پروژه آزمایشگاه ریزپردازنده

اتصال ترمینال متلب (MATLAB) به میکروکنترلر

انجام دستورات متلب

دانشگاه آزاد اسلامی واحد تهران مرکز

نام استاد : محمدرضا پرویزی

سید شایان امیرشاه کرمی 39910141054072

تایم دوشنبه 14:45

ترم بهمن 1402

فهرست

پروژه اتصال ترمینال متلب (MATLAB) به میکروکنترلر

توضیحات

برنامه ی متلب در صنایع مختلف بسیار مورد استفاده قرار میگیرد. **سیستم‌های ماهواره‌ای :**از ابزارهای متلب برای طراحی مدار ماهواره، کنترل موقعیت، پردازش سیگنال و توسعه سیستم‌های ارتباطی استفاده می‌شود.

حال بیاندیشید بتوانیم با استفاده از متلب و اتصال آن به میکرو کنترل از صرف زمان و هزینه بیشتر در ساخت و نوشتن برنامه بپرهیزیم. این گونه که کاربر با دانش متلب اطلاعات را وارد کرده و در میکروکنترلر پردازش می شوند.

عملکرد اصلی و مهم پروژه ترمینال متلب

- اتصال ترمینال COM1 و COM2
- دریافت ورودی COM1 از خروجی COM2
- نوشتن برنامه و ساخت ترمینال متلب
- اتصال متلب به proteus
- نمایش ورودی ترمینال متلب در خروجی LCD

نحوه عملکرد پروژه

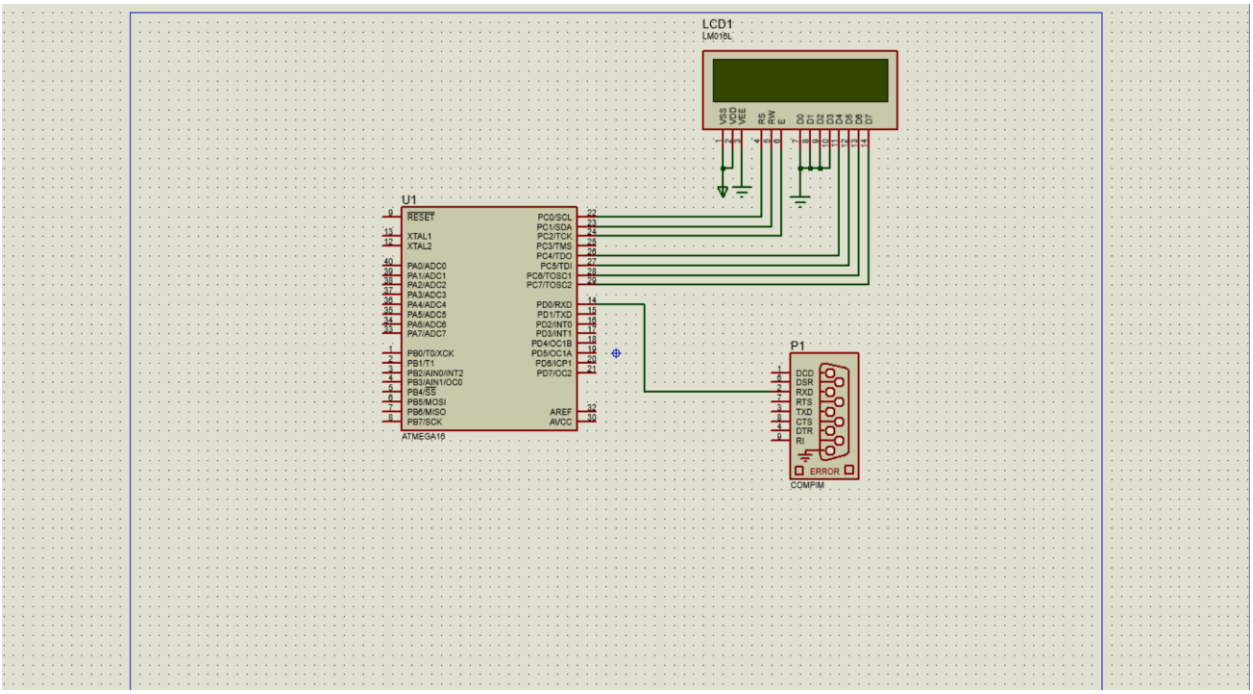
عملکرد پروژه بدین صورت است که کاربر یک ترمینال در متلب میسازد. به وسیله virtual serial port که یک اتصال مجازی سریال بین COM1 و COM2 (COM1 پروتووس و پرت مجازی آن – COM2 متلب و پرت مجازی آن) اتصال ایجاد میکند. ترمینال متلب را اجرا می کند و دستور خود را تایپ می کند. با فرستادن دستور، دستور در پروتووس در LCD نمایش داده می شود.

