Single Wire Data Transmission Protocol’s Manchester Coding using the PIC18F4520 with the EasyPIC5 Development Board

ELEC471
Spring 2009

Andrew DeMedeiros
Ben Allen
May 2, 2009
Single wire communications protocols are mainly used in RF data transmission applications because of the fact that no clock signal is required. This allows for transmission of data only, as opposed to having multiple transmitters transmitting data and clock signals.

Manchester code is one such protocol; each bit sent in Manchester is denoted by a transition, which takes place within a fixed time. So that means that each bit is sent within a set period of time, now as long as the transmitting and receiving microcontroller have the same time intervals communication can begin. In addition to the data being sent a start marker and end marker are sent to let the M.C.U. know when data is available and when it is over.

Manchester code can be setup in two ways, these are the G.E. Thomas Standard and the IEEE 802.3 standard. With the G.E. Thomas standard a low-to-high transition (i.e. rising edge) means a zero (0) and a high-to-low transition means a one (1). The IEEE 802.3 standard is the opposite a low-to-high means a one (1) and a high-to-low means a zero (0).
**Equipment:**

1-EasyPIC5 Development Board
2-PIC18F4520 or 16F877
1-Breadboard
Jumper Wires
1-Alligator to Alligator wire
1+5V supply

**Procedure MCU #1 (Transmitter)**

1. Make sure the appropriate MCU is installed in the EasyPIC 5 board, if not **carefully** remove the one that’s there and replace it with either the 18F4520 or the 16F877, other MCU’s might be possible as well.
2. Let’s quickly right a small program to transmit a small quote. Open up MikroC and start a new project, go to “Project”-> “New Project”. Type in a project name, and select your MCU, click “Default” to keep default settings and lastly select _OSC_INTI07_1H so we can use the internal oscillator. As Shown in Figure 1.

![New Project](image-url)
3. In the editor window declare two variable shorts, “i” is just a counter for a loop and “ch” will be each character in the quote. “s1[]” is the string of characters to be sent.

```c
unsigned short
i, ch;
char s1[] = "Cap'n Johnson!"; //Declaring what you want sent
```

4. Next start the program, both of these following routines are built into the MikroC libraries. Declare the port and pin that you want the output to be, in `Man_Send_Config()`

```c
void main() {
    INTCON.GIE = 0; // Disable interrupts
    Man_Send_Config(&PORTD,0); // Initialize manchester sender
}
```

5. Now let’s write the while loop. The first while loop just repeats the entire process, it allows the same data to be sent over and over. The start marker lets the receiving MCU know that data is available. Then set “ch” to the first character in the string and reset “i” back to zero. The second while loop, goes through the string and sends each character at one time.

```c
while (1)
{
    Man_Send(0x0B); // Send start marker
    Delay_ms(100); // Wait for a while
    ch = s1[0];
    i = 0;
    while (ch != 0)
    {
        Man_Send(ch); // Send char
        Delay_ms(90);
        i++;
        ch = s1[i];
    }
}
```
6. To end our program we send the end marker, and we wait 1s before we send again.

   Man_Send(0x0E);       // Send end marker
   Delay_ms(1000);
   
7. Compile the program, and then download it. (Ctrl +F11). Once downloading is complete you should notice one led flashing, that is the Manchester code.

**MCU #2 (Receiver)**

8. Turn the board off, and carefully remove and replace the MCU with another one.
9. Now lets write a receiver program that will display the send data on an LCD screen.
10. Repeat step #2 and setup a new project. Make sure that you set the receiver clock speed to the same as the first.
11. Now copy in the receiver code at the end of this paragraph, I wont get into the code specifics since its pretty straightforward.
12. Place the transmitter MCU in the breadboard and provide the appropriate voltage (5V) to the right pins.
13. Connect PortD.0 (pin 19) on the transmitter to PortD.6 (pin 29) with the alligator clip.
15. Turn everything on and you should see the displayed message(“Cap'n Johnson!”) on the LCD screen.
**Transmitter Code:**

/*Example MikroE Program to show the use of Manchester Coding*/

/*Modified By Andrew DeMedeiros and Ben Allen */

unsigned short
i, ch;
char s1[] = "Cap'n Johnson!0";    //Declaring what you want sent

void main() {
    INTCON.GIE = 0;        // Disable interrupts
    Man_Send_Config(&PORTD,0);    // Initialize manchester sender

    while (1)
    {
        Man_Send(0x0B);        // Send start marker
        Delay_ms(100);        // Wait for a while
        ch = s1[0];
i = 0;
        while (ch != 0)        // string ends with zero
        {
            Man_Send(ch);        // Send char
            Delay_ms(90);
i++;
            ch = s1[i];
        }
        Man_Send(0x0E);        // Send end marker
        Delay_ms(1000);
    }
}
Receiver Code:
/*Example MikroE Program to show the use of Manchester Coding*/
/*Modified By Andrew DeMedeiros and Ben Allen */
unsigned short
ERR,
*error,
ErrorCount,
temp;

void main() {
    ERR = 0;
    error = &ERR;
    ErrorCount = 0;
    ADCON1 = 0x0F;       // Set AN pins to Digital I/O
    Lcd_Config(&PORTB, 4, 5, 6, 3, 2, 1, 0); // Lcd_Init_EPS, see Autocomplete
    Lcd_Cmd(LCD_CLEAR);
    Man_Receive_Config(&PORTD, 6);          // Configure and synchronize receiver
    while (1) {
        Lcd_Cmd(LCD_FIRST_ROW);
        while (1) { // Wait for the start marker
            temp = Man_Receive(error);
            if (temp == 0x0B)   // Start marker, see Transmitter example
                break;
            if (ERR)        // We got the starting sequence
                break;
        }
        do {
            temp = Man_Receive(error);   // Attempt byte receive
            if (ERR)
                {
                    LcdChr_CP('?');
                    ErrorCount++;
                    if (ErrorCount > 20)
                        {
                        //Man_Receive_Init(&PORTD);
                        // alternative:
                        temp = Man_Synchro();
                        ErrorCount = 0;
                        }
                    }
            else
                { // End marker, see Transmitter example
                    LcdChr_CP(temp); // Don't write the end marker on LCD
                    Delay_ms(25);
                }
        } while (temp != 0x0E);
    }
}