RF Monolithics, Inc.

TRC101/102

FHSS

By

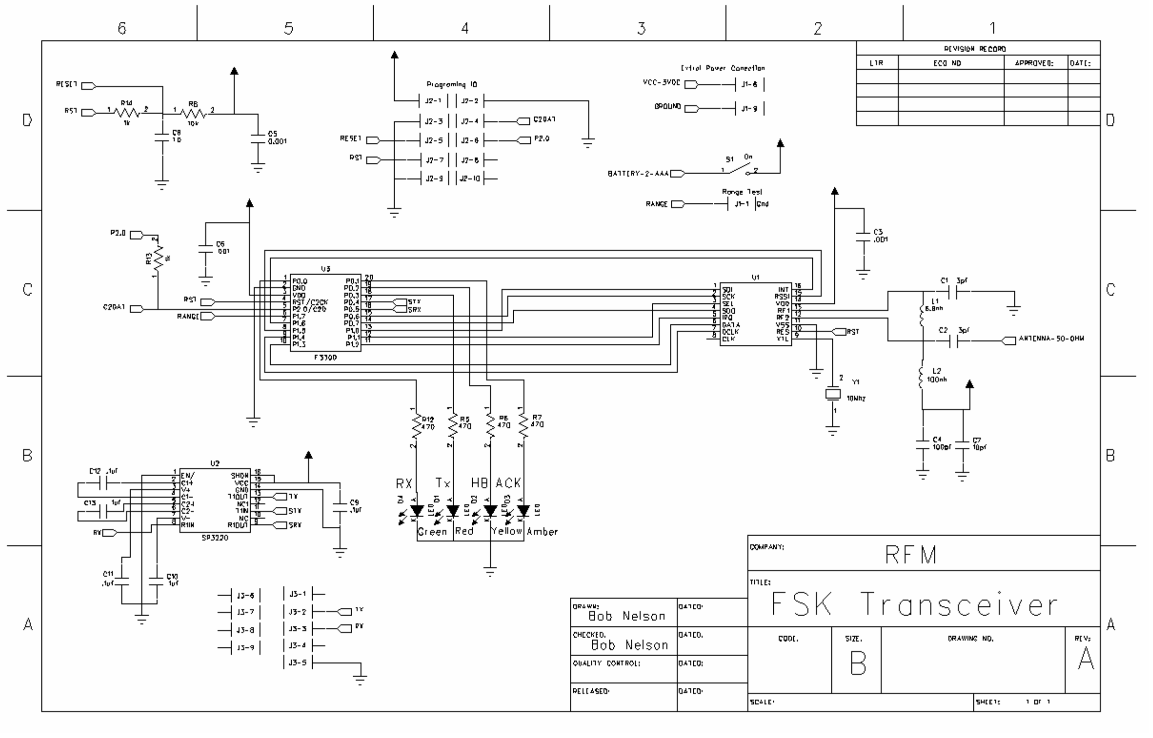
Bob Nelson

2/7/2007

Scope:

This application will demonstrate the ease of implementing a FHSS radio when using the TRC101/2. This document will give you the schematic and firmware to implement the design. Keep in mind that the TRC101/2 will meet or exceed FCC 15.249 power requirements and this application note is intended to use the higher power out while the radio is being used has a FHSS mode meeting FCC 15.247 rules and regulations.

Schematic:



Theory:

According to FCC 15.247 rules, the transmitter can not dwell on any giver channel longer than 400ms. Also the FCC requires you to hop to at lease 25 channels during your operation.

The firmware below uses the TRC101/102 RFIC to accomplish this with ease do to the fast channel switching time within the transceiver (13us or faster). The firmware is sending a short range test packet which you can modify to add your data. During operation the channel that the transmitter is going to switch to will follow the packet number being sent within the packet protocol to make the synchronization simple and fast.

The micro that is used in this demo is the Silabs 330F utilizing the internal clock oscillator.

Firmware:

```
A51 MACRO ASSEMBLER  TRC101/102FH                                                     01/01/2007 07:24:25 PAGE     1


MACRO ASSEMBLER A51 V7.04a
OBJECT MODULE PLACED IN CSL_330-trc101fh.OBJ
ASSEMBLER INVOKED BY: C:\SiLabs\MCU\IDEfiles\C51\BIN\a51.exe CSL_330-trc101fh.asm XR GEN DB EP NOMOD51

LOC  OBJ            LINE     SOURCE

                             ;*****************************************************************************
                             ;        TRC101 FSK FHSS 25 Channel Range Test
                             ;  Running in Analog Mode
                             ;  Using RFM Symbols(although its not needed)
                             ;  RS232 running at 19.2
                             ;  RF Data Rate = 22.6kbs raw
                             ;  Reciever Bandwidth = 200khz
                             ;  Trtansmitter Dev. is 105Khz
                             ;  Range test data = "FHSS Test" plus pre-amble and CRC16 at the end
                             ;
                             ;*****************************************************************************
                             ;
                             ;  CSL_330-trc101fh.ASM
                             ;
                             ;  Range Test Enabled when Jumper is installed
                             ;        Jumper is only read at power up
                             ;
                             ;  D1 LED = RX      (White)
                             ;  D2 LED = TX      (RED)
                             ;  D3 LED = HB                  (Amber)
                             ;  D4 LED = TX ACK  (Yellow)
                             ;
                             ;  Experimental software - NO representation is
                             ;  made that this software is suitable for any purpose
                             ;  Copyright(c) 2000 - 2007, RF Monolithics, Inc.
                             ;  SiLabs 8051C330 assembler source code file
                             ;  Low signal-to-noise protocol for RFM ASH transceiver
                             ;  type byte added, host & RF commands added
                             ;

                             ;  This software provides physical
                             ;  layer control of a 2nd generation
                             ;  ASH transceiver, an OSI data link layer communication utility, an
                             ;  interface to a host computer through a UART port, and examples of
                             ;  command processing for RF network related messages and application
                             ;  related messages.

                             ;  The software architecture includes two major components, a main
                             ;  loop which calls subroutines based on state flags, and an
                             ;  interrupt service routine (ISR) that is called by a timer.

                             ;  The ISR handles real time processes such as RF transmission bit
                             ;  stream generation, clock and data recovery from a received RF bit
                             ;  stream, packet start vector detection, symbol framing of received
                             ;  message bits, running timers such as transmit retry timers, etc.
                             ;  The ISR tick rate is three times the bit rate for RF transmissions
                             ;  in this code version. (Four to eight ticks per bit are used in other
                             ;  versions.)

                             ;  The main loop calls subroutines that load the transmit buffer, generate
                             ;  FCS error detection bits, initiate a message transmission, load the
                             ;  receive buffer, test FCS bits, generate ACKs, check packet addressees,
                             ;  service the hardware UART that communicates with the host, process
                             ;  commands received from the host or RF link, etc.

                             ;  The subroutine called from the main loop that feeds symbols to the ISR
                             ;  for transmission and the subroutine that gets received symbols from
                             ;  the ISR are tightly synchronized using flags set by the ISR. Also, the
```

```
                      59     ;  timeout on messages received from the host (UART) are synchronized by
                      60     ;  the ISR. Other main loop subroutines do not require tight synchronization
                      61     ;  to the ISR.
                      62     
                      63     ;  The ISR always runs and can interrupt the main loop and any subroutine
                      64     
                      65     ;  the main loop calls. From the main loop’s point of view, the ISR can
                      66     ;  change the value of certain flags and certain byte buffers at any time.
                      67     ;  The main loop and all related subroutines must be developed with this
                      68     ;  in mind.
                      69     
                      70     ;  The tick ISR is never interrupted by any other process. However,
                      71     ;  parts of the ISR can be activated or idled by state flags set/reset
                      72     ;  by main loop subroutines. Also, the ISR shares data with main loop
                      73     ;  using several common buffers.
                      74     
                      75     ;$include (c8051f330.inc) ; SiLabs 8051C330 include file
                 +1   76     ;---------------------------------------------------------------------------
                 +1   77     ;
                 +1   78     ;
                 +1   79     ;
                 +1   80     ;
                 +1   81     ; FILE NAME: C8051F330.INC
                 +1   82     ; TARGET MCUs: C8051F330, F331
                 +1   83     ; DESCRIPTION: Register/bit definitions for the C8051F330 product family.
                 +1   84     ;
                 +1   85     ; REVISION 1.0
                 +1   86     ;
                 +1   87     ;---------------------------------------------------------------------------
                 +1   88     
                 +1   89     ;REGISTER DEFINITIONS
                 +1   90     ;
  0080           +1   91     P0             DATA 080H    ; PORT 0 LATCH
  0081           +1   92     SP             DATA 081H    ; STACK POINTER
  0082           +1   93     DPL            DATA 082H    ; DATA POINTER LOW
  0083           +1   94     DPH            DATA 083H    ; DATA POINTER HIGH
  0087           +1   95     PCON           DATA 087H    ; POWER CONTROL
  0088           +1   96     TCON           DATA 088H    ; TIMER/COUNTER CONTROL
  0089           +1   97     TMOD           DATA 089H    ; TIMER/COUNTER MODE
  008A           +1   98     TL0            DATA 08AH    ; TIMER/COUNTER 0 LOW
  008B           +1   99     TL1            DATA 08BH    ; TIMER/COUNTER 1 LOW
  008C           +1  100     TH0            DATA 08CH    ; TIMER/COUNTER 0 HIGH
  008D           +1  101     TH1            DATA 08DH    ; TIMER/COUNTER 1 HIGH
  008E           +1  102     CKCON          DATA 08EH    ; CLOCK CONTROL
  008F           +1  103     PSCTL          DATA 08FH    ; PROGRAM STORE R/W CONTROL
  0090           +1  104     P1             DATA 090H    ; PORT 1 LATCH
  0091           +1  105     TMR3CN         DATA 091H    ; TIMER/COUNTER 3 CONTROL
  0092           +1  106     TMR3RLL        DATA 092H    ; TIMER/COUNTER 3 RELOAD LOW
  0093           +1  107     TMR3RLH        DATA 093H    ; TIMER/COUNTER 3 RELOAD HIGH
  0094           +1  108     TMR3L          DATA 094H    ; TIMER/COUNTER 3 LOW
  0095           +1  109     TMR3H          DATA 095H    ; TIMER/COUNTER 3 HIGH
  0096           +1  110     IDA0L          DATA 096H    ; CURRENT MODE DAC0 LOW
  0097           +1  111     IDA0H          DATA 097H    ; CURRENT MODE DAC0 HIGH
  0098           +1  112     SCON0          DATA 098H    ; UART0 CONTROL
  0099           +1  113     SBUF0          DATA 099H    ; UART0 DATA BUFFER
  009B           +1  114     CPT0CN         DATA 09BH    ; COMPARATOR0 CONTROL
  009D           +1  115     CPT0MD         DATA 09DH    ; COMPARATOR0 MODE SELECTION
  009F           +1  116     CPT0MX         DATA 09FH    ; COMPARATOR0 MUX SELECTION
  00A0           +1  117     P2             DATA 0A0H    ; PORT 2 LATCH
  00A1           +1  118     SPI0CFG        DATA 0A1H    ; SPI CONFIGURATION
  00A2           +1  119     SPI0CKR        DATA 0A2H    ; SPI CLOCK RATE CONTROL
  00A3           +1  120     SPI0DAT        DATA 0A3H    ; SPI DATA
  00A4           +1  121     P0MDOUT        DATA 0A4H    ; PORT 0 OUTPUT MODE CONFIGURATION
  00A5           +1  122     P1MDOUT        DATA 0A5H    ; PORT 1 OUTPUT MODE CONFIGURATION
  00A6           +1  123     P2MDOUT        DATA 0A6H    ; PORT 2 OUTPUT MODE CONFIGURATION
  00A8           +1  124     IE             DATA 0A8H    ; INTERRUPT ENABLE
  00A9          +1   125     CLKSEL         DATA 0A9H    ; CLOCK SELECT
  00AA          +1   126     EMI0CN         DATA 0AAH    ; EXTERNAL MEMORY INTERFACE CONTROL
  00B1          +1   127     OSCXCN         DATA 0B1H    ; EXTERNAL OSCILLATOR CONTROL
  00B2          +1   128     OSCICN         DATA 0B2H    ; INTERNAL OSCILLATOR CONTROL
  00B3          +1   129     OSCICL         DATA 0B3H    ; INTERNAL OSCILLATOR CALIBRATION
  00B6          +1   130     FLSCL          DATA 0B6H    ; FLASH SCALE
  00B7          +1   131     FLKEY          DATA 0B7H    ; FLASH LOCK AND KEY
  00B8          +1   132     IP             DATA 0B8H    ; INTERRUPT PRIORITY
  00B9          +1   133     IDA0CN         DATA 0B9H    ; CURRENT MODE DAC0 CONTROL
  00BA          +1   134     AMX0N          DATA 0BAH    ; AMUX0 NEGATIVE CHANNEL SELECT
  00BB          +1   135     AMX0P          DATA 0BBH    ; AMUX0 POSITIVE CHANNEL SELECT
  00BC          +1   136     ADC0CF         DATA 0BCH    ; ADC0 CONFIGURATION
  00BD          +1   137     ADC0L          DATA 0BDH    ; ADC0 LOW
  00BE          +1   138     ADC0H          DATA 0BEH    ; ADC0 HIGH
  00C0          +1   139     SMB0CN         DATA 0C0H    ; SMBUS CONTROL
  00C1          +1   140     SMB0CF         DATA 0C1H    ; SMBUS CONFIGURATION
  00C2          +1   141     SMB0DAT        DATA 0C2H    ; SMBUS DATA
  00C3          +1   142     ADC0GTL        DATA 0C3H    ; ADC0 GREATER-THAN COMPARE LOW
  00C4          +1   143     ADC0GTH        DATA 0C4H    ; ADC0 GREATER-THAN COMPARE HIGH
  00C5          +1   144     ADC0LTL        DATA 0C5H    ; ADC0 LESS-THAN COMPARE WORD LOW
  00C6          +1   145     ADC0LTH        DATA 0C6H    ; ADC0 LESS-THAN COMPARE WORD HIGH
  00C8          +1   146     TMR2CN         DATA 0C8H    ; TIMER/COUNTER 2 CONTROL
  00CA          +1   147     TMR2RLL        DATA 0CAH    ; TIMER/COUNTER 2 RELOAD LOW
  00CB          +1   148     TMR2RLH        DATA 0CBH    ; TIMER/COUNTER 2 RELOAD HIGH
  00CC          +1   149     TMR2L          DATA 0CCH    ; TIMER/COUNTER 2 LOW
  00CD          +1   150     TMR2H          DATA 0CDH    ; TIMER/COUNTER 2 HIGH
  00D0          +1   151     PSW            DATA 0D0H    ; PROGRAM STATUS WORD
  00D1          +1   152     REF0CN         DATA 0D1H    ; VOLTAGE REFERENCE CONTROL
  00D4          +1   153     P0SKIP         DATA 0D4H    ; PORT 0 SKIP
  00D5          +1   154     P1SKIP         DATA 0D5H    ; PORT 1 SKIP
  00D8          +1   155     PCA0CN         DATA 0D8H    ; PCA CONTROL
  00D9          +1   156     PCA0MD         DATA 0D9H    ; PCA MODE
  00DA          +1   157     PCA0CPM0       DATA 0DAH    ; PCA MODULE 0 MODE REGISTER
  00DB          +1   158     PCA0CPM1       DATA 0DBH    ; PCA MODULE 1 MODE REGISTER
  00DC          +1   159     PCA0CPM2       DATA 0DCH    ; PCA MODULE 2 MODE REGISTER
  00E0          +1   160     ACC            DATA 0E0H    ; ACCUMULATOR
  00E1          +1   161     XBR0           DATA 0E1H    ; PORT I/O CROSSBAR CONTROL 0
  00E2          +1   162     XBR1           DATA 0E2H    ; PORT I/O CROSSBAR CONTROL 1
  00E3          +1   163     OSCLCN         DATA 0E3H    ; LOW-FREQUENCY OSCILLATOR CONTROL
  00E4          +1   164     IT01CF         DATA 0E4H    ; INT0/INT1 CONFIGURATION
  00E6          +1   165     EIE1           DATA 0E6H    ; EXTENDED INTERRUPT ENABLE 1
  00E8          +1   166     ADC0CN         DATA 0E8H    ; ADC0 CONTROL
  00E9          +1   167     PCA0CPL1       DATA 0E9H    ; PCA CAPTURE 1 LOW
  00EA          +1   168     PCA0CPH1       DATA 0EAH    ; PCA CAPTURE 1 HIGH
  00EB          +1   169     PCA0CPL2       DATA 0EBH    ; PCA CAPTURE 2 LOW
  00EC          +1   170     PCA0CPH2       DATA 0ECH    ; PCA CAPTURE 2 HIGH
  00EF          +1   171     RSTSRC         DATA 0EFH    ; RESET SOURCE CONFIGURATION/STATUS
  00F0          +1   172     B              DATA 0F0H    ; B REGISTER
  00F1          +1   173     P0MDIN         DATA 0F1H    ; PORT 0 INPUT MODE CONFIGURATION
  00F2          +1   174     P1MDIN         DATA 0F2H    ; PORT 1 INPUT MODE CONFIGURATION
  00F6          +1   175     EIP1           DATA 0F6H    ; EXTENDED INTERRUPT PRIORITY 1
  00F8          +1   176     SPI0CN         DATA 0F8H    ; SPI CONTROL
  00F9          +1   177     PCA0L          DATA 0F9H    ; PCA COUNTER LOW
  00FA          +1   178     PCA0H          DATA 0FAH    ; PCA COUNTER HIGH
  00FB          +1   179     PCA0CPL0       DATA 0FBH    ; PCA CAPTURE 0 LOW
  00FC          +1   180     PCA0CPH0       DATA 0FCH    ; PCA CAPTURE 0 HIGH
  00FF          +1   181     VDM0CN         DATA 0FFH    ; VDD MONITOR CONTROL
                +1   182
                +1   183     ;
                +1   184     ;-----------------------------------------------------------------------------
                +1   185     ;BIT DEFINITIONS
                +1   186     ;
                +1   187     ; TCON 088H
  008F          +1   188     TF1            BIT 08FH     ; TIMER 1 OVERFLOW FLAG
  008E          +1   189     TR1            BIT 08EH     ; TIMER 1 ON/OFF CONTROL
  008D          +1   190     TF0            BIT 08DH     ; TIMER 0 OVERFLOW FLAG
  008C          +1   191     TR0            BIT 08CH     ; TIMER 0 ON/OFF CONTROL
  008B          +1   192     IE1            BIT 08BH     ; EXT. INTERRUPT 1 EDGE FLAG
  008A          +1   193     IT1            BIT 08AH     ; EXT. INTERRUPT 1 TYPE
  0089          +1   194     IE0            BIT 089H     ; EXT. INTERRUPT 0 EDGE FLAG
  0088          +1   195     IT0            BIT 088H     ; EXT. INTERRUPT 0 TYPE
                +1   196
                +1   197     ; SCON0 098H
  009F          +1   198     S0MODE         BIT 09FH     ; UART 0 MODE
  009D          +1   199     MCE0           BIT 09DH     ; UART 0 MCE
  009C          +1   200     REN0           BIT 09CH     ; UART 0 RX ENABLE
  009B          +1   201     TB80           BIT 09BH     ; UART 0 TX BIT 8
  009A          +1   202     RB80           BIT 09AH     ; UART 0 RX BIT 8
  0099          +1   203     TI0            BIT 099H     ; UART 0 TX INTERRUPT FLAG
  0098          +1   204     RI0            BIT 098H     ; UART 0 RX INTERRUPT FLAG
                +1   205
                +1   206     ; IE 0A8H
  00AF          +1   207     EA             BIT 0AFH     ; GLOBAL INTERRUPT ENABLE
  00AE          +1   208     ESPI0          BIT 0AEH     ; SPI0 INTERRUPT ENABLE
  00AD          +1   209     ET2            BIT 0ADH     ; TIMER 2 INTERRUPT ENABLE
  00AC          +1   210     ES0            BIT 0ACH     ; UART0 INTERRUPT ENABLE
  00AB          +1   211     ET1            BIT 0ABH     ; TIMER 1 INTERRUPT ENABLE
  00AA          +1   212     EX1            BIT 0AAH     ; EXTERNAL INTERRUPT 1 ENABLE
  00A9          +1   213     ET0            BIT 0A9H     ; TIMER 0 INTERRUPT ENABLE
  00A8          +1   214     EX0            BIT 0A8H     ; EXTERNAL INTERRUPT 0 ENABLE
                +1   215
                +1   216     ; IP 0B8H
  00BE          +1   217     PSPI0          BIT 0BEH     ; SPI0 PRIORITY
  00BD          +1   218     PT2            BIT 0BDH     ; TIMER 2 PRIORITY
  00BC          +1   219     PS0            BIT 0BCH     ; UART0 PRIORITY
  00BB          +1   220     PT1            BIT 0BBH     ; TIMER 1 PRIORITY
  00BA          +1   221     PX1            BIT 0BAH     ; EXTERNAL INTERRUPT 1 PRIORITY
  00B9          +1   222     PT0            BIT 0B9H     ; TIMER 0 PRIORITY
  00B8          +1   223     PX0            BIT 0B8H     ; EXTERNAL INTERRUPT 0 PRIORITY
                +1   224
                +1   225     ; SMB0CN 0C0H
  00C7          +1   226     MASTER         BIT 0C7H     ; SMBUS 0 MASTER/SLAVE
  00C6          +1   227     TXMODE         BIT 0C6H     ; SMBUS 0 TRANSMIT MODE
  00C5          +1   228     STA            BIT 0C5H     ; SMBUS 0 START FLAG
  00C4          +1   229     STO            BIT 0C4H     ; SMBUS 0 STOP FLAG
  00C3          +1   230     ACKRQ          BIT 0C3H     ; SMBUS 0 ACKNOWLEDGE REQUEST
  00C2          +1   231     ARBLOST        BIT 0C2H     ; SMBUS 0 ARBITRATION LOST
  00C1          +1   232     ACK            BIT 0C1H     ; SMBUS 0 ACKNOWLEDGE FLAG
  00C0          +1   233     SI             BIT 0C0H     ; SMBUS 0 INTERRUPT PENDING FLAG
                +1   234
                +1   235     ; TMR2CN 0C8H
  00CF          +1   236     TF2H           BIT 0CFH     ; TIMER 2 HIGH BYTE OVERFLOW FLAG
  00CE          +1   237     TF2L           BIT 0CEH     ; TIMER 2 LOW BYTE OVERFLOW FLAG
  00CD          +1   238     TF2LEN         BIT 0CDH     ; TIMER 2 LOW BYTE INTERRUPT ENABLE
  00CC          +1   239     TF2CEN         BIT 0CCH     ; TIMER 2 LFO CAPTURE ENABLE
  00CB          +1   240     T2SPLIT        BIT 0CBH     ; TIMER 2 SPLIT MODE ENABLE
  00CA          +1   241     TR2            BIT 0CAH     ; TIMER 2 ON/OFF CONTROL
  00C8          +1   242     T2XCLK         BIT 0C8H     ; TIMER 2 EXTERNAL CLOCK SELECT
                +1   243
                +1   244     ; PSW 0D0H
  00D7          +1   245     CY             BIT 0D7H     ; CARRY FLAG
  00D6          +1   246     AC             BIT 0D6H     ; AUXILIARY CARRY FLAG
  00D5          +1   247     F0             BIT 0D5H     ; USER FLAG 0
  00D4          +1   248     RS1            BIT 0D4H     ; REGISTER BANK SELECT 1
  00D3          +1   249     RS0            BIT 0D3H     ; REGISTER BANK SELECT 0
  00D2          +1   250     OV             BIT 0D2H     ; OVERFLOW FLAG
  00D1          +1   251     F1             BIT 0D1H     ; USER FLAG 1
  00D0          +1   252     P              BIT 0D0H     ; ACCUMULATOR PARITY FLAG
                +1   253
                +1   254     ; PCA0CN 0D8H
  00DF          +1   255     CF             BIT 0DFH     ; PCA 0 COUNTER OVERFLOW FLAG
  00DE          +1   256     CR             BIT 0DEH     ; PCA 0 COUNTER RUN CONTROL BIT
  00DA          +1   257     CCF2           BIT 0DAH     ; PCA 0 MODULE 2 INTERRUPT FLAG
  00D9          +1   258     CCF1           BIT 0D9H     ; PCA 0 MODULE 1 INTERRUPT FLAG
  00D8          +1   259     CCF0           BIT 0D8H     ; PCA 0 MODULE 0 INTERRUPT FLAG
                +1   260                                 ; ADC 0 WINDOW INTERRUPT FLAG
                +1   261     ; ADC0CN 0E8H
  00EF          +1   262     AD0EN          BIT 0EFH     ; ADC 0 ENABLE
  00EE          +1   263     AD0TM          BIT 0EEH     ; ADC 0 TRACK MODE
  00ED          +1   264     AD0INT         BIT 0EDH     ; ADC 0 EOC INTERRUPT FLAG
  00EC          +1   265     AD0BUSY        BIT 0ECH     ; ADC 0 BUSY FLAG
  00EB          +1   266     AD0WINT        BIT 0EBH     ; ADC 0 WINDOW INTERRUPT FLAG
  00EA          +1   267     AD0CM2         BIT 0EAH     ; ADC 0 CONVERT START MODE BIT 2
  00E9          +1   268     AD0CM1         BIT 0E9H     ; ADC 0 CONVERT START MODE BIT 1
  00E8          +1   269     AD0CM0         BIT 0E8H     ; ADC 0 CONVERT START MODE BIT 0
                +1   270
                +1   271     ; SPI0CN 0F8H
  00FF          +1   272     SPIF           BIT 0FFH     ; SPI 0 INTERRUPT FLAG
  00FE          +1   273     WCOL           BIT 0FEH     ; SPI 0 WRITE COLLISION FLAG
  00FD          +1   274     MODF           BIT 0FDH     ; SPI 0 MODE FAULT FLAG
  00FC          +1   275     RXOVRN         BIT 0FCH     ; SPI 0 RX OVERRUN FLAG
  00FB          +1   276     NSSMD1         BIT 0FBH     ; SPI 0 SLAVE SELECT MODE 1
  00FA          +1   277     NSSMD0         BIT 0FAH     ; SPI 0 SLAVE SELECT MODE 0
  00F9          +1   278     TXBMT          BIT 0F9H     ; SPI 0 TX BUFFER EMPTY FLAG
  00F8          +1   279     SPIEN          BIT 0F8H     ; SPI 0 SPI ENABLE
                     280
                     281     ; tick constant:
                     282
  00E1               283     ITICK     EQU      225            ; osc = 3mhz and rfdata rate is 10kbs
                     284
                                                                          ; 185 = 9600
                     285
                                                                          ; 221 = 19200
                     286
                                                                          ; 225 = 22600
                     287
                                                                          ; 230 = 27222
                     288
                     289     ; memory address constants
                     290
                     291     ;AKMB      EQU      038H           ;  ACK buffer start address (future)
  0039               292     LNAK      EQU      039H           ;  ACK length byte
  003A               293     TPAK      EQU      03AH           ;  ACK type byte
  003B               294     TFAK      EQU      03BH           ;  ACK TO/FROM byte
  003C               295     IDAK      EQU      03CH           ;  ACK packet ID byte
                     296
  0040               297     TXMB      EQU      040H           ;  TX buffer start address (future)
  0041               298     LNTX      EQU      041H           ;  TX length byte
  0042               299     TPTX      EQU      042H           ;  TX type byte
  0043               300     TFTX      EQU      043H           ;  TX TO/FROM byte
  0044               301     IDTX      EQU      044H           ;  TX packet ID byte
  0045               302     CMTX      EQU      045H           ;  TX (host) command byte
                     303
                     304
  0060               305     RXMB      EQU      060H           ;  RX buffer start address (future)
  0061               306     LNRX      EQU      061H           ;  RX length byte
  0062               307     TPRX      EQU      062H           ;  RX type byte
  0063               308     TFRX      EQU      063H           ;  RX TO/FROM byte
  0064               309     IDRX      EQU      064H           ;  RX packet ID byte
  0065               310     CMRX      EQU      065H           ;  RX command byte
                     311
                     312
                     313     ;*******************************************************
  00F8               314     SPIOCN    EQU      0f8h                                         ; spio stat
  00A1               315     SPIOCFG   EQU      0a1h                                         ; spio config
  00A2               316     SPIOCKR         EQU                     0a2h                                       ;
s
                             pio clock speed
  00A3               317     SPIODAT   EQU      0a3h           ; spi data read/write
```

```
                ;*******************************************************


                ; type, framing and RangeTest constants

0000            RSVT      EQU      000H           ;  reserved type
0010            ACKT      EQU      010H           ;  RF ACK type
0020            MSGT      EQU      020H           ;  RF message type
0040            HTCT      EQU      040H           ;  host command type
0080            RXCT      EQU      080H           ;  RF command type

0002            STX       EQU      002H           ;  ASCII STX control character (future)
0006            TACK      EQU      006H           ;  ASCII ACK control character
0015            TNAK      EQU      015H           ;  ASCII NAK control character (future)
00C0            FEND      EQU      0C0H           ;  FEND framing character

00E2            SOPL      EQU      0E2H           ;  SOP low correlator pattern
00E2            SOPH      EQU      0E2H           ;  SOP high correlator pattern

0002            RTTM      EQU      002H           ;  0.20 s RangeTest interval 14

                ; frame check sequence (FCS) constants

00FF            FCSS      EQU      0FFH           ;  FCS seed
0084            FCSH      EQU      084H           ;  FCS high XOR mask
0008            FCSL      EQU      008H           ;  FCS low XOR mask
00F0            FCVH      EQU      0F0H           ;  FCS valid high byte pattern
00B8            FCVL      EQU      0B8H           ;  FCS valid low byte pattern

                ; stack:  08H - 015H (14 bytes)

                ; bit labels (bytes 020H - 023H)

0000            DCRON     EQU      000H           ;  RX data & clock recovery flag
0001            RXSMP     EQU      001H           ;  RX input sample
0002            LRXSM     EQU      002H           ;  last RX input sample
0003            RXBIT     EQU      003H           ;  RX input bit
0004            SOPFLG    EQU      004H           ;  SOP detect flag
0005            RXSFLG    EQU      005H           ;  RX symbol flag
0006            FCSB      EQU      006H           ;  FCS message bit
0007            OKFLG     EQU      007H           ;  RX FCS OK flag
0008            SIHLD     EQU      008H           ;  RX enable host flag
0009            NHFLG     EQU      009H           ;  no RX header flag
000A            SNFLG     EQU      00AH           ;  send NAK to host flag

000B            RAFLG     EQU      00BH           ;  RF ACK type flag
000C            RMFLG     EQU      00CH           ;  RF message type flag
000D            RCFLG     EQU      00DH           ;  RF command type flag

000E            SIFLG     EQU      00EH           ;  TX enable host flag
000F            TOFLG     EQU      00FH           ;  host timeout flag

0010            HTFLG     EQU      010H           ;  host traffic type flag
0011            HCFLG     EQU      011H           ;  host command type flag

0012            TXFLG     EQU      012H           ;  TX active flag
0013            TMFLG     EQU      013H           ;  TX message flag
0014            TSFLG     EQU      014H           ;  output TX sample flag
0015            TXBIT     EQU      015H           ;  TX message bit

0016            RTFLG     EQU      016H           ;  RangeTest active flag
0017            RSFLG     EQU      017H           ;  RangeTest message flag

0018            PIFLG     EQU      018H           ;  ping active flag
0019            PMFLG     EQU      019H           ;  ping message flag