با استفاده از 4 نرم افزار proteus و codevision و Virtual Serial Port Driver و MATLAB میتوانیم این سنسور را شبیه سازی کنیم .

بخش Proteus

به کمک نرم افزار proteus شماتیک مدار را طراحی میکنیم.

برای پیاده سازی این پروژه از میکروکنترولر atmega16 و LCD(LM016L) و COMPIM برای درست کردن virtual COM استفاده می کنیم.

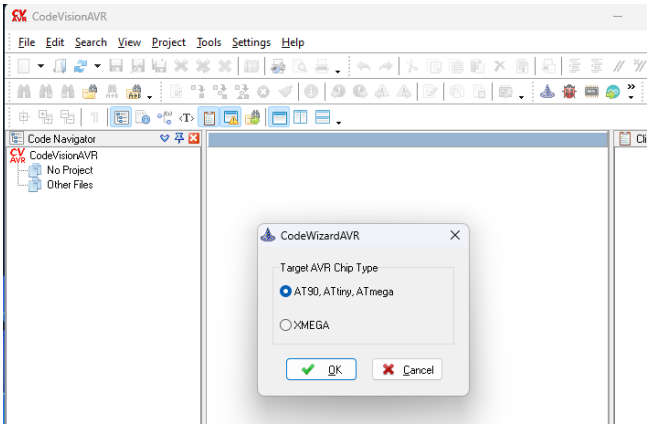


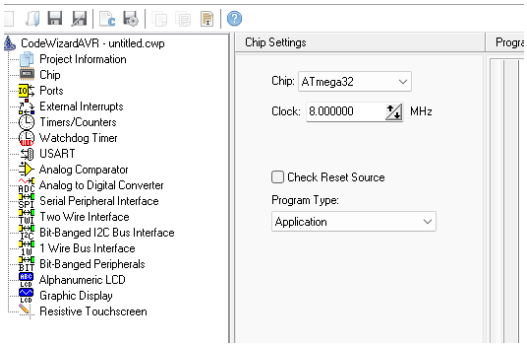
LCD را به PORTC میکرو وصل میکنیم. COMPIM را به PORTD RXD وصل می کنیم.

