12
Radio transmitters and receivers

Radio circuits used to frighten me but now with the introduction of low cost modules the radio novice like myself can transmit data easily.

This section details the use of the 418 MHz Radio Transmitter and Receiver Modules (RT1-418 and RR3-418). They do not need a license to operate and there are many varieties available. The transmitters only have 3 connections, 2 power supply and one data input, the transmitting aerial is incorporated on the unit. The receiver has 4 connections, 2 power supply, 1 aerial input and 1 data output. The receiving aerial only needs to be a piece of wire about 25cm long.

The basic circuit diagram of the radio system is shown in Figure 12.1.

The microcontroller generates the data and then passes the data pulses to the transmitter. The receiver receives the data pulses and a microcontroller decodes the information and processes it.

A microcontroller-radio system could measure the temperature outside and transmit this temperature to be displayed on a unit inside.

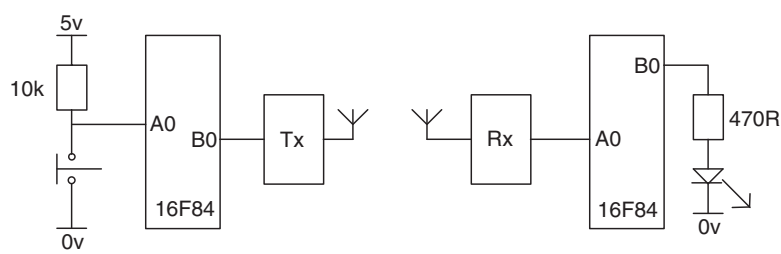

Figure 12.1 Radio data transmission system

How does it work?

The transmitter

Data is generated by the microcontroller say by pressing a switch or from a temperature sensor via the 16F818 doing an A/D conversion. Suppose this data is 27H, this would then be stored in a user file, called, say, NUMA.

So file NUMA would appear as shown in Figure 12.2.

NUMA,7	NUMA,6	NUMA,5	NUMA,4	NUMA,3	NUMA,2	NUMA,1	NUMA,0
0	0	1	0	0	1	1	1

Figure 12.2 File NUMA containing 27H

The data then needs to be passed from the micro to the data input of the transmitter. The transmitter output will then be turned on and off by the data pulses. The length of time the transmitter is on will indicate if the data was a 1, a 0 or the transmission start pulse.

I have decided to use a start bit that is 7.5ms wide, a 5ms pulse to represent a logic 1 and a 2.5ms pulse to represent a logic 0. All pulses are separated by a space of 2.5ms. The pulse train for NUMA is then as shown in Figure 12.3.

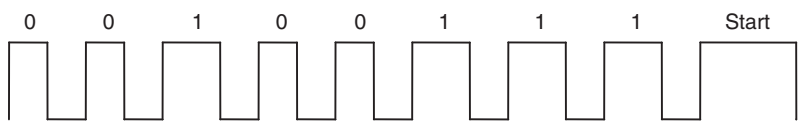

Figure 12.3 NUMA pulse train

In order to generate this train the software turns the output on for the 7.5ms start pulse, off for 2.5ms, on for 5ms for the first 1, off for 2.5ms, on for 5ms for the next logic 1, off for 2.5ms, on for 5ms for the next logic 1, off for 2.5ms, on for 2.5ms for the logic 0, etc.

To generate the data each bit in the file NUMA is tested in turn. If the bit is 0 then the output is turned on for 2.5ms, if the bit is 1 then the output is turned on for 5ms. The code for this data would be:

```
BSF     PORTB,0    ;Transmit start pulse
CALL    DELAY3     ;7.5ms Start pulse
BCF     PORTB,0    ;Transmit space
CALL    DELAY1     :Delay 2.5ms
```

```
TESTA0    BTFSC    NUMA,0    ;Test NUMA,0
          GOTO     SETA0     ;If NUMA0 = 1
          GOTO     CLRA0     ;If NUMA0 = 0

SETA0     BSF      PORTB,0   ;Transmit 1
          CALL     DELAY2    ;Delay 5ms
          GOTO     TESTA1

CLRA0     BSF      PORTB,0   ;Transmit 0
          CALL     DELAY1    ;Delay 2.5ms
          GOTO     TESTA1

TEASTA1   BCF      PORTB,0   ;Transmit space
          CALL     DELAY1
          BTFSC    NUMA,1    ;Test NUMA,1
          GOTO     SETA1     ;If NUMA0 = 1
          GOTO     CLRA1     ;If NUMA0 = 0

SETA1     BSF      PORTB,0
          CALL     DELAY2
          GOTO     TESTA2

CLRA1     BSF      PORTB,0
          CALL     DELAY1
          GOTO     TESTA2
```

•
•
•

This bit testing is repeated until all 8 bits are transmitted.

