key XMTR and delay w/sidetone
    BTFSC   PROCLAT, DAHPDL  ; DAH paddle still pressed ?
    GOTO    KEYDLP       ; Yes: Loop
    GOTO    WAIT4OFF     ; No: Leave via debounce

;=========================================
;       Set Extra Letterspace
; Uses: AH, CL
;=========================================

FARNSWORTH
    MOVLW   M_E             ; Ask for Number
    CALL    OSCHAR
    CALL    L_GETCW         ; Get user response in AH
    MOVLW   M_TO            ; Did user just sit there ?
    SUBWF   AH, W
    BTFSC   STATUS, Z
    GOTO    QUERET          ; Yes, end it
    MOVLW   80H             ; Validity check value
    MOVWF   CL              ;   save it for later
    CALL    L_CONV_LO       ; Convert AH to 1's
    BCF     STATUS, PC9     ; Clear page bit
```

```
    ADDWF   CL, W           ; Check value
    BTFSS   STATUS, C       ; Valid if Carry = 1
    GOTO    QUERET
    MOVWF   FARNS           ; Set new Farnsworth adj.
    GOTO    COMRET_R

;=========================================
;       Set New CW Sending Speed
; Uses: AH, CL, CH
;=========================================

SPEED
    CLRF    CH              ; Init pass count
SPLOOP
    MOVLW   M_E             ; Ask for Number
    CALL    OSCHAR
    CALL    L_GETCW         ; Get user response in AH
    MOVLW   M_TO            ; Did user just sit there ?
    SUBWF   AH, W
    BTFSC   STATUS, Z
    GOTO    QUERET          ; Yes, end it
    BTFSC   CH, 0           ; Skip on 1st pass
    GOTO    SPSUM
    CALL    L_CONV_HI       ; Convert AH to 10's
    BCF     STATUS, PC9     ; Clear page bit
    MOVWF   CL              ; Save MSN
    BSF     CH, 0           ; Set pass flag
    GOTO    SPLOOP
SPSUM
    CALL    L_CONV_LO       ; Convert AH to 1's
    BCF     STATUS, PC9     ; Clear page bit
    ADDWF   CL, F           ; Add in MSN
    BTFSS   STATUS, C       ; If both are valid both MSBs = 1
    GOTO    QUERET          ; So if Carry=1 we have good values
    MOVLW   -5D             ; Adjust for indexing
    ADDWF   CL, W           ; And verify low limit
    BTFSS   STATUS, C       ; C==1 if >= 5, error if not
    GOTO    QUERET
    MOVWF   SPEEDIDX
    CALL    L_IDX2SPD       ; Convert index to delay constant
    BCF     STATUS, PC9     ; Clear page bit
    MOVWF   TIMEBASE        ; Store it away
    GOTO    COMRET_R
QUERET
    MOVLW   M_QUE           ; Send ? for error
    GOTO    COMRET

;=========================================
;       Message Selection
;=========================================

MSGSELECT
    CALL    SELDELAY
MSGLOOP
    MOVLW   MSG2-MSGBASE    ; Select message 2: Short CQ
    MOVWF   CURMSG
    MOVLW   M_C             ; C
    CALL    OSCHAR
    MOVLW   M_Q             ; Q
    CALL    OSCHAR
    CALL    SELDELAY
    BTFSC   CL, 0           ; Paddle closure ?
    GOTO    MSGDONE         ; Yes, all done

    MOVLW   MSG1-MSGBASE    ; Select message 1: Variable
    MOVWF   CURMSG
 ifdef CQTEST
    MOVLW   M_C             ; C
    CALL    OSCHAR
    MOVLW   M_Q             ; Q
    CALL    OSCHAR
    MOVLW   M_T             ; T
    CALL    OSCHAR
 endif
 ifdef CONTEST
    MOVLW   M_T             ; T
```

```
    CALL    OSCHAR
    MOVLW   M_S             ; S
    CALL    OSCHAR
    MOVLW   M_T             ; T
    CALL    OSCHAR
 endif
 ifdef QRP
    MOVLW   M_Q             ; Q
    CALL    OSCHAR
    MOVLW   M_R             ; R
    CALL    OSCHAR
    MOVLW   M_P             ; P
    CALL    OSCHAR
 endif
 ifdef STANDARD
    MOVLW   M_C             ; C
    CALL    OSCHAR
    MOVLW   M_Q             ; Q
    CALL    OSCHAR
    MOVLW   M_L             ; L
    CALL    OSCHAR
 endif
    CALL    SELDELAY
    BTFSC   CL, 0           ; Paddle closure ?
    GOTO    MSGDONE         ; Yes, all done

    MOVLW   MSG3-MSGBASE    ; No, select message 3
    MOVWF   CURMSG
    MOVLW   M_D             ; D
    CALL    OSCHAR
    MOVLW   M_X             ; X
    CALL    OSCHAR
    CALL    SELDELAY
    BTFSC   CL, 0           ; Paddle closure ?
    GOTO    MSGDONE         ; Yes, all done