```

```
 001A           THFLG     EQU      01AH           ;  TO here flag
 001B           RHFLG     EQU      01BH           ;  Repeat here flag

 001C           FLG1      EQU      01CH           ;  spare flag 1
 001D           FLG2      EQU      01DH           ;  spare flag 2
 001E           RTFLG1      EQU      01EH           ;  spare flag 3
 001F           RSFLG2      EQU      01FH           ;  spare flag 4

                ; register usage

                ;  R0                                  RX data pointer
                ;  R1                                  TX data pointer
                ;  R2                                  data & clock recovery (DCR)
                ;  R3                                  RX FCS buffer A
                ;  R4                                  not used
                ;  R5                                  TX FCS buffer A
                ;  R6                                  TX FCS buffer B
                ;  R7                                  RX FCS buffer B

                ; byte labels

 0016           RXBL      EQU      016H           ;  RX low buffer, SOP correlator, etc.
 0017           RXBH      EQU      017H           ;  RX high buffer, SOP correlator, etc.
 0018           RXBB      EQU      018H           ;  RX symbol decode byte buffer
 0019           RMBYC     EQU      019H           ;  RX message byte counter (LNRX 1st byte)
 001A           RMFCS     EQU      01AH           ;  RX FCS byte buffer
 001B           RMSBC     EQU      01BH           ;  RX symbol bit counter
 001C           RMFCC     EQU      01CH           ;  RX message FCS counter, etc.

 001D           TXSI      EQU      01DH           ;  TX start symbol index
 001E           TMFCC     EQU      01EH           ;  TX timer & loop counter
 001F           TXSMC     EQU      01FH           ;  TX output sample counter
 0024           TMBIC     EQU      024H           ;  TX message bit counter
 0025           TMBYT     EQU      025H           ;  TX message byte buffer
 0026           TMBYC     EQU      026H           ;  TX message byte counter (including FENDs)
 0027           TXSL      EQU      027H           ;  TX message symbol low buffer
 0028           TXSH      EQU      028H           ;  TX message symbol high buffer
 0029           TMFCS     EQU      029H           ;  TX FCS byte buffer
 002A           TXTL      EQU      02AH           ;  TX timer low byte
 002B           TXTH      EQU      02BH           ;  TX timer high byte
 002C           TXCNT     EQU      02CH           ;  TX retry counter
 002D           IDBUF     EQU      02DH           ;  Packet ID buffer
 002E           TFBUF     EQU      02EH           ;  TO/FROM address buffer
 002F           TEMPB     EQU      02FH           ;  Temp buffer

 0030           RTID      EQU      030H           ;  RangeTest ID buffer
 0031           RTTH      EQU      031H           ;  RangeTest timer high byte

 0032           BUF01     EQU      032H           ;  spare buffer 1
 0033           BUF02     EQU      033H           ;  spare buffer 2
 0034           BUF03     EQU      034H           ;  spare buffer 3
 0035           BUF04     EQU      035H           ;  spare buffer 4
 0036           BUF05     EQU      036H           ;  spare buffer 5
 0037           BUF06     EQU      037H           ;  spare buffer 6
 0038           fhss      EQU      038h
 003D           fhcnt     EQU      03dh           ;  searching counter if lost

                ; I/O pins for SDIO demo

 0091           CL1       EQU      P1.1           ;  ASH Radio CNTRL1  *** 0 = TX, 1 = RX
 0090           CL0       EQU      P1.0           ;  ASH Radio CNTRL0  *** 1 = TX, 1 = RX

                ;The Same for Analog mode
 0094           RXPIN     EQU      P1.4           ;  RX input pin (non-inverted)
 0093           TXPIN     EQU      P1.3           ;  TX output pin (on = 1) Was 3