The receiver

The receiver works the opposite way round. The data is received and stored in a file NUMA. Several data bytes could be transmitted depending on how many switches are used. Or the data may be continually varying from a temperature sensor. In this example we are only looking for one byte i.e. the number 27H which was transmitted. The data is passed from the receiver to the input A0 of the microcontroller.

We wait to receive the 7.5ms start bit. When this is detected we then measure the next 8 pulses.

If a pulse is 5ms wide then a one has been transmitted and we SET the relative bit in the file NUMA. If the pulse is only 2.5ms long then we leave the bit CLEAR.

Measuring the received pulse width

Measuring the width of a pulse is a little more difficult than setting a pulse width. Consider the pulse in Figure 12.4.

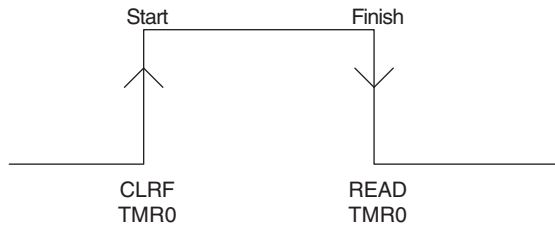

Figure 12.4 Measuring the width of a pulse

The input is continually tested until it goes high and then the timer, TMR0, is cleared to start timing. The input is continually tested until it goes low and then the value of TMR0 is read. This is done by:

MOVF TMR0,W which puts the value of TMR0 into W.

We can then check to see if the pulse is 5ms long i.e. a logic 1, if not then a shorter pulse means a logic 0 was transmitted. If the pulse is greater than 3.5ms then it must be a logic1, at 5ms. If the pulse is less than 3.5ms then it must be a logic0. TMR0 will hold a value of 3 after a time of 3.5ms, so we check to see if the width of the pulse is greater or less than 3.

The code for this is:

```
TESTA0H   BTFSS    PORTA,0    ;wait for Hi transmission
          GOTO     TESTA0H
          CLRF     TMR0       ;start timing
TESTA0L   BTFSC    PORTA,0    ;wait for Lo transmission
          GOTO     TESTA0L
          MOVF     TMR0,W     ;read value of TMR0
          SUBLW    .3         ;3-W or 3-TMR0
          BTFSC    STATUS,
                   CARRY      ;Is TMR0 > 3 i.e. a logic1
          BSF      NUMA,0     ;Yes.
```

•
•
•

This measuring of the pulse width continues until all 8 pulses are read and the relevant bits stored in the file NUMA. A TMR0 value >6 indicates the pulse was a Start pulse.

We then check to see if the number stored in the file NUMA is 27H. This is done as we have done before by subtracting 27H from it, if the answer is zero, i.e. $27-27=0$, then the number transmitted was 27H and we turn on the LED. It seems such a waste to go to all this trouble to turn an LED on. I hope you can be a little more imaginative – this is only an example.

The complete codes for the transmitter and receiver are shown below as TX.ASM and RX.ASM.

The OPTION register has been set to produce timing pulses of 1ms.