    MOVLW   MSG4-MSGBASE    ; No, select message 4
    MOVWF   CURMSG
    MOVLW   M_C             ; C
    CALL    OSCHAR
    MOVLW   M_Q             ; Q
    CALL    OSCHAR
    MOVLW   M_C             ; C
    CALL    OSCHAR
    CALL    SELDELAY
    BTFSC   CL, 0           ; Paddle closure ?
    GOTO    MSGDONE         ; Yes, all done

    MOVLW   MSG5-MSGBASE    ; No, select message 5
    MOVWF   CURMSG
    MOVLW   M_M             ; M
    CALL    OSCHAR
    MOVLW   M_S             ; S
    CALL    OSCHAR
    MOVLW   M_G             ; G
    CALL    OSCHAR
    CALL    SELDELAY
    BTFSS   CL, 0           ; Paddle closure ?
    GOTO    MSGLOOP         ; No: loop till user picks one
MSGDONE
    MOVLW   M_R             ; Yes: Send an 'R' for acknowledgement
    CALL    OSCHAR
    GOTO    WAIT4OFF        ; Common "wait then return"

;=========================================
;       Paddle Swap Toggle
;=========================================

SWAP_TOGGLE
    MOVLW   SWAPBIT
    GOTO    COMXOR          ; Shared XOR

;=========================================
;       Sidetone Toggle
;=========================================
```

```
SIDETONE
    MOVLW   STRQBIT
    GOTO    COMXOR          ; Shared XOR

;=========================================
;       Toggle Keyer Mode
; Default is keyer mode, everytime this
; is called the mode will toggle between
; keyer and straight key mode.
;=========================================

KEYMODE
    MOVLW   SKMBIT
COMXOR
    XORWF   MODEREG, F      ; Shared XOR
    GOTO    MSGDONE         ; Common return, R for ack
;
;=========================================
;       Toggle Iambic Mode
; Toggle the iambic mode between Mode A
; which is not self completing and Mode B
; which is. Self completing means an extra
; alternate element is sent after both
; paddles are released.
;=========================================

IAMBIC
    MOVLW   IABBIT
    XORWF   MODEREG, F
    MOVLW   M_A             ; Assume Iambic mode A
    BTFSS   MODEREG, IAM_A  ; A=1, B=0
    MOVLW   M_B             ; Assumed wrong, it's B
    GOTO    COMRET

;=========================================
;       Autospace Toggle
;=========================================

ASP_TOGGLE
    MOVLW   ASPBIT
    XORWF   MODEREG, F
    MOVLW   M_A             ; Assume Autospace mode
    BTFSS   MODEREG, ASP    ; Autospace=1, Normal=0
    MOVLW   M_N
    GOTO    COMRET

;=========================================
;       Toggle DE Mode
; Toggle the user defined iambic enable
; In iambic mode, when both paddles are
; pressed alternating dits/dahs are sent.
; I_MODE off will allow the use of the
; automatic 'DE' message.
;=========================================

DE_TOGGLE
    MOVLW   IAMBIT
    XORWF   MODEREG, F
    MOVLW   M_D             ; Assume DE mode
    BTFSC   MODEREG, I_MODE
    MOVLW   M_I             ; Assumed wrong, it's iambic mode
    GOTO    COMRET          ; Use shared return

;=========================================
;       Code Practice
; Send a stream of random CW characters
;=========================================

PRACTICE
    BSF     STATUS, PC9      ; Set PAGE bit for long GOTO
    GOTO    PRAC_RUN

;=========================================
;       Load RAM Routine
; Two entry points, one for loading callsign
; and a second for loading user message.
```

```
;
; Uses AH, CL
;=========================================

LOADCALL
    MOVLW   USRCALL         ; Point to start of callsign string
    MOVWF   FSR
    MOVLW   08H             ; Max callsign length (8 decimal)
    GOTO    LU00
LOADUSER

 ifdef __16F84A
    MOVLW   USERRAM         ; Point to start of user string
 else
    MOVLW   USERRAM-20H     ; Point to start of user string
    MOVWF   FSR
    BSF     MODEREG, FSRMSB ; Set FSR MS address flag
 endif
    MOVLW   0FH             ; Max string length (15 decimal)
LU00
    MOVWF   CL              ; Length in reg CL
;       CLRF    INDF            ; In case nothing is entered
LULOOP
    MOVLW   M_E             ; Signal user to enter code
    CALL    OSCHAR
    CALL    L_GETCW         ; Get a letter in AH
    MOVLW   M_PER           ; Did user explicitly end ?
    SUBWF   AH, W
    BTFSC   STATUS, Z
    GOTO    LUDONE          ; Yep, leave
    MOVLW   M_TO            ; Did user just sit there ?
    SUBWF   AH, W
    BTFSS   STATUS, Z
    GOTO    LU0             ; No, we have a valid character
    MOVLW   M_SP            ; Yes, they want a space
    GOTO    LU1
LU0
    MOVF    AH, W           ; Get char
LU1
 ifndef __16F84A
    BTFSC   MODEREG, FSRMSB ; Upper reg access ?
    BSF     FSR, REGBANK    ; Select upper bank
 endif
    MOVWF   INDF            ; Store in RAM
    INCF    FSR, F          ; Bump counter
    MOVLW   M_END           ; Append an EOS token
    MOVWF   INDF            ; Store in RAM
 ifndef __16F84A
    BCF     FSR, REGBANK    ; Select lower bank
 endif
    DECFSZ  CL, F           ; Decrement length count
    GOTO    LULOOP
LUDONE
 ifndef __16F84A
    BCF     MODEREG, FSRMSB ; Reset FSR address flag
 endif
                ; Fall thru to common return

