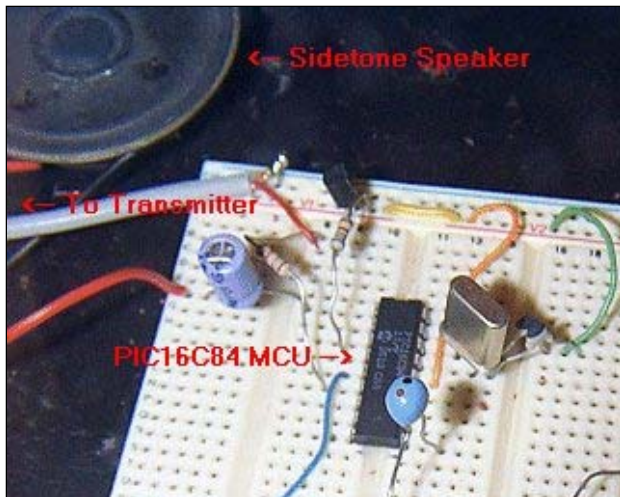


## ZL1HIT Hellschreiber / PIC Beacon



This is a simple circuit that does a fair bit of work. The core of the circuit is my favourite IC - the Microchip PIC 16F84 Microcontroller / RISC Microprocessor.

This project was inspired by my new friend Murray, ZL1BPU. Murray is very interested in H.F. band "digital" modes. He and a group of others are responsible for resurrecting the 1927 Hellschreiber visually received "fuzzy" digital mode of communication. More information about Hellschreiber can be read [here](#). I highly recommend you take a look. It's fascinating stuff to be sure.

This project produces base band Hellschreiber "digital" output which directly keys an HF radio transmitter. In my case I simply connect the output transistor to the CW Key input on my ICOM IC-706 Mk II transceiver. Since this rig is electronically keyed it works very well for this task.

Here is a sample of the Hellschreiber sent by this project:

```
>> HELLSCHREIBER BEACON VIA PIC16C84 MCU DE ZL1HIT <<
>> HELLSCHREIBER BEACON VIA PIC16C84 MCU DE ZL1HIT <<
```

Following is the PIC MPASM Source Code for Version 0.03 of the Beacon:

```
*****
; Feld Hellschreiber / CW Beacon Base Band Beacon
; Author: Bryan J. Rentoul
; Date: 10-April-1999
;
; All rights reserved
;
; Description: This project was inspired, yeh "specced" even, by Murray,
; ZL1BPU. It purpose is to send a mixture of CW and Hellschreiber via
; a simple CW transmitter based around a 3.58 MHz Xtal oscillator in the
; classic style. This is an early version and lacks any "user friendly"
; interface for loading new messages etc.
;
; This version uses the 5x5 pixel all upper case font set provided
; by ZL1BPU. (Letters 'C', 'V' an 'O' slightly modified for clarity.
;
;
*****
    include "p16c84.inc"
    include "bits.inc"          ; Bit processing Macros

;*** SET CONFIG FUSES ***

    __CONFIG _CP_OFF & _WDT_OFF & _HS_OSC

;For Assembler PORTAbility

W      EQU      0                ;For file,W
w      EQU      0                ;For file,W
F      EQU      1                ;For file,F
f      EQU      1                ;For file,F
```

```

;*****
;* REGISTER DECLARATIONS
;*****

ind      equ      0          ;0=pseudo-reg 0 for indirect (FSR)
RTCC     equ      1          ;1=Real Time Clock/Counter
TMR0     equ      1          ;1=Timer0 (same as above)
PC       equ      2          ;2=PC
STATUS   equ      3          ;3=Status Reg
INTCON   equ      0Bh
PCLATH   equ      0Ah

;* Status reg bits
      BIT      B_C,0,STATUS   ;Carry
      BIT      B_DC,1,STATUS  ;Half carry
      BIT      B_Z,2,STATUS   ;Zero
      BIT      B_PD,3,STATUS  ;Power down
      BIT      B_TO,4,STATUS  ;Timeout
      BIT      B_PA0,5,STATUS ;Page select (56/57 only)
      BIT      B_PA1,6,STATUS ;Page select (56/57 only)
      BIT      B_PA2,7,STATUS ;GP flag

;----
;* OPTION reg bits
      BIT      B_RBPU,7,1

;----
;* INTCON reg bits (defined in p16c84.inc)
      BIT      I_GIE,7,INTCON
      BIT      I_EEIE,6,INTCON
      BIT      I_T0IE,5,INTCON
      BIT      I_INTE,4,INTCON
      BIT      I_RBIE,3,INTCON
      BIT      I_T0IF,2,INTCON
      BIT      I_INTF,1,INTCON
      BIT      I_RBIF,0,INTCON

;-----

FSR      equ      4          ;4=file select reg 0-4=indirect address
fsr      equ      4          ;4=file select reg 0-4=indirect address
PORTA    equ      5          ;5=port A I/O register (4 bits)
PORTB    equ      6          ;6=port B I/O register
porta    equ      5          ;5=port A I/O register (4 bits)
portb    equ      6          ;6=port B I/O register

;-----

      BIT      RA0,0,PORTA
      BIT      RA1,1,PORTA
      BIT      RA2,2,PORTA
      BIT      RA3,3,PORTA
      BIT      RA4,4,PORTA

      BIT      RB0,0,PORTB
      BIT      RB1,1,PORTB
      BIT      RB2,2,PORTB
      BIT      RB3,3,PORTB
      BIT      RB4,4,PORTB
      BIT      RB5,5,PORTB
      BIT      RB6,6,PORTB
      BIT      RB7,7,PORTB

      BIT      PTT,0,PORTA   ; PTT Output
      BIT      SPAREA1,1,PORTA
      BIT      SIDETONE,2,PORTA
      BIT      TIMECAP,3,PORTA ; Timer capacitor output
      BIT      TIMEIN,4,PORTA ; Timer cap. input

