Homebrewing Black-Box Style

Take advantage of tried-and-true technology at a higher level. Kenwood's TM-741/742 FM-band modules make a great start for a home-brew FM transceiver project.

By Hugh Duff, VA3TO (ex-VE3OYH)

Designing and building a stable, broad-banded, digitally tuned transceiver can be quite a challenge. It takes months for teams of engineers to come up with a working prototype of a modern commercial transceiver. The days of homebrewing a rig to match the reliability and performance of these rigs are closing. Not many hams have the RF and digital expertise or equipment to accomplish this on their own. However, you can take advantage of the proven technology of readily available components by using them as building blocks. For example, the optional RF-band modules used in some multiband transceivers can be put to use in a home-brew transceiver. Harnessing the capabilities of these RF black-boxes still presents us with a challenge. The difficulty lies in providing the necessary input signals and "gluing" the additional hardware together to form a working transceiver. This article describes the findings of my experimentation with such a project and will hopefully give others a helpful head start.

Kenwood's innovative TM-741 and TM-742 modular triband FM transcievers are fine examples of modern RF-meets-digital technology in the world of Amateur Radio. Sold as dual-band 2 m/70 cm radios with space for an optional third band, these units have four other band modules available: 28 MHz, 50 MHz, 220 MHz and 1.2 GHz. Any combination of three band modules can be installed at one time. The modules perform all of the RF functions. All other functions, including power regulation and distribution, frequency control and display and audio amplification and squelch gating are performed by the main chassis. For anyone willing to do some experimenting, one of these 4×6×0.5-inch modules can be assembled into a monoband FM transceiver. With the proper considerations, a nice repeater could be assembled using two of these modules. Used modules often turn up at flea markets and on swap-net listings at pretty reasonable prices. I picked up the 2-m module from a friend who replaced it with another module in his TM-741. Just for fun, I decided to see if I could control it independently of the main chassis. By studying the service manual and schematics, and with the aid of a logic analyzer, I was able to construct a test platform that performs some of the basic functions required to get the module on the air.[1]

136 Baronwood Court
Brampton, ON L6V 3H8 Canada
e-mail: hduff@humnet.humberc.on.ca

[1]Notes appear on page17.

