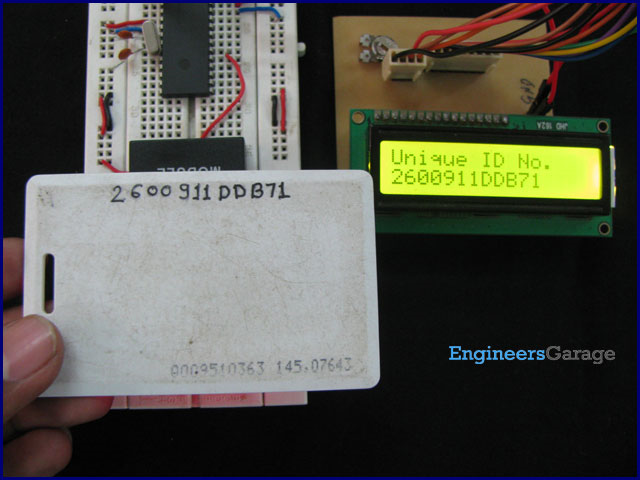
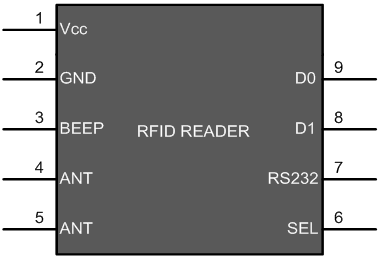
**[How to interface RFID with PIC18F4550 Microcontroller](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/rfid-interfacing-circuit-code" \o "How to interface RFID with PIC18F4550 Microcontroller)**



[RFID](http://www.engineersgarage.com/rfid-radio-frequency-identification-and-detection) (Radio Frequency Identification and Detection) is widely used everywhere from highly secured defense laboratories to school attendance system. By employing RFID, much secured entry systems can be developed without incurring huge costs. These are the reasons of excessive use of RFID technology. In this article, interfacing of an RFID reader module has been explained with [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller). The USART [interrupt](http://www.engineersgarage.com/tutorials/interrupts-8051-interrupt-programming), an internal [PIC interrupt](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/pic-external-hardware-interrupts-circuit), has also been explained. (For more details on USART, refer [PIC EUSART](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/eusart-circuit))

As explained earlier (refer [RFID interfacing with 8051](http://www.engineersgarage.com/microcontroller/8051projects/interface-rfid-AT89C51-circuit) & [with AVR](http://www.engineersgarage.com/embedded/avr-microcontroller-projects/rfid-interfacing-circuit)), an [RFID](http://www.engineersgarage.com/rfid-radio-frequency-identification-and-detection) module consists of an RFID Reader, a line converter (usually [MAX232](http://www.engineersgarage.com/electronic-components/max232-datasheet)) and a COM port. The line converter of this module converts the TTL logic voltage of RFID Reader to RS232 logic. Therefore, to convert the voltage level from such an RFID module, another MAX232 is used to interface it with a [microcontroller](http://www.engineersgarage.com/microcontroller).

One can also use an RFID Reader directly to interface with the controller, thus avoiding the need of voltage level converters. Here both the MAX232s have been eliminated from the circuit and RFID reader is directly connected with the [PIC microcontroller](http://www.engineersgarage.com/articles/pic-microcontroller-tutorial).



The following table explains the pin diagram of the RFID Reader module.

|  |  |  |
| --- | --- | --- |
| **Pin No.** | **Name** | **Description** |
| 1 | Vcc | Supply Voltage; 5V |
| 2 | GND | Ground (0V) |
| 3 | BEEP | Beep or LED drive |
| 4 | ANT | No Use |
| 5 | ANT | No Use |
| 6 | SEL | High is RS232, Low is Weigand |
| 7 | RS232 | TTL output data |
| 8 | D1 | Weigand Data 1 |
| 9 | D0 | Weigand Data 0 |

Another part of the RFID system is RFID tag, which contains 12 bytes of unique data. As the tag comes in the range of the Reader Module, it gets activated and transmits this unique code. (For more detail on working of RFID system, refer the article on [RFID](http://www.engineersgarage.com/rfid-radio-frequency-identification-and-detection)) The objective here is to receive this 12 byte unique code and display on a [16x2 LCD](http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet) using [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller).

With the RFID interfacing, this article also explains the USART [interrupt](http://www.engineersgarage.com/tutorials/interrupts-8051-interrupt-programming) which is an internal interrupt. (For external interrupts, refer [PIC Hardware interrupts](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/pic-external-hardware-interrupts-circuit)) The internal interrupts, unlike hardware interrupts, are associated with internal peripherals of the controller. To use the USART interrupt, following registers have to be configured accordingly.

