```
;=============pic based 1200 baud fsk telemetry=========
;
; (c) 1998, P.Kerckhoff
;
;
; the lower bits of port b form a simple DAC
;
; pb0 - lsb analog      (39k resistor) --/\/\/--|
; pb1 - analog          (22k resistor) --/\/\/--|
; pb2 - msb analog      (10k resistor) --/\/\/--+---||--- out
;                                   |  0.1 uf
;                                    ---
;                                    --- 0.1 uf
;                                   |
;                                   gnd
;
; the pic's internal timer is used to interrupt at regular intervals
; upon interrupt the analog output (adc) is changed in a fashion to
; simulate a sine wave (4, 6, 7, 6, 4, 1, 0, 1).  Note that a capacitor
; should be connected to the output to block the dc offset.
;
; for 1200 Hz, the counter is set to the value listed below (255->0 int)
; for 2200 Hz, the counter is set to the value listed below
;
; At 1200 baud the bit time is 833uS.  A counter is decremented in the
; interrupt routine.  Setting the counter to 8 and waiting for zero results
; in 833uS bit time for 1200 Hz.  Set the counter to 15 and wait for
; zero for 2200 Hz.
;
;
;
; AN0, AN1, AN2, and AN3 are the analog input bits (0-5v)
; Port A, bit 4 is used as an 'operate' input, 0=off, 1=on
; Port B, bits 3-7 are digital input bits
;
; The program starts sending packets of information at 1200 baud
; when RA4 goes high.  The packets are in the following form...
;
; packet count (16 bits)
; AN0 value (8 bits)
; AN1 value (8 bits)
; AN2 value (8 bits)
; AN3 value (8 bits)
; Digital value (8 bits, upper 5 bits are valid, ignore b0-b2)
; checksum (16 bits)
; CR (8 bits, $0D)
;
; Each packet consists of 10 bytes of information.  At 1200 baud
; this gives between 10 and 12 packets per second.

_CP_ON                       EQU     H'3FEF'
_CP_OFF                      EQU     H'3FFF'
_PWRTE_ON                    EQU     H'3FFF'
_PWRTE_OFF                   EQU     H'3FF7'
_WDT_ON                      EQU     H'3FFF'
_WDT_OFF                     EQU     H'3FFB'
_LP_OSC                      EQU     H'3FFC'
_XT_OSC                      EQU     H'3FFD'
_HS_OSC                      EQU     H'3FFE'
_RC_OSC                      EQU     H'3FFF'

      list  p=16c71
      radix hex
      __config  _CP_OFF & _PWRTE_ON & _WDT_OFF & _XT_OSC
      __idlocs 0x8b04


w     equ   0x00
f     equ   0x01
pc    equ   0x02
status      equ   0x03
zerof equ   0x02
caryf equ   0x00
dc    equ   0x01
porta equ   0x05
portb equ   0x06
trisa equ   0x85
trisb equ   0x86
opton equ   0x81
rp0   equ   0x05
tmr0  equ   0x01
intcon      equ   0x0B
toif  equ   0x02
adres equ   0x09
adcon0      equ   0x08
adcon1      equ   0x88

;==== timing constants, change if not using 4MHz clock=========

mrkfq equ   d'178'            ; counter freq for mark
spcfq equ   d'225'            ; counter freq for space
mrktm equ   d'8'        ; number of interrupts for mark
spctm equ   d'15'       ; number of interrupts for space


;==============================================================


digout      equ   0x03        ; bit for digital output
testop      equ   0x05        ; test / operate bit
mrkspc      equ   0x04        ; mark / space test bit

psech equ   0x2c        ; timing for pack/sec
psecl equ   0xeb        ; about 5 packets / sec
                        ; or 200mS per packet

; register use -------------------

dbyte equ   0x0c        ; byte to send in fsk
bitcnt      equ   0x0d        ; count of bits to send

fsktm equ   0x0e        ; cycle counter for fsk
fskfq equ   0x0f        ; frequency for int counter

wint  equ   0x10        ; w and status 'stack'
sint  equ   0x11
tint  equ   0x12

sinecnt     equ   0x13        ; sine wave table pos counter

; packet construction registers

cnth  equ   0x14        ; packet count, high byte
cntl  equ   0x15        ; packet count, low byte
an0   equ   0x16        ; analog input 0 results
an1   equ   0x17        ; analog input 1 results
an2   equ   0x18        ; analog input 2 results
an3   equ   0x19        ; analog input 3 results
```

```
dig   equ   0x1a        ; digital input results
chkh  equ   0x1b        ; checksum, high byte
chkl  equ   0x1c        ; checksum, low byte

temp  equ   0x1d        ; general purpose register
temp2 equ   0x1e        ; ditto

;------------------------------------------------------
      org   0x00
      goto  start       ; redirection for start of pgm

;------------------------------------------------------
      org   0x04
      goto  intrtn            ; redirection for int rtn

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

start ;
      movlw b'00001000' ; set options for...