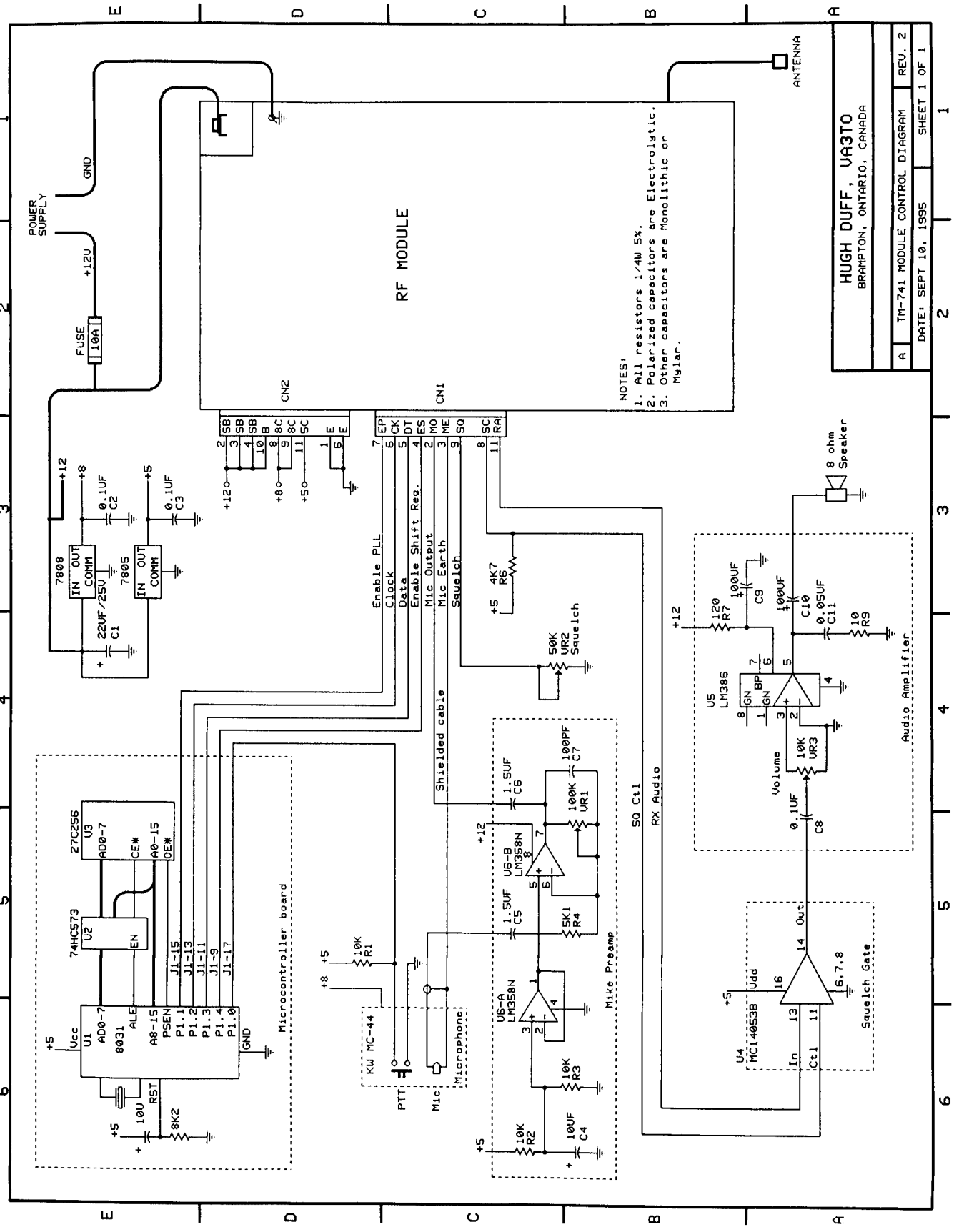


Fig 1—(left) Interconnections and additional circuitry used to interface to and control a Kenwood 2-m RF module.

The following information describes the basic hardware and programming required to use one of these modules as a stand-alone FM transceiver.

The Heart and Soul

The nucleus of the TM-741 and TM-742 main chassis is the microcontoller. It is the microcontroller that offers all of the bells and whistles found in most modern transceivers. The Kenwood RF module uses a serial shift register to control its power-level adjustment and the TX/RX switching. It also uses a serially programmed PLL unit for frequency control. The main chassis sends digital data to the module to control these functions. On my test platform, the RF module is controlled by an L. S. Electronics ELC-31, an 8031-based microcontroller.[2] An on-board EPROM was programmed to send the correct data to the module.

Four parallel I/O lines are connected to the enable shift register (ES), clock (CK), data (DT) and enable PLL (EP) lines on connector CN-1 of the module. These lines are toggled appropriately by firmware, as shown in "Programming Word Description." To send data to the shift register, 8 bits are clocked in using CK and DT, one bit at a time. Then the ES line is toggled to latch the new data into the output stage of the shift register. The PLL requires two 21-bit words. The *reference* word sets up the step size, and the *comparison* word controls the frequency. To program the PLL, the EP line must be enabled, then the first 21-bit word is clocked in one bit at a time. The EP line must be disabled and enabled again before the second 21-bit word is clocked in. Then the EP line must be disabled to terminate the transfer. The data must be sent from the microcontroller to the RF module at start-up and any time a function is altered, such as a change in frequency, or whenever PTT is pressed or released. Note the logic levels of the programming lines at rest in "Programming Word Description." The 8031 firmware (Listing 1) is well commented and may help to explain the method of programming the module. It can be downloaded from the ARRL BBS (860-594-0306) or via the Internet at http://www.arrl.org/qexfiles or ftp://ftp.arrl.org/pub/qex as file blkbox.zip.