```

```

  0082              PCRCV     EQU      P0.2           ;  LED (on = 1) 5
  0082              RFRCV     EQU      P0.2           ;  LED (on = 1) Was 1.6
  0083              RXI       EQU      P0.3           ;  LED (on = 1) Was 7,3

  0097              ID0                             EQU                      P1.7
                                     ; Range Test Jumper


  0083              Closed    EQU      P0.3           ; LED White
  0081              Opening   EQU      P0.1           ; LED Amber
  0082              Open      EQU      P0.2           ; LED yellow
  0080              Closing   EQU      P0.0           ; LED RED


                    ; start of code

0000                          ORG      00H             ;  hardware reset
0000 01FF           reset:    AJMP      start          ;  jump to start

000B                          ORG      0BH             ;  timer 0 interrupt vector

                    ; timer 0 interrupt service routine, 3 ticks/RF bit period

000B C0D0           tick:     PUSH      PSW            ;  push status
000D C0E0                     PUSH      ACC            ;  push accumulator
000F 30006D                   JNB       DCRON,tic0     ;  skip if DCR idle

                    ; RX data & clock recovery

0012 A201           dcr:      MOV       C,RXSMP        ;  store RX sample
0014 9202                     MOV       LRXSM,C        ;  as last RX sample
0016 A294                     MOV       C,RXPIN        ;  read RX input pin (non-inverted)
0018 9201                     MOV       RXSMP,C        ;  and store new RX sample
001A 300201         dcr0:     JNB       LRXSM,dcr1     ;  if last sample 1
001D B3                       CPL       C              ;  complement new sample
001E 5004           dcr1:     JNC       dcr2           ;  if no edge jump to dcr2
0020 7A00                     MOV       R2,#0          ;  else clear R2
0022 017C                     AJMP      dcr_d          ;  dcr done this tick
0024 0A             dcr2:     INC       R2             ;  else increment R2
0025 BA0106                   CJNE      R2,#1,dcr3     ;  if not 1 jump to dcr3
0028 A201                     MOV       C,RXSMP        ;  else load RXSMP
002A 9203                     MOV       RXBIT,C        ;  into RXBIT
002C 0135                     AJMP      dcr4           ;  jump to dcr4
002E BA034B         dcr3:     CJNE      R2,#3,dcr_d    ;  if not 3 jump dcr_d
0031 7A00                     MOV       R2,#0          ;  else clear R2
0033 017C                     AJMP      dcr_d          ;  done this tick
0035 20042B         dcr4:     JB        SOPFLG,dcr6    ;  skip after SOP detect

                    ; detect start-of packet-symbol (set SOP flag)

0038 E516                     MOV       A,RXBL         ;  else get RXBL
003A C3                       CLR       C              ;  clear carry
003B 33                       RLC       A              ;  rotate left through carry
003C 300302                   JNB       RXBIT,dcr5     ;  if bit 0 jump to dcr5
003F D2E0                     SETB      ACC.0          ;  else set lsb
0041 F516           dcr5:     MOV       RXBL,A         ;  store RXBL
0043 E517                     MOV       A,RXBH         ;  get RXBH
0045 33                       RLC       A              ;  shift and pull in carry
0046 F517                     MOV       RXBH,A         ;  store RXBH
0048 B4E231                   CJNE      A,#SOPH,dcr_d  ;  done if not SOPH
004B E516                     MOV       A,RXBL         ;  else get low buffer
004D B4E22C                   CJNE      A,#SOPL,dcr_d  ;  done if not SOPL
0050 751600                   MOV       RXBL,#0        ;  clear RX low buffer
0053 751B06                   MOV       RMSBC,#6       ;  set counter, 6 bits/half-symbol
```

```
0056 90078B                    MOV       DPTR,#rx_smbl  ;  point to RX symbol table
0059 C205                      CLR       RXSFLG         ;  clear RX symbol flag
005B C208                      CLR       SIHLD          ;  hold off host serial port
005D D204                      SETB      SOPFLG         ;  set SOP detected flag
005F D283                      setb       RXI            ;  turn RXI LED on
0061 017C                      AJMP      dcr_d          ;  done for now


                     ;  compile 6-bit half-symbols, synchronize rxmsg subroutine with RXSFLG flag

0063 E516            dcr6:     MOV       A,RXBL         ;  get RXBL
0065 C3                        CLR       C              ;  clear carry
0066 33                        RLC       A              ;  shift left
0067 300302                    JNB       RXBIT,dcr7     ;  if 0 bit jump to dcr7
006A D2E0                      SETB      ACC.0          ;  else set lsb
006C F516            dcr7:     MOV       RXBL,A         ;  store RXBL
006E D51B0B                    DJNZ      RMSBC,dcr_d    ;  if not 0 jump to dcr_d
0071 851618                    MOV       RXBB,RXBL      ;  get symbol
0074 751600                    MOV       RXBL,#0        ;  clear RXBL
0077 751B06                    MOV       RMSBC,#6       ;  reset counter
007A D205                      SETB      RXSFLG         ;  set symbol flag for rxmsg
007C 20044B          dcr_d:    JB        SOPFLG,tick_d  ;  skip if SOP detected

                     ; output TX bit samples (TXSMC = 0 synchronizes txmsg subroutine)

007F 30140A          tic0:     JNB       TSFLG,tic1     ;  skip if TX sample out idle
0082 E51F                      MOV       A,TXSMC        ;  else get sample count
0084 6006                      JZ        tic1           ;  skip if 0
0086 A215                      MOV       C,TXBIT        ;  else load TX bit
0088 9293                      MOV       TXPIN,C        ;  into TX output pin
008A 151F                      DEC       TXSMC          ;  decrement sample count

                     ; timeout on serial character stream from host (clear hang-ups)

008C 300F0C          tic1:     JNB       TOFLG,tic2     ;  skip if host message timeout idle
008F 051E                      INC       TMFCC          ;  else bump timeout counter
0091 E51E                      MOV       A,TMFCC        ;  get counter
0093 B49605                    CJNE      A,#150,tic2    ;  skip if counter not 150
0096 C20F                      CLR       TOFLG          ;  else reset timeout flag
0098 751E00                    MOV       TMFCC,#0       ;  reset counter

                     ; event timers

009B 052A            tic2:     INC       TXTL           ;  bump TX timer low
009D E52A                      MOV       A,TXTL         ;  load TX timer low
009F 7029                      JNZ       tick_d         ;  done if no rollover (8.89 ms cycle)

                     ; RangeTest event
00A1                 tic2_c:
                     ;          MOV       C,ID0          ;  read ID0
                     ;          JC        tic3           ;  skip if no ID0 jumper
00A1 20140B                    JB        TSFLG,tic3     ;  skip if TX busy
00A4 301608                    JNB       RTFLG,tic3     ;  skip if RangeTest idle
00A7 D53105                    DJNZ      RTTH,tic3      ;  decrement RTTH, jump if not 0
00AA D217                      SETB      RSFLG          ;  else set RangeTest message flag,
00AC 753102                    MOV       RTTH,#RTTM     ;  reload RTTH high

                     ; TX retry event

00AF 301218          tic3:     JNB       TXFLG,tick_d   ;  skip if send TX packet idle
00B2 D52B15                    DJNZ      TXTH,tick_d    ;  decrement TXTH, done if not 0
00B5 D213                      SETB      TMFLG          ;  else set TM message flag

                     ; load semi-random TX retry interval

00B7 9007CB                    MOV       DPTR,#delay    ;  point to delay table
```

```
00BA E58B                    MOV       A,TL1          ;  get random table offset
00BC 5407                    ANL       A,#07H         ;  mask out upper 5 bits
00BE 93                      MOVC      A,@A+DPTR      ;  load byte from table
00BF F52B                    MOV       TXTH,A         ;  into TX delay high

                   ; shut down TX and NAK host after 8 tries

00C1 E52C                    MOV       A,TXCNT        ;  load retry count
00C3 B40904                  CJNE      A,#9,tick_d    ;  if not 9 jump to tick_d
00C6 C213                    CLR       TMFLG          ;  else reset send TX message
00C8 D20A                    SETB      SNFLG          ;  set send TX NAK flag

                   ; restore and return from isr

00CA D5361B        tick_d:   DJNZ      BUF05,tick_d1              ; Hart beat delay counter 1
00CD D53718                                      DJNZ      BUF06,tick_d1   ; hart beat delay
counter
                    2
00D0 20800D                  JB        Closing,tick_d2 ; flash the led, if set clr it
00D3 7404                    MOV       A,#4            ; on short period of time
00D5 F537                    MOV                         BUF06,A         ; l
                   oad counter
00D7 D280                    SETB      Closing         ; led on

00D9 30160C                  JNB       RTFLG,tick_d1   ;  skip if RangeTest idle
00DC C204                    clr       SOPFLG

00DE 01E8                    AJMP      tick_d1
00E0 C280          tick_d2:  CLR       Closing         ; led off
00E2 C283                    CLR       Closed
00E4 7480                    MOV       A,#128          ; off longer than on
00E6 F537                    MOV                         BUF06,A         ; l
                   oad counter
00E8 200500        tick_d1:  JB        RXSFLG,tick_d3  ; prevent lock ups on symbol detect

00EB               tick_d3:
00EB D0E0                    POP       ACC             ;  pop accumulator
00ED D0D0                    POP       PSW             ;  pop status
00EF 32                      RETI                      ;  interrupt done

                   ;***************** RX fhss CHANNEL CHANGE **************************
00F0 75A3A1        chan:     mov       SPIODAT,#0a1h  ; Sent MSB of freq
00F3 F10F                    ACALL     spio_wait      ; WAIT FOR SPI
                   ;         mov       fhss,CMRX      ; get current channel
00F5 0538                    inc       fhss           ; bump it
00F7 E538                    MOV       A,fhss         ; fh offset count
00F9 B41901                  cjne      a,#25,h4a       ; 25 channels max now
00FC E4                      clr       a              ; da
00FD F538          h4a:      MOV       fhss,a         ; save channel
00FF AC38                    mov       R4,fhss        ; make loop count
0101 E4                      CLR       A              ; clear acc to make channe
                   l
0102 0C                      inc       r4
                   ; inc loop count
0103 240A          h2a:      add       a,#10          ; channel 10* R4
0105 DCFC                    djnz      r4,h2a          ; loop count
0107 F5A3                    mov                SPIODAT,A      ; Send LSB of Freq
0109 22                                          ret
                   ;                                     ACALL     spio_wait      ; wait for spi to finish
                   ;                                     ACALL     wait           ; ~20us to change
channels
                   ;                                     ACALL     wait
                   ;*******************************************************************

                   ;********************* restart FHSS Channel *************
010A               chan0:
010A 75A3A1                                      mov                SPIODAT,#0a1h ;a
                         ;  Sent MSB of freq
```

```
010D F10F                                                           ACALL     spio_wait
010F 7C01                                                           mov       R4,#1
0111 E4                                                             CLR       A
0112 0C                                                             inc       r4
0113 240A                    h3z:      add       a,#10
0115 DCFC                                                           djnz      r4,h3z
0117 F5A3                              mov                 SPIODAT,A ;#063h ; Send LSB of Freq
0119 F10F                              ACALL     spio_wait
011B 716E                              ACALL     wait
011D 22                                ret
                             ;******************************************************************


00FF                                   ORG       0FFH            ;  above interrupt code space
00FF D101                    start:    ACALL     setup           ;  initialization code

0101 753D08                            mov      fhcnt,#8         ; 16 tries


                                                                  ;  Flash the leds
0104 D283                              SETB      Closed

0106 715F                              ACALL     led                                       ; led lamp test delay
0108 D281                                                           SETB      Opening
010A 715F                              ACALL     led                                       ; led lamp test delay
010C D282                                                           SETB      Open
010E 715F                              ACALL     led                                       ; led lamp test delay
0110 D280                                                           SETB      Closing
0112 715F                              ACALL     led                                       ; led lamp test delay
0114 715F                              ACALL     led
0116 715F                              ACALL     led
0118 715F                              ACALL     led
011A 715F                              ACALL     led
011C C283                              CLR       Closed

011E 715F                              ACALL     led                                       ; led lamp test delay
0120 C281                                                           CLR       Opening
0122 715F                              ACALL     led                                       ; led lamp test delay
0124 C282                                                           CLR       Open
0126 715F                              ACALL     led                                       ; led lamp test delay
0128 C280                                                           CLR       Closing

012A D19E                              ACALL     rfic

012C 715F                              ACALL     led
012E 715F                              ACALL     led
0130 715F                              ACALL     led

0132 F104                              ACALL     tx_off

0134 715F                              ACALL     led
0136 715F                              ACALL     led
0138 715F                              ACALL     led

013A 75A3A1                            mov                 SPIODAT,#0a1h ;a
                                   ;  Sent MSB of freq
013D F10F                              ACALL     spio_wait
013F 7C01                              mov       R4,#1
0141 E4                                CLR       A
0142 0C                                inc       r4
0143 240A                    h3:       add       a,#10
0145 DCFC                              djnz      r4,h3
0147 F5A3                              mov                 SPIODAT,A ;#063h ; Send LSB of Freq
0149 F10F                              ACALL     spio_wait
014B 716E                              ACALL     wait
```

```


                             ; main program loop

014D                         main:



014D 301704                                                    JNB       RSFLG,mn0      ;  skip if RangeTest not queued
0150 71D0                              ACALL     do_ra          ;  else do RangeTest
0152 216E                              AJMP      mn3                 ;  jump to RX SOP detect
0154 30980B                  mn0:      JNB       RI0,mn1        ;  skip if RI clear
0157 300808                            JNB       SIHLD,mn1      ;  skip if serial in idled by RX
015A 300E05                            JNB       SIFLG,mn1      ;  skip if serial in idled by TX
015D 201802                            JB        PIFLG,mn1      ;  skip if ping active
0160 71EA                              ACALL     do_ht          ;  else service host
0162 301302                  mn1:      JNB       TMFLG,mn2      ;  skip if transmit not queued
0165 9124                              ACALL     do_rt          ;  else do transmit retry
0167 300A04                  mn2:      JNB       SNFLG,mn3      ;  skip if send NAK flag clear
016A 711C                              ACALL     aksnd          ;  else send NAK to host
016C B1A9                              ACALL     txmrs          ;  and master rest TX
016E 3004DC                  mn3:      JNB       SOPFLG,main    ;  if not SOP loop to main
0171 2016D9                                                          JB        RTFLG,main
0174 317D                              ACALL     do_rx          ;  else do RX message
0176 3019D4                  mn4:      JNB       PMFLG,main     ;  if ping idle loop to main
0179 910F                              ACALL     do_pi          ;  else do ping
017B 214D                    mn_d:     AJMP      main           ;  loop to main

                             ; receive message, transmit ACK, do command or send message to host

017D 31E0                    do_rx:    ACALL     rxmsg          ;  receive packet
017F C200                              CLR       DCRON          ;  idle RX DCR
0181 511A                              ACALL     rxfcs          ;  test packet FCS
0183 30074C                            JNB       OKFLG,rx4      ;  jump to reset RX if error
0186 201649                            JB        RTFLG,rx4      ;  jump if RangeTest active
0189 5128                              ACALL     rxto           ;  test packet TO address
018B 301A44                            JNB       THFLG,rx4      ;  jump to reset if not TO here
018E 5141                              ACALL     rxtyp          ;  else get RX packet type
0190 300B09                            JNB       RAFLG,rx1      ;  skip if not an RX ACK
0193 201802                            JB        PIFLG,rx0      ;  skip if RX ACK for ping
0196 711C                              ACALL     aksnd          ;  else ACK host
0198 B1A9                    rx0:      ACALL     txmrs          ;  master reset TX
019A 21D2                              AJMP      rx4            ;  jump to reset RX
019C                         rx1:                          ;**** add FH data to here ****
                             ;                               ACALL     ackrx          ;  transmit ACK to sender


                             ;***************** RX fhss CHANNEL CHANGE **************************
019C 75A3A1                            mov       SPIODAT,#0a1h  ; Sent MSB of freq

019F F10F                              ACALL     spio_wait      ; WAIT FOR SPI
01A1 856538                            mov       fhss,CMRX      ; get current channel
01A4 0538                              inc       fhss           ; bump it
01A6 E538                              MOV       A,fhss         ; fh offset count
01A8 B41801                            cjne      a,#24,h4       ; 25 channels max now
01AB E4                                clr       a              ; da
01AC F538                    h4:       MOV       fhss,a         ; save channel
01AE AC38                              mov       R4,fhss        ; make loop count
01B0 E4                                CLR       A              ; clear acc to make channe
                             l
01B1 0C                                inc       r4
                             ; inc loop count
01B2 240A                    h2:       add       a,#10          ; channel 10* R4
01B4 DCFC                              djnz      r4,h2          ; loop count
01B6 F5A3                              mov                 SPIODAT,A      ; Send LSB of Freq
01B8 F10F                              ACALL     spio_wait      ; wait for spi to finish
```

```
                   ;*****************************************************************

                   ;             ACALL     ackrx            ;  transmit ACK to sender

01BA 300D04                  JNB       RCFLG,rx2      ;  skip if not an RX command
01BD 51A5                    ACALL     rxcmd          ;  else do command
01BF 21D2                    AJMP      rx4            ;  and jump to reset RX
01C1 301809        rx2:      JNB       PIFLG,rx3      ;  skip if ping disabled
01C4 201206                  JB        TXFLG,rx3      ;  skip if TXFLG set
01C7 51B6                    ACALL     ld_tx          ;  else load TX from RX, idle SIFLG
01C9 51DB                    ACALL     ad_tx          ;  address TX, set PMFLG
01CB 21D2                    AJMP      rx4            ;  jump to reset RX
01CD 300C02        rx3:      JNB       RMFLG,rx4      ;  skip if not RX message
01D0 51EC                    ACALL     htsnd          ;  else send RX message to host
01D2 301602        rx4:      JNB       RTFLG,rx5      ;  skip if RangeTest idle
01D5 715F                    ACALL     led            ;  else delay to light LED
01D7 717D          rx5:      ACALL     rxrst          ;  reset RX, set SIHLD
01D9 D200                    SETB      DCRON          ;  enable RX DCR
01DB C299                    CLR       TI0            ;  clear TI0 flag
01DD C298                    CLR       RI0            ;  clear RI flag
01DF 22            rx_d:     RET                      ;  RX done

                   ; receive and de-symbolize message

01E0 3005FD        rxmsg:    JNB       RXSFLG,rxmsg   ;  wait for RX symbol flag from isr
01E3 C205                    CLR       RXSFLG         ;  clear RX symbol flag
01E5 E518                    MOV       A,RXBB         ;  index into symbol table
01E7 93                      MOVC      A,@A+DPTR      ;  get table entry
01E8 C4                      SWAP      A              ;  swap to high nibble
01E9 F517                    MOV       RXBH,A         ;  into RXBH
01EB 3005FD        rxm2:     JNB       RXSFLG,rxm2    ;  wait for symbol flag from isr
01EE C205                    CLR       RXSFLG         ;  clear flag
01F0 E518                    MOV       A,RXBB         ;  index into symbol table
01F2 93                      MOVC      A,@A+DPTR      ;  get table entry
01F3 4517                    ORL       A,RXBH         ;  add RXBH low
01F5 F517                    MOV       RXBH,A         ;  store in RXBH
01F7 A617                    MOV       @R0,RXBH       ;  and store in RX message buffer
01F9 B86115                  CJNE      R0,#LNRX,rxm4  ;  skip if not length byte
01FC E517                    MOV       A,RXBH         ;  else get first byte
01FE 543F                    ANL       A,#63          ;  mask upper 2 bits
0200 6009                    JZ        rxm3           ;  0 is an error
0202 F519                    MOV       RMBYC,A        ;  load message byte counter
0204 F51C                    MOV       RMFCC,A        ;  and RX message loop counter
0206 C3                      CLR       C              ;  clear borrow
0207 941F                    SUBB      A,#31          ;  compare number of bytes to 31
0209 4006                    JC        rxm4           ;  skip if < 31
020B 751904        rxm3:     MOV       RMBYC,#4       ;  else force byte counter to 4 (force error)
020E 751C04                  MOV       RMFCC,#4       ;  and force loop counter to 4
0211 08            rxm4:     INC       R0             ;  bump pointer
0212 D51CCB                  DJNZ      RMFCC,rxmsg    ;  if not 0 get another byte
0215 7861                    MOV       R0,#LNRX       ;  reset RX message pointer
0217 C283                    clr      RXI            ;  turn LED off
0219 22            rxm_d:    RET                      ;  RX message done

                   ; build and test FCS

021A 85191C        rxfcs:    MOV       RMFCC,RMBYC    ;  move byte count to loop counter
021D 861A          rxf0:     MOV       RMFCS,@R0      ;  get next message byte
021F 08                      INC       R0             ;  bump pointer
0220 719B                    ACALL     b_rfcs         ;  build FCS
0222 D51CF8                  DJNZ      RMFCC,rxf0     ;  loop for next byte
0225 71BF                    ACALL     a_rfcs         ;  test FCS
0227 22            rxf_d:    RET                      ;  RX FCS done

                   ; determine if packet is to here, and if so is it single or multi-hop

