An 80 Meter Direct Conversion Receiver by Andy Palm N1KSN

Basic Receiver Kit:

Ten-Tec T-KIT No. 1056 CW-SSB Receiver. An "Any Band" NE612 direct conversion design with all parts for choice of 160-10 meter ham bands, plus variable bandpass & fine tuning controls. Covers the entire 80 meter band 3.5 to 4.0 MHz with the appropriately selected parts. $32.

Enclosure:

Ten-Tec TP-19 Enclosure. $6.45. Previously part of the Ten-Tec Enclosure kit 1000-C (no longer available). You must supply mounting hardware, connectors, speaker, etc., and drill all holes. After drying from a wash with soap and water the case can be painted with Krylon or similar paints in spray cans.

Optional Pre-Amp:

Ten-Tec T-KIT No. 1001, Universal Low-Noise Broadband DC to 1 GHz RF Preamp. Added (with a cut-out switch) to allow the use of smaller antennas with receiver. If you plan on using the receiver only with a normal amateur antenna, a pre-amp isn't needed. $11.

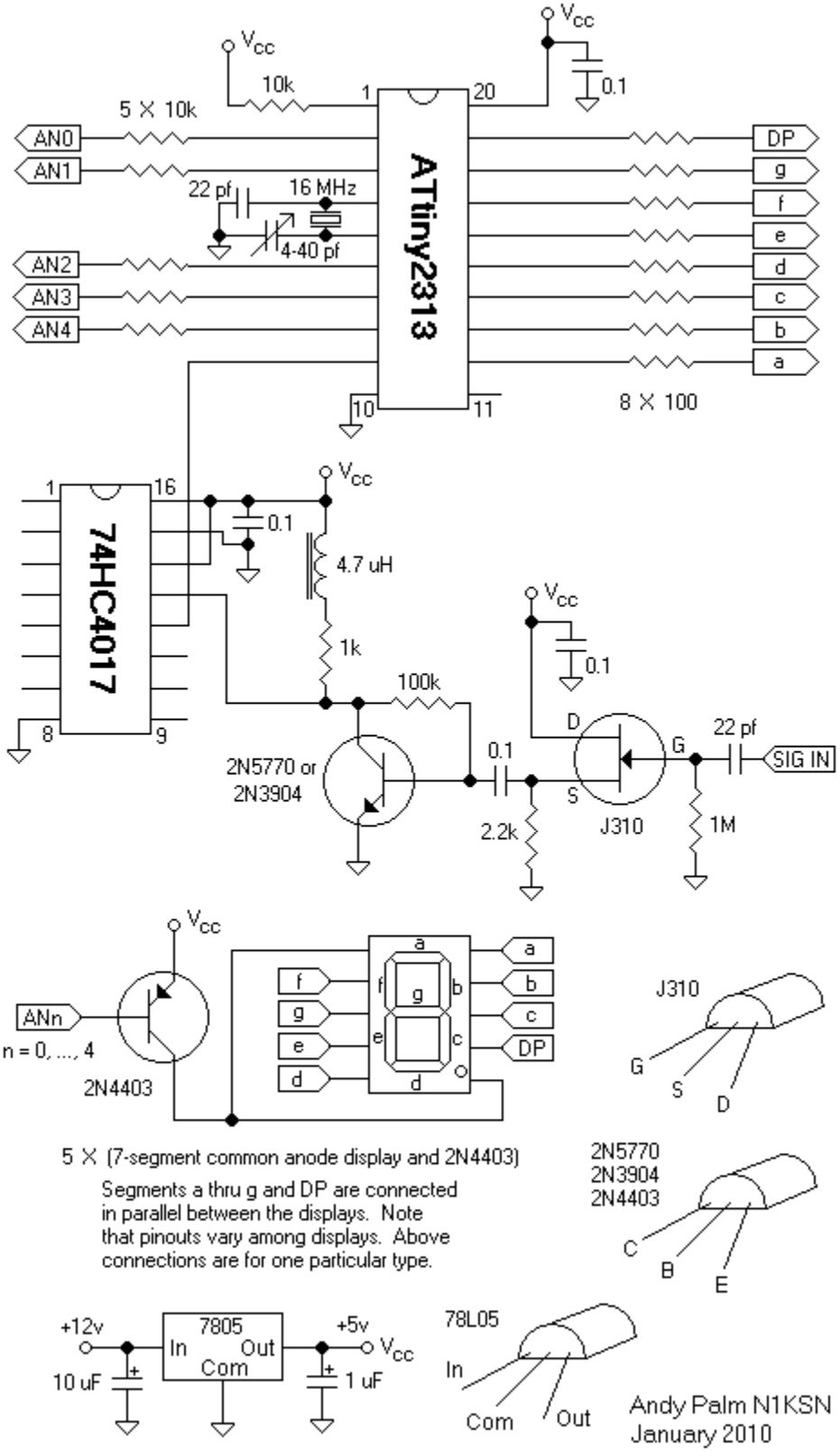
Optional Digital Frequency Readout:

Homebrew unit using Atmel AVR ATtiny2313 microcontoller, five small multiplexed seven-segment displays, and a divide-by-ten prescaler. The prescaler is based on a Steve Weber KD1JV circuit. This frequency readout unit can be replaced with the Digital Dial kit (4 digit readout) available from Hendricks QRP Kits at www.qrpkits.com (also designed by KD1JV). This kit is $30.

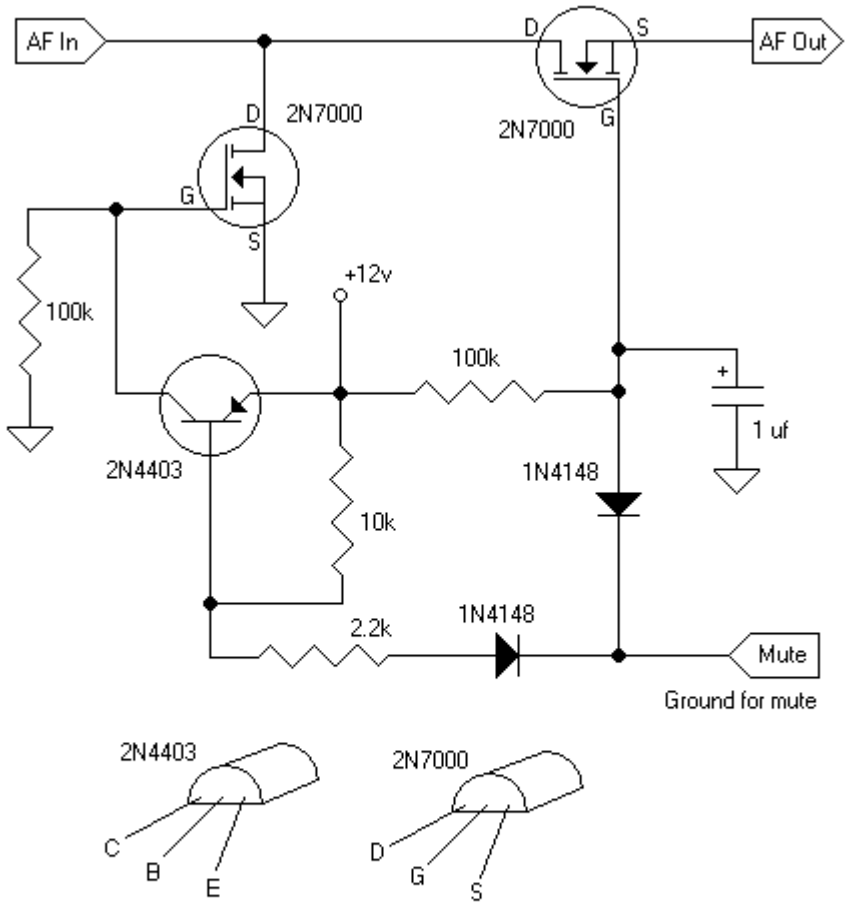
If you search the Internet with "Ten Tec 1056" you will find YouTube and other videos of this receiver in action. Unlike a regenerative receiver of comparable complexity, a direct conversion receiver is not very good for shortwave AM broadcasts due to the AM station's carrier. However, it works very well for CW and SSB reception.