بخش Code Vision

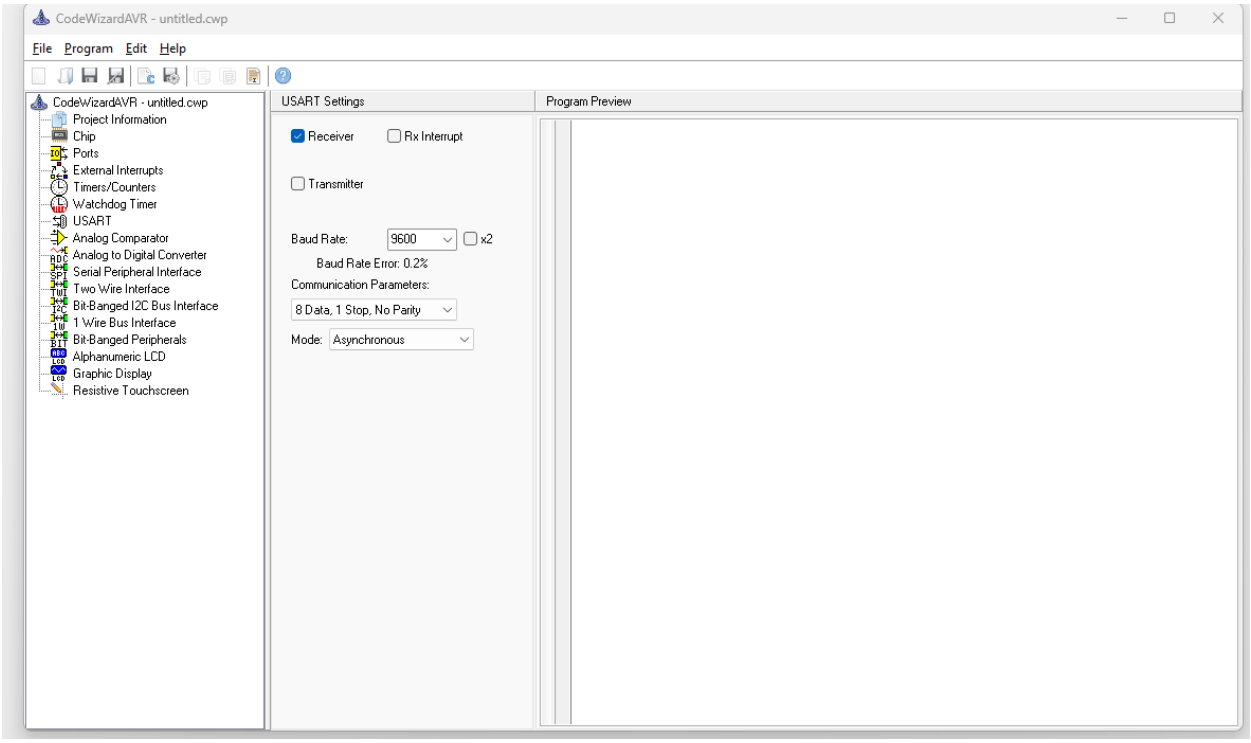
بخش code wizard

در تنظیمات code wizard میکرو atmega16 را انتخاب میکنیم.

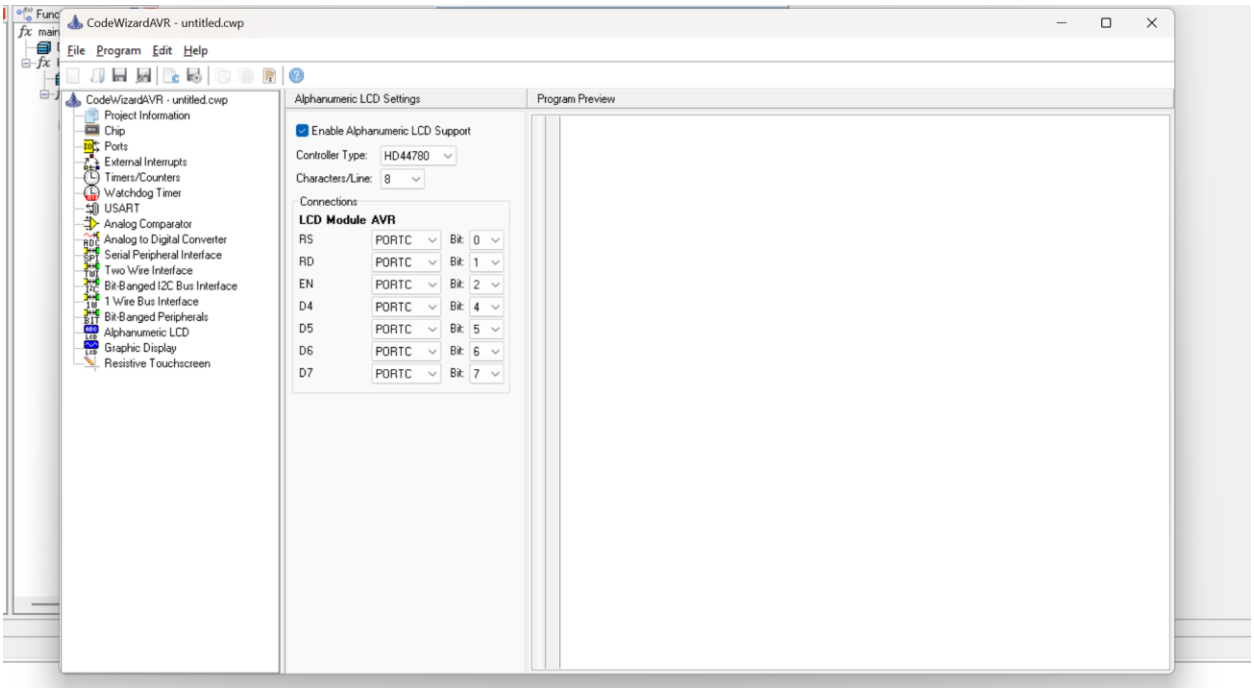




در قسمت USART گیرنده (Receiver) را فعال می کنیم. و Baud Rate 9600



سپس alphanumeric LCD را فعال کرده و برروی PORTB تنظیم میکنیم.



بخش کد

ابتدا کتابخانه های مورد نیاز را include میکنیم. و variable های global را تعریف میکنیم.

```
#include <mega16.h>

#include <delay.h>

// Alphanumeric LCD functions
#include <alcd.h>

// Declare your global variables here

// Standard Input/Output functions
#include <stdio.h>


char mychar;    //Declare 8 bit character variable
bit i=0;        //TO declare single bit variable
```

فایل های هدر لازم برای عملکردهای میکروکنترلر، توابع تاخیر، ورودی و خروجی استاندارد (stdio.h) و کتابخانه LCD سفارشی (alcd.h) را شامل می شود.

