Računske vežbe iz Projektovanja Elektronskih Sistema

cas 4

Doc.dr Borisav Jovanović

Sadržaj:

- Struktura programa, osnovne funkcije *main()* i *interrupt()*
- Korišćenje flegova
- Primer realizacije konačnih automata.
- Rad sa UART modulom
- Povezivanje i rad sa eksternim LCD16x2 displejem
- SPI modul.
- Rad sa Ethernet modulom.

Struktura programa

```
// deklaracija promenljivih
void init() {  ….
}
void init_variables() { ….
}

void main() {
   init();
        init_variables();
        while(1) { ….
        // provera flegova I pozivanje funkcija
        }
}
void interrupt() {
    if ((PIE1.RCIE==1) && (PIR1.RCIF==1))   { ….
    }
     …..
    if ( (PIE1.TMR1IE==1) && (PIR1.TMR1IF==1)) {
    }
}
```

važno: Korišćenje flegova

- U funkciji **interrupt()** smeju da se izvršavaju samo delovi programa koji kratko traju (nema čekanja).
- Ako treba izvršavati duge sekvence programa, ili pak pozivati funkcije koje recimo prenose podatke preko SPI, UART i sl., u funkciji **interrupt()** treba postaviti **Flag=1** (oznaka da nešto tek treba da se uradi), koji treba potom ispitati u beskonačnoj petlji glavnog programa **while(1)**.
- ako je u **while(1)** uslov **if (Flag==1)** ispunjen, tada se izvrsavaju dugotrajne funkcije. Takođe, Flag se resetuje.

primer korišćenja flegova

```
void main() {
      init();
       init_variables();
       while(1) {    // provera flegova i pozivanje  funkcija  koje TRAJU
            if (Flag1==1) {
               Flag1=0;
                ….
            }
            else if (Flag2==1){
               Flag2=0;
                ….
            }
      }
}
}
```

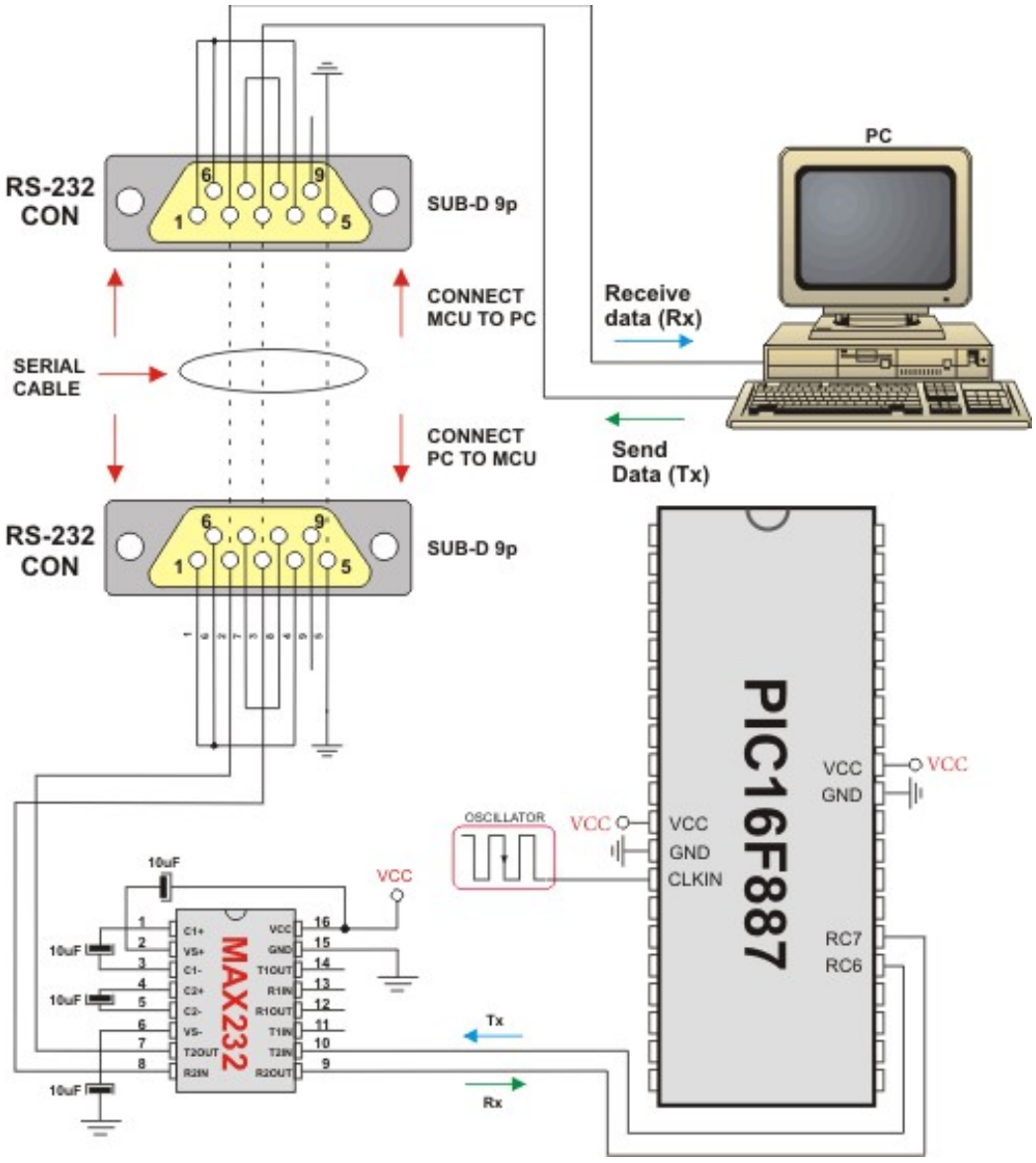
```
void interrupt() {
   if ((PIE1.RCIE==1) && (PIR1.RCIF==1))  {
         Flag1=1;     .........
   }
   else if ( (PIE1.TMR1IE==1) && (PIR1.TMR1IF==1)) {
         Flag2=1;      ........
   }
```