This is a good beginner kit, as no toroidal inductors are used. The basic kit's printed circuit board has all pots and an audio output jack mounted on it, so the rig could be used (with care) without an enclosure. After viewing a couple of YouTube videos I decided to give the kit the "full treatment." Besides the optional modules above, I added a muting circuit and jacks for muting and sidetone inputs. This allows the receiver to be used with a transmitter in a ham station.

By replacing one of the microcontoller crystal's caps with a trim cap, I was able to tweak the chip's speed to give a frequency readout accurate to 20 Hz. I could have achieved 10 Hz accuracy, but only with a longer one second count capture interval that makes tuning to a specific frequency more difficult. A half-second capture interval works well as a compromise. Unfortunately, the display circuit results in some hum in the audio, but it is only noticeable if the RF gain is turned way down or signals are very weak.



Schematic for Frequency Counter/Display



Mute circuit from VRX-1, WA0ITP and NT7S, from QRP Quarterly, Vol. 50, No. 4, Fall 2009, pp. 22-25.

Andy Palm N1KSN
January 2010

Schematic for Frequency Counter/Display

Note that the schematic in the Ten-Tec 1056 Kit instructions does not coincide with the actual position of the T-R mute jumper on the circuit board. The position of blocking capacitor C18 on the circuit board should be modified to place the mute circuit between C18 and R18 (AF gain pot) as shown in the instruction's schematic.

```
;-----------------------------------------------------------------------
; asmFreqCounter4a.asm - Frequency counter with divide by 10 prescaler
;                        for direct conversion shortwave receiver
;
; Timer1 plus an additional byte form a 24-bit counter with Timer1
; counts externally triggered by pin T1.  The Timer1 overflow ISR is
; used to increment the upper 8 bits.
;
; This version scales the count assuming an external prescaler divides
; the signal by 10.  With a clock frequency of 16 MHz the counter
; is good for range of 0.45 to 72 MHz (before prescaling) using a
; one half second capture interval.
;
; Hardware:
;   ATtiny2313 with 16.000 MHz crystal.
;   One crystal cap is fixed 22 pf, the other is a 9 to 40 pf trim cap
;      adjusted to calibrate the counter reading.
;   Five 7-segment common anode LED displays with the segments a to g
;      and decimal point lines wired in parallel.  The display format
;      is set as X.XXX.X, the last digit being the hundreds Hz digit.
;
; AVR pin assignments:
;   PORTD0:4      - Anode lines of the five 7 segment displays,
;                   D0 = lowest digit
;   PORTD5        - T1 pin for prescaler input
;   PORTB0:7      - Segments a to g and decimal point of displays
;
; Notes:  The code handles counts of up to 8 decimal digits but
;         displays only the 5 digits needed for this application.
;         For frequencies higher than 9.9999 MHz, an additional
;         10 MHz digit can be added and the spare PORTD6 pin used for
;         its anode control (with suitable modifications to the code).
;         Due to the divide-by-ten prescaler and 0.5 second capture
;         interval, the resolution of this version is 20 Hz, so
;         the display of the 10 Hz digit would not be very useful.
;
;         If the receiver is to be used for CW only or is a superhet,
;         then additional code can be included to add or substract an
;         appropriate CW sidetone or intermediate frequency offset to
;         the displayed frequency.
;
;         If the code is used as a stand-along frequency counter, one
;         can easily change to a one second capture interval by
;         setting COUNT_IVL to 500 and eliminating the multiply by two