```

```
0228 E52E           rxto:     MOV       A,TFBUF        ;  get local TO/FROM address
022A 540F                     ANL       A,#15          ;  mask to get local FROM address
022C F5F0                     MOV       B,A            ;  store FROM address
022E E563                     MOV       A,TFRX         ;  get T/F address from RX buffer
0230 C4                       SWAP      A              ;  swap - FROM/TO
0231 540F                     ANL       A,#15          ;  mask to get TO address
0233 B5F00A                   CJNE      A,B,rxto_d     ;  done if not TO here
0236 D21A                     SETB      THFLG          ;  else set TO here flag
0238 E562                     MOV       A,TPRX         ;  get RX type byte
023A 540F                     ANL       A,#15          ;  mask out upper nibble, get hops
023C 6002                     JZ        rxto_d         ;  done if single hop
023E D21B                     SETB      RHFLG          ;  else set repeat here flag
0240 22             rxto_d:   RET                      ;  done

                    ; determine RF packet type

0241 E562           rxtyp:    MOV       A,TPRX         ;  get RX type byte
0243 54F0                     ANL       A,#240         ;  mask out lower nibble, get type
0245 B42004                   CJNE      A,#MSGT,rxt0   ;  skip if not message
0248 D20C                     SETB      RMFLG          ;  else set RX message flag
024A 4160                     AJMP      rxt_d          ;  done
024C B48004         rxt0:     CJNE      A,#RXCT,rxt1   ;  skip if not RX command
024F D20D                     SETB      RCFLG          ;  else set RX command flag
0251 4160                     AJMP      rxt_d          ;  done
0253 B4100A         rxt1:     CJNE      A,#ACKT,rxt_d  ;  done if not RX ACK
0256 E52E                     MOV       A,TFBUF        ;  else get local TO/FROM
0258 C4                       SWAP      A              ;  swap for FROM/TO
0259 B56304                   CJNE      A,TFRX,rxt_d   ;  done if not RX TO/FROM
025C E52D                     MOV       A,IDBUF        ;  else get TX packet ID
                    ;          CJNE      A,IDRX,rxt_d   ;  done if not TX ID
025E D20B                     SETB      RAFLG          ;  else set RX ACK flag
0260 22             rxt_d:    RET

                    ; transmit ACK back to sending node

0261 D282           ackrx:    SETB      Open
0263 7939                     MOV       R1,#LNAK       ;  load ACK pointer
0265 7706                     MOV       @R1,#6         ;  ACK length is 6 bytes
0267 752906                   MOV       TMFCS,#6       ;  load TX message FCS byte
026A B1D1                     ACALL     b_tfcs         ;  and build FCS
026C 09                       INC       R1             ;  bump pointer
026D 7710                     MOV       @R1,#ACKT      ;  store ACK type byte
026F 752910                   MOV       TMFCS,#ACKT    ;  load TX message FCS byte
0272 B1D1                     ACALL     b_tfcs         ;  and build FCS
0274 09                       INC       R1             ;  bump pointer
0275 E563                     MOV       A,TFRX         ;  get TO/FROM byte
0277 C4                       SWAP      A              ;  swap TO/FROM addresses
0278 F7                       MOV       @R1,A          ;  add to ACK buffer
0279 F529                     MOV       TMFCS,A        ;  load TX message FCS byte
027B B1D1                     ACALL     b_tfcs         ;  and build FCS
027D 09                       INC       R1             ;  bump pointer
027E E564                     MOV       A,IDRX         ;  get packet ID byte
0280 F7                       MOV       @R1,A          ;  add ID to ACK message
0281 F529                     MOV       TMFCS,A        ;  load TX message FCS byte
0283 B1D1                     ACALL     b_tfcs         ;  and build FCS
0285 09                       INC       R1             ;  bump pointer
0286 B1F5                     ACALL     a_tfcs         ;  add FCS
0288 7939                     MOV       R1,#LNAK       ;  reset ACK pointer
028A 85262F                   MOV       TEMPB,TMBYC    ;  store TX message TMBYC
028D 752606                   MOV       TMBYC,#6       ;  6 bytes in ACK
0290 B129                     ACALL     txmsg          ;  send TX message
0292 E4                       CLR       A              ;  reset for next TX
0293 F525                     MOV       TMBYT,A        ;  clear TX message byte
0295 F51F                     MOV       TXSMC,A        ;  clear TX out count
0297 F527                     MOV       TXSL,A         ;  clear TX symbol low
0299 F528                     MOV       TXSH,A         ;  clear TX symbol high
```

```
029B 7941                     MOV       R1,#LNTX       ;  point R1 to message start
029D 852F26                   MOV       TMBYC,TEMPB    ;  restore TX message TMBYC
02A0 C282           arx0:     clr       RFRCV          ;  turn FCS LED off
02A2 C282                     CLR       Open
02A4 22             arx_d:    RET                      ;  RX ACK done

                    ; do RX command (bounce on/off), reset RCFLG

02A5 E565           rxcmd:    MOV       A,CMRX         ;  get RX command
02A7 B44204                   CJNE      A,#66,rxc0     ;  skip if not bounce on
02AA D218                     SETB      PIFLG          ;  else set RX message flag
02AC 41B3                     AJMP      rxc1           ;  done
02AE B44E02         rxc0:     CJNE      A,#78,rxc1     ;  skip if not bounce off
02B1 C218                     CLR       PIFLG          ;  disable ping
02B3 C20D           rxc1:     CLR       RCFLG          ;  reset RX command flag
02B5 22             rxc_d:    RET

                    ; load TX buffer from RX buffer, idle SIFLG to block host input

02B6 C282           ld_tx:    CLR       PCRCV          ;  turn PC LED on
02B8 1519                     DEC       RMBYC          ;  adjust number of bytes to copy
02BA 851926                   MOV       TMBYC,RMBYC    ;  transfer RX count to TX count
02BD 7861                     MOV       R0,#LNRX       ;  reset RX buffer pointer
02BF 7941                     MOV       R1,#LNTX       ;  reset TX buffer pointer
02C1 77C0                     MOV       @R1,#FEND      ;  load first FEND
02C3 08                       INC       R0             ;  bump RX pointer past FEND
02C4 09                       INC       R1             ;  bump TX pointer past FEND
02C5 1519                     DEC       RMBYC          ;  decrement byte count
02C7 E6             ld0:      MOV       A,@R0          ;  get RX byte
02C8 F7                       MOV       @R1,A          ;  load TX buffer
02C9 08                       INC       R0             ;  bump RX pointer
02CA 09                       INC       R1             ;  bump TX pointer
02CB D519F9                   DJNZ      RMBYC,ld0      ;  loop to load message
02CE 77C0                     MOV       @R1,#FEND      ;  add 2nd FEND
02D0 7861                     MOV       R0,#LNRX       ;  reset RX buffer pointer
02D2 7941                     MOV       R1,#LNTX       ;  reset TX buffer pointer
02D4 C20E                     CLR       SIFLG          ;  TX loaded, idle host
02D6 C282           ld1:      clr      RFRCV          ;  turn FCS LED off
02D8 D282                     SETB      PCRCV          ;  turn PC LED off
02DA 22             ld_d:     RET                      ;  load TX buffer done

                    ; set TX TO/FROM address, ID, set PMFLG (ping message)

02DB E563           ad_tx:    MOV       A,TFRX         ;  get RX TO/FROM
02DD C4                       SWAP      A              ;  swap TO/FROM
02DE F543                     MOV       TFTX,A         ;  load TX TO/FROM
02E0 856444                   MOV       IDTX,IDRX      ;  load TX ID from RX ID
02E3 85432E                   MOV       TFBUF,TFTX     ;  update local TO/FROM buffer
02E6 85442D                   MOV       IDBUF,IDTX     ;  update local ID buffer
02E9 D219                     SETB      PMFLG          ;  enable ping message
02EB 22             ad_d:     RET                      ;  TX address done

                    ; send received message to host with FEND framing

02EC C282           htsnd:    CLR       PCRCV          ;  turn PC LED on
02EE 1519                     DEC       RMBYC          ;  don't send the
02F0 1519                     DEC       RMBYC          ;  2 FCS bytes
02F2 7861                     MOV       R0,#LNRX       ;  reset RX message pointer
02F4 76C0                     MOV       @R0,#FEND      ;  replace number of bytes with FEND
                    ;          JNB       NHFLG,hts0     ;  skip if no FEND/header flag reset
02F6 08                       INC       R0             ;  bump past FEND
02F7 1519                     DEC       RMBYC          ;  decrement byte count
02F9 08                       INC       R0             ;  bump past FEND
02FA 1519                     DEC       RMBYC          ;  decrement byte count
02FC 08                       INC       R0             ;  bump past Type
02FD 1519                     DEC       RMBYC          ;  decrement byte count
```

```
02FF 08                       INC       R0             ;  bump past TO/FROM
0300 1519                     DEC       RMBYC          ;  decrement byte count
0302 08                       INC       R0             ;  bump past ID
0303 1519                     DEC       RMBYC          ;  decrement byte count
0305 C299           hts0:     CLR       TI0            ;  clear TI0 flag
0307 8699           hts1:     MOV       SBUF0,@R0      ;  send byte
0309 3099FD         hts2:     JNB       TI0,hts2       ;  wait until byte sent
030C C299                     CLR       TI0            ;  clear TI0 flag
030E 08                       INC       R0             ;  bump pointer
030F D519F5                   DJNZ      RMBYC,hts1     ;  loop to send message
0312 200902                   JB        NHFLG,hts4     ;  skip if no FEND/header flag set
                    ;          MOV       SBUF0,#FEND    ;  add 2nd FEND
                    ;hts3:     JNB       TI0,hts3       ;  wait until byte sent
0315 C299                     CLR       TI0            ;  clear TI0 flag
0317 C282           hts4:     clr       RFRCV          ;  turn FCS LED off
0319 D282                     SETB      PCRCV          ;  turn PC LED off
031B 22             hts_d:    RET                      ;  send RX message done

                    ; send ACK/NAK to host

031C D280           aksnd:          SETB      Closing
031E C282                     CLR       PCRCV          ;  turn PC LED on
0320 E52D                     MOV       A,IDBUF        ;  get local ID
0322 5407                     ANL       A,#7           ;  mask unused bits
0324 C4                       SWAP      A              ;  swap ID to upper IDS nibble
0325 252C                     ADD       A,TXCNT        ;  add retry count to IDS
0327 300B02                   JNB       RAFLG,aks0     ;  skip if not ACK
032A 2480                     ADD       A,#128         ;  else set ACK bit
032C F5F0           aks0:     MOV       B,A            ;  hold IDS in B
032E E52E                     MOV       A,TFBUF        ;  get local TO/FROM
0330 C4                       SWAP      A              ;  switch TO and FROM
0331 C299                     CLR       TI0            ;  clear TI0 flag
0333 7599C0                   MOV       SBUF0,#FEND    ;  send first FEND
0336 3099FD         aks1:     JNB       TI0,aks1       ;  wait until byte sent
0339 C299                     CLR       TI0            ;  clear TI0 flag
033B 759910                   MOV       SBUF0,#ACKT    ;  send ACK type byte
033E 3099FD         aks2:     JNB       TI0,aks2       ;  wait until byte sent
0341 C299                     CLR       TI0            ;  clear TI0 flag
0343 F599                     MOV       SBUF0,A        ;  send TO/FROM
0345 3099FD         aks3:     JNB       TI0,aks3       ;  wait until byte sent
0348 C299                     CLR       TI0            ;  clear TI0 flag
034A 85F099                   MOV       SBUF0,B        ;  send IDS
034D 3099FD         aks4:     JNB       TI0,aks4       ;  wait until byte sent
0350 C299                     CLR       TI0            ;  clear TI0 flag
0352 7599C0                   MOV       SBUF0,#FEND    ;  send 2nd FEND
0355 3099FD         aks5:     JNB       TI0,aks5       ;  wait until byte sent
0358 C282                     clr       RFRCV          ;  turn FCS LED off
035A D282                     SETB      PCRCV          ;  turn PC LED off
035C C280                                                CLR       Closing
035E 22             aks_d:    RET                      ;  send ACK message done

                    ; delay to show FCS LED

035F 75F000         led:      MOV       B,#0           ;  load delay value
0362 7C58                                                MOV                              r4,#58h
0364 00             led0:     NOP                      ;  NOP delay
0365 00                       NOP                      ;  NOP delay
0366 00                       NOP                      ;  NOP delay
0367 00                       NOP                      ;  NOP delay
0368 D5F0F9                   DJNZ      B,led0         ;  loop to delay
036B DCF7                                                DJNZ      r4,led0
036D 22             led_d:    RET

                    ;***********************************

036E 75F000         wait:     MOV       B,#0           ;  load delay value
```

```
0371 7C08                                           MOV                             r4,#8
0373 00             wait0:    NOP                       ;  NOP delay
0374 00                       NOP                       ;  NOP delay
0375 00                       NOP                       ;  NOP delay
0376 00                       NOP                       ;  NOP delay
0377 D5F0F9                   DJNZ      B,wait0         ;  loop to delay
037A DCF7                                                  DJNZ      r4,wait0
037C 22             wait_d:   RET


                    ; reset receive state, set SIHLD

037D E4             rxrst:    CLR       A               ;  clear A
037E F517                     MOV       RXBH,A          ;  clear buffer
0380 F516                     MOV       RXBL,A          ;  clear buffer
0382 F518                     MOV       RXBB,A          ;  clear buffer
0384 F519                     MOV       RMBYC,A         ;  clear RX byte count
0386 F51C                     MOV       RMFCC,A         ;  clear loop counter
0388 7861                     MOV       R0,#LNRX        ;  point R0 to message start
038A C207                     CLR       OKFLG           ;  clear FCS OK flag
038C C21A                     CLR       THFLG           ;  clear TO here flag
038E C20C                     CLR       RMFLG           ;  clear RF message flag
0390 C20D                     CLR       RCFLG           ;  clear RF command flag
0392 C204                     CLR       SOPFLG          ;  enable SOP test
0394 D208                     SETB      SIHLD           ;  take serial in off hold
0396 C282                     clr       RFRCV           ;  turn FCS LED off
0398 C283                     clr       RXI             ;  turn RXI LED off
039A 22             rxr_d:    RET                       ;  RX reset done

                    ; build FCS from each received byte

039B 75F008         b_rfcs:   MOV       B,#8            ;  load loop count of 8
039E C3             brf0:     CLR       C               ;  clear carry bit
039F E51A                     MOV       A,RMFCS         ;  load RX message byte
03A1 13                       RRC       A               ;  shift lsb into carry
03A2 F51A                     MOV       RMFCS,A         ;  store shifted message byte
03A4 9206                     MOV       FCSB,C          ;  load FCSB with lsb
03A6 C3                       CLR       C               ;  clear carry bit
03A7 EB                       MOV       A,R3            ;  load high FCS byte
03A8 13                       RRC       A               ;  shift right
03A9 FB                       MOV       R3,A            ;  store shifted high FCS
03AA EF                       MOV       A,R7            ;  load low FCS byte
03AB 13                       RRC       A               ;  shift and pull in bit for FCS high
03AC FF                       MOV       R7,A            ;  store shifted low FCS
03AD 300601                   JNB       FCSB,brf1       ;  if lsb of low FCS = 0, jump to brf1
03B0 B3                       CPL       C               ;  else complement carry bit
03B1 5008           brf1:     JNC       brf2            ;  if FCSB XOR (low FCS lsb) = 0 jump to brf2
03B3 EB                       MOV       A,R3            ;  else load high FCS
03B4 6484                     XRL       A,#FCSH         ;  and XOR with high FCS poly
03B6 FB                       MOV       R3,A            ;  store high FCS
03B7 EF                       MOV       A,R7            ;  load low FCS
03B8 6408                     XRL       A,#FCSL         ;  XOR with low FCS poly
03BA FF                       MOV       R7,A            ;  store low FCS
03BB D5F0E0         brf2:     DJNZ      B,brf0          ;  loop through bits in message byte
03BE 22             brfcs_d:  RET                       ;  done this pass

                    ; test received message FCS

03BF EB             a_rfcs:   MOV       A,R3            ;  load FCS high
03C0 B4F008                   CJNE      A,#FCVH,arf0    ;  jump if not 0F0H
03C3 EF                       MOV       A,R7            ;  load FCS low
03C4 B4B804                   CJNE      A,#FCVL,arf0    ;  jump if not 0B8H
03C7 D282                     setb      RFRCV           ;  else turn FCS LED on
03C9 D207                     SETB      OKFLG           ;  set FCS OK flag
03CB 7BFF           arf0:     MOV       R3,#FCSS        ;  reseed FCS high
03CD 7FFF                     MOV       R7,#FCSS        ;  reseed FCS low
```

```
03CF 22            arfcs_d:  RET                       ;  RX FCS done


                   ; RangeTest message (timer activated, host not required)

03D0 201216        do_ra:    JB        TXFLG,ra_d     ;  skip is TX active
03D3 C200                    CLR       DCRON          ;  idle RX DCR
03D5 C282                    CLR       PCRCV          ;  PCRCV LED on
03D7 D282                                                SETB      Open
03D9 F12D                    ACALL     hello2         ;  get RangeTest message
03DB B111                    ACALL     txfcs          ;  build and add FCS
03DD D282                    SETB      PCRCV          ;  PCRCV LED off
03DF B129                    ACALL     txmsg          ;  send TX message
03E1 B1BA                    ACALL     txrst          ;  reset TX (retry)
03E3 C217                    CLR       RSFLG          ;  clear RangeTest msg flag
03E5 D200                    SETB      DCRON          ;  enable RX DCR
03E7 C282                                                CLR       Open
03E9 22            ra_d:     RET                      ;  RangeTest message done

                   ; get message from host, do command or transmit once

03EA C282          do_ht:    CLR       PCRCV          ;  turn PC LED on
03EC C200                    CLR       DCRON          ;  idle RX DCR
03EE 9135                    ACALL     htget          ;  get message from host
03F0 918A                    ACALL     httyp          ;  determine message type
03F2 301104                  JNB       HCFLG,ht0      ;  skip if not host command
03F5 91B4                    ACALL     htcmd          ;  else do command
03F7 810A                    AJMP      ht3            ;  and enable DCR
03F9 30100A        ht0:      JNB       HTFLG,ht1      ;  skip if not transfer type
03FC B111                    ACALL     txfcs          ;  else build and add FCS
03FE B129                    ACALL     txmsg          ;  send TX message
0400 D212                    SETB      TXFLG          ;  set TX active flag
0402 052C                    INC       TXCNT          ;  increment TX count
0404 8108                    AJMP      ht2            ;  and reset TX
0406 D20E          ht1:      SETB      SIFLG          ;  set SIFLG if unknown type
0408 B1BA          ht2:      ACALL     txrst          ;  reset TX (retry)
040A D200          ht3:      SETB      DCRON          ;  enable RX DCR
040C D282                    SETB      PCRCV          ;  turn PC LED off
040E 22            ht_d:     RET                      ;  TX message done

                   ; ping message once

040F C282          do_pi:    CLR       PCRCV          ;  turn PC LED on
0411 C200                    CLR       DCRON          ;  idle RX DCR
0413 B111                    ACALL     txfcs          ;  build and add FCS
0415 B129                    ACALL     txmsg          ;  send TX message
0417 D212                    SETB      TXFLG          ;  set TX active flag
0419 052C                    INC       TXCNT          ;  increment TX count
041B B1BA                    ACALL     txrst          ;  reset TX (retry)
041D C219                    CLR       PMFLG          ;  clear ping msg flag
041F D200                    SETB      DCRON          ;  enable RX DCR
0421 D282                    SETB      PCRCV          ;  turn PC LED off
0423 22            pi_d:     RET                      ;  TX message done

                   ; transmit message again

0424 C282          do_rt:    CLR       PCRCV          ;  else turn PCRCV LED on
0426 C200                    CLR       DCRON          ;  idle RX DCR
0428 B129                    ACALL     txmsg          ;  send TX message
042A 052C                    INC       TXCNT          ;  increment TX count
042C B1BA                    ACALL     txrst          ;  reset TX (retry)
042E C213                    CLR       TMFLG          ;  clear TX msg flag
0430 D200                    SETB      DCRON          ;  enable RX DCR
0432 D282                    SETB      PCRCV          ;  turn PC LED off
0434 22            rt_d:     RET                      ;  TX message done