The Bloodlines

Refer to the schematic diagram, Fig 1. This shows the hardware functions that must be provided to support the RF module. Three dc voltages are required: 12 V, 8 V and 5 V. The main external supply is 12 V. 8 V and 5 V are derived using 78xx 3-pin regulators that are supplied from the main 12 V. Ground (or earth) must be connected to pins 1 and 6 of the module, the (E) lines of connector CN-2. Ground should also be attached to the chassis of the module through the use of a heavy gauge wire with an eyelet terminal. Pins 2, 3, 4 (SB) and 10 (B) of CN-2 require 12-V dc. The exposed terminal near the back of the module should also be supplied with 12 V through a heavy gauge wire. This supplies power to the RF amplifier brick within the module. Pins 8 and 9 (8C) of connector CN-2 require 8 V. Pin 11 (5C) of connector CN-2 requires 5 V. The microcontroller will also require 5 V. The low-level receiver audio comes from the RA line of connector CN-1. This line must be gated through a 4053 CMOS analog switch IC to provide squelch muting. The analog switch is controlled by the SC line. The SQ line must be connected to the wiper and high side of a 50-kΩ potentiometer. The low side of the potentiometer should be connected to ground. This is the squelch control. Gated audio goes into a simple LM386 amplifier that drives an 8-Ω speaker.

I use a Kenwood MC-44 amplified condenser microphone that's wired into the module through an 8-pin chassis-mount connector. An LM358 op amp is used to amplify the TX audio since the mic audio alone proved to be too low. A trimmer is used to adjust for optimum transmitted audio. The output of this amplifier is connected to the MO (mic output) and ME (mic earth) pins on connector CN-1. The PTT line of the mic gets grounded on transmit. This is sensed by the microcontroller, which in turn sends the appropriate serial data to the shift register to put the module into transmit mode. The microphone used here has a built-in DTMF pad that requires an 8-V supply. The remaining lines of CN-1 and CN-2, such as the SM line (S-meter output), are unused at this time and may be left open.

It's Alive!

I encounted RF feedback once I got the module to transmit. Proper shielding and the liberal use of bypass capacitors on all dc lines is required. I fabricated a metal cover and attached it to the bottom of the module since it is normally exposed. This reduced my RFI problems significantly.

Consideration must be given to overheating since the 2-m module is capable of transmitting 50 W of power on the high setting. The die-cast housing of the module is not sufficient to dissipate the amount of heat generated by the power amplifier module. The TM-741 and TM-742 transceivers use the other installed band modules as additional heat sinking, as well as a small fan to keep the active band module cool. An appropriate heat sink should be attached to the stand-alone module, especially if running on the high power setting.

The RF performance of the project is as expected. Signal and audio reports have been good! The only problem I had was the poor quality of audio coming out of the 2-inch PC speaker that I originally used (what was I expecting?). A proper extension speaker is now used, and the audio quality is superb.

The scope of this article is to demonstrate how to calculate and load the registers to bring the module to life, so the 8031 firmware as listed performs only the most rudimentary of functions to get the module up and running on a single frequency. The next step will be to add frequency agility and display, memory channels, multiple VFOs and other bells and whistles through further programming of the microcontroller and additional hardware. Of course, a 68HC11 or PIC16Cxx, or any other microcontroller that you may be familiar with, can be used instead of the 8031—as long as the basic programming core is followed. Although the register data in this article is specific to the 2-m module, the programming concept and hardware remain the same for all of the others. Consult the service manual.

Other Tidbits

Mating header connectors are used at CN1 and CN2 to avoid soldering directly to the main PC board of the module. These were obtained from Kenwood Parts under part number E40-5452-05. Curiosity had me wondering if the PLL chips used in these modules are generic parts or custom to Kenwood. I could not determine the actual part number or manufacturer of the chip since the PLL and VCO are sealed into a functional block. The service manual refers to this block by a

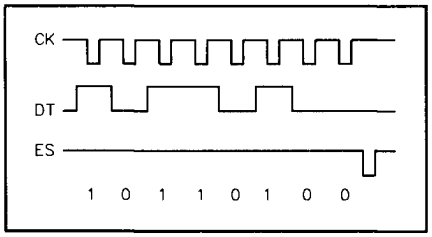
Programming Word Description

The serial *shift register* inside the module controls power level settings and TX/RX switching. A surface-mount 4094 CMOS part is used. Data is sent MSB first.