;         code in the main loop.
;
; Andy Palm
; 2009.12.15
;
;-----------------------------------------------------------------------
;------------------ includes, defines, equates ------------------------
.nolist
.include "tn2313def.inc"
.list

.equ  FREQ          = 16000000    ; Clock frequency in Hz

.equ  TICK_PRESCALE = 256               ; Tick timer prescaler value
.equ  TICKS_PER_S   = 500               ; Ticks per second for 2 ms
.equ  TIMER0_START  = 256 - (FREQ/(TICKS_PER_S*TICK_PRESCALE))
```

```
                                        ; Tick timer count

.equ  COUNT_IVL     = 250         ; Counter capture interval in ticks
                                  ; Equals TICKS_PER_S/2 for 0.5 sec
                                  ; count capture interval.
                                  ; Change to 500 for a 1 sec capture
                                  ; interval.

.equ  MAX_DIGIT     = 5           ; Max digit index for display
.equ  MIN_DIGIT     = 1           ; Min digit index for display

.equ  DIG_ADDR      = 2           ; Address of first digit register

; Register variables
.def  tempa         = r16
.def  tempb         = r17
.def  dig_index     = r18         ; Current digit to display
.def  timer1_msb    = r19         ; Most sig 8 bits for freq counter

.def  remain_L      = r20         ; Bottom 8 bits of division remainder
.def  remain_M      = r21         ; Middle 8 bits of division remainder
.def  remain_H      = r22         ; Top 8 bits of division remainder

.def  count_tick    = r24         ; Tick counter

.def  freq_digit_0  = r2          ; Eight digits of frequency
.def  freq_digit_1  = r3
.def  freq_digit_2  = r4
.def  freq_digit_3  = r5
.def  freq_digit_4  = r6
.def  freq_digit_5  = r7
.def  freq_digit_6  = r8
.def  freq_digit_7  = r9

.def  count_L       = r11         ; Bottom 8 bits of freq count
.def  count_M       = r12         ; Middle 8 bits of freq count
.def  count_H       = r13         ; Top 8 bits of freq count

;------------------ SRAM assignments ---------------------------------
.dseg
.org SRAM_START

;------------------ macros -------------------------------------------
.macro  Calc_Digit

; Divide 24-bit frequency count by power of ten 10^n to get digit
; for display.  Used with series of successive calls only.
;
; Method of division taken from "Electrical Engineering 101" by
; Darren Ashby, 2006, Elsevier/Newnes, p. 122
;
; Call is
;         Calc_Digit digit_reg, H, M, L
; where
;         digit_reg = register to store digit
;         H, M, L = High, mid, low bytes of divisor as constants

  clr   @0                      ; Clear registers for division by
  clr   remain_L                ; 10^n to get digit which is stored
  clr   remain_M                ; in freq_count_n
```

```
  clr   remain_H
  ldi   tempb, 24              ; Load counter with number of bits
Calc_Digit_A:
  lsl   @0
  lsl   count_L                ; Rotate dividend left into remainder
  rol   count_M
  rol   count_H
  rol   remain_L
  rol   remain_M
  rol   remain_H
  push  remain_L               ; Save copy of remainder
  push  remain_M
  push  remain_H
  subi  remain_L, @3           ; Subtract divisor 10^n from remainder
  sbci  remain_M, @2
  sbci  remain_H, @1
  brsh  Calc_Digit_B           ; Compare remainder and divisor
  pop   remain_H               ; Remainder < divisor so restore
  pop   remain_M               ; remainder and do nothing else
  pop   remain_L
  rjmp  Calc_Digit_C
Calc_Digit_B:
  pop   tempa                  ; Remainder >= divisor so discard
  pop   tempa                  ; old remainder and keep new value
  pop   tempa
  inc   @0                     ; Add one to result digit
Calc_Digit_C:
  dec   tempb
  brne  Calc_Digit_A           ; Continue through all bits of dividend