```

```

        BIT     TXOUT,0,PORTB
        BIT     FSK0,1,PORTB
        BIT     FSK1,2,PORTB
        BIT     FSK2,3,PORTB
        BIT     DXMODE,4,PORTB ; DX Mode input. Jumper to GND for DX mode enable

;-----
SAVEW   equ     0Ch       ; Used in INT routine to save W and STATUS register
SAVES   equ     0Dh
FLAGS   equ     0Eh       ; Inputs status register
DL1     equ     0Fh       ; Delay routine counters
DL2     equ     10h       ;      "      "      "
DL3     equ     11h       ;      "      "      "
INTCNT  equ     12h       ; Interrupt call counter to count 64 calls then reset
TONECNT equ     13h       ; Tone counter using to toggle output every 8 int calls
COLCNT  equ     14h       ; Column counter
ROWCNT  equ     15h       ; Row counter
CURCOL  equ     16h       ; Current column data for rotating each bit into carry flag
CHAR    equ     17h       ; Character code to send (ASCII values from 32 to 95 OK)
TEMP    equ     18h
TEMP1   equ     19h
TIMER   equ     20h       ; We use this as a free running timer incremented
                          ; by the Int routine. It is for CW dot / dash timing etc

SAVE1   equ     21h       ; Just another TEMP variable
SAVE2   equ     22h       ;
OUTBUF  equ     23h       ; CW Output buffer

; 24h to 2B free

        ; Define some status bits (flags)
        BIT     BUSY,0,FLAGS ; 0 = Buffer awaits new character to send (1 = busy)
        BIT     F_SIDETONE,1,FLAGS
        BIT     F_TABLE,2,FLAGS ; 0 = Font Table 1, 1 = Font Table 2
        BIT     F_CW,3,FLAGS ; In CW mode (1) the Int does not toggle TXOUT

;*** Reset Vector *****

        ORG     0000
reset   GOTO    INIT

; =====
; == INTERRUPT PROCEDURE ==
; =====
        ORG     0004
TIMERINT ; Interrupt routine called (17.5 * 5 * 60) times per second providing
          ; pixel timing (5 pixels/rows per column)

        ; Save W and Status
        CLB     I_T0IE      ; Disable this interrupt
        CLB     I_T0IF      ; clear the interrupt
        movwf   SAVEW       ; save w reg in Buffer
        swapf   SAVEW, f    ; swap it
        swapf   STATUS,w    ; get status (swapped)
        movwf   SAVES       ; and save it

        movlw   18          ; Reset Counter to 18 (instead of around zero or so)
        movwf   TMR0        ; /

        ; Interrupt procedure propper starts here

        decfsz  TONECNT, f   ; Time for side tone click?
        goto    INEXT1

        movlw   4           ; Toggle the Sidetone output every 6 calls
        movwf   TONECNT     ; /
        SKBC    F_SIDETONE  ; ...but only if the F_SIDETONE flag is set