;=========================================
;       Common Return
;=========================================

COMRET_R
    MOVLW   M_R             ; Send a "roger"
COMRET
    CALL    OSCHAR
    GOTO    SERVICE
;
;=========================================
;       Dump User String
;=========================================

DUMPUSER
    MOVLW   MSG5-MSGBASE    ; Point to USER message
QVRET
    CALL    L_SENDMSG       ; We will return to caller from there
```

```
    MOVLW   BEACON_TIME
    MOVWF   BX
    GOTO    SERVICE

;=========================================
;       Report Morse speed
;=========================================

OUTWPM
    MOVLW   5D              ; Slowest speed
    ADDWF   SPEEDIDX, W     ; Add index

OUTDEC
    CLRF    CH              ; Clear tens register
    MOVWF   CL              ; Copy decimal number
    MOVLW   10D             ; Base 10
ODLOOP
    SUBWF   CL, F           ; Extract 10's
    BTFSS   STATUS, C       ; C==0 : Underflow
    GOTO    ODNEG
    INCF    CH, F           ; Incr 10's
    GOTO    ODLOOP
ODNEG
    ADDWF   CL, F           ; Restore ones to positive
    MOVF    CH, W           ; Output 10's first
    BTFSC   STATUS, Z       ; Suppress leading zero
    GOTO    ODONES
    CALL    DEC2CW
    BCF     STATUS, PC9     ; Clear page bit
    CALL    OSCHAR
ODONES
    MOVF    CL, W           ; Output 1's
    CALL    DEC2CW
    BCF     STATUS, PC9     ; Clear page bit
    CALL    OSCHAR
    MOVLW   MSG8-MSGBASE    ; Send WPM string
    GOTO    QVRET           ; Use shared return

;*****************************************
;*      Initialization Routine           *
;*****************************************

INIT
 ifdef __16F84A

 ifdef BEACON
    MOVLW   B'11111000'     ; IN:RB7,6,5,4,3 OUT:RB2,1,0
 else
    MOVLW   B'11110000'     ; IN:RB7,6,5,4 OUT:RB3,2,1,0
 endif
    BSF STATUS, RP0

    MOVWF   TRISB ^ 0x80

    MOVLW   0
    MOVWF   TRISA ^ 0x80    ; A = all outputs

    MOVLW   B'00001110'     ; Enable PUs, Pscl->WDT
    MOVWF   OPTION_REG ^ 0x80   ; Timer Clk=Internal, Prescale=1:64

    BCF STATUS, RP0

    CLRF    GPIO        ; Note: Beacon Request is asserted
                ; but not active since RB3 is an input

    MOVLW   B'00001000' ; Enable RB port change interrupt (but NOT GIE)
    MOVWF   INTCON

 else
 ifdef BEACON
    MOVLW   B'00101011'     ; IN:GP5,3,1,0 OUT:GP4,2
 else
    MOVLW   B'00001011'     ; IN:GP3,1,0 OUT:GP5,4,2
 endif
    TRIS    GPIO            ;
    CLRF    GPIO            ; Init PIC Outputs
```

```
                ; Note: Beacon Request is asserted
                ; but not active since GP5 is an input
    MOVLW   B'00001110'     ; Wake-up on GPIO, Enable PUs, Pscl->WDT
    OPTION                  ; Timer Clk=Internal, Prescale=1:64
 endif

 ifdef BEACON
    BTFSS   STATUS, PD      ; Is this is a power up or dead man WDT ?
                ; Finish reset if so
    GOTO    WAKEUP      ; else: Go do wakeup stuff
 else
 ifndef __16F84A
    BTFSC   STATUS, GPWUF   ; Were we asleep ?
    GOTO    SERVICE         ; Go directly to main loop if so
 else
    ; TODO - implement wake-on-interrupt
 endif
 endif

 ifdef SETFARNS
        MOVLW   1       ; CAN ONLY USE THIS IN NON_BEACON MODE !!!
        MOVWF   FARNS
 else
    CLRF    FARNS           ; Initialize Farnsworth adj.
 endif
    MOVLW   SPEED_DEFAULT   ; Initialize Morse Timebase
    MOVWF   SPEEDIDX
    CALL    L_IDX2SPD       ; Convert index to delay constant
    BCF     STATUS, PC9     ; Clear page bit
    MOVWF   TIMEBASE        ; Store it away

    MOVLW   MODE_DEFAULT    ; Load MODEREG default
    MOVWF   MODEREG
    MOVLW   USERRAM
    MOVWF   FSR             ; Use indirect addressing
    CLRF    INDF            ; To init user message empty
    MOVLW   USRCALL         ; Load callsign into RAM
    MOVWF   FSR             ; Use indirect addressing
    BSF     STATUS, PC9     ; Set PAGE bit for long GOTO
    GOTO    XFRSTUB
XFRRET
    MOVLW   MSG2-MSGBASE    ; Select message 2: Short CQ
    MOVWF   CURMSG          ;  as default
    BSF     PROCLAT, TXSQ   ; Don't transmit, just run sidetone
                ; SERVICE will reset TXSQ
    MOVLW   M_R             ; Send R for hello
    BTFSS   GPIO, XDAHPDL    ; Hook to get version
    MOVLW   M_G             ; Send Version ID
    GOTO    COMRET          ; Use common return entry