```

```
                    ; get message from host, clear SIFLG if valid

0435 E599           htget:    MOV       A,SBUF0        ;  get byte
0437 C298                     CLR       RI0            ;  clear RI flag
0439 B4C048                   CJNE      A,#FEND,htg8   ;  done if not FEND
043C 7941                     MOV       R1,#LNTX       ;  reset TX buffer index
043E F7                       MOV       @R1,A          ;  store first FEND
043F 09                       INC       R1             ;  bump TX buffer index
0440 0526                     INC       TMBYC          ;  bump TX byte counter
0442 751E00         htg0:     MOV       TMFCC,#0       ;  reset timeout counter
0445 D20F                     SETB      TOFLG          ;  set timeout flag
0447                htg1:;     JNB       TOFLG,htg8    ;  jump to htg8 if TOFLG resets
                    ;          JNB       RI0,htg1      ;  else loop until next byte
0447 E599                     MOV       A,SBUF0        ;  get byte
0449 C298                     CLR       RI0            ;  clear RI flag
044B F7                       MOV       @R1,A          ;  store byte
044C 09                       INC       R1             ;  bump TX buffer index
044D 0526                     INC       TMBYC          ;  bump TX byte counter
044F B4C002                   CJNE      A,#FEND,htg2   ;  skip if not FEND
0452 815B                     AJMP      htg3           ;  else jump to htg3
0454 E526           htg2:     MOV       A,TMBYC        ;  load A with number of bytes + 1
0456 B41DE9                   CJNE      A,#29,htg0     ;  loop again if not 29th byte
0459 77C0                     MOV       @R1,#FEND      ;  else force 2nd FEND
045B 7941           htg3:     MOV       R1,#LNTX       ;  reset TX buffer index
045D E526                     MOV       A,TMBYC        ;  get byte count
045F B40202                   CJNE      A,#2,htg4      ;  if not 2 jump to htg4
0462 8184                     AJMP      htg8           ;  else jump to htg8
0464 C20E           htg4:     CLR       SIFLG          ;  disable serial in
0466 C20F                     CLR       TOFLG          ;  clear timeout flag
0468 C299                     CLR       TI0            ;  clear TI0 flag
046A 7599C0                   MOV       SBUF0,#FEND    ;  send first FEND
046D 3099FD         htg5:     JNB       TI0,htg5       ;  wait until byte sent
0470 C299                     CLR       TI0            ;  clear TI0 flag
0472 759906                   MOV       SBUF0,#6       ;  send TACK byte
0475 3099FD         htg6:     JNB       TI0,htg6       ;  wait until byte sent
0478 C299                     CLR       TI0            ;  clear TI0 flag
047A 7599C0                   MOV       SBUF0,#FEND    ;  send 2nd FEND
047D 3099FD         htg7:     JNB       TI0,htg7       ;  wait until byte sent
0480 C299                     CLR       TI0            ;  clear TI0 flag
0482 8189                     AJMP      htg_d          ;  jump to done
0484 7941           htg8:     MOV       R1,#LNTX       ;  reset TX buffer index
0486 752600                   MOV       TMBYC,#0       ;  reset byte counter
0489 22             htg_d:    RET                      ;  get TX message done

                    ; determine host message type; update T/F if transfer

048A E526           httyp:    MOV       A,TMBYC        ;  get message length
048C 6025                     JZ        htt_d          ;  0 not valid, done
048E E542                     MOV       A,TPTX         ;  get host type byte
0490 54F0                     ANL       A,#240         ;  mask out lower nibble, get type
0492 B44006                   CJNE      A,#HTCT,htt0   ;  jump to htt0 if not host command
0495 C210                     CLR       HTFLG          ;  clear host transfer flag
0497 D211                     SETB      HCFLG          ;  set host command flag
0499 81B3                     AJMP      htt_d          ;  done
049B 6420           htt0:     XRL       A,#MSGT        ;  compare with RF message type
049D 600A                     JZ        htt1           ;  jump to htt1 if message
049F E542                     MOV       A,TPTX         ;  get host type byte
04A1 54F0                     ANL       A,#240         ;  mask out lower nibble, get type
04A3 6480                     XRL       A,#RXCT        ;  compare with RF command type
04A5 6002                     JZ        htt1           ;  jump to htt1 if RF command
04A7 81B3                     AJMP      htt_d          ;  done
04A9 85432E         htt1:     MOV       TFBUF,TFTX     ;  update local TO/FROM buffer
04AC 85442D                   MOV       IDBUF,IDTX     ;  update local ID buffer
04AF D210                     SETB      HTFLG          ;  set host RF traffic flag
04B1 C211                     CLR       HCFLG          ;  clear host command flag
04B3 22             htt_d:    RET
```

```

                    ; ACK host, do command, set SIFLG if not TX

04B4 91D3           htcmd:    ACALL     akhcm          ;  send host command ACK
04B6 D282                     setb      RFRCV        ;  flash RFRCV & RXI
04B8 D283                     setb      RXI          ;  to demo a command
04BA 7400                     MOV       A,#0           ;  outer delay value
04BC 75F000         htc0:     MOV       B,#0           ;  inner delay value
04BF 00             htc1:     NOP                      ;  NOP to delay
04C0 00                       NOP                      ;  NOP to delay
04C1 00                       NOP                      ;  NOP to delay
04C2 D5F0FA                   DJNZ      B,htc1         ;  inner delay loop
04C5 D5E0F4                   DJNZ      ACC,htc0       ;  outer delay loop
04C8 715F                     ACALL     led
04CA C282                     clr       RFRCV          ;  turn off RFRCV
04CC C283                     clr       RXI            ;  and RXI
04CE D20E           htc2:     SETB      SIFLG          ;  TX enable host
04D0 C211           htc3:     CLR       HCFLG          ;  reset host command flag
04D2 22             htc_d:    RET

                    ; ACK host command

04D3 C282           akhcm:    CLR       PCRCV          ;  turn ON PCRCV
04D5 E52D                     MOV       A,IDBUF        ;  get local ID
04D7 5407                     ANL       A,#7           ;  mask unused bits
04D9 C4                       SWAP      A              ;  swap ID to upper IDS nibble
04DA 2401                     ADD       A,#1           ;  add count to IDS
04DC 2480                     ADD       A,#128         ;  set ACK bit
04DE F5F0           akh0:     MOV       B,A            ;  hold IDS in B
04E0 E52E                     MOV       A,TFBUF        ;  get local TO/FROM
04E2 C4                       SWAP      A              ;  switch TO and FROM
04E3 C299                     CLR       TI0            ;  clear TI0 flag
04E5 7599C0                   MOV       SBUF0,#FEND    ;  send first FEND
04E8 3099FD         akh1:     JNB       TI0,akh1       ;  wait until byte sent
04EB C299                     CLR       TI0            ;  clear TI0 flag
04ED 759910                   MOV       SBUF0,#ACKT    ;  send ACK type byte
04F0 3099FD         akh2:     JNB       TI0,akh2       ;  wait until byte sent
04F3 C299                     CLR       TI0            ;  clear TI0 flag
04F5 F599                     MOV       SBUF0,A        ;  send TO/FROM
04F7 3099FD         akh3:     JNB       TI0,akh3       ;  wait until byte sent
04FA C299                     CLR       TI0            ;  clear TI0 flag
04FC 85F099                   MOV       SBUF0,B        ;  send IDS
04FF 3099FD         akh4:     JNB       TI0,akh4       ;  wait until byte sent
0502 C299                     CLR       TI0            ;  clear TI0 flag
0504 7599C0                   MOV       SBUF0,#FEND    ;  send 2nd FEND
0507 3099FD         akh5:     JNB       TI0,akh5       ;  wait until byte sent
050A C299                     CLR       TI0            ;  clear TI0 flag
050C C298                     CLR       RI0            ;  clear RI flag
050E D282                     SETB      PCRCV          ;  turn OFF PCRCV
0510 22             akh_d:    RET                      ;  host command ACK done

                    ; build FCS for transmit

0511 0526           txfcs:    INC       TMBYC          ;  number of bytes + 2 (FCS 2 bytes)
0513 A726                     MOV       @R1,TMBYC      ;  replace FEND with number of bytes
0515 85261E                   MOV       TMFCC,TMBYC    ;  move byte count to loop counter
0518 151E                     DEC       TMFCC          ;  loop count is 2 less than
051A 151E                     DEC       TMFCC          ;  number of bytes including FCS
051C 8729           txf0:     MOV       TMFCS,@R1      ;  get next message byte
051E 09                       INC       R1             ;  bump pointer
051F B1D1                     ACALL     b_tfcs         ;  build FCS
0521 D51EF8                   DJNZ      TMFCC,txf0     ;  loop for next byte
0524 B1F5                     ACALL     a_tfcs         ;  add FCS
0526 7941                     MOV       R1,#LNTX       ;  reset TX message pointer
0528 22             txf_d:    RET                      ;  TX FCS done