```

```

        TBIT      SIDETONE      ;

INEXT1
    decfsz  INTCNT, f          ; Time to send a dot pixel?
    goto   IEND              ; Nope...

    ;-----
    decf    TIMER, f          ; decrement out free running timer used in DELAY function
    ;-----

    movlw   60                ; Yes...
    movwf   INTCNT           ; Reset INTCNT to 60

    ; Check the BUSY FLAG. If zero do nothing
    BBC     BUSY, IEND

    movfw   ROWCNT
    BBS     B_Z, IENDCOL     ; If ROWCNT is zero then we are done

    ; Other wise send next bit

    movfw   CURCOL
    rrf     CURCOL, f        ; Rotate bit into carry flag for testing
    BBS     B_C, ITONEON

    BBS     F_CW, INEXT2     ; Dont touch TXOUT or F_SIDETONE if in CW mode
    CLB     TXOUT            ; Turn TX carrier OFF
    CLB     F_SIDETONE       ; flag side tone to OFF
    goto   INEXT2

ITONEON
    BBS     F_CW, INEXT2     ; Dont touch TXOUT or F_SIDETONE if in CW mode
    SEB     TXOUT            ; Turn TX carrier ON
    SEB     F_SIDETONE       ; flag side tone to ON

INEXT2
    decf    ROWCNT, f        ; Decrement row counter
    goto   IEND              ; Exit the int routine

IENDCOL
    CLB     BUSY             ; Clear the BUSY flag to indicate column sent

IEND
    swapf   SAVES, w         ; get status (swapped back)
    movwf   STATUS          ; and restore it
    swapf   SAVEW, w        ; restore W reg (swapped back to restore Z,C,DC flags)
    SEB     I_T0IE          ; Re-enable the interrupt
    retfie

;---- END OF INTERRUPT ROUTINE -----
;-----

;*****
;**  INIT
;**  Hardware reset entry point
;**
;*****

INIT    ;Power-on entry

;*****
;**  RESET
;**  software reset entry point
;**
;*****

RESET  ;Soft reset

```

```

;Init ports
movlw 10h      ; All Port A bits to outputs except RA4 (Timer Cap input)
tris  PORTA

movlw 0F0h    ; Initialize Port B I/O pins
movwf PORTB
movlw 0F0h    ; Port B<0-3> Output, Port B<4-7> Input
tris  PORTB

movlw b'00011001' ; 0 Turn ON Port B pullups
                    ; 0 External Int triggers on falling edge
                    ; 0 TOCS low (timer run from internal clock)
                    ; 1 TOSE - Counter timer/counter triggers on rising edge
                    ; 1 Prescaler switched OUT (to WDT) ...
                    ; 110 ...with div. 64 (WDT) or 128 (TMR0)

option ; Store W in the Option register.

CLB  BUSY          ; Clear busy flag
CLB  F_SIDETONE
CLB  F_CW          ; Start in Hell' mode

clrf  COLCNT
clrf  ROWCNT

; Set up the timer interrupt to give us the required 122.5 dots per second
; (17.5 columns per second with 7 pixel rows)
; We reset the counter to 18 each time to yield 240 counts per
; interrupt = (7.3728 / 4 / 240) = 7680 ints. per second.
; 7680 / (17.5 x 7 row pixels) = 62.69. So every 63rd call to the int. should
; send one pixel.
;
; NOTE: In practice it was found tat every 60th call produced the right timing!

movlw 6
movwf TONECNT ; Init tone counter reg. Toggle TXOUT every 6 calls to int.
movlw 60
movwf INTCNT ; Initialize the INTCNT counter reg.
SEB  I_TOIE ; enable timer overflow interrupt
SEB  I_GIE  ; global interrupt enable
                    ; The TIMERINT interrupt routine is now running

goto  MAIN_LOOP

;=====

DELAY ; Arbitrary delay routine for CW dash/dot timing or whatever

; Make a delay send one blank column
clrf  CURCOL
call  TXCOL
return

;-----

;=====
; Subroutines for various things shall go here..
;=====

TXCOL ; Sets the Int routine up to send a column - 5 rows
; (bits stretched over usual 14 rows)
; The CURCOL variable hold the column dat to send

movfw CURCOL ; Save column data (the Int routine destroys it)
movwf SAVE1

movlw 6