;=============================================
;      Page Boundary
;=============================================
    ORG 0200h
;=============================================

;=============================================
;       Move W to (FSR++)
;=============================================

XFRIDX
    MOVWF   INDF
    INCF    FSR, F
    RETLW   0

;=============================================
; Message Table
;=============================================
; Call with message pointer in DH reg
; Table entry *DH++ is returned in W.
; We start at a new page to allow us to have
; a long message table which we can be sure
; will never cross a page boundary.
; Note: CALLs and computed jumps will always
; clear the PC[8] bit. PC[9] is sourced from
; STATUS reg bit 5.
```

```
GETMSG
    MOVF    DH, W
    INCF    DH, F
 ifdef __16F84A
    MOVWF   WTEMP
    MOVLW   HIGH($)
    MOVWF   PCLATH
    MOVF    WTEMP,W
 endif
    ADDWF   PC, F           ; Jump through table

;--------------------------------------------------

MSGBASE
MSG1
 ifdef CONTEST
    RETLW   M_T     ; Contest Message
    RETLW   M_E
    RETLW   M_S
    RETLW   M_T
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_T     ; Contest Message
    RETLW   M_E
    RETLW   M_S
    RETLW   M_T
    RETLW   M_SP
    RETLW   M_END
 endif
 ifdef CQTEST
    RETLW   M_C     ; CQ TEST
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_T
    RETLW   M_E
    RETLW   M_S
    RETLW   M_T
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_T
    RETLW   M_E
    RETLW   M_S
    RETLW   M_T
    RETLW   M_SP
    RETLW   M_D
    RETLW   M_E
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_K
    RETLW   M_END
 endif
 ifdef QRP
    RETLW   M_C     ; QRP
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_D
    RETLW   M_E
    RETLW   M_SP
```

```
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_CALL
;   RETLW   M_SP
    RETLW   M_DN
    RETLW   M_Q
    RETLW   M_R
    RETLW   M_P
    RETLW   M_SP
 ifdef AR_ON
    RETLW   M_AR
    RETLW   M_SP
 endif
    RETLW   M_K
    RETLW   M_END
 endif
 ifdef STANDARD
    RETLW   M_C     ; Long CQ Message
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_D
    RETLW   M_E
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
 endif
;
MSG2
    RETLW   M_C     ; CQ Message
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_D
    RETLW   M_E
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
;   RETLW   M_CALL  ;extra
;   RETLW   M_SP    ;extra
 ifdef AR_ON
    RETLW   M_AR
    RETLW   M_SP
 endif
    RETLW   M_K
    RETLW   M_END
;
MSG3
    RETLW   M_C     ; CQ DX Message
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_C
    RETLW   M_Q
    RETLW   M_SP
    RETLW   M_D
    RETLW   M_X
    RETLW   M_SP
    RETLW   M_D
    RETLW   M_E
```

```
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_D
    RETLW   M_X
    RETLW   M_SP
 ifdef AR_ON
    RETLW   M_AR
    RETLW   M_SP
 endif
    RETLW   M_K
    RETLW   M_END
;
MSG4
    RETLW   M_USER  ; Custom Contest Message
    RETLW   M_SP
    RETLW   M_D
    RETLW   M_E
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
    RETLW   M_CALL
    RETLW   M_SP
 ifdef AR_ON
    RETLW   M_AR
    RETLW   M_SP
 endif
    RETLW   M_K
    RETLW   M_END
;
MSG5
    RETLW   M_USER  ; User Message
    RETLW   M_SP
    RETLW   M_END
;
MSG7
 ifdef NO_DE
    RETLW   CALL_1
    RETLW   CALL_2
    RETLW   CALL_3
 else
    RETLW   M_E
    RETLW   M_SP
    RETLW   M_CALL
 endif
    RETLW   M_SP
    RETLW   M_END
;
MSG8
    RETLW   M_SP    ; WPM suffix
    RETLW   M_W
    RETLW   M_P
    RETLW   M_M
    RETLW   M_END

;=========================================
; Randomized Code Practice Table
;=========================================

GETPRAC
     ifdef __16F84A
    MOVWF   WTEMP
    MOVLW   HIGH($)
    MOVWF   PCLATH
    MOVF    WTEMP,W
 endif
    ADDWF   PC, F   ; Jump through table
CWTBL
    RETLW   0xE4    ; 8 8  1110 0100 = DAH-DAH-DAH-DI-DIT
    RETLW   0x20    ;25 I  0010 0000 = DI-DIT
    RETLW   0x70    ;39 W  0111 0000 = DI-DAH-DAH
    RETLW   0x50    ;34 R  0101 0000 = DI-DAH-DIT
    RETLW   0xA8    ;19 C  1010 1000 = DAH-DI-DAH-DIT
    RETLW   0xD8    ;33 Q  1101 1000 = DAH-DAH-DI-DAH
```