SHIFT REGISTER FORMAT
1011XXXX
^ ^
Bit 8 Bit 1

Note that this data gets complemented (inverted) inside the module before reaching the 4094 shift register.
Bit 1: TX/RX switching: 0=RX, 1=TX
Bits 2 and 3: Output power selection: 00=low, 10=medium, 01=high (bit3, bit2)
Bit 4: Special function register. Selects AM or FM mode on receive for the 2-m module: AM mode below 136 MHz=1, FM mode 136 MHz and above=0.
Bit 5: =1
Bit 6: =1
Bit 7: =0
Bit 8: =1
} These bits not used or function unknown. Set to specified levels for normal loperation.

Example: To set the module to FM receive mode with the power setting set to medium, send the following data: 10110100
Waveforms:



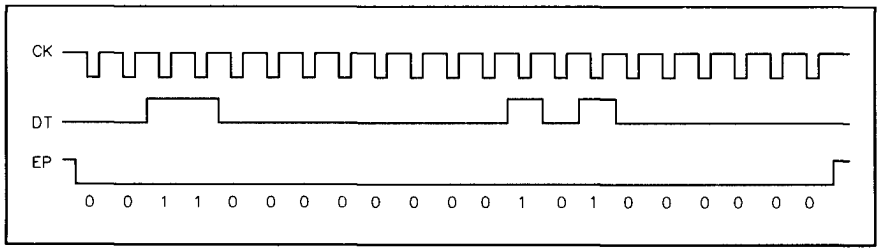
The *reference register* sets the frequency step size of the PLL. Data is sent MSB first.

REFERENCE REGISTER FORMAT
0011 XXXXXXXXXXXXXXXXX
^ ^
Bit 21 Bit 1

Bits 1-17: PLL reference oscillator frequency division ratio. Sets the step size. 17-bit word length.
Bit 18: Binary 1 identifies this stream as the reference register data.
Bit 19: =1
Bit 20: =0
Bit 21: =0
} These bits not specified or function unknown. Set to specified levels for normal operation.

The following equation determines the binary value to be loaded into the reference oscillator divider to set up the step size.

Division ratio = F_{osc}/(N <prescalar> x desired step size <kHz>)
For a 5-kHz step size:
Ref = 12800/(8 x 5)
= 320
= 00000000101000000 binary
Waveforms:



The *comparison register* comprises the A and N counters, which determine the operating frequency of the PLL. Data is sent MSB first.

COMPARISON REGISTER FORMAT
0010 XXXXXXXXXX XXXXXXX
^ ^
Bit 21 Bit 1

Bits 1-7: PLL A counter register. 7-bit word length.
Bits 8-17: PLL N counter register. 10-bit word length.
Bit 18: Binary 0 identifies this stream as the comparison register data.
Bit 19: =1
Bit 20: =0
Bit 21: =0
} These bits not used or function unknown. Set to specified levels for normal operation.

The following equations determine the binary values to be loaded into the N and A registers for a particular operating frequency (all frequencies in kHz).

For RX: $F_{vco}=F_{rx}-10700$ (IF)

For TX: $F_{vco}=F_{tx}$

N register value=$(F_{vco}/5)/128$ (whole number only)

A register value=$(F_{vco}/5)-(N \times 128)$

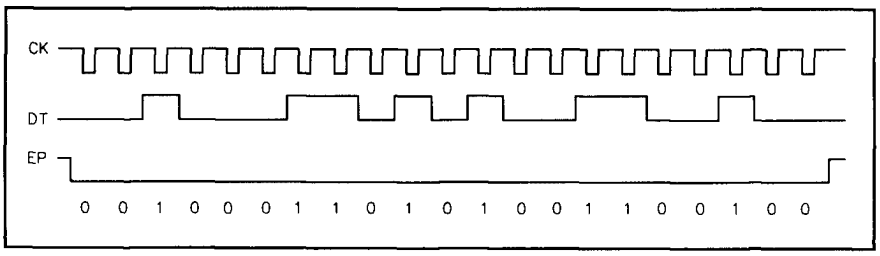
For example, to receive 146.88 MHz;

$F_{vco}=146880-10700=136180$ (kHz):

N=(136180/5)/128
= 212 (whole number only)
= 0011010100 binary

A=(136180/5)–(212 x 128)
=27236–27136=100
= 1100100 binary (first 7 bits only)

Waveforms:



Kenwood part number. However, the programming method is similar to that used on some Motorola PLL chips.[3]

An alternate use for these modules could be to use two of them as a repeater. Both modules could run off a single microcontroller and power supply. A decent repeater controller could take care of the switching and audio functions. Of course, a large heat sink would have to be fixed to the TX module to support the heavy duty cycle. This could make for an affordable alternative to buying a commercially available repeater.