  mov   count_L, remain_L      ; Remainder is dividend in next step
  mov   count_M, remain_M
  mov   count_H, remain_H

.endmacro

;------------------ interrupt vectors --------------------------------
.cseg
.org 0x0000
  rjmp  Reset       ; Reset service
  reti              ; INT0 external interrupt
  reti              ; INT1 external interrupt
  reti              ; TIMER1 CAPT capture event
  reti              ; TIMER1 COMPA compare match A
  rjmp  Timer1_OVF  ; TIMER1 OVF overflow
  reti              ; TIMER0 OVF overflow
  reti              ; USART, RXC rx complete
  reti              ; USART, UDRE data register empty
  reti              ; USART, TXC tx complete
  reti              ; ANA_COMP analog comparator
  reti              ; PCINT pin change
  reti              ; TIMER1 COMPB compare match B
  reti              ; TIMER0 COMPA compare match A
  reti              ; TIMER0 COMPB compare match B
  reti              ; USI START USI start condition
  reti              ; USI OVERFLOW USI overflow
  reti              ; EE READY EEPROM ready
  reti              ; WDT OVERFLOW Watchdog timer overflow

;------------------ device initialization ----------------------------
Reset:
```

```
  ldi   tempa, RAMEND              ; Set up stack
  out   SPL, tempa

; Port setup
  ldi   tempa, 0xFF
  out   DDRB, tempa                ; PORTB0:7 for segs a-g and DP
  out   PORTB, tempa
  ldi   tempa, 0b00011111
  out   DDRD, tempa                ; PORTD0:4 for 5 display anodes
  out   PORTD, tempa               ; PORTD5 is set as T1 input below

; Set up TIMER0 for system tick
  ldi   tempa, TIMER0_START
  out   TCNT0, tempa
  ldi   tempa, (1<<CS02)|(0<<CS01)|(0<<CS00)
  out   TCCR0, tempa    ; Start with 256x prescaler

;------------------ main program ------------------------------------
Main:
  ldi   count_tick, COUNT_IVL      ; Counter for freq counter
  ldi   dig_index, MIN_DIGIT       ; Counter for digit to display

; Set up Timer1 as 24-bit frequency counter
  ldi   timer1_msb, 0              ; Clear high 8 bits of freq counter
  out   TCNT1H, timer1_msb         ; Clear Timer1
  out   TCNT1L, timer1_msb
  ldi   tempa, (1<<TOIE1)          ; Enable overflow interrupt
  out   TIMSK, tempa
  ldi   tempa, (1<<CS12)|(1<<CS11)|(1<<CS10)
  out   TCCR1B, tempa              ; T1 pin as clock, rising edge
  sei                              ; Global interrupt enable

Main_Loop:

  rcall Wait_for_Tick           ; Wait for system tick

; Store frequency counter value and convert to decimal digits after
; COUNT_IVL ticks have passed.
  dec   count_tick              ; Decr tick counter
  brne  Count_Done              ; Check if count interval has passed
  in    count_L, TCNT1L         ; Store lower 16 bits of counter
  in    count_M, TCNT1H
  cli                           ; Suspend interrupt
  mov   count_H, timer1_msb     ; Store upper 8 bits of counter
  ldi   tempa, 0                ; Reset 24-bit counter to zero
  out   TCNT1H, tempa
  out   TCNT1L, tempa
  ldi   timer1_msb, 0
  sei                           ; Restore interrupt

; Multiply count by 2 to compensate for 0.5 sec capture interval
; Drop these three lines if a one sec capture interval is used.
  lsl   count_L
  rol   count_M
  rol   count_H

  rcall Convert_Count           ; Calculate digits for display
  ldi   dig_index, MIN_DIGIT    ; Reset display digit counter

; ** Insert freq offset addition or subtraction code here if needed **
```

```
  ldi   count_tick, COUNT_IVL   ; Reload tick counter
Count_Done:

; Display frequency on multiplexed 7-segment displays
  rcall Disp_7seg               ; Display freq value

  rjmp  Main_Loop

;------------------ interrupt service routines ------------------------
Timer1_OVF:
  inc   timer1_msb              ; Incr upper 8 bits if Timer 1 overflow