```
    RETLW   0x32    ;15 ?  0011 0010 = DI-DI-DAH-DAH-DI-DIT
    RETLW   0x3C    ; 2 2  0011 1100 = DI-DI-DAH-DAH-DAH
    RETLW   0x94    ;16 /  1001 0100 = DAH-DI-DI-DAH-DIT
    RETLW   0x60    ;17 A  0110 0000 = DI-DAH
    RETLW   0x68    ;32 P  0110 1000 = DI-DAH-DAH-DIT
    RETLW   0xFC    ; 0 0  1111 1100 = DAH-DAH-DAH-DAH-DAH
    RETLW   0x78    ;26 J  0111 1000 = DI-DAH-DAH-DAH
    RETLW   0xCE    ;13 ,  1100 1110 = DAH-DAH-DI-DI-DAH-DAH
    RETLW   0x0C    ; 4 4  0000 1100 = DI-DI-DI-DI-DAH
    RETLW   0x10    ;35 S  0001 0000 = DI-DI-DIT
    RETLW   0x8C    ;14 BT 1000 1100 = DAH-DI-DI-DI-DAH
    RETLW   0x54    ;10 AR 0101 0100 = DI-DAH-DI-DAH-DIT
    RETLW   0x28    ;22 F  0010 1000 = DI-DI-DAH-DIT
    RETLW   0xD0    ;23 G  1101 0000 = DAH-DAH-DIT
    RETLW   0x90    ;20 D  1001 0000 = DAH-DI-DIT
    RETLW   0x04    ; 5 5  0000 0100 = DI-DI-DI-DI-DIT
    RETLW   0x88    ;18 B  1000 1000 = DAH-DI-DI-DIT
    RETLW   0x18    ;38 V  0001 1000 = DI-DI-DI-DAH
    RETLW   0xF4    ; 9 9  1111 0100 = DAH-DAH-DAH-DAH-DIT
    RETLW   0x56    ;12 .  0101 0110 = DAH-DAH-DI-DI-DAH
    RETLW   0xC0    ;36 T  1100 0000 = DAH
    RETLW   0x30    ;37 U  0011 0000 = DI-DI-DAH
    RETLW   0xB8    ;41 Y  1101 1000 = DAH-DAH-DI-DAH
    RETLW   0xE0    ;29 M  1110 0000 = DAH-DAH
    RETLW   0xC4    ; 7 7  1100 0100 = DAH-DAH-DI-DI-DIT
    RETLW   0x84    ; 6 6  1000 0100 = DAH-DI-DI-DI-DIT
    RETLW   0x16    ;11 SK 0001 0110 = DI-DI-DI-DAH-DI-DAH
    RETLW   0x1C    ; 3 3  0001 1100 = DI-DI-DI-DAH-DAH
    RETLW   0x08    ;24 H  0000 1000 = DI-DI-DI-DIT
    RETLW   0x98    ;40 X  1001 1000 = DAH-DI-DI-DAH
    RETLW   0xC8    ;42 Z  1100 1000 = DAH-DAH-DI-DIT
    RETLW   0x48    ;28 L  0100 1000 = DI-DAH-DI-DIT
    RETLW   0x7C    ; 1 1  0111 1100 = DI-DAH-DAH-DAH-DAH
    RETLW   0xB0    ;27 K  1011 0000 = DAH-DI-DAH
    RETLW   0xA0    ;30 N  1010 0000 = DAH-DIT
    RETLW   0x40    ;21 E  0100 0000 = DIT
    RETLW   0xF0    ;31 O  1111 0000 = DAH-DAH-DAH

;=========================================
; Code Speed Table
;=========================================

;-----------------------------------------
;       TIMEBASE Defaults
;
; Using ARRL Handbook as a reference, a 5 WPM dit
; is 240 msec, a dah is 720 ms.
; Formula:
;
; Count= (((5/WPM) * 240) / SIDETONE_PERIOD)
;
; For Sidetone Freq of 870 Hz (1.15 ms):
;
; Count = ((5/WPM) * (240/1.15))
;
SVAL   EQU      (5D * SPD_CONST)    ; Fudged By oscilloscope verification
;
;-----------------------------------------

IDX2SPD
 ifdef __16F84A
    MOVWF   WTEMP
    MOVLW   HIGH($)
    MOVWF   PCLATH
    MOVF    WTEMP,W
 endif
    ADDWF   PC, F    ; Jump through table
SPEEDTBL
    RETLW   SVAL/5D  ; WPM_5
    RETLW   SVAL/6D  ; WPM_6
    RETLW   SVAL/7D  ; WPM_7
    RETLW   SVAL/8D  ; WPM_8
    RETLW   SVAL/9D  ; WPM_9
    RETLW   SVAL/10D ; WPM_10
    RETLW   SVAL/11D ; WPM_11
    RETLW   SVAL/12D ; WPM_12
```