The information in this article should help get you on your way to running one of the Kenwood modules as a monoband FM transceiver on any ham band from 28 MHz to 1.2 GHz. I hope it encourages some experimentation and homebrewing from a different perspective. My thanks to Francis, VE3TDL, and Steve, VE3TZC, for their assistance. For further discussion, comments or questions I can be reached via Internet (address above), packet radio: VA3TO@VA3BBS.#SON.ON.CAN.NA or by snail-mail (replies by SASE only).

Notes

[1]Familiarization with these modules through the use of the TM-741 or TM-742 service manual is recommended.

[2]L. S. Electronics, 2280 Camilla Rd, Mississauga, ON L5A 2J8 Canada, tel: 905-277-4893. LSE carries a wide range of MCS-51-based microcontroller products, including a microcontroller beginners book and application notes.

[3]For a greater understanding of phase-locked-loop circuity, I suggest reading the "Radio Frequency Oscillators and Synthesizers" section in a late edition of *The ARRL Handbook*.

Listing 1

```
; Programs the PLL and Shift registers of a Kenwood TM-741/742 2 meter
; module to control the frequency and TX/RX control.
; This program runs on an L.S.Electronics ELC-31 8031 microcontroller.
; Clock xtal = 11.0592 MHz.
; Assembled using the PseudoSam(TM) 51 Assembler V1.6.02
;
; By Hugh Duff VA3TO April 1995 Brampton, Ontario, Canada
;
;*************************************************************
; Port pin definitions

      .equ   PTT,p1.0 ; Push-To-Talk input...Lo=TX, Hi=RX
      .equ   EP,p1.1  ; PLL enable output
      .equ   CK,p1.2  ; Clock output
      .equ   DT,p1.3  ; Data output
      .equ   ES,p1.4  ; Shift Register enable output

;*************************************************************
; Start of program..micro initialization
; PSW (Program Status Word) BANK 0 contains the RX data, PSW BANK 1
; contains the TX data.
; R7 to R5 contain the Reference register word
; R4 to R2 contain the Comparsion register word
; Both words are 21 bits in length. The top 3 bits of R7 and R4 are
; unused. These get purged at the "bit_5" subroutine.

INIT:  mov   SP,#0x2F ; setup stack
       clr   PSW.3    ; set PSW to register to point at bank 0
       clr   PSW.4    ;          "

; Load PSW BANK0 with RX step,frequency and keying data
       mov   R7,#0x06 ; REF data for 5kHz step,   1st byte
       mov   R6,#0x01 ;     "           2nd byte
       mov   R5,#0x40 ;     "           3rd byte
       mov   R4,#0x04 ; COMP data for 146.88 MHz RX, 1st byte
       mov   R3,#0x6A ;     "           2nd byte
       mov   R2,#0x64 ;     "           3rd byte
       mov   R1,#0xB4 ; Shift Reg. data for RX

; Load PSW BANK1 with TX step,frequency and keying data
       setb  PSW.3    ; select TX bank in PSW
       mov   R7,#0x06 ; REF data for 5 kHz step,   1st byte
       mov   R6,#0x01 ;     "           2nd byte
       mov   R5,#0x40 ;     "           3rd byte
       mov   R4,#0x08 ; COMP data for 146.28 MHz TX, 1st byte
       mov   R3,#0x72 ;     "           2nd byte
       mov   R2,#0x48 ;     "           3rd byte
       mov   R1,#0xB5 ; Shift Reg. data for TX

; Set module control lines to appropriate resting levels
       setb  EP       ; PLL enable line normally HI
       setb  CK       ; PLL clock line normally HI
       clr   DT       ; PLL data line normally LO
       setb  ES       ; Shift Register enable normally HI
```

```
;*****************************************************************
; Main routine loads receive data,waits for a PTT then loads
; transmit data and waits for PTT to drop...

MAIN:   clr   PSW.3    ; select RX bank in PSW
        acall LD_SHFT  ; load RX shift register data
        acall LD_REF   ; load RX Reference data into PLL