<mega16.h> این فایل سربرگ، تعاریف مربوط به میکروکنترلر ATmega16 را در اختیار برنامه قرار می دهد.

<delay.h> این فایل کتابخانه تاخیر را برای ایجاد وقفه های زمانی در برنامه فراهم می کند.

<alcd.h> این فایل کتابخانه توابع مربوط به نمایشگر LCD الفبا-عددی را در اختیار برنامه قرار می دهد.

<stdio.h> این فایل سربرگ استاندارد ورودی/خروجی را برای توابعی مانند printf و scanf(در صورت نیاز) شامل می شود.

Global variables

- Mychar به صورت char که 8 بیت کاراکتری که از سریال دریافت می شه.
- i یک متغیر بیتی برای تشخیص وقوع وقفه دریافت سریال

:Interrupt

```
// Interrupt Service Routine (ISR) for serial recieve completion

interrupt [USART_RXC] void myinterrupt (void)
{
    mychar = getchar();     // Get character from data stored in serial register (UDR)
    i=1;                    //Interrupt has occured
}
```

اینتراپت برای دریافت سریال به نام myinterrupt:

Getchar() در C تابع برای دریافت ورودی

I نشان دهنده به وجود آمدن interrupt. در اصل نشان دهنده این است یک کاراکتر دریافت شده.

:Main

```
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) | (0<<DDD0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);
```

تنظیم پورت ها:

- کد، پورت هایA ، B، Cو D را به عنوان ورودی (Input) تعریف می کند.
- پورت C به عنوان خروجی (Output) برای اتصال به نمایشگر LCD تنظیم می شود.(`PORTC=0xff;`)

تنظیم USART:

- این بخش مربوط به پیکربندی واحد ارسال و دریافت سریال (USART) است.
- پارامترهای ارتباطی شامل 8 بیت داده، 1 بیت توقف و بدون برابری تنظیم می شوند.
- گیرنده سریال روشن و فرستنده سریال خاموش است.
- سرعت Baud Rate برای USART روی 9600 تنظیم می شود.

غیرفعال کردن وقفه های خارجی:

- کد، وقفه های خارجیINT0 ، INT1وINT2 را غیرفعال می کند.(`MCUCR=0x00;`)

```
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);
```

فعال کردن وقفه های سراسری:

- دستور `sei` در اسمبلی، وقفه های سراسری را فعال می کند

```
#asm ("sei");
// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
```

تنظیم نمایشگر LCD:

- تابع `lcd_init(16)` کتابخانه LCD را برای نمایش 16 کاراکتر در هر خط پیکربندی می کند.

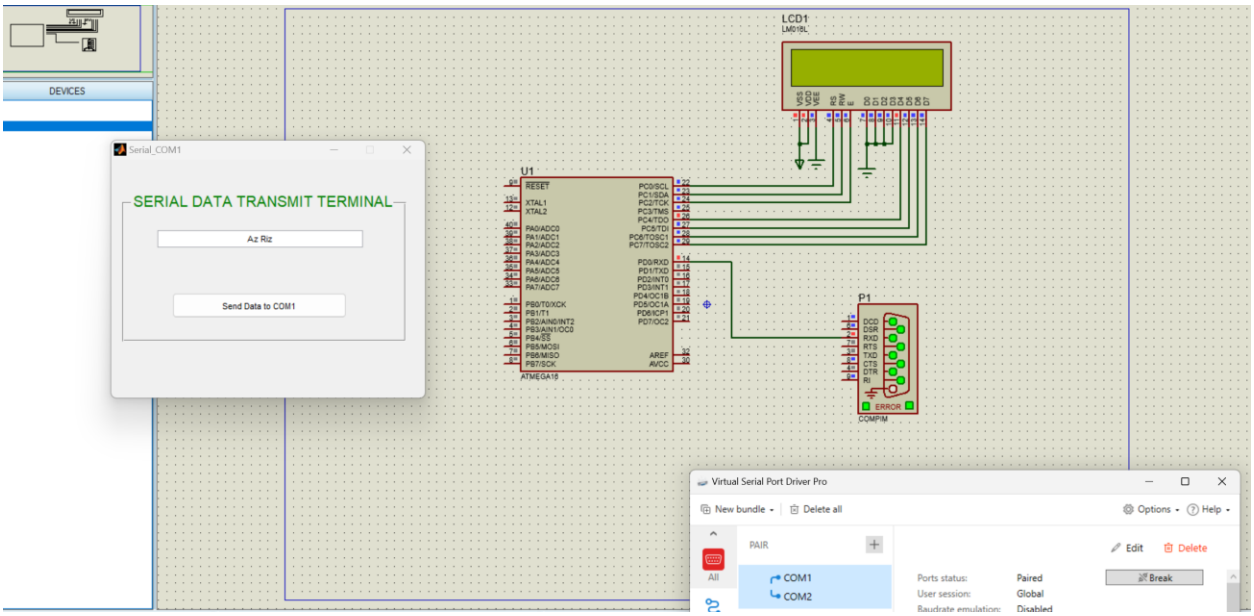
```
lcd_init(16);   // Characters/line: 16
```