```

```

        movwf    ROWCNT          ; ROWCNT will process 5 rows (0 not processed)
        SEB     BUSY             ; Setting this bit signals Int routine to send a column
        LOOPBS  BUSY            ; sit in loop waiting for BUSY to clear

        movfw   SAVE1           ; Restore column data (in case we want to send it again)
        movwf   CURCOL

        return

;-----

TXCHAR  ; Send character in "W" using Hellscheiber
        movwf   TEMP
        movlw   32              ; make the char code zero based
        subwf   TEMP,f          ; and store in TEMP

        ; The FONT table is split into two parts. Characters 0 to 31 are in part 1
        ; The rest are in part 2 (FONTTAB2)

        movlw   32              ; Is the character to send less than 32?
        subwf   TEMP, w         ; Subtract W (32) from TEMP
        BBS     B_C, USETABLE2  ; Carry Flag is SET for a positive result (TEMP > 32)
USETABLE1
        movlw   HIGH FONTTAB1   ; Get the jump page right and stick
        movwf   PCLATH          ; it in PCLATH as per the book :)
        CLB     F_TABLE         ; Clear the F_TABLE flag to indicate FONTTAB1 below
        goto    SENDCHAR       ; --> go lookup and send this character

USETABLE2
        movlw   32              ; Subtract addition 32 to correct char offset for table 2
        subwf   TEMP, f         ; /
        movlw   HIGH FONTTAB2   ; Load PCLATH to correct memory page is used
        movwf   PCLATH         ; /
        SEB     F_TABLE         ; Set the F_TABLE flag to indicate FONTTAB2 below

SENDCHAR
        movfw   TEMP            ; Retrieve char offset
        addwf   TEMP, f         ; multiply by 5 to get to start of
        addwf   TEMP, f         ; char in font table
        addwf   TEMP, f         ; /
        addwf   TEMP, f         ; /

        clrf    COLCNT          ; Reset COLCNT (column counter)
TAB1LOOP
        movfw   TEMP            ; Retrieve char font table offset
        addwf   COLCNT, w       ; Add the column number
        CBC     F_TABLE, FONTTAB1 ; Call the right table based on F_TABLE flag ...
        CBS     F_TABLE, FONTTAB2 ; ...set above to get the column data
        movwf   CURCOL

        call    TXCOL           ; Flag the Interrupt routine to send this column
        SKBS   DXMODE          ; If no jumper on for DX mode then exit
        call    TXCOL           ; otherwise send the column a 2nd time

        incf    COLCNT, f       ; Increment COLCNT
        movfw   COLCNT          ; Last column (5) ?
        sublw   5
        BBC     B_Z, TAB1LOOP   ; If not then send next column

        clrf    CURCOL          ; Send one blank column
        call    TXCOL           ; /

        return

;-----
;--- CW ROUTINES START HERE -----
;-----
SENDDASH

```

```

    SEB    F_SIDETONE
    SEB    TXOUT          ; PTT on
    call   DELAY          ; Dash delay
    call   DELAY
    call   DELAY
    call   DELAY
    call   DELAY
    CLB    TXOUT          ; PTT off
    CLB    F_SIDETONE
    call   DELAY
    return

;-----
SENDDOT
    SEB    F_SIDETONE
    SEB    TXOUT          ; PTT on
    call   DELAY          ; Dot delay
    call   DELAY
    CLB    TXOUT          ; PTT off
    CLB    F_SIDETONE
    call   DELAY          ; "space between" delay
    return

;-----

SENDCW  ; Sends CW character held in W. RRF's each bit out until W = 1
        ; 00h in W means send space between words

    SEB    F_CW          ; Tell the Int routine we are in CW mode

    movwf OUTBUF
    xorlw  0              ; Trigger Z flag (maybe)
    BBS    B_Z, SSPACE   ; Its zero so just send a space

SILOOP
    movfw  OUTBUF
    sublw  1
    BBS    B_Z, SIEND    ; If just one then we are done

    CLB    B_C
    rrf    OUTBUF, f     ; Get next bit to send

    BBC    B_C, SDOT     ; Branch to DOT if Carry clear. Otherwise its a DASH
    call   SENDDASH
    goto   SILOOP        ; loop back

SDOT
    call   SENDDOT
    goto   SILOOP        ; loop back

SSPACE
    call   DELAY          ; Send word space (just wait)
    call   DELAY
    call   DELAY

SIEND
    call   DELAY          ; Inter char delay
    call   DELAY
    call   DELAY

    CLB    F_CW          ; Tat's enough CW for now thanks :)
    return

;-----

;*****
;***** M A I N   L O O P   *****
;*****

MAIN_LOOP

    ; Send the beacon string from the "T_BEACON" table
    clrf  TEMP1