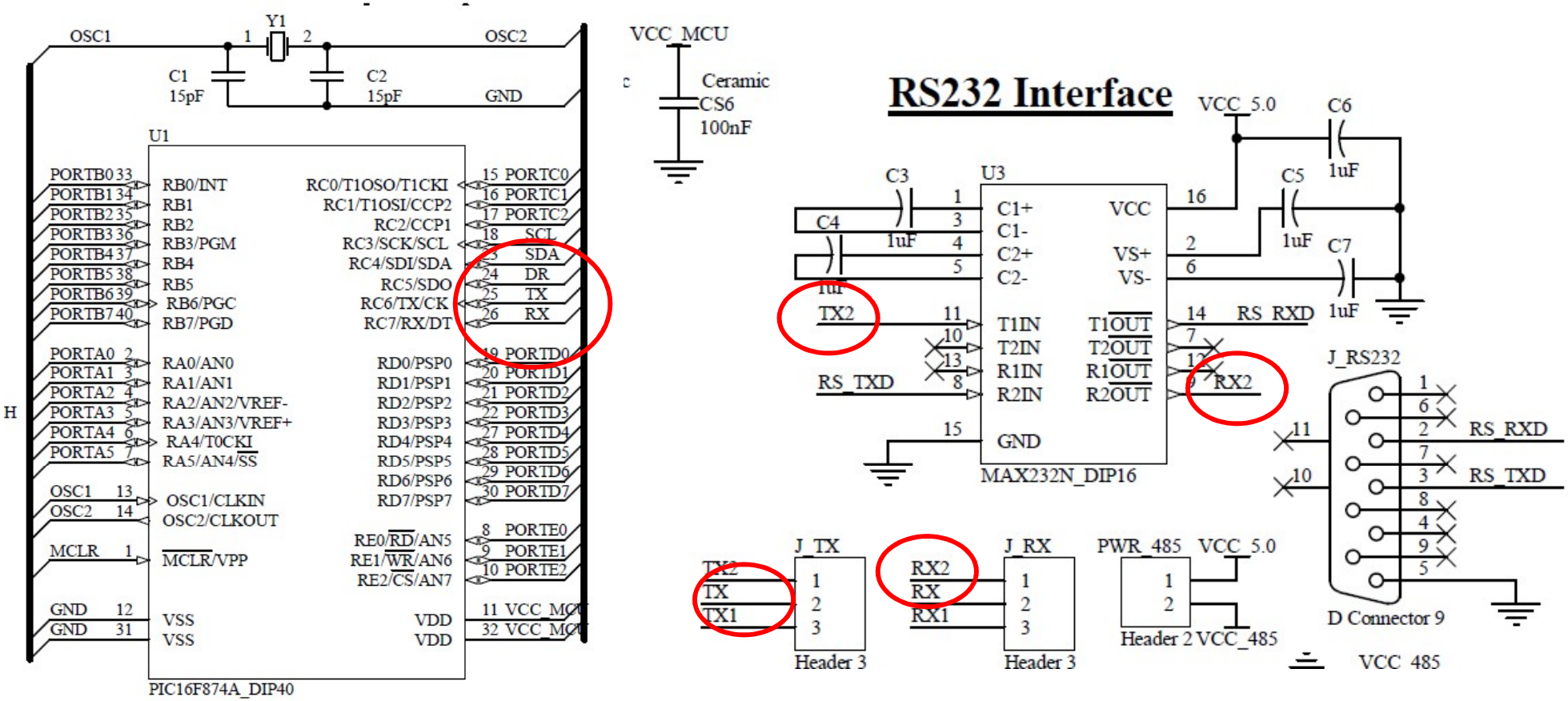
primer korišćenja konačnih automata

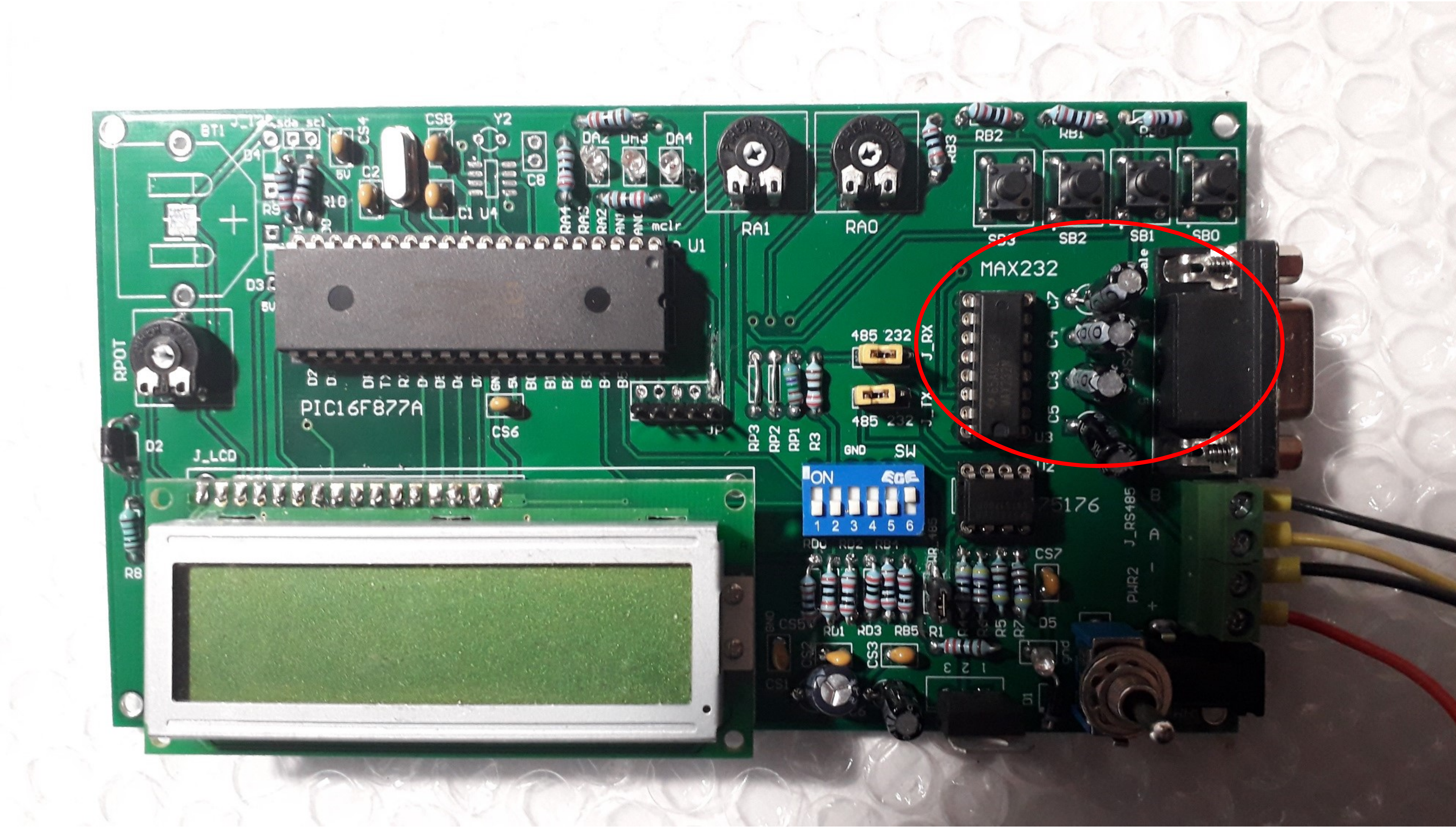
```
void main() {
    …….
     while(1) {
         if (Flag1==1) {  // 1 sekunda
            Flag1=0;
            switch (state) {
            case 0:  if (uslov0==1) state=1;
                       else  state=0;
                       ……
                     break;
            case 1:  if (uslov1==1) state=2;
                       else  state=1;
                     ……
                     break;
            ……..
            default: state=0;
          }
            …..
         }
    }
}
```

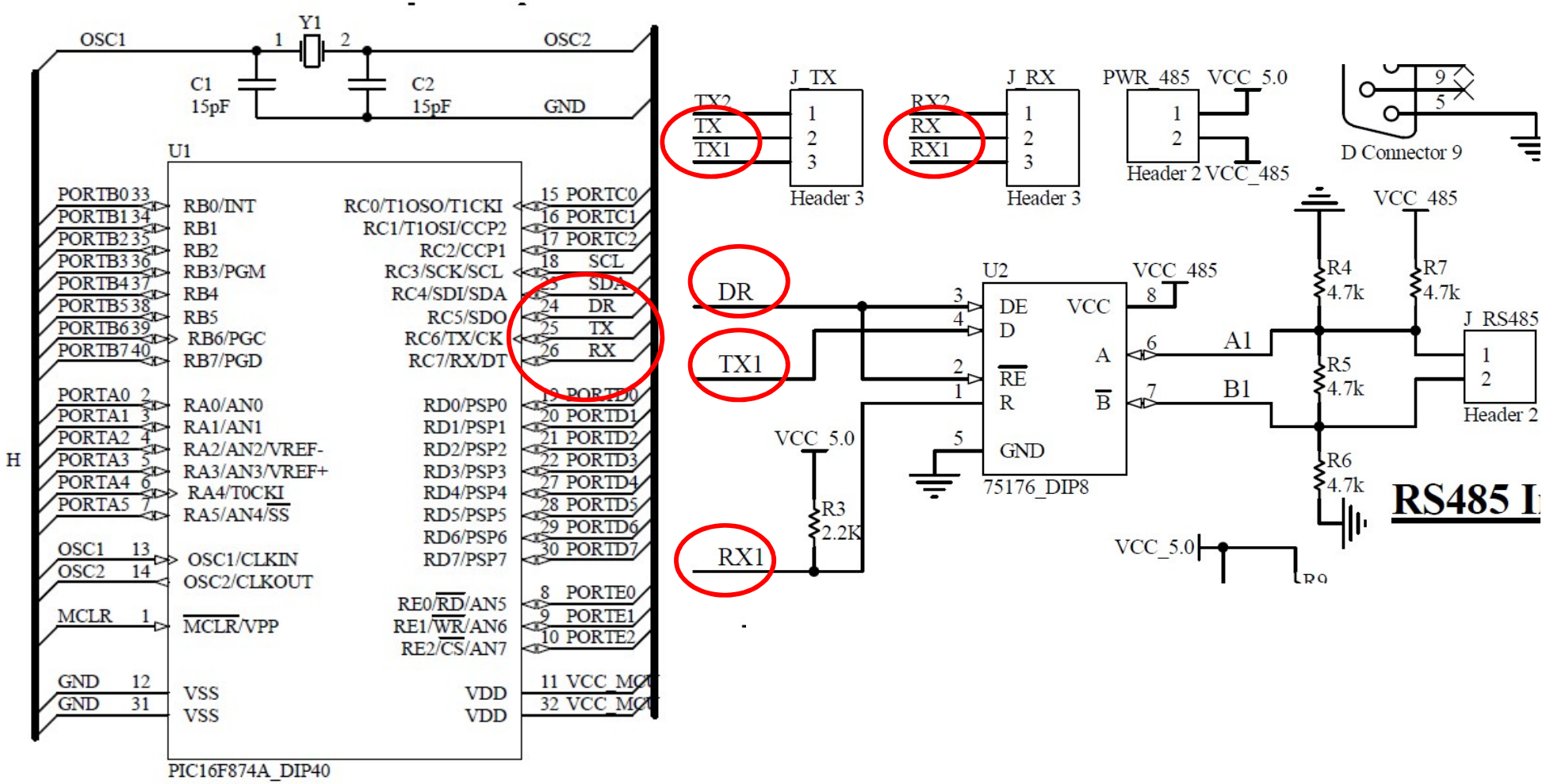
```
void interrupt() {
  if ( (PIE1.TMR1IE==1) && (PIR1.TMR1IF==1)) { //
na svakih 100 ms
      if  (counter<9) counter++;
      else counter=0;
      if (counter==0) Flag1=1;
      ……..
   }
}
```