Transmitter program code

TX.ASM

```
;tx.asm transmits code from a switch.

TMR0        EQU         1           ;means TMR0 is file 1.
STATUS      EQU         3           ;means STATUS is file 3.
PORTA       EQU         5           ;means PORTA is file 5.
PORTB       EQU         6           ;means PORTB is file 6.
TRISA       EQU         85H         ;TRISA (the PORTA I/O selection)
                                    ;is file 85H
TRISB       EQU         86H         ;TRISB (the PORTB I/O selection)
                                    ;is file 86H
OPTION_R    EQU         81H         ;the OPTION register is file 81H
ZEROBIT     EQU         2           ;means ZEROBIT is bit 2.
COUNT       EQU         0CH         ;COUNT is file 0C, a register to
                                    ;count events.
NUMA        EQU         0DH
;*****************************************************

LIST        P=16F84     ; we are using the 16F84.
ORG         0           ;the start address in memory is 0
GOTO        START       ; goto start!

;*****************************************************
;Configuration Bits

__CONFIG H'3FF0'        ;selects LP oscillator, WDT off, PUT on,
                        ; Code Protection disabled.
```

```
;*******************************************************
;SUBROUTINE SECTION.

;2.5ms SECOND DELAY
DELAY1  CLRF    TMR0                ;Start TMR0
LOOPA   MOVF    TMR0,W              ;Read TMR0 into W
        SUBLW   .1                  ;TIME-W
        BTFSS   STATUS,ZEROBIT      ;Check TIME-W = 0
        GOTO    LOOPA
                RETLW 0             ;Return after TMR0 = 32

;5ms SECOND DELAY
DELAY2  CLRF    TMR0                ;Start TMR0
LOOPB   MOVF    TMR0,W              ;Read TMR0 into W
                SUBLW .3            ;TIME-W
        BTFSS   STATUS,ZEROBIT      ;Check TIME-W = 0
        GOTO    LOOPB
        RETLW   0                   ;Return after TMR0 = 2

;7.5ms SECOND DELAY
DELAY3  CLRF    TMR0                ;Start TMR0
LOOPC   MOVF    TMR0,W              ;Read TMR0 into W
                SUBLW .6            ;TIME-W
        BTFSS   STATUS,ZEROBIT      ;CHECK TIME-W = 0
        GOTO    LOOPC
        RETLW    0                  ;Return after TMR0 = 3

;*******************************************************
;CONFIGURATION SECTION

START   BSF     STATUS,5    ;Turns to Bank1.
        MOVLW   B'00011111' ;5bits of PORTA are I/P
        MOVWF   TRISA
        MOVLW   B'00000000'
        MOVWF   TRISB       ;PORTB is OUTPUT
        MOVLW   B'00000010' ;Prescaler is /256
        MOVWF   OPTION_R    ;PRESCALER is /8,1ms

        BCF     STATUS,5    ;Return to Bank0.
        CLRF    PORTA       ;Clears PortA.
        CLRF    PORTB       ;Clears PortB.
```

```
;*******************************************************
;Program starts now.

BEGIN   BTFSC   PORTA,0     ;wait for switch press
        GOTO    BEGIN
        MOVLW   27H         ;Put 27H into W
        MOVWF   NUMA        ;PUT 27H into NUMA

        BCF     PORTB,0
        CALL    DELAY1
        BSF     PORTB,0     ;Transmit START
        CALL    DELAY3      ;wait 7.5ms

TESTA0  BCF     PORTB,0     ;Transmit space
        CALL    DELAY1      ;wait 2.5ms
        BTFSC   NUMA,0      ;Test NUMA,0
        GOTO    SETA0       ;If NUMA0 = 1
        GOTO    CLRA0       ;If NUMA0 = 0

SETA0   BSF     PORTB,0     ;Transmit 1
        CALL    DELAY2      ;wait 5ms
        GOTO    TESTA1

CLRA0   BSF     PORTB,0     ;Transmit 0
        CALL    DELAY1      ;wait 2.5ms

TESTA1  BCF     PORTB,0
        CALL    DELAY1
        BTFSC   NUMA,1
        GOTO    SETA1
        GOTO    CLRA1

SETA1   BSF     PORTB,0
        CALL    DELAY2
        GOTO    TESTA2

CLRA1   BSF     PORTB,0
        CALL    DELAY1

TESTA2  BCF     PORTB,0
        CALL    DELAY1
        BTFSC   NUMA,2
        GOTO    SETA2
        GOTO    CLRA2
```

```
SETA2   BSF     PORTB,0
        CALL    DELAY2
        GOTO    TESTA3

CLRA2   BSF     PORTB,0
        CALL    DELAY1

TESTA3  BCF     PORTB,0
        CALL    DELAY1
        BTFSC   NUMA,3
        GOTO    SETA3
        GOTO    CLRA3

SETA3   BSF     PORTB,0
        CALL    DELAY2
        GOTO    TESTA4

CLRA3   BSF     PORTB,0
        CALL    DELAY1

TESTA4  BCF     PORTB,0
        CALL    DELAY1
        BTFSC   NUMA,4
        GOTO    SETA4
        GOTO    CLRA4

SETA4   BSF     PORTB,0
        CALL    DELAY2
        GOTO    TESTA5

CLRA4   BSF     PORTB,0
        CALL    DELAY1

TESTA5  BCF     PORTB,0
        CALL    DELAY1
        BTFSC   NUMA,5
        GOTO    SETA5
        GOTO    CLRA5

SETA5   BSF     PORTB,0
        CALL    DELAY2
        GOTO    TESTA6
```

```
CLRA5     BSF     PORTB,0
          CALL    DELAY1

TESTA6    BCF     PORTB,0
          CALL    DELAY1
          BTFSC   NUMA,6
          GOTO    SETA6
          GOTO    CLRA6

SETA6     BSF     PORTB,0
          CALL    DELAY2
          GOTO    TESTA7

CLRA6     BSF     PORTB,0
          CALL    DELAY1

TESTA7    BCF     PORTB,0
          CALL    DELAY1
          BTFSC   NUMA,7
          GOTO    SETA7
          GOTO    CLRA7

SETA7     BSF     PORTB,0
          CALL    DELAY2
          CLRF    PORTB
          GOTO    BEGIN

CLRA7     BSF     PORTB,0
          CALL    DELAY1
          CLRF    PORTB
          GOTO    BEGIN
END
```