        acall LD_COMP  ; load RX Comparison data into PLL
RX_IDLE:jb    PTT,*    ; wait for PTT to ground
        acall TX       ; go load TX data and return
TX_IDLE:jnb   PTT,*    ; wait for PTT release
        acall RX       ; go load RX data and return
        ajmp  RX_IDLE  ; repeat this loop

;*****************************************************************
; Selects receive databank and calls load routines

RX:     clr   PSW.3    ; select receive data bank
        acall LD_COMP  ; load com.freq. data to PLL
        acall LD_SHFT  ; disable TX in shift reg.
        ret            ; return

;*****************************************************************
; Selects transmit databank and calls load routines

TX:     setb  PSW.3    ; select transmit data bank
        acall LD_COMP  ; load comp.freq. data to PLL
        acall LD_SHFT  ; enable TX in shift reg.
        ret            ; return

;*****************************************************************
; Loads PLL Reference register

LD_REF: clr   EP       ; enable PLL
        acall DELAY    ; wait
        mov   A,R7     ; get 1st PLL REF data byte
        acall bit_5    ; send it out
        mov   A,R6     ; get 2nd PLL REF data byte
        acall bit_8    ; send it out
        mov   A,R5     ; get 3rd PLL REF data byte
        acall bit_8    ; send it out
        acall DELAY    ; wait
        setb  EP       ; disable PLL
        ret            ;

;*****************************************************************
; Loads PLL Comparison register

LD_COMP:clr   EP       ; enable PLL
        acall DELAY    ; wait
        mov   A,R4     ; get 1st PLL COMP data byte
        acall bit_5    ; send it out
        mov   A,R3     ; get 2nd PLL COMP data byte
        acall bit_8    ; send it out
        mov   A,R2     ; get 3rd PLL COMP data byte
        acall bit_8    ; send it out
```

```
      acall  DELAY    ; wait
       setb  EP       ; disable PLL
       ret           ;

;*************************************************************
; Loads Shift Register

LD_SHFT:mov  A,R1     ; get data
       acall bit_8    ; send it out
       clr   ES       ; strobe data into shift register
       acall DELAY    ; wait
       setb  ES       ; return ES to rest level
       ret           ;

;*************************************************************
; Rotates 8 bits of data in reg.A out to DATA line

BIT_8: mov   B,#0X08 ; setup reg.B as an 8 bit counter
LOOP8: rlc   A       ; rotate left 1 bit thru carry
       mov   DT,C    ; move carry status to DATA port line
CLK8:  acall CLOCK   ; clock the data
       djnz  B,LOOP8 ; keep looping until 8 bits sent
       ret           ;

;*************************************************************
; Rotates 5 bits of data in reg.A out to DATA line
; The top 3 bits of R7 and R4 are not sent since the registers
; are only 21 bits in length.

BIT_5: mov   B,#0X05 ; setup reg.B as a 5 bit counter
       rlc   A       ; waste unused MSB
       rlc   A       ;     "
       rlc   A       ;     "
LOOP5: rlc   A       ; rotate left 1 bit thru carry
       mov   DT,C    ; move carry status to DATA port line
CLK5:  acall CLOCK   ; clock the data
       djnz  B,LOOP5 ; keep looping until 5 bits sent
       ret           ;

;*************************************************************
; Toggles the clock line

CLOCK: clr   CK       ; Clock line LO
       acall DELAY    ; wait
       setb  CK       ; Clock line HI
       acall DELAY    ; wait
       ret           ;

;*************************************************************
; Delay 200 us based on 11.0592 MHz xtal

DELAY: mov   R0,#0X5C; load count
DLY1:  djnz  R0,DLY1 ; kill time
       ret

       .end
```