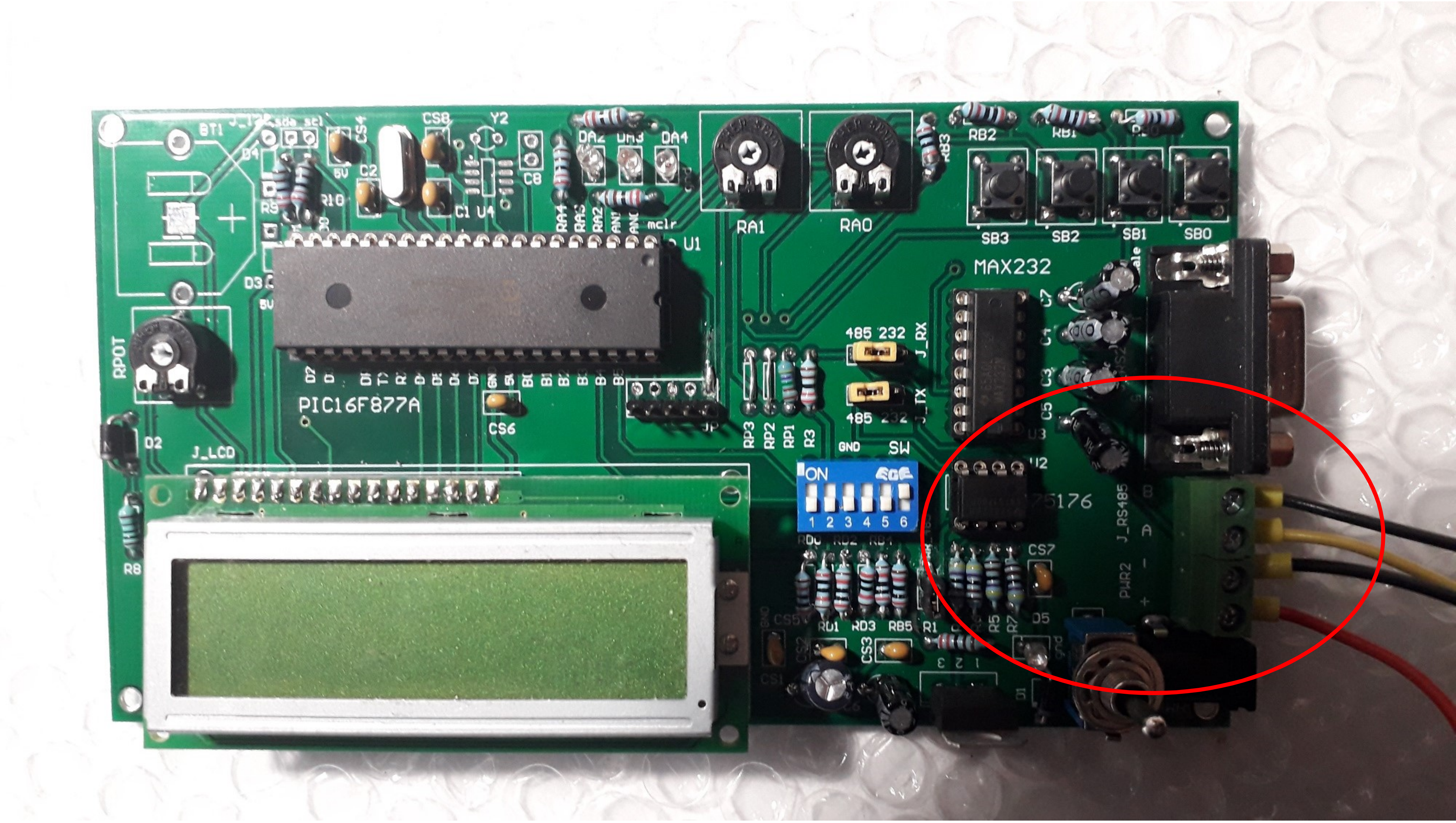
Rad sa UART modulom



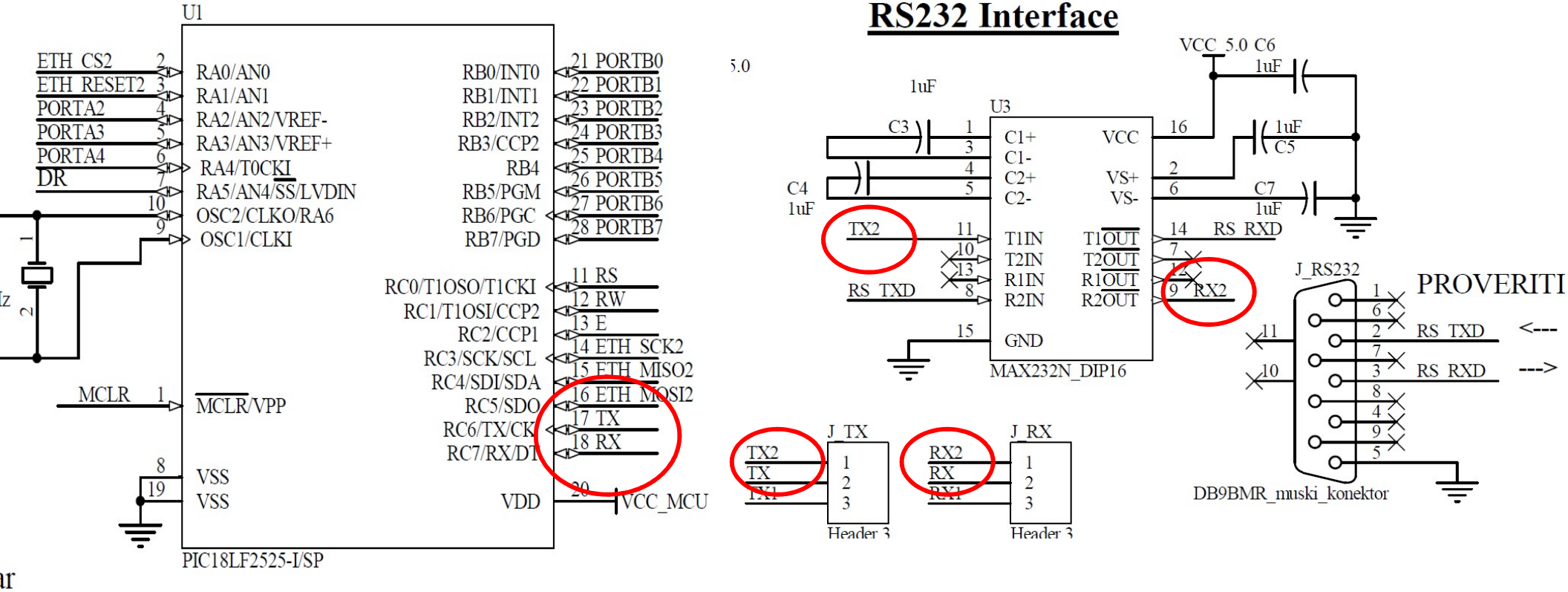

OSC1
Y1
OSC2
C1
15pF
C2
15pF
GND
U1
PIC16F874A_DIP40
VCC_MCU
Ceramic
CS6
100nF
RS232 Interface
VCC_5.0
C6
1uF
C3
C4
C5
C7
U3
MAX232N_DIP16
TX2
RS_TXD
RS_RXD
RX2
J_RS232
D Connector 9
J_TX
J_RX
PWR_485
Header 3
Header 2
VCC_485