Receiver program code:

```
;RX.ASM

TMR0      EQU   1     ;means TMR0 is file 1.
STATUS    EQU   3     ;means STATUS is file 3.
PORTA     EQU   5     ;means PORTA is file 5.
PORTB     EQU   6     ;means PORTB is file 6.
TRISA     EQU   85H   ;TRISA (the PORTA I/O selection) is file 85H
TRISB     EQU   86H   ;TRISB (the PORTB I/O selection) is file 86H
OPTION_R  EQU   81H   ;the OPTION register is file 81H
```

```
ZEROBIT   EQU   2       ;means ZEROBIT is bit 2.
CARRY     EQU   0
COUNT     EQU   0CH     ;COUNT is file 0C, a register to count events.
NUMA      EQU   0DH
;*******************************************************

LIST      P=16F84       ;we are using the 16F84.
ORG       0             ;the start address in memory is 0
GOTO      START         ;goto start!

;*******************************************************
;Configuration Bits

__CONFIG H'3FF0'    ;selects LP oscillator, WDT off, PUT on,
                    ;Code Protection disabled.

;*******************************************************
;CONFIGURATION SECTION.

START   BSF         STATUS,5    ;Turns to Bank1.

        MOVLW       B'00011111' ;5bits of PORTA are I/P
        MOVWF       TRISA
        MOVLW       B'00000000'
        MOVWF       TRISB       ;PORTB is OUTPUT

        MOVLW       B'00000010' ;Prescaler is /256
        MOVWF       OPTION_R    ;PRESCALER is /8,1ms

        BCF         STATUS,5    ;Return to Bank0.
        CLRF        PORTA       ;Clears PortA.
        CLRF        PORTB       ;Clears PortB.
        BCF         STATUS,5    ;Return to BANK0
        CLRF        PORTA       ;Clears PORTA
        CLRF        PORTB       ;Clears PORTB

;*******************************************************
;Program starts now.

BEGIN       CLRF        NUMA

WAITHI  BTFSS       PORTA,0         ;Wait for HI Transmission
```

```
         GOTO    WAITHI
         CLRF    TMR0
TESTST   BTFSC   PORTA,0          ;Wait for LOW Transmission
         GOTO    TESTST          ;Test for START PULSE
         MOVF    TMR0,W
         SUBLW   .5              ;5-W or 5-TMR0
         BTFSC   STATUS,CARRY    ;SKIP IF TIME>5
         GOTO    WAITHI          ;NOT START BIT
TESTA0H  BTFSS   PORTA,0          ;wait for Hi transmission
         GOTO    TESTA0H
         CLRF    TMR0            ;start timing
TESTA0L  BTFSC   PORTA,0          ;wait for Lo transmission
         GOTO    TESTA0L
         NOP
         MOVF    TMR0,W          ;read value of TMR0
         SUBLW   .3              ;3-W or 3-TMR0
         BTFSS   STATUS,CARRY    ;Is TMR0>3 i.e. a logic1
         BSF     NUMA,0          ;Yes, 1 was transmitted.

TESTA1H  BTFSS   PORTA,0          ;Wait for pulse
         GOTO    TESTA1H
         CLRF    TMR0
TESTA1L  BTFSC   PORTA,0          ;Wait for LO.
         GOTO    TESTA1L
         NOP
         MOVF    TMR0,W
         SUBLW   .3
         BTFSS   STATUS,CARRY
         BSF     NUMA,1          ;1 was transmitted

TESTA2H  BTFSS   PORTA,0          ;Wait for pulse
         GOTO    TESTA2H
         CLRF    TMR0
