Chapter 12

Radio Transmitters and Receivers

There are many low cost radio transmitters and receivers on the market from companies like Quasar, Radiotronix, and RF Solutions, transmitting on 433 MHz and costing as little as £3 from Farnell.com. They do not need a license to operate and are fairly straightforward to use.

The transmitters only have three connections, two power supplies, and one data input, the transmitting aerial is incorporated on the unit. The receiver has four connections, two power supplies, one aerial input, and one data output. The receiving aerial only needs to be a piece of wire about 25 cm long.

Consider the transmitter and receiver circuits in which a code is transmitted when a button is pressed. At the receiving end an LED lights when the correct code is received. The block diagram is shown in Figure 12.1.

THE TRANSMITTER

The data need not be generated by pushing a button. It could come from a keypad or be generated when making an analogue measurement. In this example an 8 bit code is used to store a number 0 × 37 in the file NUMA when the button is pressed. When the 0 × 37 is received in file NUMA the LED will turn on. The file NUMA is shown in Figure 12.2.

In order to transmit the data a start pulse of 5 ms duration is sent followed by the data with each bit being 2.5 ms wide as shown in Figure 12.3.

The 0 × 37 data train is shown in Figure 12.4.

The program to generate this pulse is shown in **Tx.C**:

```
1.    // Tx.C by DW Smith 6-6-12
2.    #include <p18f1220.h>
3.    #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
4.    #include <delays.h>

5.    // put variables here
6.    char NUMA, NUMAbit;

7.    void Tx (void)
8.    {
9.    PORTBbits.RB0=0;          //zero Tx
10.   Delay10TCYx(125);         //5 ms, 1TCY=4 µs
```

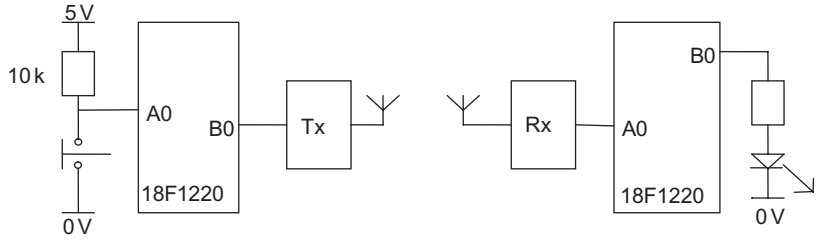


FIGURE 12.1 The radio transmitter and receiver system.

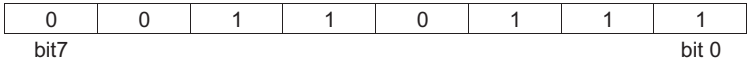


FIGURE 12.2 Showing 0×37 in the file NUMA.

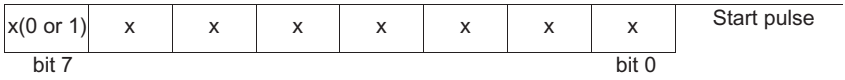


FIGURE 12.3 The basic data pulse train.

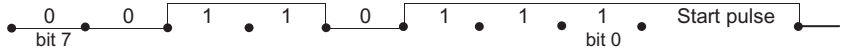


FIGURE 12.4 The 0×37 data train.

```
11.   PORTBbits.RB0=1;          //start pulse
12.   Delay10TCYx(125);          //5 ms

13.   NUMAbit=NUMA & 0b00000001;          // determines NUMA bit0.
14.   if (NUMAbit==0) PORTBbits.RB0=0;
15.   else PORTBbits.RB0=1;
16.   Delay10TCYx(62);          //2.5 ms

17.   NUMAbit=NUMA & 0b00000010;          // determines NUMA bit1.
18.   if (NUMAbit==0) PORTBbits.RB0=0;
19.   else PORTBbits.RB0=1;
20.   Delay10TCYx(62);          //2.5 ms

21.   NUMAbit=NUMA & 0b00000100;          // determines NUMA bit 2.
22.   if (NUMAbit==0) PORTBbits.RB0=0;
23.   else PORTBbits.RB0=1;
24.   Delay10TCYx(62);          //2.5 ms

25.   NUMAbit=NUMA & 0b00001000;          // determines NUMA bit 3.
26.   if (NUMAbit == 0) PORTBbits.RB0=0;
27.   else PORTBbits.RB0=1;
28.   Delay10TCYx(62);          //2.5 ms
```