```
    RETLW   SVAL/13D ; WPM_13
    RETLW   SVAL/14D ; WPM_14
    RETLW   SVAL/15D ; WPM_15
    RETLW   SVAL/16D ; WPM_16
    RETLW   SVAL/17D ; WPM_17
    RETLW   SVAL/18D ; WPM_18
    RETLW   SVAL/19D ; WPM_19
    RETLW   SVAL/20D ; WPM_20
    RETLW   SVAL/21D ; WPM_21
    RETLW   SVAL/22D ; WPM_22
    RETLW   SVAL/23D ; WPM_23
    RETLW   SVAL/24D ; WPM_24
    RETLW   SVAL/25D ; WPM_25
    RETLW   SVAL/26D ; WPM_26
    RETLW   SVAL/27D ; WPM_27
    RETLW   SVAL/28D ; WPM_28
    RETLW   SVAL/29D ; WPM_29
    RETLW   SVAL/30D ; WPM_30
    RETLW   SVAL/31D ; WPM_31
    RETLW   SVAL/32D ; WPM_32
    RETLW   SVAL/33D ; WPM_33
    RETLW   SVAL/34D ; WPM_34
    RETLW   SVAL/35D ; WPM_35
    RETLW   SVAL/36D ; WPM_36
    RETLW   SVAL/37D ; WPM_37
    RETLW   SVAL/38D ; WPM_38
    RETLW   SVAL/39D ; WPM_39
    RETLW   SVAL/40D ; WPM_40
    RETLW   SVAL/41D ; WPM_41
    RETLW   SVAL/42D ; WPM_42
    RETLW   SVAL/43D ; WPM_43
    RETLW   SVAL/44D ; WPM_44
    RETLW   SVAL/45D ; WPM_45
    RETLW   SVAL/46D ; WPM_46
    RETLW   SVAL/47D ; WPM_47
    RETLW   SVAL/48D ; WPM_48
    RETLW   SVAL/49D ; WPM_49

;=========================================
; SENDMSG:
;    Sends message pointed to by W
; Be careful of case where message table
; grows so large that it pushes this code
; above 0xff or a CALL instruction will
; not be able to reach it.
; Uses AL, DH (DH = MSGIDX)
;=========================================

SENDMSG
    MOVWF   DH              ; Store index
S0LOOP
    BTFSC   PROCLAT, USERON ; Determine data source
    GOTO    S0USER
    CALL    GETMSG          ; Get *DH++ in W
    GOTO    S0CONT
S0USER
 ifndef __16F84A
    BTFSC   MODEREG, FSRMSB ; Upper register set access ?
    BSF     FSR, REGBANK    ; Select upper bank
 endif
    MOVF    INDF, W         ; Get *USERRAM++
    INCF    FSR, F
 ifndef __16F84A
    BCF     FSR, REGBANK    ; Select lower bank
 endif

S0CONT
    MOVWF   AL
    MOVLW   M_END           ; End delimiter ?
    SUBWF   AL, W
    BTFSS   STATUS, Z       ; Skip if no compare
    GOTO    S1CONT
    BTFSS   PROCLAT, USERON ; End of user or end of message ?
    GOTO    S1DONE
    BCF     PROCLAT, USERON
 ifndef __16F84A
```

```
    BCF     MODEREG, FSRMSB ; Clear FSR upper address flag
 endif
    GOTO    S0LOOP
S1DONE
    BCF     STATUS, PC9     ; Clear PAGE bit, we are done here
    BCF     PROCLAT, USERON ; Clear User mode (case: user msg aborted)
 ifndef __16F84A
    BCF     MODEREG, FSRMSB ; Clear FSR flag (case: user msg aborted)
 endif

    RETLW   0               ; All done, return
S1CONT
    MOVLW   M_USER          ; Toggle to User Mode ?
    SUBWF   AL, W
    BTFSC   STATUS, Z       ; Skip if no compare
    GOTO    S1USER
    MOVLW   M_CALL          ; Toggle to Callsign Mode ?
    SUBWF   AL, W
    BTFSC   STATUS, Z       ; Skip if no compare
    GOTO    S1CALL
    MOVLW   M_SKIP          ; Skip entry ?
    SUBWF   AL, W
    BTFSC   STATUS, Z       ; Skip if no compare
    GOTO    S0LOOP
    MOVLW   M_SP            ; Word space ?
    SUBWF   AL, W
    BTFSS   STATUS, Z       ; Skip if compare
    GOTO    S2LOOP
    BCF     STATUS, PC9     ; Clear PAGE bit for low page call
    CALL    WORDSPACE       ; Word Space = 7 time bits
    BSF     STATUS, PC9     ; Set PAGE bit on return
    GOTO    S0LOOP
S1CALL
    MOVLW   USRCALL         ; Point to start of callsign string
    GOTO    S100
S1USER
 ifdef __16F84A
    MOVLW   USERRAM         ; Point to start of user string
 else
    BSF     MODEREG, FSRMSB ; Set upper register set access flag
    MOVLW   USERRAM-20H     ; Point to start of user string
 endif
S100
    MOVWF   FSR
    BSF     PROCLAT, USERON
    GOTO    S0LOOP
S2LOOP
    BTFSS   GPIO, XDAHPDL
    GOTO    ABORT1
    BTFSC   GPIO, XDITPDL
    GOTO    NOABORT
ABORT1
 ifdef BEACON
    CLRWDT          ; Reset watchdog timer
 endif
    CLRF    PROCLAT
    BTFSC   GPIO, XDAHPDL    ; Wait till hands off paddle
    BTFSS   GPIO, XDITPDL
    GOTO    ABORT1
    GOTO    S1DONE
NOABORT
    MOVF    AL, W           ; Get coded morse
    ADDWF   AL, F           ; AL*2, if (Z==0) DONE else C=DIT/DAH
    BTFSS   STATUS, Z       ; Skip if zero
    GOTO    S2CONT
    BCF     STATUS, PC9     ; Clear PAGE bit for low page call
    CALL    LETTERSPACE     ; Inter-letter space
    BSF     STATUS, PC9     ; Set PAGE bit on return
    GOTO    S0LOOP
S2CONT
    BTFSC   STATUS, C       ; (C==1) then DAH else DIT
    GOTO    S2DAH
    BCF     STATUS, PC9     ; Clear PAGE bit for low page call
    CALL    DITOUT
    BSF     STATUS, PC9     ; Set PAGE bit on return
    GOTO    S2LOOP
```