```

```

BEACONL1
    movlw    HIGH T_BEACON    ; Stick the MSByte of T_BEACON in PCLATH
    movwf   PCLATH
    movwf   TEMP1            ; Retrieve the beacon character offset
    call    T_BEACON        ; Go get the character
    xorlw   0                ; To update the Z flag
    BBS     B_Z, MLEND1     ; Zero? If so we are done so jump out to MLEND1
    call    TXCHAR          ; Otherwise send this character
    incf    TEMP1, f        ; Increment to next character
    goto    BEACONL1        ; Loopy de loop

MLEND1
    call    DELAY
    call    DELAY

;-----

; Same again but this time CW
    clrf    TEMP1
BEACONL2
    movlw    HIGH T_CALLSIGN    ; Stick the MSByte of T_BEACON in PCLATH
    movwf   PCLATH
    movwf   TEMP1            ; Retrieve the beacon character offset
    call    T_CALLSIGN        ; Go get the character
    xorlw   0                ; To update the Z flag
    BBS     B_Z, MLEND2     ; Zero? If so we are done so jump out to MLEND1
    movwf   TEMP            ; Save W
    movlw   HIGH CWT         ; Get the dot and dashes from the CWT
    movwf   PCLATH          ; /
    movwf   TEMP            ; Restore W
    call    CWT              ; /
    call    SENDCW          ; Send this character
    incf    TEMP1, f        ; Increment to next character
    goto    BEACONL2        ; Loopy de loop

MLEND2
    call    DELAY
    call    DELAY

    goto    MAIN_LOOP        ; Luckily PIC MCU's do not get bored!
;-----

    ORG     0200h

FONTTAB1
    addwf   PC, f

    retlw   0                ; 32 -
    retlw   0
    retlw   0
    retlw   0
    retlw   0

    retlw   0                ; 33 - !
    retlw   0
    retlw   29
    retlw   0
    retlw   0

    retlw   0                ; 34 - "
    retlw   24
    retlw   0
    retlw   24
    retlw   0

    retlw   10               ; 35 - #
    retlw   31