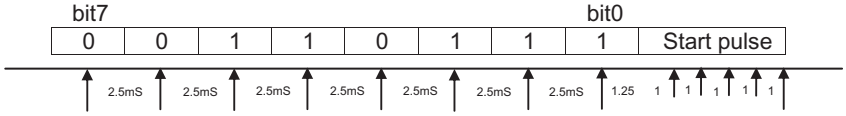
```
29.   NUMAbit=NUMA & 0b00010000;              // determines NUMA bit 4.
30.   if (NUMAbit==0) PORTBbits.RB0=0;
31.   else PORTBbits.RB0=1;
32.   Delay10TCYx(62);          //2.5 ms

33.   NUMAbit=NUMA & 0b00100000;          // determines NUMA bit 5.
34.   if (NUMAbit == 0) PORTBbits.RB0=0;
35.   else PORTBbits.RB0=1;
36.   Delay10TCYx(62);          //2.5 ms

37.   NUMAbit=NUMA & 0b01000000;          // determines NUMA bit 6.
38.   if (NUMAbit == 0) PORTBbits.RB0=0;
39.   else PORTBbits.RB0=1;
40.   Delay10TCYx(62);          //2.5 ms

41.   NUMAbit=NUMA & 0b10000000;          // determines NUMA bit 7.
42.   if (NUMAbit == 0) PORTBbits.RB0=0;
43.   else PORTBbits.RB0=1;
44.   Delay10TCYx(62);          //2.5 ms
45.   PORTBbits.RB0=0;
46.   }

47.   void main (void)
48.   {
49.   //SET UP
50.   OSCCON=0b01001000;              //osc is 1 MHz, 1TCY=4 µs
51.   ADCON1=0×7F;                    //all IO are digital or 0b01111111 in binary
52.   TRISA=0b11111111;               //sets PORTA as all inputs
53.   PORTA=0b00000000;               //turns off PORTA outputs, not required, no outputs
54.   TRISB=0b00000000;               //sets PORTB as all outputs
55.   PORTB=0b00000000;               //turns off PORTB outputs, good start position
56.   NUMA=0×37;

57.   while(1)
58.     {
59.   if (PORTAbits.RA0 ==0) Tx();
60.     }
61.   }
```

- The transmit code is shown between lines 7 and 46.
- Lines 9 and 10 set the output low as a reset for 5 ms.
- Lines 11 and 12 set the output high generating the Start pulse for 5 ms.
- Lines 13–16 transmit bit0 of NUMA for 2.5 ms. In order to determine if bit0 is a 1 or a 0 NUMA is anded with 0b00000001. So bits 7–1 are anded with 0, anything anded with 0=0, i.e., 0&0=0 0&1=0. Bits 7–1 are said to be masked out, they are 0 whatever they were before. Bit0 is anded with 1, i.e., 1&0=0, 1&1=1, so the value is unchanged. NUMAbit takes on the value of NUMA bit0.
- Lines 17–20 transmit bit1 of NUMA for 2.5 ms. Here NUMA is anded with 0b00000010. Now bit1 is unchanged the other bits will be 0. NUMAbit takes on the value of NUMA bit1.
- Lines 21–44 transmit the other bits in a similar fashion.

- Line 45 resets the output to 0.
- Line 50 sets the internal oscillator to 1 MHz because we require a time faster than 31 kHz for our timing.
- Lines 57–61 continuously checks the state of the switch on PORTA bit0 and transmits the code in the file NUMA, if PORTA bit0=0.

THE RECEIVER

The receiver has to detect each of the bits that arrive and determine if they are 1 or 0.



In order to determine the start pulse has arrived, once the leading edge goes high the input is checked after 1 ms to see if it is still high and then three times more to establish this was a pulse not a glitch. The program then waits a further 1 ms to the end of the Start pulse and then a further 1.25 ms would be in the middle of bit0. Further 2.5 ms would locate in the middle of the other bits.