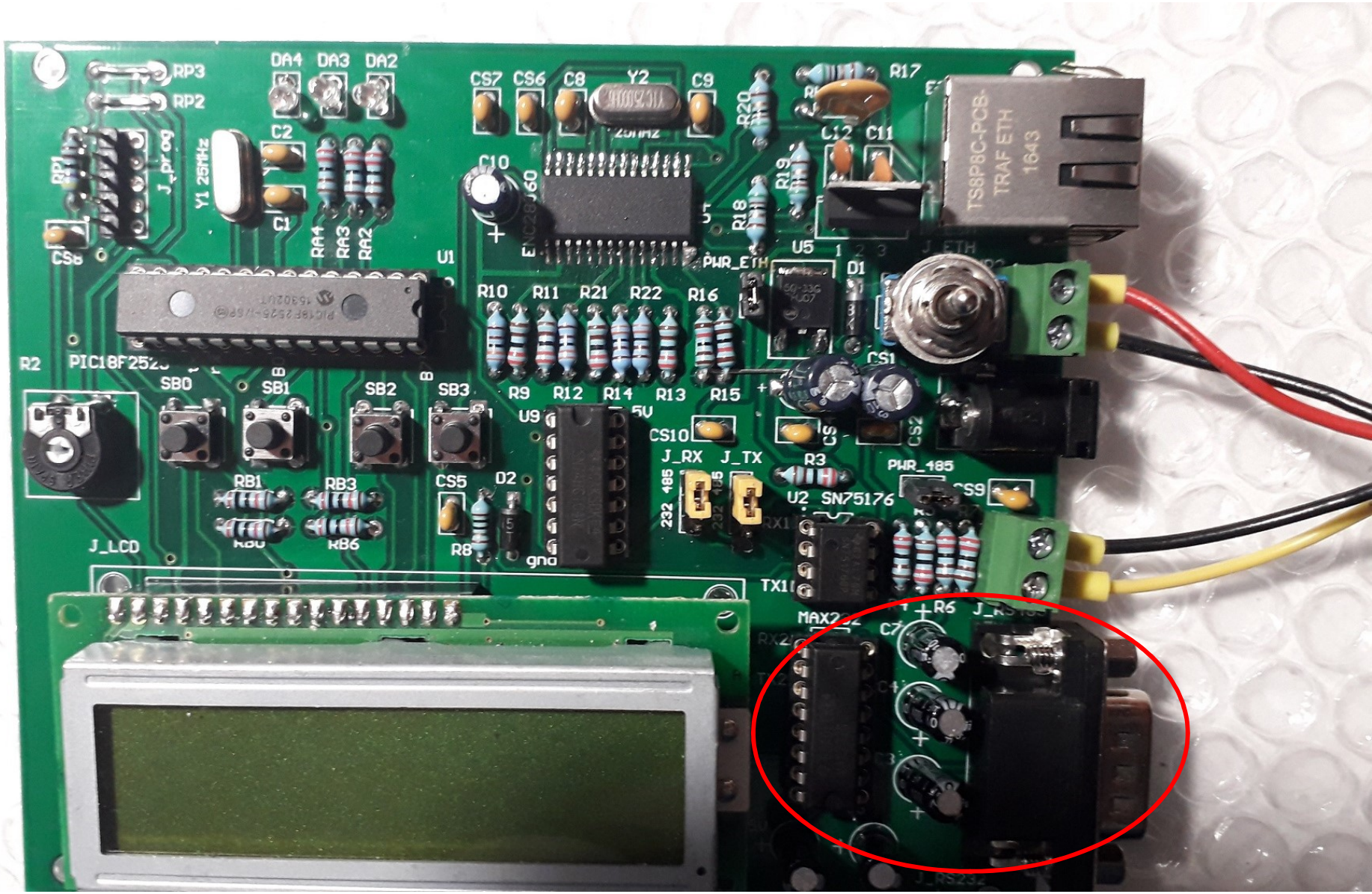

MAX232
PIC16F877A
RPOT
J_LCD
CS6
RA1
RA0
RB2
SB2
SB1
SB0
485 232
J_RX
J_TX
485 232
RP3
RP2
RP1
R3
GND
SW
ON
1 2 3 4 5 6
75176
J_RS485
PWR2
CS7
D5
R5
R7
RD1
RD3
RB5
R1
D2
R8
C1 U4
Y2
CS8
DA2
DA4
C8
U1
mclr
C7
C4
C3
C5
U3


OSC1
Y1
OSC2
C1
15pF
C2
15pF
GND
U1
PIC16F874A_DIP40
J_TX
J_RX
PWR_485
VCC_5.0
Header 3
Header 2
VCC_485
D Connector 9
DR
TX1
RX1
U2
75176_DIP8
R3
2.2K
R4
4.7k
R5
4.7k
R6
4.7k
R7
4.7k
A1
B1
J_RS485
RS485 I

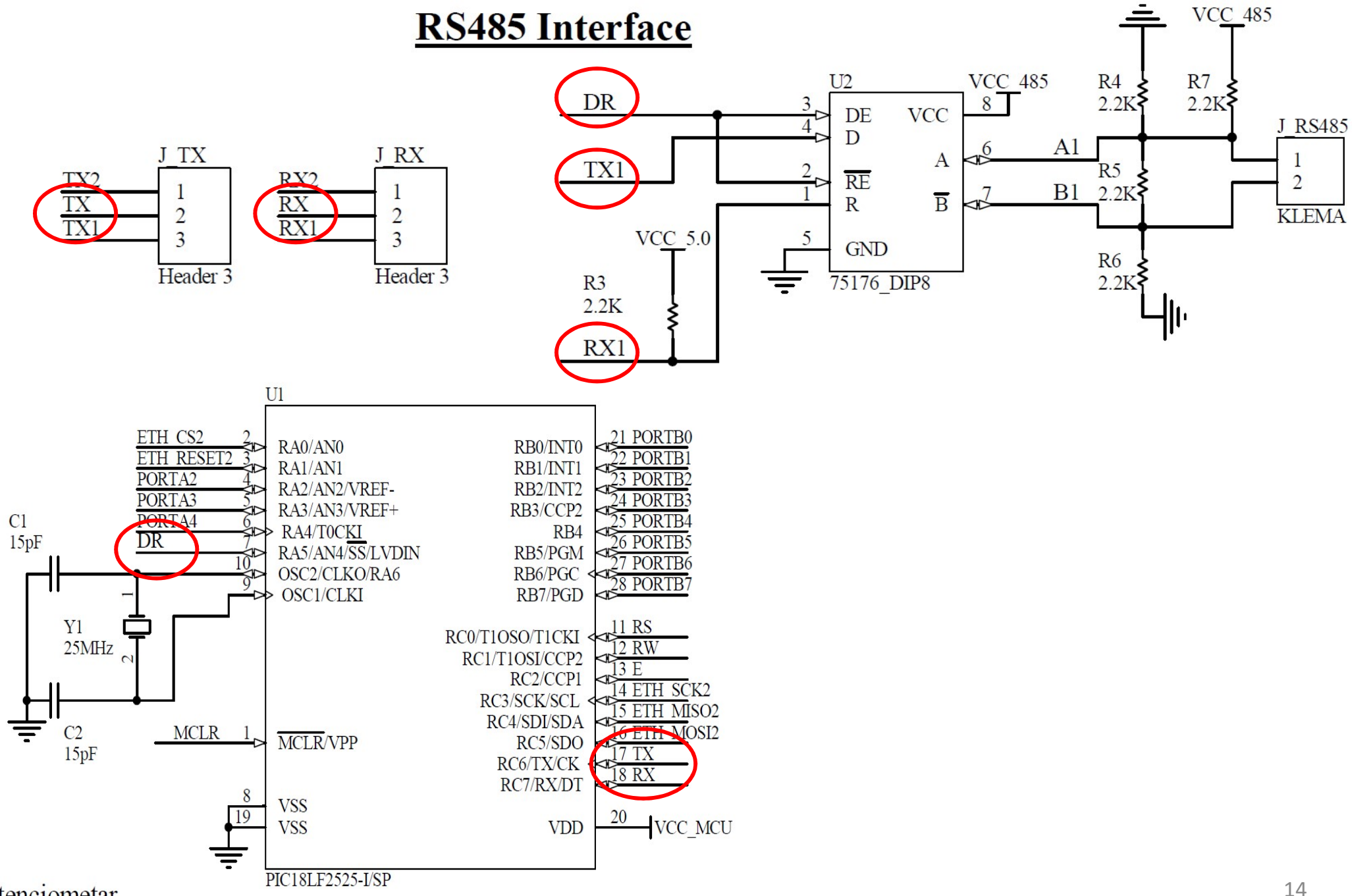


PIC16F877A
MAX232
75176
J_RS485
PWR2
RA1
RA0
SB3
SB2
SB1
SB0
RB2
RB3
J_LCD
RPOT
BT1
Y2
CS8
C8
C2
C1 U4
DA2
DA4
U1
mclr
D2
D3
R8
R9
R10
CS6
CS7
JP
RP3
RP2
RP1
R3
485 232
J_RX
J_TX
GND
SW
ON
1 2 3 4 5 6
U2
U3
C7
C4
C3
C5
R5
R7
D5
RD1
RD3
RB5
R1
CS3
CS1
B
A