                        ; b7 0 = pull ups enabled
                        ; b6 0 = int on falling rb0/int pin (na)
                        ; b5 0 = tmr0 clock is cycle clock
                        ; b4 0 = tmr0 inc on falling edge of ra4/tocki (na)
                        ; b3 1 = prescaler set to wdt (na)
                        ; b2-b0 000 = 1:1 rate
      bsf   status,rp0  ; bank 1
      movwf opton       ; set option into place

      movlw b'00011111' ; port a, all in
      movwf trisa       ; set port a direction

      movlw b'01111000' ; port b, 0=out 1=in
      movwf trisb       ; set port b direction

      movlw b'00000000' ; port a is an0-an3 (pa4 = digital input)
      movwf adcon1            ; setup port a

      bcf   status,rp0  ; bank 0

      movlw b'00000001' ; setup adc for...
                        ; b7,6      00 = Fosc/2 conv clock
                        ; b5  0  = n/a
                        ; b4,3      00 = chn an0
                        ; b2  0  = go/done reset
                        ; b1  0  = conv not complete
                        ; b0  1  = adc on
      movwf adcon0            ; tell the adc

      movlw mrkfq       ; setup for mark'ing time
      movwf fskfq       ; send to the int rtn's freq. byte

      bcf   intcon,toif ; clear tmr0 interrupt bit
      bsf   intcon,7    ; enable interrupts (global)
      bsf   intcon,5    ; enable tmr0 interrupts

      ; at this point the processor is interrupting at regular intervals
      ; and sending out a sine wave at a frequency is determined by
      ; the value stored in the fskfq register (in this case, a mark freq.)
      ; the adc subsystem is on, currently looking at an0

      clrf  cnth        ; clear the packet counter
      clrf  cntl

packlop     ; the packet loop
```

```
      ; wait for a high on pa4
      btfss porta,4           ; test bit 4
      goto  packlop           ; loop until high

      ; start with an0, work towards an3
      ; read the adc, saving the results
      ; then read the digital
      ; compute a checksum
      ; and transmit

      movlw b'00000001' ; an0
      movwf adcon0            ; tell the adc
      call  w12uS       ; wait 12uS for sampling

      ; start the an0 conversion
      bsf   adcon0,2    ; start the conversion
convl0      btfss adcon0,1    ; wait until conv complete
      goto  convl0
      movf  adres,w           ; get the results for an0
      movwf an0         ; and save it

      movlw b'00001001' ; an1
      movwf adcon0            ; tell the adc
      call  w12uS       ; wait 12uS for sampling

      ; start the an1 conversion
      bsf   adcon0,2    ; start the conversion
convl1      btfss adcon0,1    ; wait until conv complete
      goto  convl1
      movf  adres,w           ; get the results for an0
      movwf an1         ; and save it

      movlw b'00010001' ; an2
      movwf adcon0            ; tell the adc
      call  w12uS       ; wait 12uS for sampling

      ; start the an2 conversion
      bsf   adcon0,2    ; start the conversion
convl2      btfss adcon0,1    ; wait until conv complete
      goto  convl2
      movf  adres,w           ; get the results for an0
      movwf an2         ; and save it

      movlw b'00011001' ; an3
      movwf adcon0            ; tell the adc
      call  w12uS       ; wait 12uS for sampling

      ; start the an3 conversion
      bsf   adcon0,2    ; start the conversion
convl3      btfss adcon0,1    ; wait until conv complete
      goto  convl3
      movf  adres,w           ; get the results for an0
      movwf an3         ; and save it

      ; read port b and shift the bits down
      rrf   portb,w           ; shift once
      movwf dig         ; save it
      rrf   portb,f           ; shift again
      rrf   portb,f           ; lower 5 bits are digital
      movf  portb,w           ; get the bits
      andlw b'00001111' ; mask
      movwf dig         ; save
      ; ready to compute checksum
      movf  cnth,w            ; get count hi
      movwf chkh        ; save it
      movf  cntl,w            ; get count low
      addwf an0,w       ; add in an0
      btfsc status,caryf      ; if carry then
      incf  chkh,f            ; carry into chk high
      addwf an1,w       ; do the same for an1
      btfsc status,caryf
      incf  chkh,f
      addwf an2,w       ; and an2
      btfsc status,caryf
      incf  chkh,f
      addwf an3,w       ; and an3
      btfsc status,caryf
      incf  chkh,f
      addwf dig,w       ; and finally the digital
      btfsc status,caryf
      incf  chkh,f
      movwf chkl        ; save the low byte of checksum

;----------------------------------------------------
spacket     ; send a preformed packet of information
      movf  cnth,w            ; just get the packet bytes
      call  sascii            ; and send them as
      movf  cntl,w            ; ascii (sascii) or raw (sbyte)
      call  sascii
      movf  an0,w
      call  sascii
      movf  an1,w
      call  sascii
      movf  an2,w
      call  sascii
      movf  an3,w
      call  sascii
      movf  dig,w
      call  sascii
      movf  chkh,w
      call  sascii
      movf  chkl,w
      call  sascii
      movlw 0x0d        ; final byte is a CR
      call  sbyte

      ; done sending the packet, bump packet number
      incfsz      cntl,f            ; bump low
      goto  nohbmp
      incf  cnth,f            ; bump high if needed
nohbmp;

      ; do a timing loop for specific packets/second
      movlw psech       ; load constant (high)
      movwf temp2       ; save it
pseclh      movlw psecl       ; get the low part
      movwf temp        ; into register
pseclp      decfsz      temp,f            ; loop
      goto  pseclp            ; for timing
      decfsz      temp2,f           ; do the high byte
      goto  pseclh            ; loop til done

      ; do more packets!