The program to achieve this is **Rx.C**:

```
1.    //Rx.C by DW Smith 26-11-11
2.    #include <p18f1220.h>
3.    #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
4.    #include <delays.h>

5.    // put variables here
6.    char NUMA;

7.    void Rx(void)          // check for start pulse 5 ms hi.
8.    {
9.    wait:
10.   if (PORTAbits.RA0 ==1) Delay10TCYx(25);          //1 ms
11.   else          goto wait;

12.   if (PORTAbits.RA0 ==1) Delay10TCYx(25);          //1 ms
13.   else          goto wait;

14.   if (PORTAbits.RA0 ==1) Delay10TCYx(25);          //1 ms
15.   else          goto wait;

16.   if (PORTAbits.RA0 ==1) Delay10TCYx(25);          //1 ms
17.   else          goto wait;

18.   Delay10TCYx(56);          //1 ms + 1.25 ms middle of bit 0
19.   if (PORTAbits.RA0 ==1) NUMA=1;

20.   Delay10TCYx(62);//2.5 ms middle of bit 1
21.   if (PORTAbits.RA0 ==1) NUMA=NUMA+2;
```

```
22.     Delay10TCYx(62);            //2.5 ms middle of bit 2
23.     if (PORTAbits.RA0 ==1) NUMA=NUMA+4;
24.     Delay10TCYx(62);            //2.5 ms middle of bit 3
25.     if (PORTAbits.RA0 ==1) NUMA=NUMA+8;

26.     Delay10TCYx(62);            //2.5 ms middle of bit 4
27.     if (PORTAbits.RA0 ==1) NUMA=NUMA+16;

28.     Delay10TCYx(62);            //2.5 ms middle of bit 5
29.     if (PORTAbits.RA0 ==1) NUMA=NUMA+32;

30.     Delay10TCYx(62);            //2.5 ms middle of bit 6
31.     if (PORTAbits.RA0 ==1) NUMA=NUMA+64;

32.     Delay10TCYx(62);            //2.5 mS middle of bit 7
33.     if (PORTAbits.RA0 ==1) NUMA=NUMA+128;
34.     }

35.     void main (void)
36.     {
37.     //SET UP
38.     OSCCON=0b01001000;          //osc is 1 MHz
39.     ADCON1=0×7F;                //all IO are digital or 0b01111111 in binary
40.     TRISA=0b11111111;           //sets PORTA as all inputs
41.     PORTA=0b00000000;           //turns off PORTA outputs, not required, no outputs
42.     TRISB=0b00000000;           //sets PORTB as all outputs
43.     PORTB=0b00000000;           //turns off PORTB outputs, good start position
44.     while(1)
45.     {
46.     Rx();
47.     if (NUMA == 0×37) PORTBbits.RB0=1;
48.     }

49.     }
```

- The receive routine lines 7–34 detect the incoming pulses and store the bits in the file NUMA.
- Lines 10–17 check first of all for the arrival of the +ve edge of the Start pulse and then check to see if it lasts 4 ms.
- If it does then we wait a further 1 + 1.25 ms to place us in the middle of bit0, line 18.
- Line 19 checks for a 1. If the pulse is high then bit0 was1 and a 1 is added to NUMA, i.e., NUMA=1. If the bit is low then nothing is added to NUMA.
- Line 20 waits for 2.5 ms putting the time in the middle of bit1.
- Line 21 if the bit is high then 2 is added to NUMA.
- Lines 22–34, in this way the bits are checked in the middle and the corresponding bit value is added to NUMA.

In order to test out your code I recommend you hard wire the Tx and Rx circuits together without the radio modules. When you are happy that works then include the Rf modules.

EXAMPLES

- In order to test your understanding use two switches on the input. One to turn the LED on, another to turn the LED off. Use different codes for each and look for them in the receiver.
- Connect the keypad to the receiver circuit and transmit key presses. Show the characters on an LCD in the receiver.
- Transmit temperature measurements remotely.