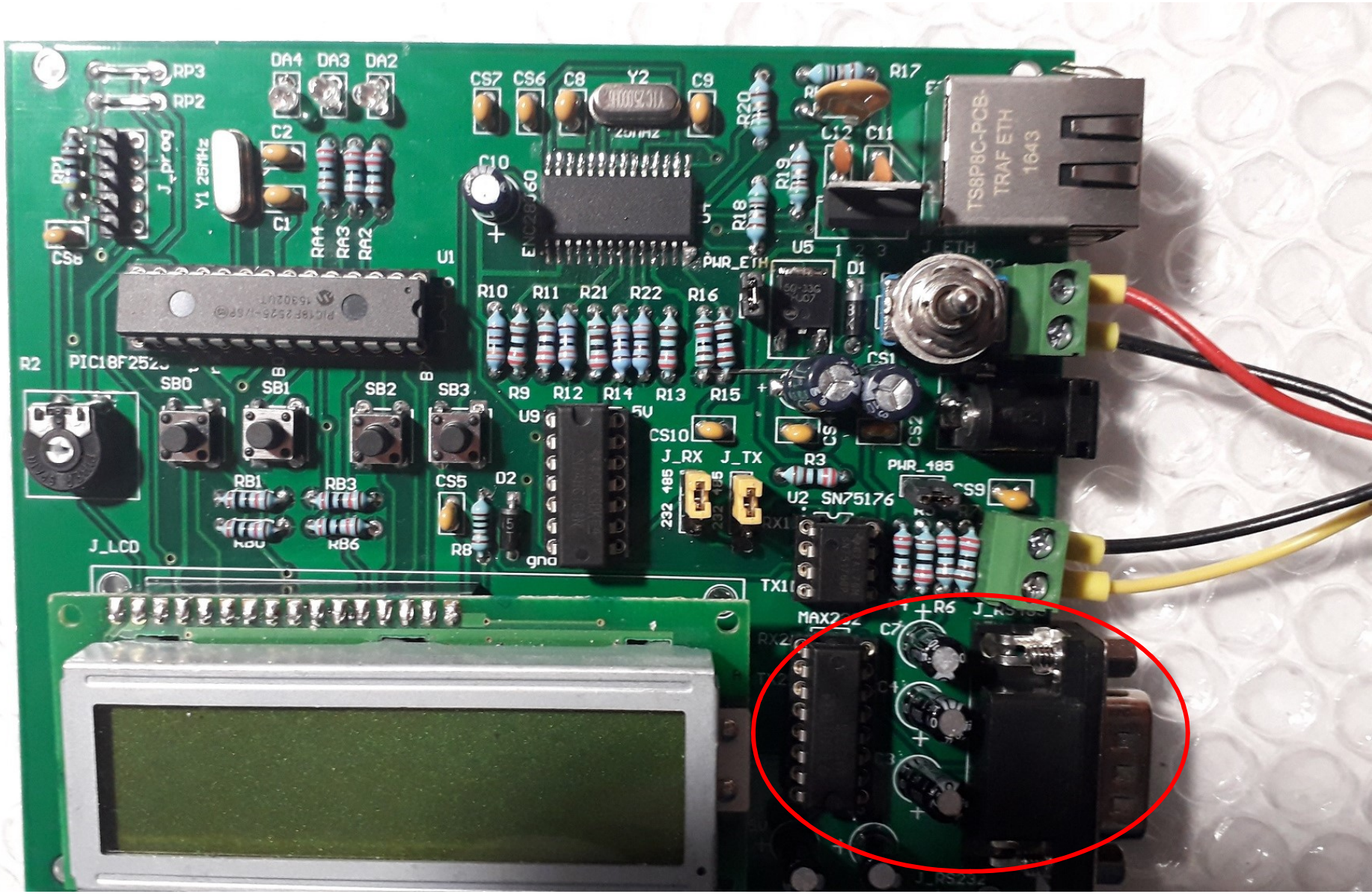

RS232 Interface
U1
ETH_CS2
ETH_RESET2
PORTA2
PORTA3
PORTA4
DR
RA0/AN0
RA1/AN1
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/SS/LVDIN
OSC2/CLKO/RA6
OSC1/CLKI
MCLR
MCLR/VPP
VSS
VSS
RB0/INT0
RB1/INT1
RB2/INT2
RB3/CCP2
RB4
RB5/PGM
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
VDD
PORTB0
PORTB1
PORTB2
PORTB3
PORTB4
PORTB5
PORTB6
PORTB7
RS
RW
E
ETH_SCK2
ETH_MISO2
ETH_MOSI2
TX
RX
VCC_MCU
PIC18LF2525-I/SP
VCC_5.0
C6
1uF
C3
C4
C5
C7
U3
C1+
C1-
C2+
C2-
VCC
VS+
VS-
T1IN
T2IN
R1IN
R2IN
T1OUT
T2OUT
R1OUT
R2OUT
GND
TX2
RS_TXD
RS_RXD
RX2
MAX232N_DIP16
J_TX
J_RX
Header 3
J_RS232
DB9BMR_muski_konektor
PROVERITI
<---
-->


RP3
RP2
DA4
DA3
DA2
CS7
CS6
C8
Y2
C9
R17
C12
C11
TS8P8C-PCB-
TRAF ETH
1643
C2
C10
R20
R19
R18
U5
D1
J_ETH
PWR_ETH
RP1
J_prog
Y1 25MHz
C1
RA4
RA3
RA2
U1
CS8
R10
R11
R21
R22
R16
R9
R12
R14
R13
R15
CS1
R2
PIC18F252
SB0
SB1
SB2
SB3
U9
5V
CS10
J_RX
J_TX
R3
PWR_485
U2 SN75176
RB1
RB3
CS5
D2
R8
gnd
J_LCD
TX1
R6
C7
MAX232

RS485 Interface

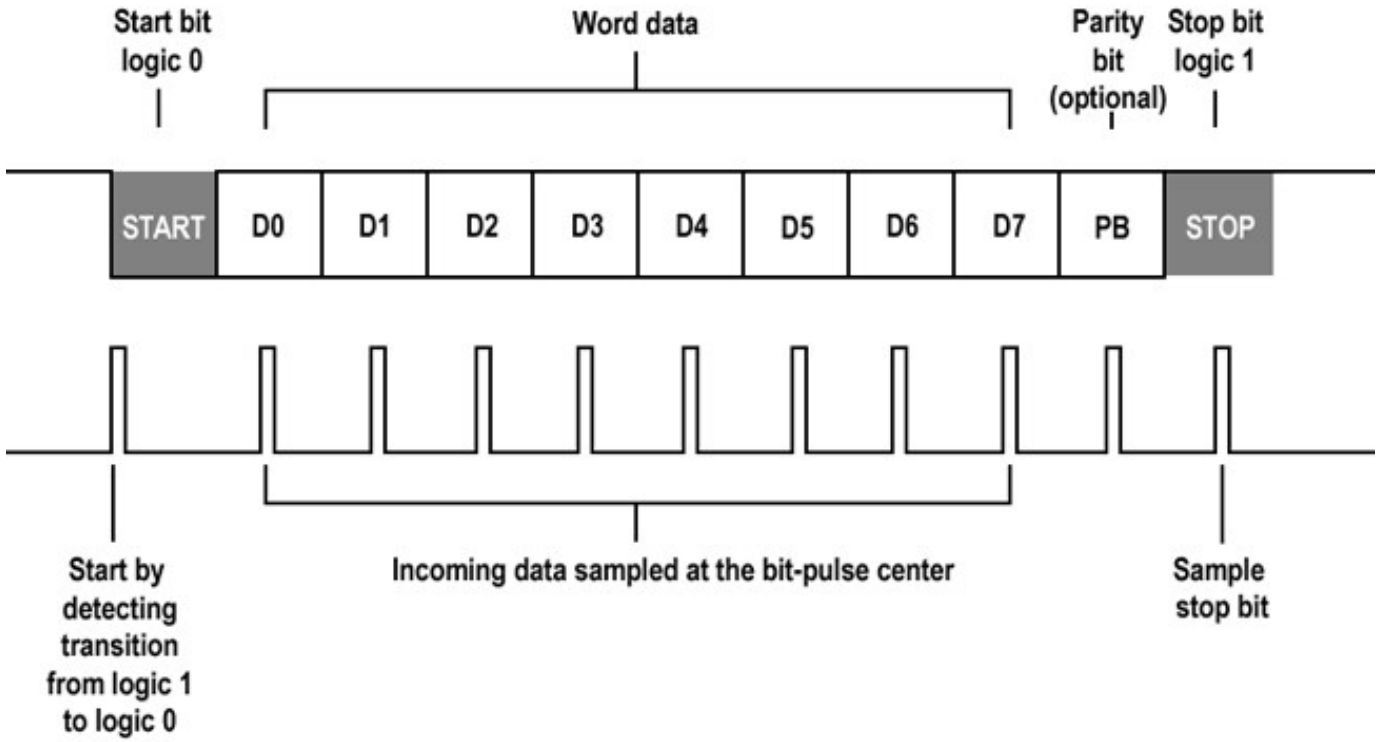


tenciometar


RP3
RP2
DA4
DA3
DA2
CS7
CS6
C8
Y2
C9
R17
C12
C11
TS8P8C-PCB-
TRAF ETH
1643
RP1
J_prog
Y1 25MHz
C2
C1
RA4
RA3
RA2
C10
R20
R19
R18
U5
D1
CS8
U1
PWR_ETH
R10
R11
R21
R22
R16
R2
PIC18F252
SB0
SB1
SB2
SB3
R9
R12
R14
R13
R15
CS1
U9
5V
CS10
J_RX
J_TX
R3
PWR_485
U2 SN75176
RB1
RB3
CS5
D2
J_LCD
R8
gnd
TX1
R6
C7
C4

Konfiguracija serijskog UART porta – deo **init()** funkcije:

```
……
TRISC=0xC0;// pinovi 6 i 7 su vezani za RS232 ili RS485
……

Uart1_Init(19200);
// funkcija je definisana u UART bibliteci Mikro C kompajlera
// konfigurišemo serijski port na brzinu od 19200
// konfigurišemo dodatne bitove za serijski prenos
……
PIE1.RCIE=1; // dozvola prekida
PIR1.RCIF=0; // brisanje flega
…….
INTCON.GIE=1;
// globalna dozvola prekida
```