```

```
                  ; transmit preamble, SOP and message

0529 D1F9         txmsg:    CALL               tx_on                              ; spi command to
rf
                  ic

                  ;                                    clr       CL1           ;  turn TX on
052B 75F064                 MOV       B,#100         ;  load TX delay count
052E D5F0FD       txp0:     DJNZ      B,txp0         ;  loop to delay
0531 900775                 MOV       DPTR,#tstrt    ;  point to start symbol table
0534 751D00                 MOV       TXSI,#0        ;  clear index
0537 75F006                 MOV       B,#6           ;  load 6 into loop counter
053A 751F00                 MOV       TXSMC,#0       ;  load 0 into sample count (fall through)
053D D214                   SETB      TSFLG          ;  flag to output TX samples
053F E51D         txs0:     MOV       A,TXSI         ;  load index into A
0541 051D                   INC       TXSI           ;  increment index
0543 93                     MOVC      A,@A+DPTR      ;  load table entry
0544 F525                   MOV       TMBYT,A        ;  into TMBYT
0546 752408                 MOV       TMBIC,#8       ;  load 8 into bit counter
0549 E525         txs1:     MOV       A,TMBYT        ;  load start symbol byte into A
054B 33                     RLC       A              ;  shift bit left into carry
054C F525                   MOV       TMBYT,A        ;  store shifted message byte
054E E51F         txs2:     MOV       A,TXSMC        ;  get sample count
0550 70FC                   JNZ       txs2           ;  loop until 0 (synchronize with tick isr)
0552 9215                   MOV       TXBIT,C        ;  load next TX bit
0554 751F03                 MOV       TXSMC,#3       ;  reload sample count
0557 D524EF                 DJNZ      TMBIC,txs1     ;  send 8 bits/byte
055A D5F0E2                 DJNZ      B,txs0         ;  send 6 bytes
055D 90077B       txm0:     MOV       DPTR,#tx_smbl  ;  point to symbol table;
0560 8526F0                 MOV       B,TMBYC        ;  load B with byte count
0563 E7           txm1:     MOV       A,@R1          ;  get TX message byte
0564 09                     INC       R1             ;  bump message index
0565 F525                   MOV       TMBYT,A        ;  copy byte into TMBYT
0567 540F                   ANL       A,#0FH         ;  mask upper nibble
0569 93                     MOVC      A,@A+DPTR      ;  get 6-bit symbol
056A F527                   MOV       TXSL,A         ;  move to TXSL
056C E525                   MOV       A,TMBYT        ;  get TMBYT
056E C4                     SWAP      A              ;  swap nibbles
056F 540F                   ANL       A,#0FH         ;  mask swapped lower nibble
0571 93                     MOVC      A,@A+DPTR      ;  get 6-bit symbol
0572 F528                   MOV       TXSH,A         ;  move to TXSH
0574 75240C                 MOV       TMBIC,#12      ;  set bit count to 12
0577 E524         txm2:     MOV       A,TMBIC        ;  get bit count
0579 C3                     CLR       C              ;  clear carry
057A 9407                   SUBB      A,#7           ;  subtract 7
057C 4007                   JC        txm3           ;  if < 7 jump to txm3
057E E528                   MOV       A,TXSH         ;  get high TX symbol
0580 33                     RLC       A              ;  shift left into carry
0581 F528                   MOV       TXSH,A         ;  store shifted message byte
0583 A18A                   AJMP      txm4           ;  jump to txm4
0585 E527         txm3:     MOV       A,TXSL         ;  get low TX symbol
0587 33                     RLC       A              ;  shift left into carry
0588 F527                   MOV       TXSL,A         ;  store shifted message byte
058A E51F         txm4:     MOV       A,TXSMC        ;  get sample count
058C 70FC                   JNZ       txm4           ;  loop until 0 (synchronize with tick isr)
058E 9215                   MOV       TXBIT,C        ;  load next bit
0590 751F03                 MOV       TXSMC,#3       ;  reload sample count
0593 1524                   DEC       TMBIC          ;  decrement bit count
0595 E524                   MOV       A,TMBIC        ;  get TMBIC
0597 6002                   JZ        txm5           ;  if 0 jump to txm5
0599 A177                   AJMP      txm2           ;  else loop again
059B D5F0C5       txm5:     DJNZ      B,txm1         ;  loop to send next byte
059E E51F         txm6:     MOV       A,TXSMC        ;  get sample count
05A0 70FC                   JNZ       txm6           ;  loop until 0 (synchronize with tick isr)
05A2 C214                   CLR       TSFLG          ;  clear TX sample out flag
05A4 C293                   CLR       TXPIN          ;  clear TX out pin
05A6 F104                                              CALL                   tx_off
```

```
                         ; spi commnad to rfic
                    ;          setb      CL1            ;  turn TX off
05A8 22             txm_d:    RET                       ;  TX message done

                    ; TX master reset

05A9 C20B           txmrs:    CLR       RAFLG           ;  reset RF ACK flag
05AB C20A                     CLR       SNFLG           ;  reset TX NAK flag
05AD C212                     CLR       TXFLG           ;  reset TX active flag
05AF C21B                     CLR       RHFLG           ;  reset repeat here flag
05B1 B1BA                     ACALL     txrst           ;  full reset TX
05B3 D20E                     SETB      SIFLG           ;  enable serial in
05B5 C299                     CLR       TI0             ;  clear TI0 flag
05B7 C298                     CLR       RI0             ;  clear RI flag
05B9 22             txs_d:    RET                       ;  master TX reset done


                    ; reset transmit state (retry or full)

05BA 7941           txrst:    MOV       R1,#LNTX        ;  point R1 to length byte
05BC E4                       CLR       A               ;  clear A to
05BD F525                     MOV       TMBYT,A         ;  clear TX message byte
05BF F51E                     MOV       TMFCC,A         ;  clear TX FCS count
05C1 F51F                     MOV       TXSMC,A         ;  clear TX sample count
05C3 F527                     MOV       TXSL,A          ;  clear TX symbol low
05C5 F528                     MOV       TXSH,A          ;  clear TX symbol high
05C7 201206                   JB        TXFLG,txr_d     ;  skip if TX active

                    ; full reset if TXFLG clear

05CA F526                     MOV       TMBYC,A         ;  reset TX byte count
05CC F52C                     MOV       TXCNT,A         ;  reset TX retry count
05CE F52B                     MOV       TXTH,A          ;  clear TX timer high
05D0 22             txr_d:    RET                       ;  TX reset done

                    ; build transmit FCS from each byte

05D1 75F008         b_tfcs:   MOV       B,#8            ;  load loop count of 8
05D4 C3             btf0:     CLR       C               ;  clear carry bit
05D5 E529                     MOV       A,TMFCS         ;  load TX message byte
05D7 13                       RRC       A               ;  shift lsb into carry
05D8 F529                     MOV       TMFCS,A         ;  store shifted message byte
05DA 9206                     MOV       FCSB,C          ;  load FCSB with lsb
05DC C3                       CLR       C               ;  clear carry bit
05DD ED                       MOV       A,R5            ;  load high FCS byte
05DE 13                       RRC       A               ;  shift right
05DF FD                       MOV       R5,A            ;  store shifted high FCS
05E0 EE                       MOV       A,R6            ;  load low FCS byte
05E1 13                       RRC       A               ;  shift and pull in bit for FCS high
05E2 FE                       MOV       R6,A            ;  store shifted low FCS
05E3 300601                   JNB       FCSB,btf1       ;  if lsb of low FCS = 0, jump to btf1
05E6 B3                       CPL       C               ;  else complement carry bit
05E7 5008           btf1:     JNC       btf2            ;  if FCSB XOR (low FCS lsb) = 0 jump to btf2
05E9 ED                       MOV       A,R5            ;  else load high FCS
05EA 6484                     XRL       A,#FCSH         ;  and XOR with high FCS poly
05EC FD                       MOV       R5,A            ;  store high FCS
05ED EE                       MOV       A,R6            ;  load low FCS
05EE 6408                     XRL       A,#FCSL         ;  XOR with low FCS poly
05F0 FE                       MOV       R6,A            ;  store low FCS
05F1 D5F0E0         btf2:     DJNZ      B,btf0          ;  loop through bits in message byte
05F4 22             btfcs_d:  RET                       ;  done this pass

                    ; complement and append FCS

05F5 EE             a_tfcs:   MOV       A,R6            ;  load FCS (high/low switch)
05F6 F4                       CPL       A               ;  1's complement
```

```
05F7 F7                       MOV       @R1,A          ;  store at end of TX message
05F8 09                       INC       R1             ;  increment TX message byte pointer
05F9 ED                       MOV       A,R5           ;  load FCS (high/low switch)
05FA F4                       CPL       A              ;  1's complement
05FB F7                       MOV       @R1,A          ;  store at end of TX message
05FC 7DFF                     MOV       R5,#FCSS       ;  reseed FCS high
05FE 7EFF                     MOV       R6,#FCSS       ;  reseed FCS low
0600 22             atfcs_d:  RET                      ;  add TX FCS done

                    ; initialize software

0601 C2AF           setup:    CLR       EA             ;  disable interrupts
0603 75D900         set_ck:   MOV       PCA0MD,#000H   ;  disable watchdog
0606 75B283                   MOV       OSCICN,#083H   ;  set SiLabs CPU clock to 24.5 MHz/ without is 3mhz...
0609 758E00                   MOV       CKCON,#000H    ;  divide by 12 for timers 000
060C                set_io:
060C 75A45F                   mov       P0MDOUT,#05Fh      ; was ff, 7f
060F 75A57B                   mov       P1MDOUT,#07Bh  ; was fb
0612 75E101                   mov       XBR0,#001h
0615 75E2C0                   mov       XBR1,#0C0h

0618                new_io:
0618 75A4FF             mov  P0MDOUT,   #0FFh
061B 75A502             mov  P1MDOUT,   #002h
061E 75D40F             mov  P0SKIP,    #00Fh
0621 75E103             mov  XBR0,      #003h
0624 75E240             mov  XBR1,      #040h

0627 75A45F             mov  P0MDOUT,   #05Fh
062A 75A503             mov  P1MDOUT,   #003h
062D 75D40F             mov  P0SKIP,    #00Fh
0630 75E103             mov  XBR0,      #003h
0633 75E240             mov  XBR1,      #040h

                    ;*********************************
0636                set_spio:

0636 75A170              mov  SPI0CFG,   #070h
0639 75F809              mov  SPI0CN,    #009h
063C 75A206              mov  SPI0CKR,   #006h


                    ;          setb      CL0            ;  TR RX mode
                    ;          setb      CL1            ;  TR RX mode

                    ;    mov  TMOD,      #020h
                    ;    mov  CKCON,     #008h
                    ;    mov  TH1,       #0B1h

063F C293                     CLR       TXPIN          ;  turn TX modulation off
0641 7816                     MOV       R0,#22         ;  starting above stack
0643 75F06A                   MOV       B,#106         ;  clear 106 bytes
0646 E4                       CLR       A              ;  clear A
0647 F6             clear:    MOV       @R0,A          ;  clear flags, bytes and buffers
0648 08                       INC       R0             ;  bump pointer
0649 D5F0FB                   DJNZ      B,clear        ;  loop again
064C 758922         tick_su:  MOV       TMOD,#022H     ;  set timers T0 and T1 to mode 2 022
064F C28C                     CLR       TR0            ;  stop timer T0
0651 C28D                     CLR       TF0            ;  clear T0 overflow
0653 758CE1                   MOV       TH0,#ITICK     ;  load count for tick
0656 758AE1                   MOV       TL0,#ITICK     ;  load count for tick
0659 D2B9                     SETB      PT0            ;  give T0 first interrupt priority
065B D2A9                     SETB      ET0            ;  unmask T0 interrupt
065D D28C                     SETB      TR0            ;  start timer T0
065F C28E           uart_su:  CLR       TR1            ;  stop timer T1
0661 C28F                     CLR       TF1            ;  clear T1 overflow
```

```
0663 758DCB                   MOV       TH1,#0CBH      ;  load baud rate count for 19.2 kb/s  0cb
0666 758BCB                   MOV       TL1,#0CBH      ;  load baud rate count for 19.2 kb/s  0cb
0669 759810                   MOV       SCON0,#010H    ;  enable UART mode 1
066C C2AC                     CLR       ES0            ;  mask serial interrupt
066E D28E                     SETB      TR1            ;  start baud rate timer T1
0670 D208                     SETB      SIHLD          ;  disable serial hold off (default)
0672 D20E                     SETB      SIFLG          ;  enable serial in (default)
0674 F117                     ACALL     hello          ;  send start up message
0676 7861           buf_su:   MOV       R0,#LNRX       ;  load RX buffer pointer
0678 7941                     MOV       R1,#LNTX       ;  load TX buffer pointer
067A 7A00                     MOV       R2,#0          ;  zero DCR ramp
067C 7BFF                     MOV       R3,#FCSS       ;  seed R3
067E 7DFF                     MOV       R5,#FCSS       ;  seed R5
0680 7EFF                     MOV       R6,#FCSS       ;  seed R6
0682 7FFF                     MOV       R7,#FCSS       ;  seed R7
0684 752E22                   MOV       TFBUF,#34      ;  initialize TO/FROM 2 & 2
0687 752D00                   MOV       IDBUF,#0       ;  initialize ID = 0

068A A297           rtst_su:  MOV       C,ID0          ;  read ID0
068C 4005                     JC        isr_on         ;  skip if no ID0 jumper
068E D216                     SETB      RTFLG          ;  else set RangeTest flag
0690 753102                   MOV       RTTH,#RTTM     ;  load TX timer high


0693 C204           isr_on:   CLR       SOPFLG         ;  clear SOP detect flag
0695 D200                     SETB      DCRON          ;  enable RX DCR
0697 D2AF                     SETB      EA             ;  enable interrupts
0699 C299                     CLR       TI0            ;  clear TI0 (serial byte sent) flag
069B C298                     CLR       RI0            ;  clear RI (serial byte received) flag
069D 22             setup_d:  RET                      ;  setup done

                    ;*****************************
069E                rfic:
069E 75A380                   mov             SPIODAT,        #080h

06A1 F10F                     ACALL spio_wait
06A3 75A333                   mov             SPIODAT,        #033h
                                         ;33 = 902, 23 = 868, 13 = 433
06A6 F10F                     ACALL spio_wait
06A8 75A3A3                   mov             SPIODAT,        #0a3h

06AB F10F                     ACALL spio_wait
06AD 75A363                   mov             SPIODAT,        #063h
06B0 F10F                     ACALL spio_wait
06B2 75A392                   mov             SPIODAT,        #092h

06B5 F10F                     ACALL spio_wait
06B7 75A380                   mov             SPIODAT,        #080h ;#080h    ;#0
                    a0h,  060h = 270khz bw
06BA F10F                     ACALL spio_wait
06BC 75A398                   mov             SPIODAT,        #098h

06BF F10F                     ACALL spio_wait
06C1 75A360                   mov             SPIODAT,        #060h  ;#060h  ;#02
                    0h, 080h = 135k dev
06C4 F10F                     ACALL spio_wait
06C6 75A3C4                   mov             SPIODAT,        #0c4h

06C9 F10F                     ACALL spio_wait
06CB 75A367                   mov             SPIODAT,        #067h ;#067h
06CE F10F                     ACALL spio_wait
06D0 75A3C6                   mov             SPIODAT,        #0c6h

06D3 F10F                     ACALL spio_wait
06D5 75A311                   mov             SPIODAT,        #011h
06D8 F10F                     ACALL spio_wait
```

```
06DA 75A3C2                             mov             SPIODAT,        #0c2h

06DD F10F                               ACALL spio_wait
06DF 75A33C                             mov             SPIODAT,        #03ch
06E2 F10F                               ACALL spio_wait
06E4 75A382                             mov             SPIODAT,        #082h

06E7 F10F                               ACALL spio_wait
06E9 75A309                             mov             SPIODAT,        #009h
06EC F10F                               ACALL spio_wait
06EE 75A382                             mov             SPIODAT,        #082h

06F1 F10F                               ACALL spio_wait
06F3 75A319                             mov             SPIODAT,        #019h
06F6 F10F                               ACALL spio_wait

06F8 22                                 RET


                             ;************************
06F9                         tx_on:
06F9 75A382                             mov             SPIODAT,        #082h
                                                            ;SPIO = 82d9h to rfic, rx on tx off
06FC F10F                               ACALL spio_wait
06FE 75A339                             mov             SPIODAT,        #039h
0701 F10F                               ACALL spio_wait
0703 22                                 RET

0704                         tx_off:
0704 75A382                             mov             SPIODAT,        #082h
                                   ;SPIO = 82d9h to rfic, rx on tx off
0707 F10F                               ACALL spio_wait
0709 75A3D9                             mov             SPIODAT,        #0d9h
070C F10F                               ACALL spio_wait
070E 22                                 RET

070F                         spio_wait:
070F E5F8                            mov   A,SPIOCN
0711 5402                            ANL   A,#02h
0713 B402F9                          cjne  a,#02h,spio_wait
0716 22                              RET

                             ;************************



                             ; send SW version to host

0717 9007D3                  hello:    MOV       DPTR,#table    ;  point to table
071A 75F00E                            MOV       B,#14          ;  load loop count in B
071D 7F00                              MOV       R7,#0          ;  R7 has first table entry
071F EF                      snd_h:    MOV       A,R7           ;  move table offset into A
0720 93                                MOVC      A,@A+DPTR      ;  load table byte
0721 C299                              CLR       TI0            ;  clear TI0 flag
0723 F599                              MOV       SBUF0,A        ;  send byte
0725 3099FD                  nxt_tx:   JNB       TI0,nxt_tx     ;  wait until sent
0728 0F                                INC       R7             ;  bump index
0729 D5F0F3                            DJNZ      B,snd_h        ;  loop to send message
072C 22                      hello_d:  RET                      ;  done

                             ; load RangeTest message (not used in this SW version)

072D                         hello2:
                             ;**************
072D E538                              MOV       A,fhss         ;  load ID into A
072F 04                                INC       A              ;  bump ID for next time
```

```
0730 B41801                        cjne                    a,#024,h1b
0733 E4                            clr                             a
0734 F538                h1b:      mov       fhss,a