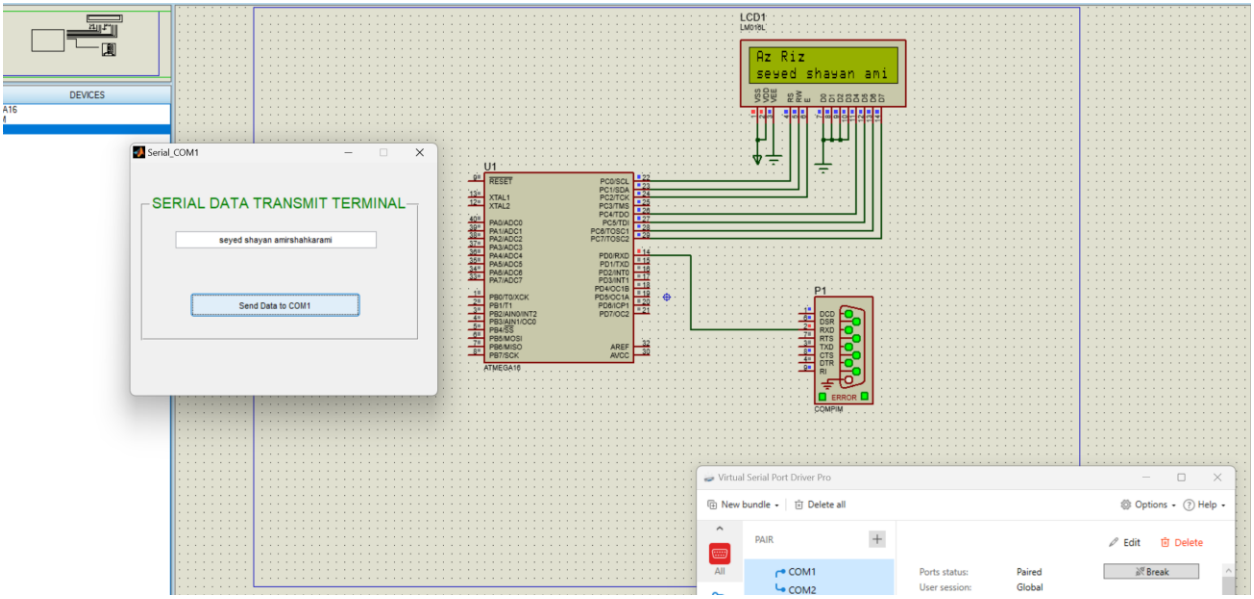
حلقه اصلی:(while)

- این حلقه به طور دائم تکرار می شود.
- کد بررسی می کند که آیا `i` برابر با 1 است (یعنی یک کاراکتر دریافت شده است).
  - اگر `i` برابر با 1 باشد، کاراکتر دریافت شده (`mychar`) با استفاده از تابع `lcd_putchar`روی نمایشگر LCD چاپ می شود.
  - سپس مقدار `i`به 0 تنظیم می شود تا نشان دهد کاراکتر نمایش داده شده است.

```
while (1)
    {
      if(i==1)
      {
          lcd_putchar(mychar);
          i=0;
      }
    }
}
```

نمایش خروجی


LCD1
LM016L
Serial_COM1
SERIAL DATA TRANSMIT TERMINAL
Az Riz
Send Data to COM1
U1
ATMEGA16
P1
COMPIM
ERROR
Virtual Serial Port Driver Pro
New bundle
Delete all
Options
Help
PAIR
COM1
COM2
Edit
Delete
Break
Ports status:
Paired
User session:
Global


LCD1
LM016L
Az Riz
seyed shayan ami
Serial_COM1
SERIAL DATA TRANSMIT TERMINAL
seyed shayan amirshahkarami
Send Data to COM1
U1
ATMEGA16
P1
COMPIM
ERROR
Virtual Serial Port Driver Pro
New bundle
Delete all
Options
Help
PAIR
COM1
COM2
Edit
Delete
Break
Ports status:
Paired
User session:
Global