  reti

;------------------ subroutines ---------------------------------------
Wait_for_Tick:
  in    tempa, TIFR             ; Check overflow flag for TIMER0
  sbrs  tempa, TOV0
  rjmp  Wait_for_Tick
  ldi   tempa, TIMER0_START     ; Load counter start value
  out   TCNT0, tempa
  ori   tempa, (1<<TOV0)        ; Clear flag by writing 1 to it
  out   TIFR, tempa
  ret

;---------------------------------------------------------------------
; Successively divide 24-bit frequency count by powers of ten 10^n to
; get digits for display.  Digits are stored in freq_digit_n, n=7,...,0
; as binary values.
; Due to divide-by-ten prescaler, ones digit is actually tens digit of
; external signal, and so on.
;
Convert_Count:
  Calc_Digit freq_digit_7, 0x98, 0x96, 0x80   ; Ten millions digit
  Calc_Digit freq_digit_6, 0x0F, 0x42, 0x40   ; Millions digit
  Calc_Digit freq_digit_5, 0x01, 0x86, 0xA0   ; Hundred thousands digit
  Calc_Digit freq_digit_4, 0x00, 0x27, 0x10   ; Tens of thousands digit
  Calc_Digit freq_digit_3, 0x00, 0x03, 0xE8   ; Thousands digit
  Calc_Digit freq_digit_2, 0x00, 0x00, 0x64   ; Hundreds digit
  Calc_Digit freq_digit_1, 0x00, 0x00, 0x0A   ; Tens digit
  mov   freq_digit_0, remain_L                ; Ones digit
  ret

;---------------------------------------------------------------------
; Display frequency digits on multiplexed 7-segment displays.
; The register dig_index determines which digit is displayed and
; is assumed to be between MIN_DIGIT and MAX_DIGIT, with 0 the ones
; digit.  Rotates through digits on successive calls.
;
; This version is for display of five digits.  Uses lowest five bits
; of PORTD for anode output (display unit selection) and PORTB
; for segment output.
;
Disp_7seg:

; Set active anode line low, all others high
  ldi   tempa, 1                ; Calculate 1<<(dig_index - MIN_DIGIT)
  mov   tempb, dig_index
  subi  tempb, MIN_DIGIT
  tst   tempb
```

```
  breq  PC+4
  lsl   tempa
  dec   tempb
  rjmp  PC-4
  com   tempa                   ; Invert bits for output
  out   PORTD, tempa            ; Set active anode line low

; Get segment codes for digit and set appropriate bits low
  ldi   ZL, LOW(DIG_ADDR)       ; Get digit value from register
  ldi   ZH, HIGH(DIG_ADDR)
  ldi   tempb, 0
  add   ZL, dig_index
  adc   ZH, tempb
  ld    tempa, Z                ; tempa now contains digit value
; Get display codes
  ldi   ZL, LOW(2*Seg_Code_Table)    ; Get segment display codes
  ldi   ZH, HIGH(2*Seg_Code_Table)   ; from table in prgm memory
  add   ZL, tempa               ; Add offset to base address
  adc   ZH, tempb               ;
  lpm   tempa, Z                ; tempa now contains PORTB code
; Set decimal point bit if digit index is 2 or 5
  ldi   tempb, (1<<7)
  cpi   dig_index, 2
  brne  PC+2
  add   tempa, tempb            ; Set bit 7 for DP on
  cpi   dig_index, 5
  brne  PC+2
  add   tempa, tempb            ; Set bit 7 for DP on
; Invert bits and output
  com   tempa
  out   PORTB, tempa

; Set counter to next digit to be displayed with wrap-around
  inc   dig_index
  cpi   dig_index, MAX_DIGIT + 1
  brne  PC+2
  ldi   dig_index, MIN_DIGIT

  ret

;------------------ rom constants and tables -------------------------
Seg_Code_Table:
; Seven-segment display code table for digits 0 to 9.
; Each line contains codes for port used for segment control.
; A one bit means segment is on.  Values may have to be inverted
; prior to placing in port output depending on hardware.
;     D           D
;     Pgfedcba    Pgfedcba
.db 0b00111111, 0b00000110     ; 0, 1
.db 0b01011011, 0b01001111     ; 2, 3
.db 0b01100110, 0b01101101     ; 4, 5
.db 0b01111101, 0b00000111     ; 6, 7
.db 0b01111111, 0b01101111     ; 8, 9

;------------------ eeprom -------------------------------------------
```