TESTA2L  BTFSC   PORTA,0          ;Wait for Lo.
         GOTO    TESTA2L
         NOP
         MOVF    TMR0,W
         SUBLW   .3
         BTFSS   STATUS,CARRY
         BSF     NUMA,2          ;1 was transmitted

TESTA3H  BTFSS   PORTA,0          ;Wait for pulse
         GOTO    TESTA3H
         CLRF    TMR0
```

```
TESTA3L BTFSC   PORTA,0         ;Wait for Lo
        GOTO    TESTA3L
        NOP
        MOVF    TMR0,W
        SUBLW   .3
        BTFSS   STATUS,CARRY
        BSF     NUMA,3          ;1 was transmitted

TESTA4H BTFSS   PORTA,0         ;Wait for pulse
        GOTO    TESTA4H
        CLRF    TMR0
TESTA4L BTFSC   PORTA,0         ;Wait for Lo
        GOTO    TESTA4L
        NOP
        MOVF    TMR0,W
        SUBLW   .3
        BTFSS   STATUS,CARRY
        BSF     NUMA,4          ;1 was transmitted

TESTA5H BTFSS   PORTA,0         ;Wait for pulse
        GOTO    TESTA5H
        CLRF    TMR0
TESTA5L BTFSC   PORTA,0         ;Wait for Lo
        GOTO    TESTA5L
        NOP
        MOVF    TMR0,W
        SUBLW   .3
        BTFSS   STATUS,CARRY
        BSF     NUMA,5          ;1 was transmitted

TESTA6H BTFSS   PORTA,0         ;Wait for pulse
        GOTO    TESTA6H
        CLRF    TMR0
TESTA6L BTFSC   PORTA,0         ;Wait for Lo
        GOTO    TESTA6L
        NOP
        MOVF    TMR0,W
        SUBLW   .3
        BTFSS   STATUS,CARRY
        BSF     NUMA,6          ;1 was transmitted

TESTA7H BTFSS   PORTA,0         ;Wait for pulse
        GOTO    TESTA7H
        CLRF    TMR0
```

```
TESTA7L  BTFSC   PORTA,0          ;Wait for Lo
         GOTO    TESTA7L
         NOP
         MOVF    TMR0,W
         SUBLW   .3
         BTFSS    STATUS,CARRY
         BSF     NUMA,7           ;1 was transmitted

         MOVLW   27H
         SUBWF   NUMA,W           ;NUMA-27
         BTFSS   STATUS,ZEROBIT
         GOTO    BEGIN            ;If NUMA is not 27
         BSF     PORTB,0          ;Turn on LED.
         GOTO    BEGIN

END
```

Using the transmit and receive subroutines

The transmit and receive subroutines may seem a little complex, but all you need to do in your code is call them.

- To transmit
  Put the data you wish to transmit in the file NUMA then CALL TRANSMIT. The data in the file NUMA is transmitted.
- To receive
  CALL RECEIVE, the received data will be present in the file NUMA for you to use.

These programs have illustrated how to switch an LED on (this could be a remote control for a car burglar alarm). You may of course want to add more lines of code to be able to turn the LED off. This could be done in the receiver section by waiting for say 2 seconds and on the next transmission turn the LED off, providing of course the code was again 27H. Other codes could of course be added for other switches or keypad buttons, the possibilities are endless.

The transmitter and receiver micros could be hard wired together first to test the software without the radio link. The radio transmitter and receiver can then replace the wire to give a wireless transmission.