Start bit
logic 0
Word data
Parity
bit
(optional)
Stop bit
logic 1
START
D0
D1
D2
D3
D4
D5
D6
D7
PB
STOP
Start by
detecting
transition
from logic 1
to logic 0
Incoming data sampled at the bit-pulse center
Sample
stop bit

Slanje informacija preko serijskog prekida

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

```
void transmit  (unsigned char data_8b) {
            TXREG = data_8b;
            while  ( !TXSTA.TRMT);
}
```

Primer poziva funkcije:

```
unsigned char dataX=0xAA;
…..
transmit(dataX);
```

```
void main(){
.....
while(1){
.....
if (CallFlag == 1){
  CallFlag=0;
  if ((Command & 0x10) == 0x10) Operation = 1;
  else Operation = 0;
  if (Error == 1){ // nema kartica
     DR = 1;
     if (Operation == 0x01) CommandModified = 0b00010000 | RAMP_ID;
     else CommandModified = 0b00000000 | RAMP_ID;
     transmit(CommandModified);
     DR = 0;
  } else if (Event == 0) {
     // taster nije pritisnut
     // vraca se prozivka
     DR = 1;
     if (Operation == 1) CommandModified = 0b00110000 | RAMP_ID;
     else CommandModified = 0b00100000 | RAMP_ID;
     transmit(CommandModified);
     DR = 0;
  }
}
}
]
```

Prijem podataka korišćenjem prekida

- Prekid UART prijemnika se nalazi u grupi periferijskih prekida, tako da podešavamo registre PIE1 i PIR1
- Prijem 8-bitnog podatka se registruje flegom RCIF (PIR1<6>).
- Ovaj prekid može da se dozvoli/ zabrani postavljanjem / resetovanjem bita RCIE (PIE1<6>).

PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

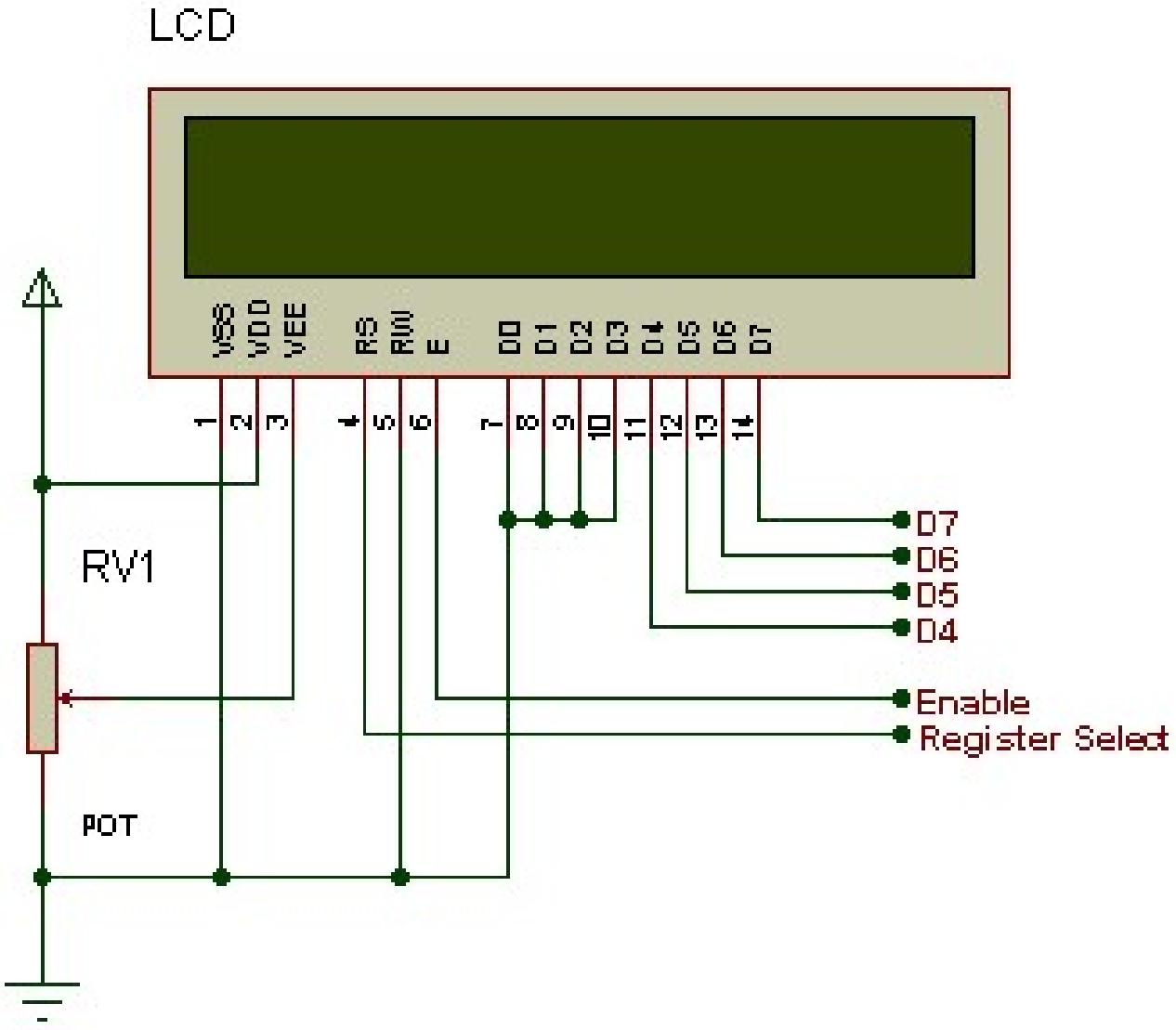
```
if ((PIE1.RCIE==1) && (PIR1.RCIF==1))  {

……
      ch=RCREG;
……

}
```

```
if ((PIE1.RCIE) && (PIR1.RCIF)){
      PIR1.RCIF = 0;
      ch = RCREG;
      if (BytesToReceive == 0x00){
         if ((ch & 0x0F) == RAMP_ID){//adresa slejva se poklapa
            Command = ch;
            if ((ch & 0xE0) == 0x20){
               // primljeni bajt je bajt prozivke
               BytesToReceive = 0x00;
               CallFlag = 1;
            }
            else if ((ch & 0xE0) == 0x60){
               // primljeni bajt je komanda za
               // podesavanje sata realnog vremena
               BytesToReceive = 0x03;
               Counter2 = 3;
               // 300 ms je vreme tokom kojeg
               // trebaju da stignu preostali bajtovi
            }
         }
      }
...
}
```

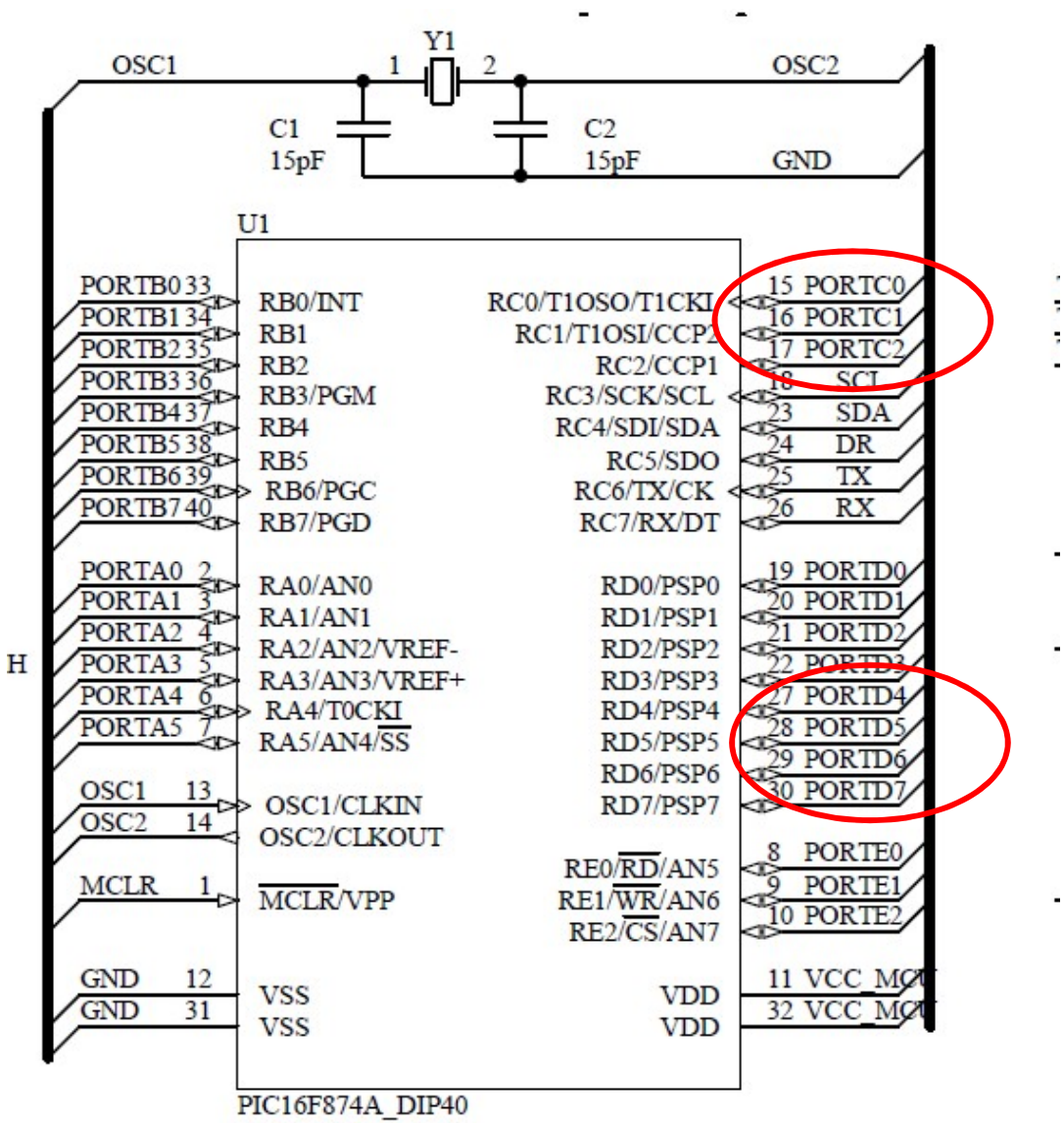
Povezivanje i rad sa eksternim LCD16x2 displejem