1. **INTCON** (Interrupt Control Register)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |

**PEIE/GIEL**: This bit is used to enable/disable all the peripheral interrupts (Internal interrupts) of the controller. But GIE/GIEH bit must be set to high first.

1 = Enables all Peripheral Interrupts

0 = Disables all Peripheral Interrupts

**GIE/GIEH**: This is Global Interrupt Enable bit. This bit is set to high to enable all interrupts of the PIC18F4550.

1 = Enables interrupts

0 = Disables all interrupts

2. **PIR1** (Peripheral Interrupt Request 1)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| SPPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF |

**TXIF**: This is Transmission interrupt flag which is set to high when TXREG**\*** is empty.

**RCIF:** This is Reception interrupt flag which is set to low when reception is complete.

**\***TXREG : [EUSART](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/eusart-circuit) Transmit Register (The data to be transmitted is stored in this register)

3. **PIE1** (Peripheral Interrupt Enable 1)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| SPPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |

**TXIE**: This bit is used to enable/disable the Transmission (Tx) interrupt.

**RCIE:** This bit is used to enable/disable the Reception (Rx) interrupt.

Refer [PIC EUSART](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/eusart-circuit) registers for more details.

The connections of RFID reader module and LCD with the microcontroller are shown in the circuit diagram tab.

**Programming Steps:**

1. Set the baud rate of PIC’s USART of to 9600 bps.

2. Set the SPEN and CREN bits to ‘1’ (RCSTA Register).

3. Set the GIE and PEIE to ‘1’ (INTCON Register).

4. Store the 12 byte data into an array when Reception Interrupt is generated.

5. Print the all stored data on the LCD. Refer [displaying text on LCD using PIC](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/lcd-interfacing-text-circuit).

// Program to interface RFID module using EUSART in PIC18F4550  
  
// Configuration bits  
/\* \_CPUDIV\_OSC1\_PLL2\_1L, // Divide clock by 2  
 \_FOSC\_HS\_1H, // Select High Speed (HS) oscillator  
 \_WDT\_OFF\_2H, // Watchdog Timer off  
 MCLRE\_ON\_3H // Master Clear on  
\*/  
  
#define FREQ 12000000  
#define baud 9600  
#define spbrg\_value (((FREQ/64)/baud)-1)  
#define rs LATA.F0  
#define rw LATA.F1  
#define en LATA.F2  
#define lcdport LATB  
  
unsigned char rx\_data();  
void lcd\_ini();  
void lcdcmd(unsigned char);  
void lcddata(unsigned char);  
unsigned char data[]="Unique ID No.";  
unsigned char card\_id[12];  
unsigned int i=0,pos;  
  
void main()  
{  
 TRISB=0; // Set Port B as output port  
 LATB=0;  
 TRISA=0;  
 LATA=0;  
 SPBRG=spbrg\_value; // Fill SPBRG register to set the baud rate  
 RCSTA.SPEN=1; // To activate serial port (Tx and Rx pins)   
 RCSTA.CREN=1; // To enable continuous reception  
 PIE1.RCIE=1; // To enable the Reception (Rx) Interrupt  
 INTCON.GIE=1;  
 INTCON.PEIE=1;  
 lcd\_ini(); // LCD initialization  
 while(data[i]!='\0')  
 {  
 lcddata(data[i]); // To send characters one by one from 'data' array  
 i++;  
 }  
 while(1)  
 {  
 i=0;  
 while(i<12);  
 lcdcmd(0xC0);  
 i=0;  
 while(i<12)  
 {  
 lcddata(card\_id[i]); // Print the 12 byte received data  
 i++;  
 }  
 }  
}  
  
  
void interrupt()  
{  
 card\_id[i]=RCREG; // Store the received data byte by byte  
 i++;  
}  
  
void lcd\_ini()  
{  
 lcdcmd(0x38); // Configure the LCD in 8-bit mode, 2 line and 5x7 font  
 lcdcmd(0x0C); // Display On and Cursor Off  
 lcdcmd(0x01); // Clear display screen  
 lcdcmd(0x06); // Increment cursor  
 lcdcmd(0x80); // Set cursor position to 1st line, 1st column  
}  
  
void lcdcmd(unsigned char cmdout)  
{  
 lcdport=cmdout; //Send command to lcdport=PORTB  
 rs=0;   
 rw=0;  
 en=1;  
 Delay\_ms(10);  
 en=0;  
}  
  
void lcddata(unsigned char dataout)  
{  
 lcdport=dataout; //Send data to lcdport=PORTB  
 rs=1;  
 rw=0;  
 en=1;  
 Delay\_ms(10);  
 en=0;  
}