      goto  packlop           ; loop forever

;----------------------------------------------------
```

```
sbyte ; send a byte (in w) of information
      movwf dbyte       ; save the type
      movlw 0x08        ; setup for eight bits of fsk
      movwf bitcnt
      call  sspace            ; send a start bit
sbylop      rrf   dbyte,f           ; get the next bit to send
      btfss status,caryf      ; if hi then
      goto  sends       ; branch to send a space
      call  smark       ; else send a low, mark
      goto  sbytec            ; and continue
sends call  sspace            ; send a space
sbytec      decfsz      bitcnt,f    ; do all 8 bits
      goto  sbylop
      call  smark       ; send a mark for stop bit
      return
;-----------------------------------------------------

sspace      ; send a space bit
      movlw spcfq       ; freq for space
      movwf fskfq       ; save it
      movlw spctm       ; number of cycles for space
      movwf fsktm
      movlw 0xff        ; cause an interrupt
      movwf tmr0        ; on ff->0 rollover
      bsf   portb,digout      ; set output bit to space
wsspace     movf  fsktm,w           ; done with interrupts yet?
      btfss status,zerof
      goto  wsspace           ; no, loop
      return                  ; yes, leave

smark ; send a mark bit
      movlw mrkfq       ; freq for space
      movwf fskfq       ; save it
      movlw mrktm       ; number of cycles for space
      movwf fsktm
      movlw 0xff        ; cause an interrupt
      movwf tmr0        ; on ff->0 rollover
      bcf   portb,digout      ; set output bit to mark
wsmark      movf  fsktm,w           ; done with interrupts yet?
      btfss status,zerof
      goto  wsmark            ; no, loop
      return                  ; yes, leave


w12uS ;
      ; delay 12 uS (or so)
      ; 2uS for call, 2uS for return, 8 uS here
      goto  $+1         ; 2uS
      goto  $+1         ; 4uS
      goto  $+1         ; 6uS
      goto  $+1         ; 8uS
      return                  ; 8+2+2=12uS


intrtn      ; interrupt routine, sends sine data out

      movwf wint        ; save the w register
      swapf status,w    ; flip nybbles stat->w
      movwf sint        ; save status register

      incf  sinecnt,f   ; bump sine pointer
      movf  sinecnt,w   ; get a table value
      call  sine
      movwf tint        ; save the new sine value
      movf  portb,w           ; get port B
      andlw b'11111000' ; mask for lower bits
      iorwf tint,w            ; merge in sine bits
      movwf portb       ; update the port

      decf  fsktm,f           ; keep count of cycle position
      movf  fskfq,w           ; get frequency value
      movwf tmr0        ; and put it into the timer
      bcf   intcon,toif ; reset timer interrupt

      swapf sint,w            ; pull status and w off of 'stack'
      movwf status
      swapf wint,f
      swapf wint,w
      retfie                  ; leave the interrupt routine

sine  andlw b'00000111' ; force to 3 bits
      addwf pc,f        ; add w to pc for lookup
      dt    4,6,7,6,4,1,0,1

sascii      ; convert the byte in W into two ascii bytes
      ; and then transmit the bytes via fsk

      movwf temp        ; save the byte
      rrf   temp,w            ; shift upper nybble down
      movwf temp2
      rrf   temp2,f
      rrf   temp2,f
      rrf   temp2,w           ; nybble in w
      andlw b'00001111' ; mask for nybble
      addlw 0x36        ; add '0' + 6 to the nybble
      btfsc status,dc   ; check for nybble carry
      addlw 0x07        ; if carry then number is a-f, so add 1 + 6
      addlw 0-6         ; finally, subtract 6 from the result
      call  sbyte       ; send the upper nybble as ascii (hex)

      movf  temp,w            ; get the original byte
      andlw b'00001111' ; mask for nybble
      addlw 0x36        ; convert to ascii (hex)
      btfsc status,dc
      addlw 0x07
      addlw 0-6
      call  sbyte

      return
      end
```