```

```
retlw 10
retlw 31
retlw 10

retlw 8 ; 36 - $
retlw 21
retlw 31
retlw 21
retlw 2

retlw 25 ; 37 - %
retlw 26
retlw 4
retlw 11
retlw 19

retlw 10 ; 38 - &
retlw 21
retlw 13
retlw 2
retlw 5

retlw 0 ; 39 - '
retlw 16
retlw 24
retlw 0
retlw 0

retlw 0 ; 40 - (
retlw 14
retlw 17
retlw 0
retlw 0

retlw 0 ; 41 - )
retlw 0
retlw 17
retlw 14
retlw 0

retlw 21 ; 42 - *
retlw 14
retlw 31
retlw 14
retlw 21

retlw 4 ; 43 - +
retlw 4
retlw 31
retlw 4
retlw 4

retlw 0 ; 44 - ,
retlw 1
retlw 6
retlw 0
retlw 0

retlw 4 ; 45 - -
retlw 4
retlw 4
retlw 4
retlw 4

retlw 0 ; 46 - .
retlw 3
retlw 3
```

```
retlw 0
retlw 0

retlw 1 ; 47 - /
retlw 2
retlw 4
retlw 8
retlw 16

retlw 14 ; 48 - 0
retlw 19
retlw 21
retlw 25
retlw 14

retlw 0 ; 49 - 1
retlw 16
retlw 31
retlw 0
retlw 0

retlw 3 ; 50 - 2
retlw 21
retlw 21
retlw 21
retlw 9

retlw 0 ; 51 - 3
retlw 21
retlw 21
retlw 21
retlw 10

retlw 0 ; 52 - 4
retlw 30
retlw 2
retlw 7
retlw 2

retlw 28 ; 53 - 5
retlw 21
retlw 21
retlw 21
retlw 2

retlw 14 ; 54 - 6
retlw 21
retlw 21
retlw 21
retlw 2

retlw 16 ; 55 - 7
retlw 16
retlw 23
retlw 24
retlw 0

retlw 10 ; 56 - 8
retlw 21
retlw 21
retlw 21
retlw 10

retlw 8 ; 57 - 9
retlw 21
retlw 21
retlw 21
```

```

retlw    14

retlw    0      ; 58 - :
retlw    5
retlw    5
retlw    0
retlw    0

retlw    0      ; 59 - ;
retlw    10
retlw    11
retlw    0
retlw    0

retlw    4      ; 60 - <
retlw    10
retlw    17
retlw    0
retlw    0

retlw    0      ; 61 - =
retlw    10
retlw    10
retlw    10
retlw    0

retlw    0      ; 62 - >
retlw    0
retlw    17
retlw    10
retlw    4

retlw    8      ; 63 - ?
retlw    16
retlw    21
retlw    20
retlw    8
;-----

CWT      ; CW Table      Call with W=ASC of char to send (EG. 'A' (65) for "di'dah" )
addlw    0
SKBC     B_Z      ; Return a zero if zero received
retlw    0

movwf    SAVE1      ; Calculate offset in table from ASCII char given
movlw    48          ; [ ASCII code for "0" ]
subwf    SAVE1, w
addwf    PC, f

retlw    b'00111111' ; 0
retlw    b'00111110' ; 1
retlw    b'00111100' ; 2
retlw    b'00111000' ; 3
retlw    b'00110000' ; 4
retlw    b'00100000' ; 5
retlw    b'00100001' ; 6
retlw    b'00100011' ; 7
retlw    b'00100111' ; 8
retlw    b'00101111' ; 9
retlw    b'00000000' ; :
retlw    b'00000000' ; ;
retlw    b'00000000' ; <
retlw    b'00000000' ; =
retlw    b'00000000' ; >
retlw    b'00000000' ; ?
retlw    b'00000000' ; @
retlw    b'00000110' ; A

```

```

retlw  b'00010001' ; B
retlw  b'00010101' ; C
retlw  b'00001001' ; D
retlw  b'00000010' ; E
retlw  b'00010100' ; F
retlw  b'00001011' ; G
retlw  b'00010000' ; H
retlw  b'00000100' ; I
retlw  b'00011110' ; J
retlw  b'00001101' ; K
retlw  b'00010010' ; L
retlw  b'00000111' ; M
retlw  b'00000101' ; N
retlw  b'00001111' ; O
retlw  b'00010110' ; P
retlw  b'00011011' ; Q
retlw  b'00001010' ; R
retlw  b'00001000' ; S
retlw  b'00000011' ; T
retlw  b'00001100' ; U
retlw  b'00011000' ; V
retlw  b'00001110' ; W
retlw  b'00011001' ; X
retlw  b'00011101' ; Y
retlw  b'00010011' ; Z

```

```

;=====

```

```

; FONT TABLE 2 (second half)

```

```

ORG      0300h    ; start table at even multiple of 256 address

```

```

FONTTAB2

```

```

addwf   PC, f

```

```

retlw   14      ; 64 - @
retlw   17
retlw   21
retlw   21
retlw   8

```

```

retlw   15      ; 65 - A
retlw   20
retlw   20
retlw   20
retlw   15

```

```

retlw   31      ; 66 - B
retlw   21
retlw   21
retlw   21
retlw   10

```

```

retlw   14      ; 67 - C
retlw   17
retlw   17
retlw   17
retlw   0      ;was 10