0736 75A3A1                        mov                 SPIODAT,#0A1h  ;  Sent MSB of f
                         req
0739 F10F                          ACALL     spio_wait
073B AC38                          mov       R4,fhss
073D 0C                            inc       r4
073E E4                            clr       a
073F 240A                h1:       add       a,#10
0741 DCFC                          djnz      r4,h1
0743 F5A3                          mov                 SPIODAT,A      ; Send LSB of Fr
                         eq
0745 F10F                          ACALL     spio_wait
0747 716E                          ACALL     wait
0749 716E                          ACALL     wait
074B 716E                          ACALL     wait
                         ;***************

074D 7941                         MOV       R1,#LNTX       ;  reset TX buffer pointer
074F 9007E1                       MOV       DPTR,#tbl_2    ;  point to table 2
0752 75F00F                       MOV       B,#15          ;  load loop count in B
0755 752600                       MOV       TMBYC,#0       ;  offset for first table entry
0758 E526                snd_h2:  MOV       A,TMBYC        ;  move table offset into A
075A 93                           MOVC      A,@A+DPTR      ;  load table byte
075B F7                           MOV       @R1,A          ;  into TX buffer
075C 0526                         INC       TMBYC          ;  increment TMBYC
075E 09                           INC       R1             ;  increment R1
075F D5F0F6                       DJNZ      B,snd_h2       ;  loop to load message
0762 7941                         MOV       R1,#LNTX       ;  reset TX pointer
0764 853044                       MOV       IDTX,RTID      ;  get RangeTest ID
0767 E544                         MOV       A,IDTX         ;  load ID into A
0769 04                           INC       A              ;  bump ID for next time
076A B40701                       cjne                    a,#7,h1a
076D E4                           clr                             a
076E F530                h1a:     mov       RTID,a
0770 E538                         mov       a,fhss
0772 F545                         mov       CMTX,A

0774 22                  helo2_d: RET

                         ;  tables:

0775 AA                  tstrt:   DB        170            ;  preamble/SOP table
0776 AA                           DB        170            ;  table data
0777 AA                           DB        170            ;  table data
0778 AA                           DB        170            ;  table data
0779 E2                           DB        226            ;  table data
077A E2                           DB        226            ;  table data

077B 34                  tx_smbl: DB         52            ;  4-to-6 TX table (flush left)
077C 38                           DB         56            ;  table data
077D 4C                           DB         76            ;  table data
077E 54                           DB         84            ;  table data
077F 58                           DB         88            ;  table data
0780 64                           DB        100            ;  table data
0781 68                           DB        104            ;  table data
0782 70                           DB        112            ;  table data
0783 8C                           DB        140            ;  table data
0784 94                           DB        148            ;  table data
0785 98                           DB        152            ;  table data
0786 A4                           DB        164            ;  table data
0787 A8                           DB        168            ;  table data
0788 B0                           DB        176            ;  table data
0789 C8                           DB        200            ;  table data
078A D0                           DB        208            ;  table data
```

```
                    1671
078B FF             1672     rx_smbl:  DB        255            ;  6-to-4 RX table (flush right)
078C FF             1673               DB        255            ;  table data
078D FF             1674               DB        255            ;  table data
078E FF             1675               DB        255            ;  table data
078F FF             1676               DB        255            ;  table data
0790 FF             1677               DB        255            ;  table data
0791 FF             1678               DB        255            ;  table data
0792 FF             1679               DB        255            ;  table data
0793 FF             1680               DB        255            ;  table data
0794 FF             1681               DB        255            ;  table data
0795 FF             1682               DB        255            ;  table data
0796 FF             1683               DB        255            ;  table data
0797 FF             1684               DB        255            ;  table data
0798 00             1685               DB          0            ;  table data
0799 01             1686               DB          1            ;  table data
079A FF             1687               DB        255            ;  table data
079B FF             1688               DB        255            ;  table data
079C FF             1689               DB        255            ;  table data
079D FF             1690               DB        255            ;  table data
079E 02             1691               DB          2            ;  table data
079F FF             1692               DB        255            ;  table data
07A0 03             1693               DB          3            ;  table data
07A1 04             1694               DB          4            ;  table data
07A2 FF             1695               DB        255            ;  table data
07A3 FF             1696               DB        255            ;  table data
07A4 05             1697               DB          5            ;  table data
07A5 06             1698               DB          6            ;  table data
07A6 FF             1699               DB        255            ;  table data
07A7 07             1700               DB          7            ;  table data
07A8 FF             1701               DB        255            ;  table data
07A9 FF             1702               DB        255            ;  table data
07AA FF             1703               DB        255            ;  table data
07AB FF             1704               DB        255            ;  table data
07AC FF             1705               DB        255            ;  table data
07AD FF             1706               DB        255            ;  table data
07AE 08             1707               DB          8            ;  table data
07AF FF             1708               DB        255            ;  table data
07B0 09             1709               DB          9            ;  table data
07B1 0A             1710               DB         10            ;  table data
07B2 FF             1711               DB        255            ;  table data
07B3 FF             1712               DB        255            ;  table data
07B4 0B             1713               DB         11            ;  table data
07B5 0C             1714               DB         12            ;  table data
07B6 FF             1715               DB        255            ;  table data
07B7 0D             1716               DB         13            ;  table data
07B8 FF             1717               DB        255            ;  table data
07B9 FF             1718               DB        255            ;  table data
07BA FF             1719               DB        255            ;  table data
07BB FF             1720               DB        255            ;  table data
07BC FF             1721               DB        255            ;  table data
07BD 0E             1722               DB         14            ;  table data
07BE FF             1723               DB        255            ;  table data
07BF 0F             1724               DB         15            ;  table data
07C0 FF             1725               DB        255            ;  table data
07C1 FF             1726               DB        255            ;  table data
07C2 FF             1727               DB        255            ;  table data
07C3 FF             1728               DB        255            ;  table data
07C4 FF             1729               DB        255            ;  table data
07C5 FF             1730               DB        255            ;  table data
07C6 FF             1731               DB        255            ;  table data
07C7 FF             1732               DB        255            ;  table data
07C8 FF             1733               DB        255            ;  table data
07C9 FF             1734               DB        255            ;  table data
07CA FF             1735               DB        255            ;  table data
                    1736
```

```
                ;  TX retry delay values (34.73 us tick)

07CB 07         delay:    DB          7            ;  0.062 second
07CC 0F                   DB         15            ;  0.133 second
07CD 0B                   DB         11            ;  0.098 second
07CE 14                   DB         20            ;  0.178 second
07CF 09                   DB          9            ;  0.080 second
07D0 12                   DB         18            ;  0.160 second
07D1 0D                   DB         13            ;  0.116 second
07D2 16                   DB         22            ;  0.196 second

07D3 20         table:    DB        ' '            ;  Start-Up message
07D4 20                   DB        32            ;  table data
07D5 22                   DB        34            ;  table data
07D6 52                   DB        'R'           ;  table data
07D7 46                   DB        'F'            ;  table data
07D8 4D                   DB        'M'            ;  table data
07D9 20                   DB        ' '            ;  table data
07DA 20                   DB        ' '            ;  table data
07DB 52                   DB        'R'            ;  table data
07DC 46                   DB        'F'            ;  table data
07DD 49                   DB        'I'            ;  table data
07DE 43                   DB        'C'            ;  table data
07DF 20                   DB        ' '            ;  table data
07E0 20                   DB        ' '            ;  table data

07E1 C0         tbl_2:    DB        192            ;  RangeTest message
07E2 20                   DB         32            ;  table data
07E3 22                   DB         34            ;  table data
07E4 30                   DB        '0'            ;  table data
07E5 31                                              DB        '1'
07E6 46                   DB        'F'            ;  table data
07E7 48                   DB        'H'            ;  table data
07E8 53                   DB        'S'            ;  table data
07E9 53                   DB        'S'            ;  table data
07EA 20                   DB        ' '            ;  table data
07EB 54                   DB        'T'            ;  table data
07EC 65                   DB        'e'            ;  table data
07ED 73                   DB        's'            ;  table data
07EE 74                   DB        't'            ;  table data
07EF 20                   DB        ' '            ;  table data
07F0 20                   DB        ' '            ;  table data



                          END                     ;  end of source code
```

```
XREF SYMBOL TABLE LISTING
---- ------ ----- -------


N A M E             T Y P E  V A L U E   ATTRIBUTES / REFERENCES

AC . . . . . . . .  B ADDR   00D0H.6 A      246#
ACC. . . . . . . .  D ADDR   00E0H   A      160# 474 504 528 612 1239
ACK. . . . . . . .  B ADDR   00C0H.1 A      232#
ACKRQ. . . . . . .  B ADDR   00C0H.3 A      230#
ACKRX. . . . . . .  C ADDR   0261H   A      866#
ACKT . . . . . . .  N NUMB   0010H   A      324# 855 872 873 998 1262
AD0BUSY. . . . . .  B ADDR   00E8H.4 A      265#
AD0CM0 . . . . . .  B ADDR   00E8H.0 A      269#
AD0CM1 . . . . . .  B ADDR   00E8H.1 A      268#
AD0CM2 . . . . . .  B ADDR   00E8H.2 A      267#
AD0EN. . . . . . .  B ADDR   00E8H.7 A      262#
AD0INT . . . . . .  B ADDR   00E8H.5 A      264#
AD0TM. . . . . . .  B ADDR   00E8H.6 A      263#
AD0WINT. . . . . .  B ADDR   00E8H.3 A      266#
ADC0CF . . . . . .  D ADDR   00BCH   A      136#
ADC0CN . . . . . .  D ADDR   00E8H   A      166#
ADC0GTH. . . . . .  D ADDR   00C4H   A      143#
ADC0GTL. . . . . .  D ADDR   00C3H   A      142#
ADC0H. . . . . . .  D ADDR   00BEH   A      138#
ADC0L. . . . . . .  D ADDR   00BDH   A      137#
ADC0LTH. . . . . .  D ADDR   00C6H   A      145#
ADC0LTL. . . . . .  D ADDR   00C5H   A      144#
AD_D . . . . . . .  C ADDR   02EBH   A      947#
AD_TX. . . . . . .  C ADDR   02DBH   A      775 940#
AKH0 . . . . . . .  C ADDR   04DEH   A      1255#
AKH1 . . . . . . .  C ADDR   04E8H   A      1260# 1260
AKH2 . . . . . . .  C ADDR   04F0H   A      1263# 1263
AKH3 . . . . . . .  C ADDR   04F7H   A      1266# 1266
AKH4 . . . . . . .  C ADDR   04FFH   A      1269# 1269
AKH5 . . . . . . .  C ADDR   0507H   A      1272# 1272
AKHCM. . . . . . .  C ADDR   04D3H   A      1230 1249#
AKH_D. . . . . . .  C ADDR   0510H   A      1276#
AKS0 . . . . . . .  C ADDR   032CH   A      989 991#
AKS1 . . . . . . .  C ADDR   0336H   A      996# 996
AKS2 . . . . . . .  C ADDR   033EH   A      999# 999
AKS3 . . . . . . .  C ADDR   0345H   A      1002# 1002
AKS4 . . . . . . .  C ADDR   034DH   A      1005# 1005
AKS5 . . . . . . .  C ADDR   0355H   A      1008# 1008
AKSND. . . . . . .  C ADDR   031CH   A      721 742 983#
AKS_D. . . . . . .  C ADDR   035EH   A      1012#
AMX0N. . . . . . .  D ADDR   00BAH   A      134#
AMX0P. . . . . . .  D ADDR   00BBH   A      135#
ARBLOST. . . . . .  B ADDR   00C0H.2 A      231#
ARF0 . . . . . . .  C ADDR   03CBH   A      1088 1090 1093#
ARFCS_D. . . . . .  C ADDR   03CFH   A      1095#
ARX0 . . . . . . .  C ADDR   02A0H   A      899#
ARX_D. . . . . . .  C ADDR   02A4H   A      901#
ATFCS_D. . . . . .  C ADDR   0600H   A      1430#
A_RFCS . . . . . .  C ADDR   03BFH   A      826 1087#
A_TFCS . . . . . .  C ADDR   05F5H   A      887 1289 1421#
B. . . . . . . . .  D ADDR   00F0H   A      172# 833 837 991 1004 1016 1022 1028 1034 1060 1082 1234 1238 1255 1268
                                           1298 1299 1302 1318 1320 1352 1394 1416 1474 1478 1590 1598 1626 1633
BRF0 . . . . . . .  C ADDR   039EH   A      1061# 1082
BRF1 . . . . . . .  C ADDR   03B1H   A      1073 1075#
BRF2 . . . . . . .  C ADDR   03BBH   A      1075 1082#
BRFCS_D. . . . . .  C ADDR   03BEH   A      1083#
BTF0 . . . . . . .  C ADDR   05D4H   A      1395# 1416
BTF1 . . . . . . .  C ADDR   05E7H   A      1407 1409#
BTF2 . . . . . . .  C ADDR   05F1H   A      1409 1416#
BTFCS_D. . . . . .  C ADDR   05F4H   A      1417#
```

```
BUF01. . . . . . .  N NUMB   0032H   A      432#
BUF02. . . . . . .  N NUMB   0033H   A      433#
BUF03. . . . . . .  N NUMB   0034H   A      434#
BUF04. . . . . . .  N NUMB   0035H   A      435#
BUF05. . . . . . .  N NUMB   0036H   A      436# 594
BUF06. . . . . . .  N NUMB   0037H   A      437# 595 598 608
BUF_SU . . . . . .  C ADDR   0676H   A      1497#
B_RFCS . . . . . .  C ADDR   039BH   A      824 1060#
B_TFCS . . . . . .  C ADDR   05D1H   A      870 874 880 885 1287 1394#
CCF0 . . . . . . .  B ADDR   00D8H.0 A      259#
CCF1 . . . . . . .  B ADDR   00D8H.1 A      258#
CCF2 . . . . . . .  B ADDR   00D8H.2 A      257#
CF . . . . . . . .  B ADDR   00D8H.7 A      255#
CHAN . . . . . . .  C ADDR   00F0H   A      617#
CHAN0. . . . . . .  C ADDR   010AH   A      638#
CKCON. . . . . . .  D ADDR   008EH   A      102# 1437
CL0. . . . . . . .  B ADDR   0090H.0 A      444#
CL1. . . . . . . .  B ADDR   0090H.1 A      443#
CLEAR. . . . . . .  C ADDR   0647H   A      1476# 1478
CLKSEL . . . . . .  D ADDR   00A9H   A      125#
CLOSED . . . . . .  B ADDR   0080H.3 A      458# 606 659 671
CLOSING. . . . . .  B ADDR   0080H.0 A      461# 596 599 605 665 677 983 1011
CMRX . . . . . . .  N NUMB   0065H   A      310# 752 905
CMTX . . . . . . .  N NUMB   0045H   A      302# 1642
CPT0CN . . . . . .  D ADDR   009BH   A      114#
CPT0MD . . . . . .  D ADDR   009DH   A      115#
CPT0MX . . . . . .  D ADDR   009FH   A      116#
CR . . . . . . . .  B ADDR   00D8H.6 A      256#
CY . . . . . . . .  B ADDR   00D0H.7 A      245#
DCR. . . . . . . .  C ADDR   0012H   A      479#
DCR0 . . . . . . .  C ADDR   001AH   A      483#
DCR1 . . . . . . .  C ADDR   001EH   A      483 485#
DCR2 . . . . . . .  C ADDR   0024H   A      485 488#
DCR3 . . . . . . .  C ADDR   002EH   A      489 493#
DCR4 . . . . . . .  C ADDR   0035H   A      492 496#
DCR5 . . . . . . .  C ADDR   0041H   A      503 505#
DCR6 . . . . . . .  C ADDR   0063H   A      496 524#
DCR7 . . . . . . .  C ADDR   006CH   A      527 529#
DCRON. . . . . . .  N NUMB   0000H   A      351# 475 733 782 1101 1110 1117 1131 1138 1145 1152 1157 1514
DCR_D. . . . . . .  C ADDR   007CH   A      487 493 495 509 511 519 530 535#
DELAY. . . . . . .  C ADDR   07CBH   A      579 1739#
DO_HT. . . . . . .  C ADDR   03EAH   A      717 1116#
DO_PI. . . . . . .  C ADDR   040FH   A      727 1137#
DO_RA. . . . . . .  C ADDR   03D0H   A      711 1100#
DO_RT. . . . . . .  C ADDR   0424H   A      719 1151#
DO_RX. . . . . . .  C ADDR   017DH   A      725 732#
DPH. . . . . . . .  D ADDR   0083H   A      94#
DPL. . . . . . . .  D ADDR   0082H   A      93#
EA . . . . . . . .  B ADDR   00A8H.7 A      207# 1434 1515
EIE1 . . . . . . .  D ADDR   00E6H   A      165#
EIP1 . . . . . . .  D ADDR   00F6H   A      175#
EMI0CN . . . . . .  D ADDR   00AAH   A      126#
ES0. . . . . . . .  B ADDR   00A8H.4 A      210# 1492
ESPI0. . . . . . .  B ADDR   00A8H.6 A      208#
ET0. . . . . . . .  B ADDR   00A8H.1 A      213# 1485
ET1. . . . . . . .  B ADDR   00A8H.3 A      211#
ET2. . . . . . . .  B ADDR   00A8H.5 A      209#
EX0. . . . . . . .  B ADDR   00A8H.0 A      214#
EX1. . . . . . . .  B ADDR   00A8H.2 A      212#
F0 . . . . . . . .  B ADDR   00D0H.5 A      247#
F1 . . . . . . . .  B ADDR   00D0H.1 A      251#
FCSB . . . . . . .  N NUMB   0006H   A      357# 1065 1073 1399 1407
FCSH . . . . . . .  N NUMB   0084H   A      342# 1077 1411
FCSL . . . . . . .  N NUMB   0008H   A      343# 1080 1414
FCSS . . . . . . .  N NUMB   00FFH   A      341# 1093 1094 1428 1429 1500 1501 1502 1503
FCVH . . . . . . .  N NUMB   00F0H   A      344# 1088
```

```
FCVL . . . . . . . .  N NUMB   00B8H   A      345# 1090
FEND . . . . . . . .  N NUMB   00C0H   A      332# 921 930 955 995 1007 1165 1179 1183 1191 1197 1259 1271
FHCNT. . . . . . . .  N NUMB   003DH   A      439# 656
FHSS . . . . . . . .  N NUMB   0038H   A      438# 620 621 624 625 752 753 754 757 758 1605 1609 1612 1641
FLG1 . . . . . . . .  N NUMB   001CH   A      387#
FLG2 . . . . . . . .  N NUMB   001DH   A      388#
FLKEY. . . . . . . .  D ADDR   00B7H   A      131#
FLSCL. . . . . . . .  D ADDR   00B6H   A      130#
H1 . . . . . . . . .  C ADDR   073FH   A      1615# 1616
H1A. . . . . . . . .  C ADDR   076EH   A      1638 1640#
H1B. . . . . . . . .  C ADDR   0734H   A      1607 1609#
H2 . . . . . . . . .  C ADDR   01B2H   A      761# 762
H2A. . . . . . . . .  C ADDR   0103H   A      628# 629
H3 . . . . . . . . .  C ADDR   0143H   A      696# 697
H3Z. . . . . . . . .  C ADDR   0113H   A      644# 645
H4 . . . . . . . . .  C ADDR   01ACH   A      755 757#
H4A. . . . . . . . .  C ADDR   00FDH   A      622 624#
HCFLG. . . . . . . .  N NUMB   0011H   A      371# 1120 1213 1225 1244
HELLO. . . . . . . .  C ADDR   0717H   A      1496 1589#
HELLO2 . . . . . . .  C ADDR   072DH   A      1104 1603#
HELLO_D. . . . . . .  C ADDR   072CH   A      1599#
HELO2_D. . . . . . .  C ADDR   0774H   A      1644#
HT0. . . . . . . . .  C ADDR   03F9H   A      1120 1123#
HT1. . . . . . . . .  C ADDR   0406H   A      1123 1129#