```
S2DAH
    BCF     STATUS, PC9     ; Clear PAGE bit for low page call
    CALL    DAHOUT
    BSF     STATUS, PC9     ; Set PAGE bit on return
    GOTO    S2LOOP

;=========================================
; Get a Morse Character from User
; Value returned in AH
; Uses AL, DH
;=========================================

GETCW
    CLRF    AH              ; Start with empty frame
    MOVLW   08H             ; Init bit count
    MOVWF   AL
    MOVLW   GC_TO_L         ; Init timeouts
    MOVWF   DELAYLO
    MOVLW   GC_TO_M         ; Init timeouts
    MOVWF   DELAYHI
    MOVLW   GC_TO_H         ; Init timeouts
    MOVWF   DH
GCLOOP
    BCF     STATUS, PC9     ; Clear PAGE bit for low page call
    CALL    SAMPLE
    BSF     STATUS, PC9     ; Set PAGE bit back for high access

    BTFSS   PROCLAT, DITPDL  ; Both Dit
    GOTO    GCSINGLE
    BTFSC   PROCLAT, DAHPDL  ;          and Dah ?
    GOTO    GCTOGGLE

GCSINGLE
    BTFSC   PROCLAT, DITPDL  ; DIT Paddle closed when = 1
    GOTO    GCDIT
    BTFSC   PROCLAT, DAHPDL  ; DAH Paddle closed when = 1
    GOTO    GCDAH

    DECFSZ  DELAYLO, F      ; Inner loop test
    GOTO    GCLOOP
    MOVLW   GC_TO_L         ; Reset inner loop
    MOVWF   DELAYLO
    DECFSZ  DELAYHI, F      ; Middle loop Test
    GOTO    GCLOOP
    MOVLW   GC_TO_M         ; Reset middle loop
    MOVWF   DELAYHI
    DECFSZ  DH, F           ; Outer loop Test
    GOTO    GCLOOP
    GOTO    GCJUST          ; Timed out, we are done, now justify

GCTOGGLE
    BTFSS   MODEREG, IAM_A
    GOTO    GCTOGL1
    MOVLW   PROCMASK        ; Mode A, kill any in latch
    ANDWF   PROCLAT, F
GCTOGL1
    BTFSC   PROCLAT, OUTLAST ; If Dit was last send a DAH
    GOTO    GCDAH
GCDIT
    BCF     STATUS, PC9     ; Clear PAGE bit for low page call
    CALL    DITOUT
    BSF     STATUS, PC9     ; Set PAGE bit on return
    BCF     STATUS, C       ; DIT is 0
    GOTO    GCLOAD
GCDAH
    BCF     STATUS, PC9     ; Clear PAGE bit for low page call
    CALL    DAHOUT
    BSF     STATUS, PC9     ; Set PAGE bit on return
    BSF     STATUS, C       ; DAH is 1
GCLOAD
    RLF     AH, F           ; Shift left 1, CARRY->LSB
    DECFSZ  AL, F           ; Decrement bit count
    GOTO    GCLD1
    GOTO    GCERR           ; Overflow, treat as null
GCLD1
    MOVLW   GC_TO_L         ; Restart timeouts
```

```
    MOVWF   DELAYLO
    MOVLW   GC_TO_M         ;
    MOVWF   DELAYHI
    MOVLW   1               ; set a shorter 2nd timeout
    MOVWF   DH
    GOTO    GCLOOP
GCJUST
    BSF     STATUS, C       ; Add tag bit
    GOTO    GCJ2
GCJ1
    BCF     STATUS, C       ; Fill in bottom
GCJ2
    RLF     AH, F           ;   with zeroes
    DECFSZ  AL, F           ; Keep going till left justified
    GOTO    GCJ1

GCDONE
    BCF     STATUS, PC9     ; Clear PAGE bit for low page return
    RETLW   0

GCERR
    MOVLW   M_TO            ; Overflow error treated as a timeout
    MOVWF   AH
    GOTO    GCDONE

;=========================================
; This was moved here from the low page to
; free up some code space. It's function is
; to load fixed callsign into RAM.
;=========================================