- The mikroC PIC poseduje biblioteku za rad sa LCD displejem, koji se povezuje preko 4-bitnog interfejsa na mikrokontroler.
- Primer povezivanja interfejsa LCD–a na pinove PIC16F877A je objašnjen, kao i upotreba osnovnih funkcija LCD biblioteke.

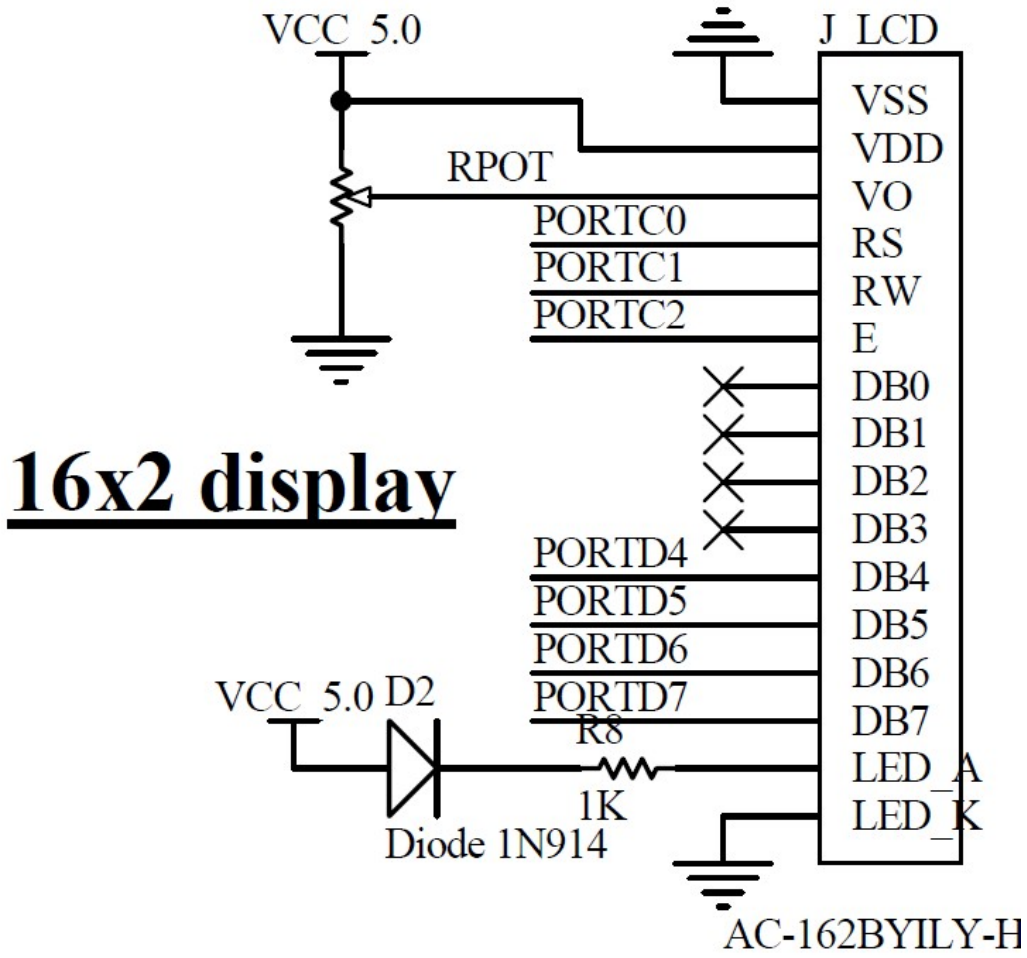
Povezivanje i rad sa eksternim LCD16x2 displejem