HT2. . . . . . . . .  C ADDR   0408H   A      1128 1130#
HT3. . . . . . . . .  C ADDR   040AH   A      1122 1131#
HTC0 . . . . . . . .  C ADDR   04BCH   A      1234# 1239
HTC1 . . . . . . . .  C ADDR   04BFH   A      1235# 1238
HTC2 . . . . . . . .  C ADDR   04CEH   A      1243#
HTC3 . . . . . . . .  C ADDR   04D0H   A      1244#
HTCMD. . . . . . . .  C ADDR   04B4H   A      1121 1230#
HTCT . . . . . . . .  N NUMB   0040H   A      326# 1211
HTC_D. . . . . . . .  C ADDR   04D2H   A      1245#
HTFLG. . . . . . . .  N NUMB   0010H   A      370# 1123 1212 1224
HTG0 . . . . . . . .  C ADDR   0442H   A      1170# 1182
HTG1 . . . . . . . .  C ADDR   0447H   A      1172#
HTG2 . . . . . . . .  C ADDR   0454H   A      1179 1181#
HTG3 . . . . . . . .  C ADDR   045BH   A      1180 1184#
HTG4 . . . . . . . .  C ADDR   0464H   A      1186 1188#
HTG5 . . . . . . . .  C ADDR   046DH   A      1192# 1192
HTG6 . . . . . . . .  C ADDR   0475H   A      1195# 1195
HTG7 . . . . . . . .  C ADDR   047DH   A      1198# 1198
HTG8 . . . . . . . .  C ADDR   0484H   A      1165 1187 1201#
HTGET. . . . . . . .  C ADDR   0435H   A      1118 1163#
HTG_D. . . . . . . .  C ADDR   0489H   A      1200 1203#
HTS0 . . . . . . . .  C ADDR   0305H   A      967#
HTS1 . . . . . . . .  C ADDR   0307H   A      968# 972
HTS2 . . . . . . . .  C ADDR   0309H   A      969# 969
HTS4 . . . . . . . .  C ADDR   0317H   A      973 977#
HTSND. . . . . . . .  C ADDR   02ECH   A      778 951#
HTS_D. . . . . . . .  C ADDR   031BH   A      979#
HTT0 . . . . . . . .  C ADDR   049BH   A      1211 1215#
HTT1 . . . . . . . .  C ADDR   04A9H   A      1216 1220 1222#
HTTYP. . . . . . . .  C ADDR   048AH   A      1119 1207#
HTT_D. . . . . . . .  C ADDR   04B3H   A      1208 1214 1221 1226#
HT_D . . . . . . . .  C ADDR   040EH   A      1133#
ID0. . . . . . . . .  B ADDR   0090H.7 A      455# 1507
IDA0CN . . . . . . .  D ADDR   00B9H   A      133#
IDA0H. . . . . . . .  D ADDR   0097H   A      111#
IDA0L. . . . . . . .  D ADDR   0096H   A      110#
IDAK . . . . . . . .  N NUMB   003CH   A      295#
IDBUF. . . . . . . .  N NUMB   002DH   A      425# 859 945 985 1223 1250 1505
IDRX . . . . . . . .  N NUMB   0064H   A      309# 882 943
IDTX . . . . . . . .  N NUMB   0044H   A      301# 943 945 1223 1635 1636
IE . . . . . . . . .  D ADDR   00A8H   A      124#
IE0. . . . . . . . .  B ADDR   0088H.1 A      194#
```

```
IE1. . . . . . . .  B ADDR   0088H.3 A      192#
IP . . . . . . . .  D ADDR   00B8H   A      132#
ISR_ON . . . . . .  C ADDR   0693H   A      1508 1513#
IT0. . . . . . . .  B ADDR   0088H.0 A      195#
IT01CF . . . . . .  D ADDR   00E4H   A      164#
IT1. . . . . . . .  B ADDR   0088H.2 A      193#
ITICK. . . . . . .  N NUMB   00E1H   A      283# 1482 1483
LD0. . . . . . . .  C ADDR   02C7H   A      925# 929
LD1. . . . . . . .  C ADDR   02D6H   A      934#
LD_D . . . . . . .  C ADDR   02DAH   A      936#
LD_TX. . . . . . .  C ADDR   02B6H   A      774 916#
LED. . . . . . . .  C ADDR   035FH   A      660 662 664 666 667 668 669 670 672 674 676 681 682 683 687 688 689 780
                                           1016# 1240
LED0 . . . . . . .  C ADDR   0364H   A      1018# 1022 1023
LED_D. . . . . . .  C ADDR   036DH   A      1024#
LNAK . . . . . . .  N NUMB   0039H   A      292# 867 888
LNRX . . . . . . .  N NUMB   0061H   A      306# 802 815 919 931 954 1047 1497
LNTX . . . . . . .  N NUMB   0041H   A      298# 897 920 932 1166 1184 1201 1290 1376 1498 1624 1634
LRXSM. . . . . . .  N NUMB   0002H   A      353# 480 483
MAIN . . . . . . .  C ADDR   014DH   A      706# 723 724 726 728
MASTER . . . . . .  B ADDR   00C0H.7 A      226#
MCE0 . . . . . . .  B ADDR   0098H.5 A      199#
MN0. . . . . . . .  C ADDR   0154H   A      710 713#
MN1. . . . . . . .  C ADDR   0162H   A      713 714 715 716 718#
MN2. . . . . . . .  C ADDR   0167H   A      718 720#
MN3. . . . . . . .  C ADDR   016EH   A      712 720 723#
MN4. . . . . . . .  C ADDR   0176H   A      726#
MN_D . . . . . . .  C ADDR   017BH   A      728#
MODF . . . . . . .  B ADDR   00F8H.5 A      274#
MSGT . . . . . . .  N NUMB   0020H   A      325# 849 1215
NEW_IO . . . . . .  C ADDR   0618H   A      1444#
NHFLG. . . . . . .  N NUMB   0009H   A      360# 973
NSSMD0 . . . . . .  B ADDR   00F8H.2 A      277#
NSSMD1 . . . . . .  B ADDR   00F8H.3 A      276#
NXT_TX . . . . . .  C ADDR   0725H   A      1596# 1596
OKFLG. . . . . . .  N NUMB   0007H   A      358# 735 1048 1092
OPEN . . . . . . .  B ADDR   0080H.2 A      460# 663 675 866 900 1103 1111
OPENING. . . . . .  B ADDR   0080H.1 A      459# 661 673
OSCICL . . . . . .  D ADDR   00B3H   A      129#
OSCICN . . . . . .  D ADDR   00B2H   A      128# 1436
OSCLCN . . . . . .  D ADDR   00E3H   A      163#
OSCXCN . . . . . .  D ADDR   00B1H   A      127#
OV . . . . . . . .  B ADDR   00D0H.2 A      250#
P. . . . . . . . .  B ADDR   00D0H.0 A      252#
P0 . . . . . . . .  D ADDR   0080H   A      91# 451 452 453 458 459 460 461
P0MDIN . . . . . .  D ADDR   00F1H   A      173#
P0MDOUT. . . . . .  D ADDR   00A4H   A      121# 1439 1445 1451
P0SKIP . . . . . .  D ADDR   00D4H   A      153# 1447 1453
P1 . . . . . . . .  D ADDR   0090H   A      104# 443 444 447 448 455
P1MDIN . . . . . .  D ADDR   00F2H   A      174#
P1MDOUT. . . . . .  D ADDR   00A5H   A      122# 1440 1446 1452
P1SKIP . . . . . .  D ADDR   00D5H   A      154#
P2 . . . . . . . .  D ADDR   00A0H   A      117#
P2MDOUT. . . . . .  D ADDR   00A6H   A      123#
PCA0CN . . . . . .  D ADDR   00D8H   A      155#
PCA0CPH0 . . . . .  D ADDR   00FCH   A      180#
PCA0CPH1 . . . . .  D ADDR   00EAH   A      168#
PCA0CPH2 . . . . .  D ADDR   00ECH   A      170#
PCA0CPL0 . . . . .  D ADDR   00FBH   A      179#
PCA0CPL1 . . . . .  D ADDR   00E9H   A      167#
PCA0CPL2 . . . . .  D ADDR   00EBH   A      169#
PCA0CPM0 . . . . .  D ADDR   00DAH   A      157#
PCA0CPM1 . . . . .  D ADDR   00DBH   A      158#
PCA0CPM2 . . . . .  D ADDR   00DCH   A      159#
PCA0H. . . . . . .  D ADDR   00FAH   A      178#
PCA0L. . . . . . .  D ADDR   00F9H   A      177#
```

```
PCA0MD . . . . . .  D ADDR   00D9H   A      156# 1435
PCON . . . . . . .  D ADDR   0087H   A      95#
PCRCV. . . . . . .  B ADDR   0080H.2 A      451# 916 935 951 978 984 1010 1102 1106 1116 1132 1137 1146 1151 1158
                                           1249 1275
PIFLG. . . . . . .  N NUMB   0018H   A      381# 716 741 772 907 910
PI_D . . . . . . .  C ADDR   0423H   A      1147#
PMFLG. . . . . . .  N NUMB   0019H   A      382# 726 946 1144
PS0. . . . . . . .  B ADDR   00B8H.4 A      219#
PSCTL. . . . . . .  D ADDR   008FH   A      103#
PSPI0. . . . . . .  B ADDR   00B8H.6 A      217#
PSW. . . . . . . .  D ADDR   00D0H   A      151# 473 613
PT0. . . . . . . .  B ADDR   00B8H.1 A      222# 1484
PT1. . . . . . . .  B ADDR   00B8H.3 A      220#
PT2. . . . . . . .  B ADDR   00B8H.5 A      218#
PX0. . . . . . . .  B ADDR   00B8H.0 A      223#
PX1. . . . . . . .  B ADDR   00B8H.2 A      221#
RAFLG. . . . . . .  N NUMB   000BH   A      363# 740 861 989 1363
RA_D . . . . . . .  C ADDR   03E9H   A      1100 1112#
RB80 . . . . . . .  B ADDR   0098H.2 A      202#
RCFLG. . . . . . .  N NUMB   000DH   A      365# 769 853 911 1051
REF0CN . . . . . .  D ADDR   00D1H   A      152#
REN0 . . . . . . .  B ADDR   0098H.4 A      200#
RESET. . . . . . .  C ADDR   0000H   A      467#
RFIC . . . . . . .  C ADDR   069EH   A      679 1521#
RFRCV. . . . . . .  B ADDR   0080H.2 A      452# 899 934 977 1009 1054 1091 1231 1241
RHFLG. . . . . . .  N NUMB   001BH   A      385# 842 1366
RI0. . . . . . . .  B ADDR   0098H.0 A      204# 713 784 1164 1175 1274 1370 1517
RMBYC. . . . . . .  N NUMB   0019H   A      408# 806 811 821 917 918 924 929 952 953 958 960 962 964 966 972 1045
RMFCC. . . . . . .  N NUMB   001CH   A      411# 807 812 814 821 825 1046
RMFCS. . . . . . .  N NUMB   001AH   A      409# 822 1062 1064
RMFLG. . . . . . .  N NUMB   000CH   A      364# 777 850 1050
RMSBC. . . . . . .  N NUMB   001BH   A      410# 513 530 533
RS0. . . . . . . .  B ADDR   00D0H.3 A      249#
RS1. . . . . . . .  B ADDR   00D0H.4 A      248#
RSFLG. . . . . . .  N NUMB   0017H   A      379# 568 710 1109
RSFLG2 . . . . . .  N NUMB   001FH   A      390#
RSTSRC . . . . . .  D ADDR   00EFH   A      171#
RSVT . . . . . . .  N NUMB   0000H   A      323#
RTFLG. . . . . . .  N NUMB   0016H   A      378# 566 601 724 736 779 1509
RTFLG1 . . . . . .  N NUMB   001EH   A      389#
RTID . . . . . . .  N NUMB   0030H   A      429# 1635 1640
RTST_SU. . . . . .  C ADDR   068AH   A      1507#
RTTH . . . . . . .  N NUMB   0031H   A      430# 567 569 1510
RTTM . . . . . . .  N NUMB   0002H   A      337# 569 1510
RT_D . . . . . . .  C ADDR   0434H   A      1159#
RX0. . . . . . . .  C ADDR   0198H   A      741 743#
RX1. . . . . . . .  C ADDR   019CH   A      740 745#
RX2. . . . . . . .  C ADDR   01C1H   A      769 772#
RX3. . . . . . . .  C ADDR   01CDH   A      772 773 777#
RX4. . . . . . . .  C ADDR   01D2H   A      735 736 738 744 771 776 777 779#
RX5. . . . . . . .  C ADDR   01D7H   A      779 781#
RXBB . . . . . . .  N NUMB   0018H   A      407# 531 791 797 1044
RXBH . . . . . . .  N NUMB   0017H   A      406# 506 508 794 799 800 801 803 1042
RXBIT. . . . . . .  N NUMB   0003H   A      354# 491 503 527
RXBL . . . . . . .  N NUMB   0016H   A      405# 500 505 510 512 524 529 531 532 1043
RXC0 . . . . . . .  C ADDR   02AEH   A      906 909#
RXC1 . . . . . . .  C ADDR   02B3H   A      908 909 911#
RXCMD. . . . . . .  C ADDR   02A5H   A      770 905#
RXCT . . . . . . .  N NUMB   0080H   A      327# 852 1219
RXC_D. . . . . . .  C ADDR   02B5H   A      912#
RXF0 . . . . . . .  C ADDR   021DH   A      822# 825
RXFCS. . . . . . .  C ADDR   021AH   A      734 821#
RXF_D. . . . . . .  C ADDR   0227H   A      827#
RXI. . . . . . . .  B ADDR   0080H.3 A      453# 518 816 1055 1232 1242
RXM2 . . . . . . .  C ADDR   01EBH   A      795# 795
RXM3 . . . . . . .  C ADDR   020BH   A      805 811#
```

```
RXM4 . . . . . . .  C ADDR   0211H   A      802 810 813#
RXMB . . . . . . .  N NUMB   0060H   A      305#
RXMSG. . . . . . .  C ADDR   01E0H   A      732 789# 789 814
RXM_D. . . . . . .  C ADDR   0219H   A      817#
RXOVRN . . . . . .  B ADDR   00F8H.4 A      275#
RXPIN. . . . . . .  B ADDR   0090H.4 A      447# 481
RXRST. . . . . . .  C ADDR   037DH   A      781 1041#
RXR_D. . . . . . .  C ADDR   039AH   A      1056#
RXSFLG . . . . . .  N NUMB   0005H   A      356# 515 534 609 789 790 795 796
RXSMP. . . . . . .  N NUMB   0001H   A      352# 479 482 490
RXT0 . . . . . . .  C ADDR   024CH   A      849 852#
RXT1 . . . . . . .  C ADDR   0253H   A      852 855#
RXTO . . . . . . .  C ADDR   0228H   A      737 831#
RXTO_D . . . . . .  C ADDR   0240H   A      837 841 843#
RXTYP. . . . . . .  C ADDR   0241H   A      739 847#
RXT_D. . . . . . .  C ADDR   0260H   A      851 854 855 858 862#
RX_D . . . . . . .  C ADDR   01DFH   A      785#
RX_SMBL. . . . . .  C ADDR   078BH   A      514 1672#
S0MODE . . . . . .  B ADDR   0098H.7 A      198#
SBUF0. . . . . . .  D ADDR   0099H   A      113# 968 995 998 1001 1004 1007 1163 1174 1191 1194 1197 1259 1262 1265
                                           1268 1271 1595
SCON0. . . . . . .  D ADDR   0098H   A      112# 1491
SETUP. . . . . . .  C ADDR   0601H   A      654 1434#
SETUP_D. . . . . .  C ADDR   069DH   A      1518#
SET_CK . . . . . .  C ADDR   0603H   A      1435#
SET_IO . . . . . .  C ADDR   060CH   A      1438#
SET_SPIO . . . . .  C ADDR   0636H   A      1458#
SI . . . . . . . .  B ADDR   00C0H.0 A      233#
SIFLG. . . . . . .  N NUMB   000EH   A      367# 715 933 1129 1188 1243 1368 1495
SIHLD. . . . . . .  N NUMB   0008H   A      359# 516 714 1053 1494
SMB0CF . . . . . .  D ADDR   00C1H   A      140#
SMB0CN . . . . . .  D ADDR   00C0H   A      139#
SMB0DAT. . . . . .  D ADDR   00C2H   A      141#
SND_H. . . . . . .  C ADDR   071FH   A      1592# 1598
SND_H2 . . . . . .  C ADDR   0758H   A      1628# 1633
SNFLG. . . . . . .  N NUMB   000AH   A      361# 590 720 1364
SOPFLG . . . . . .  N NUMB   0004H   A      355# 496 517 535 602 723 1052 1513
SOPH . . . . . . .  N NUMB   00E2H   A      335# 509
SOPL . . . . . . .  N NUMB   00E2H   A      334# 511
SP . . . . . . . .  D ADDR   0081H   A      92#
SPI0CFG. . . . . .  D ADDR   00A1H   A      118# 1460
SPI0CKR. . . . . .  D ADDR   00A2H   A      119# 1462
SPI0CN . . . . . .  D ADDR   00F8H   A      176# 1461
SPI0DAT. . . . . .  D ADDR   00A3H   A      120#
SPIEN. . . . . . .  B ADDR   00F8H.0 A      279#
SPIF . . . . . . .  B ADDR   00F8H.7 A      272#
SPIOCFG. . . . . .  N NUMB   00A1H   A      315#
SPIOCKR. . . . . .  N NUMB   00A2H   A      316#
SPIOCN . . . . . .  N NUMB   00F8H   A      314# 1578
SPIODAT. . . . . .  N NUMB   00A3H   A      317# 617 630 639 646 691 698 750 763 1522 1524 1526 1528 1530 1532 1534
                                           1536 1538 1540 1542 1544 1546 1548 1550 1552 1554 1556 1564 1566 1571 1573
                                           1610 1617
SPIO_WAIT. . . . .  C ADDR   070FH   A      618 640 647 692 699 751 764 1523 1525 1527 1529 1531 1533 1535 1537 1539
                                           1541 1543 1545 1547 1549 1551 1553 1555 1557 1565 1567 1572 1574 1577#
                                           1580 1611 1618
STA. . . . . . . .  B ADDR   00C0H.5 A      228#
START. . . . . . .  C ADDR   00FFH   A      467 654#
STO. . . . . . . .  B ADDR   00C0H.4 A      229#
STX. . . . . . . .  N NUMB   0002H   A      329#
T2SPLIT. . . . . .  B ADDR   00C8H.3 A      240#
T2XCLK . . . . . .  B ADDR   00C8H.0 A      242#
TABLE. . . . . . .  C ADDR   07D3H   A      1589 1748#
TACK . . . . . . .  N NUMB   0006H   A      330#
TB80 . . . . . . .  B ADDR   0098H.3 A      201#
TBL_2. . . . . . .  C ADDR   07E1H   A      1625 1763#
TCON . . . . . . .  D ADDR   0088H   A      96#
```

```
TEMPB. . . . . . .  N NUMB   002FH   A      427# 889 898
TF0. . . . . . . .  B ADDR   0088H.5 A      190# 1481
TF1. . . . . . . .  B ADDR   0088H.7 A      188# 1488
TF2CEN . . . . . .  B ADDR   00C8H.4 A      239#
TF2H . . . . . . .  B ADDR   00C8H.7 A      236#
TF2L . . . . . . .  B ADDR   00C8H.6 A      237#
TF2LEN . . . . . .  B ADDR   00C8H.5 A      238#
TFAK . . . . . . .  N NUMB   003BH   A      294#
TFBUF. . . . . . .  N NUMB   002EH   A      426# 831 856 944 992 1222 1256 1504
TFRX . . . . . . .  N NUMB   0063H   A      308# 834 858 876 940
TFTX . . . . . . .  N NUMB   0043H   A      300# 942 944 1222
TH0. . . . . . . .  D ADDR   008CH   A      100# 1482
TH1. . . . . . . .  D ADDR   008DH   A      101# 1489
THFLG. . . . . . .  N NUMB   001AH   A      384# 738 838 1049
TI0. . . . . . . .  B ADDR   0098H.1 A      203# 783 967 969 970 976 994 996 997 999 1000 1002 1003 1005 1006 1008
                                           1190 1192 1193 1195 1196 1198 1199 1258 1260 1261 1263 1264 1266 1267 1269
                                           1270 1272 1273 1369 1516 1594 1596
TIC0 . . . . . . .  C ADDR   007FH   A      475 539#
TIC1 . . . . . . .  C ADDR   008CH   A      539 541 548#
TIC2 . . . . . . .  C ADDR   009BH   A      548 551 557#
TIC2_C . . . . . .  C ADDR   00A1H   A      562#
TIC3 . . . . . . .  C ADDR   00AFH   A      565 566 567 573#
TICK . . . . . . .  C ADDR   000BH   A      473#
TICK_D . . . . . .  C ADDR   00CAH   A      535 559 573 574 588 594#
TICK_D1. . . . . .  C ADDR   00E8H   A      594 595 601 604 609#
TICK_D2. . . . . .  C ADDR   00E0H   A      596 605#
TICK_D3. . . . . .  C ADDR   00EBH   A      609 611#
TICK_SU. . . . . .  C ADDR   064CH   A      1479#
TL0. . . . . . . .  D ADDR   008AH   A      98# 1483
TL1. . . . . . . .  D ADDR   008BH   A      99# 580 1490
TMBIC. . . . . . .  N NUMB   0024H   A      416# 1309 1317 1332 1333 1348 1349
TMBYC. . . . . . .  N NUMB   0026H   A      418# 889 890 898 918 1169 1178 1181 1185 1202 1207 1280 1281 1282 1320
                                           1387 1627 1628 1631
TMBYT. . . . . . .  N NUMB   0025H   A      417# 893 1308 1310 1312 1323 1327 1378
TMFCC. . . . . . .  N NUMB   001EH   A      414# 549 550 553 1170 1282 1283 1284 1288 1379
TMFCS. . . . . . .  N NUMB   0029H   A      421# 869 873 879 884 1285 1396 1398
TMFLG. . . . . . .  N NUMB   0013H   A      374# 575 589 718 1156
TMOD . . . . . . .  D ADDR   0089H   A      97# 1479
TMR2CN . . . . . .  D ADDR   00C8H   A      146#
TMR2H. . . . . . .  D ADDR   00CDH   A      150#
TMR2L. . . . . . .  D ADDR   00CCH   A      149#
TMR2RLH. . . . . .  D ADDR   00CBH   A      148#
TMR2RLL. . . . . .  D ADDR   00CAH   A      147#
TMR3CN . . . . . .  D ADDR   0091H   A      105#
TMR3H. . . . . . .  D ADDR   0095H   A      109#
TMR3L. . . . . . .  D ADDR   0094H   A      108#
TMR3RLH. . . . . .  D ADDR   0093H   A      107#
TMR3RLL. . . . . .  D ADDR   0092H   A      106#
TNAK . . . . . . .  N NUMB   0015H   A      331#
TOFLG. . . . . . .  N NUMB   000FH   A      368# 548 552 1171 1189
TPAK . . . . . . .  N NUMB   003AH   A      293#
TPRX . . . . . . .  N NUMB   0062H   A      307# 839 847
TPTX . . . . . . .  N NUMB   0042H   A      299# 1209 1217
TR0. . . . . . . .  B ADDR   0088H.4 A      191# 1480 1486
TR1. . . . . . . .  B ADDR   0088H.6 A      189# 1487 1493
TR2. . . . . . . .  B ADDR   00C8H.2 A      241#
TSFLG. . . . . . .  N NUMB   0014H   A      375# 539 565 1304 1355
TSTRT. . . . . . .  C ADDR   0775H   A      1300 1648#
TXBIT. . . . . . .  N NUMB   0015H   A      376# 542 1315 1346
TXBMT. . . . . . .  B ADDR   00F8H.1 A      278#
TXCNT. . . . . . .  N NUMB   002CH   A      424# 587 988 1127 1142 1154 1388
TXF0 . . . . . . .  C ADDR   051CH   A      1285# 1288
TXFCS. . . . . . .  C ADDR   0511H   A      1105 1124 1139 1280#
TXFLG. . . . . . .  N NUMB   0012H   A      373# 573 773 1100 1126 1141 1365 1383
TXF_D. . . . . . .  C ADDR   0528H   A      1291#
TXM0 . . . . . . .  C ADDR   055DH   A      1319#
```

```
TXM1 . . . . . . .  C ADDR   0563H   A      1321# 1352
TXM2 . . . . . . .  C ADDR   0577H   A      1333# 1351
TXM3 . . . . . . .  C ADDR   0585H   A      1336 1341#
TXM4 . . . . . . .  C ADDR   058AH   A      1340 1344# 1345
TXM5 . . . . . . .  C ADDR   059BH   A      1350 1352#
TXM6 . . . . . . .  C ADDR   059EH   A      1353# 1354
TXMB . . . . . . .  N NUMB   0040H   A      297#
TXMODE . . . . . .  B ADDR   00C0H.6 A      227#
TXMRS. . . . . . .  C ADDR   05A9H   A      722 743 1363#
TXMSG. . . . . . .  C ADDR   0529H   A      891 1107 1125 1140 1153 1295#
TXM_D. . . . . . .  C ADDR   05A8H   A      1359#
TXP0 . . . . . . .  C ADDR   052EH   A      1299# 1299
TXPIN. . . . . . .  B ADDR   0090H.3 A      448# 543 1356 1472
TXRST. . . . . . .  C ADDR   05BAH   A      1108 1130 1143 1155 1367 1376#
TXR_D. . . . . . .  C ADDR   05D0H   A      1383 1390#
TXS0 . . . . . . .  C ADDR   053FH   A      1305# 1318
TXS1 . . . . . . .  C ADDR   0549H   A      1310# 1317
TXS2 . . . . . . .  C ADDR   054EH   A      1313# 1314
TXSH . . . . . . .  N NUMB   0028H   A      420# 896 1331 1337 1339 1382
TXSI . . . . . . .  N NUMB   001DH   A      413# 1301 1305 1306
TXSL . . . . . . .  N NUMB   0027H   A      419# 895 1326 1341 1343 1381
TXSMC. . . . . . .  N NUMB   001FH   A      415# 540 544 894 1303 1313 1316 1344 1347 1353 1380
TXS_D. . . . . . .  C ADDR   05B9H   A      1371#
TXTH . . . . . . .  N NUMB   002BH   A      423# 574 583 1389
TXTL . . . . . . .  N NUMB   002AH   A      422# 557 558
TX_OFF . . . . . .  C ADDR   0704H   A      685 1357 1570#
TX_ON. . . . . . .  C ADDR   06F9H   A      1295 1563#
TX_SMBL. . . . . .  C ADDR   077BH   A      1319 1655#
UART_SU. . . . . .  C ADDR   065FH   A      1487#
VDM0CN . . . . . .  D ADDR   00FFH   A      181#
WAIT . . . . . . .  C ADDR   036EH   A      648 700 1028# 1619 1620 1621
WAIT0. . . . . . .  C ADDR   0373H   A      1030# 1034 1035
WAIT_D . . . . . .  C ADDR   037CH   A      1036#
WCOL . . . . . . .  B ADDR   00F8H.6 A      273#
XBR0 . . . . . . .  D ADDR   00E1H   A      161# 1441 1448 1454
XBR1 . . . . . . .  D ADDR   00E2H   A      162# 1442 1449 1455


REGISTER BANK(S) USED: 0


ASSEMBLY COMPLETE.  0 WARNING(S), 0 ERROR(S)
```

Bob Nelson
RF Monolithics
2/7/2007