Receive.c

```c
/* Since we never know when we are going to get data, we write a
Interrupt Routine that will execute on receiving data.
   You can write different conditions depending on your convenience to
test whether data reception is complete and also
   whether it is accurate
*/


#include <avr/io.h>
#include <avr/interrupt.h>

#define USART_BAUDRATE 9600
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

void initusart(void);                                // Set the baud
rate ( communication speed in bits per second)

int i=0;                      // Global variable acts as a counter

unsigned char str[8];    // String to be sent, declared global if to be
used in interrupts

int main (void)
{
      DDRA=0xff;        // Initialise all ports
      DDRC=0xff;
      DDRB=0xff;
      DDRD=0X00;        // Set PORTD for input, since it contains RX and
TX pins
      PORTB=0x00;       // Set all ports
      PORTC=0xff;
      PORTA=0x01;
      PORTD=0XFF;       // Pull up PORTD
      initusart();      // Initialise USART
      while(1);         // Infinite loop to keep the program running
      }



ISR(USART_RXC_vect)
{
      PORTC -= 0x01;          // Decrement PORTC by one
      str[i]=UDR;             // Load the character in the receive buffer
to string variable
      if (str[6]=='g') { PORTC=0x00;}   // Condition to test accuracy of
last variable
      i++;                    // Increment counter to read next character

}


void initusart(void)
```

```
{
   UCSRB |= (1 << RXEN);                                 // Turn on the
transmission and reception circuitry
   UCSRC |= (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);  // Use 8-bit
character sizes
   UBRRL = BAUD_PRESCALE;         // Load lower 8-bits of the baud rate
value into the low byte of the UBRR register

   UBRRH = (BAUD_PRESCALE >> 8);  // Load upper 8-bits of the baud rate
value into the high byte of the UBRR register
   UCSRB |= (1 << RXCIE);                                // Enable the
USART Recieve Complete interrupt (USART_RXC)
   sei();                         // Enable global interrupts, necessary
if you want to use ANY interrupt

}
```

Transmit.c

```
/* The execution of the code below is such: After initialising USART, I
call a send function to transmit the character.
    An interrupt is triggered on completion, the counter goes to the next
character and calls the send function. So,
    the exectution will be in the form of continuous loops and will enter
the main function only on completion of
    transmission of all characters
*/



#include <avr/io.h>
#include <avr/interrupt.h>

#define USART_BAUDRATE 9600                                // Set the
baud rate ( communication speed in bits per second)
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1) //
Prescale factor,calculated automatically

void initusart(void);           // Function for USART initialisation

void send(void);                // Function to send characters

int i=0;                       // Global variable acts as a counter

unsigned char str[8]="Testing"; // String to be sent, declared global if
to be used in interrupts

int main (void)
{
   initusart();                         // Initialise USART
   send();                              // Call send function
   while(1);                            // Infinite loop to keep the
program running
   }

void send()
{  if (i<7)                             // Send the character only if i
is less than string size
      {
      UDR=str[i];                       // Loads the character into the
transmit buffer
      }

}


ISR(USART_TXC_vect)                     // Interrupt Service Routine,
fires on Transmit Complete (Tx C)
{
      i++;                              // Increments i to send next
character
```

```
      UCSRA &= (0 << TXC);              // Clears the Transmit complete
flag, Do this to avoid errors
      send();                           // Call the send function again
}


void initusart(void)
{
   UCSRB |= (1 << TXEN);                                  // Turn on the
transmission and reception circuitry
   UCSRC |= (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);  // Use 8-bit
character sizes
   UBRRL = BAUD_PRESCALE;          // Load lower 8-bits of the baud rate
value into the low byte of the UBRR register

   UBRRH = (BAUD_PRESCALE >> 8);  // Load upper 8-bits of the baud rate
value into the high byte of the UBRR register
   UCSRB |= (1 << RXCIE);                                 // Enable the
USART Transmit Complete interrupt (USART_TXC)
   sei();                          // Enable global interrupts, necessary
if you want to use ANY interrupt

}
```