XFRSTUB
    MOVLW   CALL_0
    CALL    XFRIDX
    MOVLW   CALL_1
    CALL    XFRIDX
    MOVLW   CALL_2
    CALL    XFRIDX
    MOVLW   CALL_3
    CALL    XFRIDX
    MOVLW   CALL_4
    CALL    XFRIDX
    MOVLW   CALL_5
    CALL    XFRIDX
    MOVLW   CALL_6
    CALL    XFRIDX
    MOVLW   CALL_7
    CALL    XFRIDX
    BCF     STATUS, PC9     ; Clear PAGE bit for low page goto
    GOTO    XFRRET

;=========================================
;       Code Practice
; Send a stream of random CW characters
;=========================================
;
; Generates pseudo random numbers by hopping
; though a morse character table with a constantly
; incrementing index. The index is incremented on
; every pass by an odd valued skip factor. A new
; skip factor is used periodically. The interval
; of the period is "randomly" read from the timer.
; Note that by virtue of the fact that the skip
; can cause a table bounds wrap, both positive
; and negative moves occur.
; DH = Skip,  DL = Running index, AH = Loop count

PRAC_RUN
    CLRF    DL              ; Clear index
PRCLP0
 ifdef __16F84A
    MOVF    TMR0, W       ; Get an initial skip factor
 else
    MOVF    TIMER0, W       ; Get an initial skip factor
 endif
    MOVWF   AH              ; save loop count
```

```
    IORLW   01H             ; make it odd
    ANDLW   01FH            ; 1 -> 31
    MOVWF   DH
PRCLP1
    MOVF    DH, W           ; Get skip value in W
    ADDWF   DL, F           ; Add to index
    MOVLW   -43D            ; If > 42 wrap around
    ADDWF   DL, W           ; Carry will be set on wrap
    BTFSS   STATUS, C       ; Value wrapped, use wrapped value
    MOVF    DL, W           ; Value was within limits, use it as is
    MOVWF   DL              ; Update index in case we wrapped
    CALL    GETPRAC
    BCF     STATUS, PC9     ; Clear page bit for low page call
    CALL    OSCHAR
    BSF     STATUS, PC9     ; Set PAGE bit on return
    BTFSC   GPIO, XDITPDL    ; loop till paddle hit
    BTFSS   GPIO, XDAHPDL    ; loop till paddle hit
    GOTO    PRCDONE         ; paddle hit: leave
    DECFSZ  AH, F           ; decrement loop count
    GOTO    PRCLP1          ; NZ: Continue
    GOTO    PRCLP0          ; Z:  Get new skip factor & loop count
PRCDONE
    BCF     STATUS, PC9     ; Clear page bit
    GOTO    WAIT4OFF        ; Common "wait then return"

;=========================================
; Convert CW Character Into Table Index
; Input: CW in AH
; Return: Index in AH
; Uses: AL, AH
; Note: A timeout code (M_TO) will be handled
; the same as an illegal code, .ie it won't
; be found in the table.
;=========================================

CW2IDX
    MOVF    AH, W           ; Transfer AH to AL
    MOVWF   AL
    MOVLW   M_A             ; A ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   0
    MOVLW   M_C             ; C ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   01D
    MOVLW   M_F             ; F ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   02D
    MOVLW   M_I             ; I ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   03D
    MOVLW   M_K             ; K ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   04D
    MOVLW   M_L             ; L ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   05D
    MOVLW   M_S             ; S ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   06D
    MOVLW   M_T             ; T ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   07D
    MOVLW   M_U             ; U ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   08D
    MOVLW   M_X             ; X ?
    SUBWF   AL, W           ; Compare
```

```
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   09D
    MOVLW   M_Z             ; Z ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   10D
    BCF     PROCLAT, CONVERSE ; End converse mode
    MOVLW   M_D             ; D ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   11D
    MOVLW   M_M             ; M ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   12D
    MOVLW   M_P             ; P ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   13D
    MOVLW   M_W             ; W ?
    SUBWF   AL, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   14D
    RETLW   15D             ; Unknown command

;=========================================
; Convert to ONES
; Valid numbers are returned with MSB set,
; Input: AH  Output:W
;=========================================
;
CONV_LO
    MOVLW   M_1             ; 1 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   01D+80H
    MOVLW   M_2             ; 2 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   02D+80H
    MOVLW   M_3             ; 3 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   03D+80H
    MOVLW   M_4             ; 4 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   04D+80H
    MOVLW   M_5             ; 5 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   05D+80H
    MOVLW   M_6             ; 6 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   06D+80H
    MOVLW   M_7             ; 7 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   07D+80H
    MOVLW   M_8             ; 8 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   08D+80H
    MOVLW   M_9             ; 9 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   09D+80H
                ; fall thru to pick up 0 or T
                ; or error

;=========================================
; Convert to TENs
; Input: AH  Output:W
;=========================================
;
```

```
CONV_HI
    MOVLW   M_T             ; T for 0 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   00D+80H
    MOVLW   M_0             ; 0 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   00D+80H
    MOVLW   M_1             ; 1 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   10D+80H
    MOVLW   M_2             ; 2 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   20D+80H
    MOVLW   M_3             ; 3 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   30D+80H
    MOVLW   M_4             ; 4 ?
    SUBWF   AH, W           ; Compare
    BTFSC   STATUS, Z       ; Skip if no compare
    RETLW   40D+80H
    RETLW   00D             ; Error for value > 49 or non-numeric

    END
```