```

```

retlw   31      ; 68 - D
retlw   17
retlw   17
retlw   17
retlw   14

```

```

retlw   31      ; 69 - E
retlw   21

```

```
retlw 21
retlw 21
retlw 17

retlw 31 ; 70 - F
retlw 20
retlw 20
retlw 20
retlw 16

retlw 14 ; 71 - G
retlw 17
retlw 17
retlw 19
retlw 10

retlw 31 ; 72 - H
retlw 4
retlw 4
retlw 4
retlw 31

retlw 0 ; 73 - I
retlw 17
retlw 31
retlw 17
retlw 0

retlw 2 ; 74 - J
retlw 17
retlw 31
retlw 16
retlw 0

retlw 31 ; 75 - K
retlw 4
retlw 10
retlw 17
retlw 0

retlw 31 ; 76 - L
retlw 1
retlw 1
retlw 1
retlw 0

retlw 31 ; 77 - M
retlw 8
retlw 4
retlw 8
retlw 31

retlw 31 ; 78 - N
retlw 8
retlw 4
retlw 2
retlw 31

; retlw 31 ; 79 - O
; retlw 17
; retlw 17
; retlw 17
; retlw 31

retlw 14 ; 79 - O
retlw 17
retlw 17
```

```
retlw 17
retlw 14

retlw 31 ; 80 - P
retlw 20
retlw 20
retlw 20
retlw 8

retlw 31 ; 81 - Q
retlw 17
retlw 21
retlw 19
retlw 15

retlw 31 ; 82 - R
retlw 20
retlw 22
retlw 21
retlw 9

retlw 8 ; 83 - S
retlw 21
retlw 21
retlw 21
retlw 2

retlw 16 ; 84 - T
retlw 16
retlw 31
retlw 16
retlw 16

retlw 31 ; 85 - U
retlw 1
retlw 1
retlw 1
retlw 31

; retlw 16 ; 86 - V
; retlw 8
; retlw 4
; retlw 2
; retlw 31

retlw 24 ; 86 - V
retlw 6
retlw 1
retlw 6
retlw 24

retlw 30 ; 87 - W
retlw 1
retlw 6
retlw 1
retlw 30

retlw 17 ; 88 - X
retlw 10
retlw 4
retlw 10
retlw 17

retlw 16 ; 89 - Y
retlw 8
retlw 7
retlw 8
```

```

retlw    16

retlw    17      ; 90 - Z
retlw    19
retlw    21
retlw    25
retlw    17

retlw    31      ; 91 - [
retlw    17
retlw    17
retlw    0
retlw    0

retlw    16      ; 92 - \
retlw    8
retlw    4
retlw    2
retlw    1

retlw    17      ; 93 - ]
retlw    17
retlw    31
retlw    0
retlw    0

retlw    0      ; 94 - ^
retlw    8
retlw    16
retlw    8
retlw    0

retlw    31      ; 95 - _
retlw    31
retlw    31
retlw    31
retlw    31

```

T\_BEACON

```

addwf    PC, f
retlw    ' '
retlw    '>'
retlw    '>'
retlw    ' '
retlw    'H'
retlw    'E'
retlw    'L'
retlw    'L'
retlw    'S'
retlw    'C'
retlw    'H'
retlw    'R'
retlw    'E'
retlw    'I'
retlw    'B'
retlw    'E'
retlw    'R'
retlw    ' '
retlw    'B'
retlw    'E'
retlw    'A'
retlw    'C'
retlw    'O'
retlw    'N'
retlw    ' '
retlw    'V'
retlw    'I'

```

```
retlw 'A'  
retlw ' '  
retlw 'P'  
retlw 'I'  
retlw 'C'  
retlw '1'  
retlw '6'  
retlw 'C'  
retlw '8'  
retlw '4'  
retlw ' '  
retlw 'M'  
retlw 'C'  
retlw 'U'  
retlw ' '  
retlw 'D'  
retlw 'E'  
retlw ' '  
retlw 'Z'  
retlw 'L'  
retlw '1'  
retlw 'H'  
retlw 'I'  
retlw 'T'  
retlw ' '  
retlw '<'  
retlw '<'  
retlw ' '  
retlw ' '  
retlw 0
```

## T\_CALLSIGN

```
addwf PC, f  
retlw 'Z'  
retlw 'L'  
retlw '1'  
retlw 'H'  
retlw 'I'  
retlw 'T'  
retlw 0
```

```
END
```