Povezivanje i rad sa eksternim LCD16x2 displejem

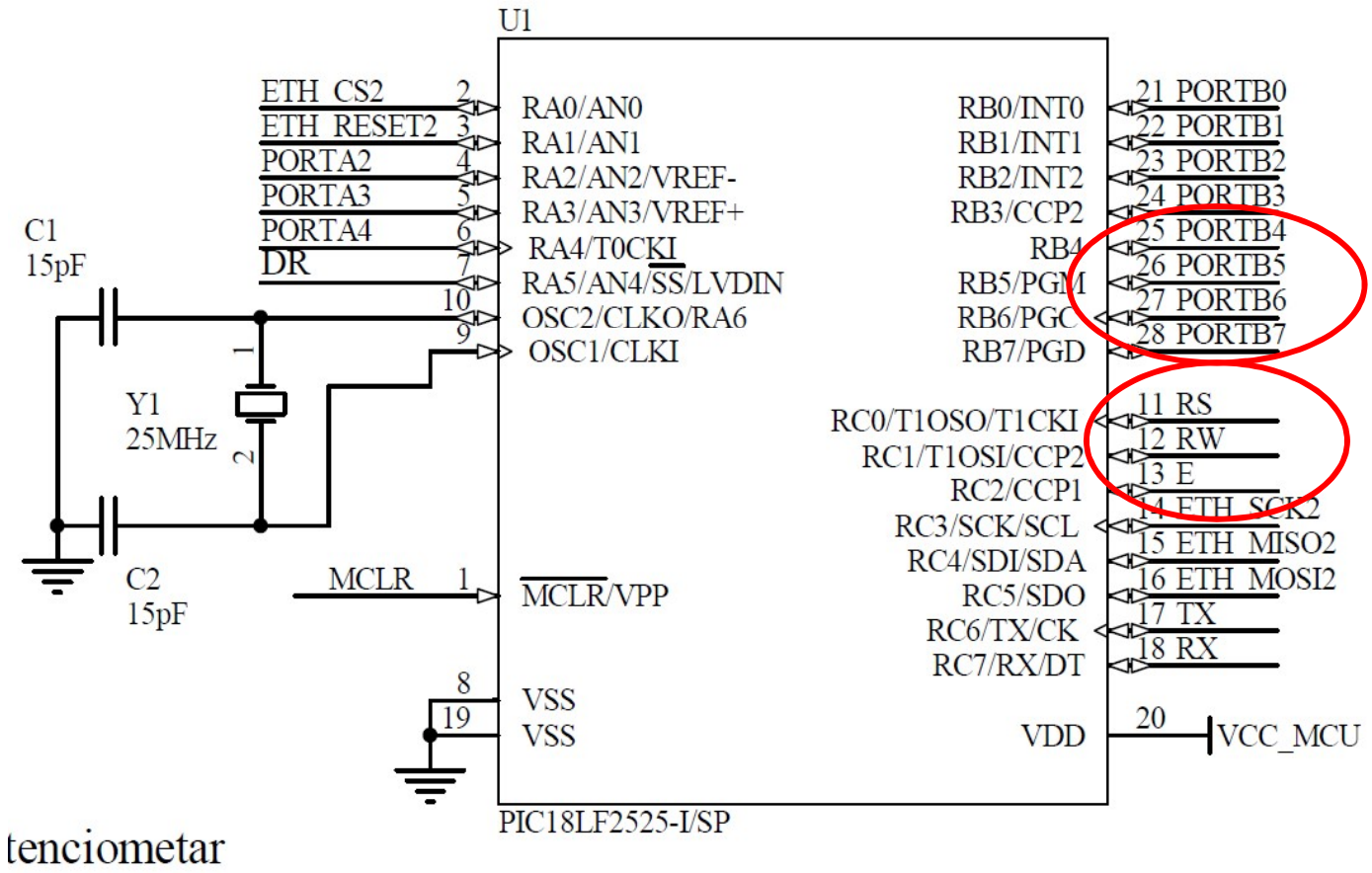


Povezivanje i rad sa eksternim LCD16x2 displejem

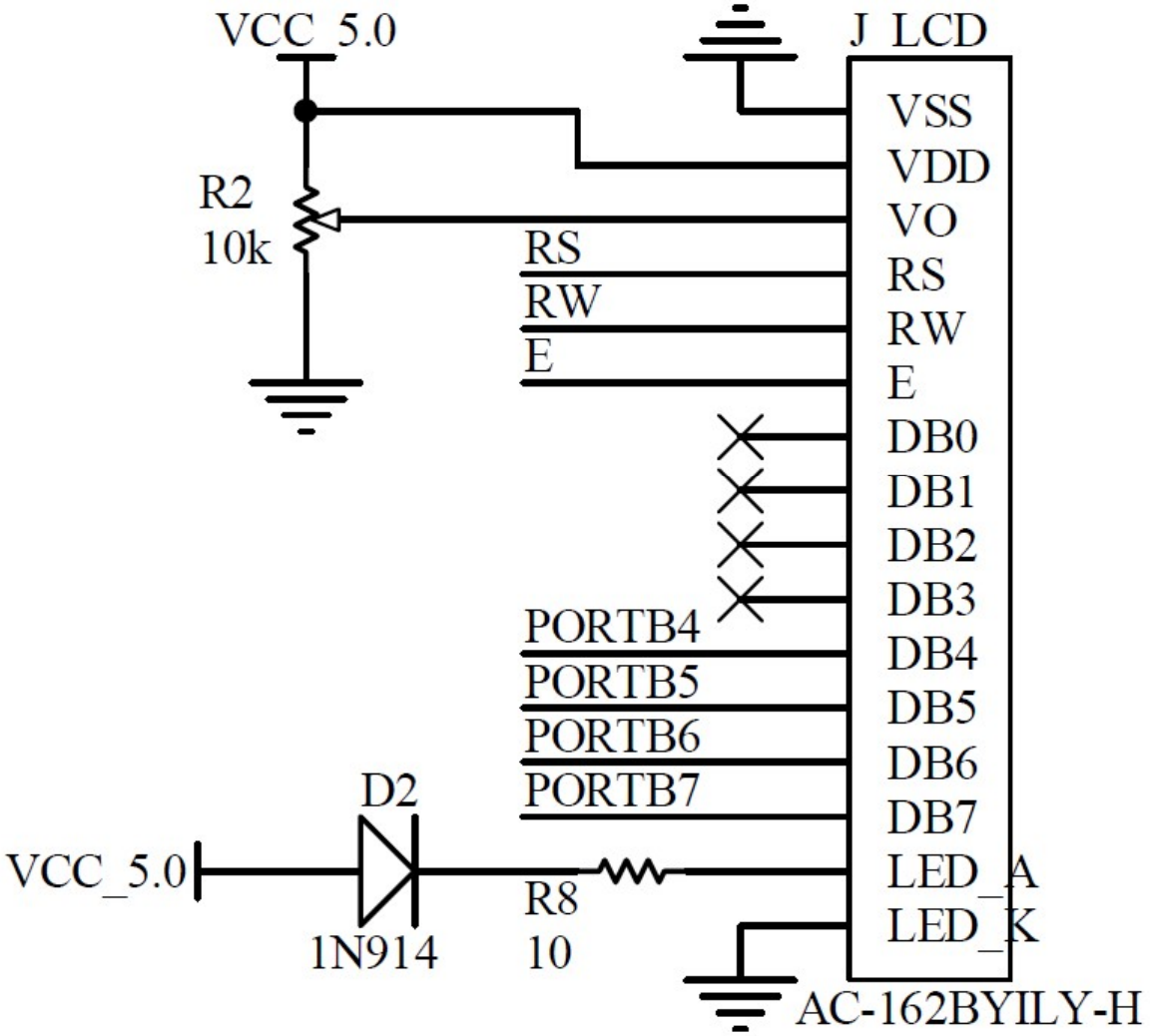


```
// Lcd pinout settings
sbit LCD_RS at RC0_bit;
sbit LCD_EN at RC2_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D4 at RD4_bit;

// Pin direction
sbit LCD_RS_Direction at TRISC0_bit;
sbit LCD_EN_Direction at TRISC2_bit;
sbit LCD_D7_Direction at TRISD7_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D4_Direction at TRISD4_bit;
```

Rad sa Ethernet modulom.



potenciometar

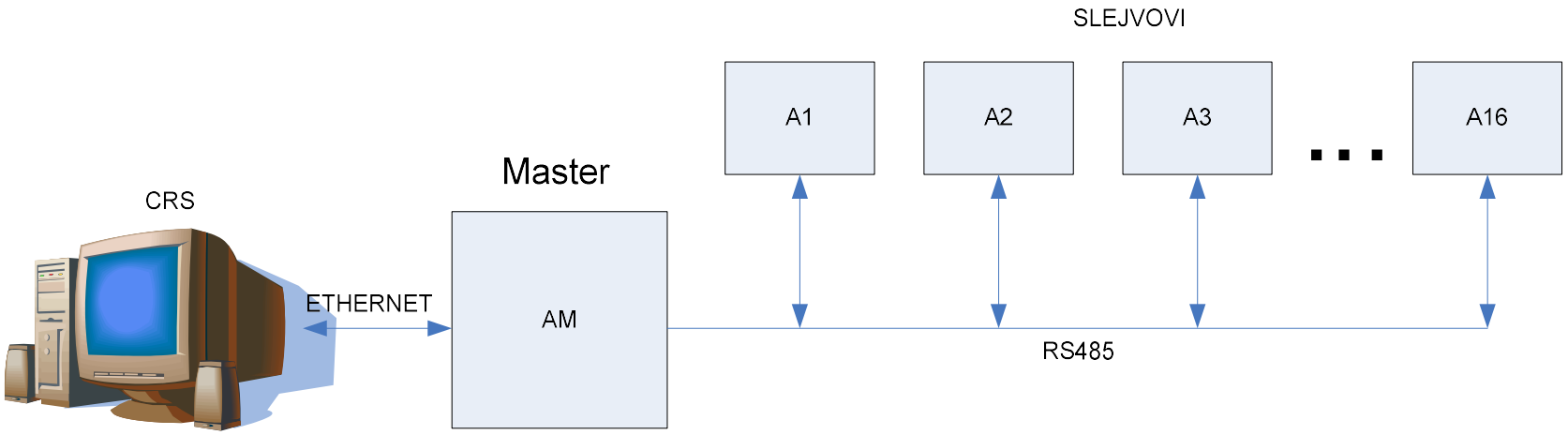


```
// Lcd pinout settings
sbit LCD_RS at RC0_bit;
sbit LCD_RW at RC1_bit; // Dodao sam LCD_RW
sbit LCD_EN at RC2_bit;
sbit LCD_D7 at RB7_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D4 at RB4_bit;

// Pin direction
sbit LCD_RS_Direction at TRISC0_bit;
sbit LCD_RW_Direction at TRISC1_bit; // Dodao sam LCD_RW
sbit LCD_EN_Direction at TRISC2_bit;
sbit LCD_D7_Direction at TRISB7_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB4_bit;
```

```
Lcd_Init(); // samo jednom pri inicijalizaciji

// Write text "Hello!" on Lcd starting from row 1, column 3:
Lcd_Out(1, 3, "Hello!");



// Write character "i" at row 2, column 3:
Lcd_Chr(2, 3, 'i');


// Clear Lcd display:
Lcd_Cmd (_LCD_CLEAR);


 // Cursor off
 Lcd_Cmd (_LCD_CURSOR_OFF);
```

```
void UpdateLCD()  {
   Lcd_Out(1, 1, "Time:");
   Lcd_Chr(1, 6, (Hour_X10 + 0x30));
   Lcd_Chr(1, 7, (Hour_X1 + 0x30));
   Lcd_Chr(1, 8, ':');
   Lcd_Chr(1, 9, (Min_X10 + 0x30));
   Lcd_Chr(1, 10, (Min_X1 + 0x30));
   Lcd_Chr(1, 11, ':');
   Lcd_Chr(1, 12, (Sec_X10 + 0x30));
   Lcd_Chr(1, 13, (Sec_X1 + 0x30));
   if (Operation2 == 0x01)
      Lcd_Out(2, 1, "Operating");
   else if (Error == 0x01)
      Lcd_Out(2, 1, "Error    ");
   else
      Lcd_Out(2, 1, "         ");
   if (RampOpen2 == 0x01)
      Lcd_Out(2, 11, "Opened");
   else
      Lcd_Out(2, 11, "Closed");
}
```

```
void UpdateLCD(){
   int i = 0;
   Lcd_Out(1, 1, "Operation    ");
   for (i = 0; i <= 15; i++)
   {
      if (Operation[i] == 1)
         Lcd_Chr(2, 16 - i, '1');
      else
         Lcd_Chr(2, 16 - i, '0');
   }
}
```

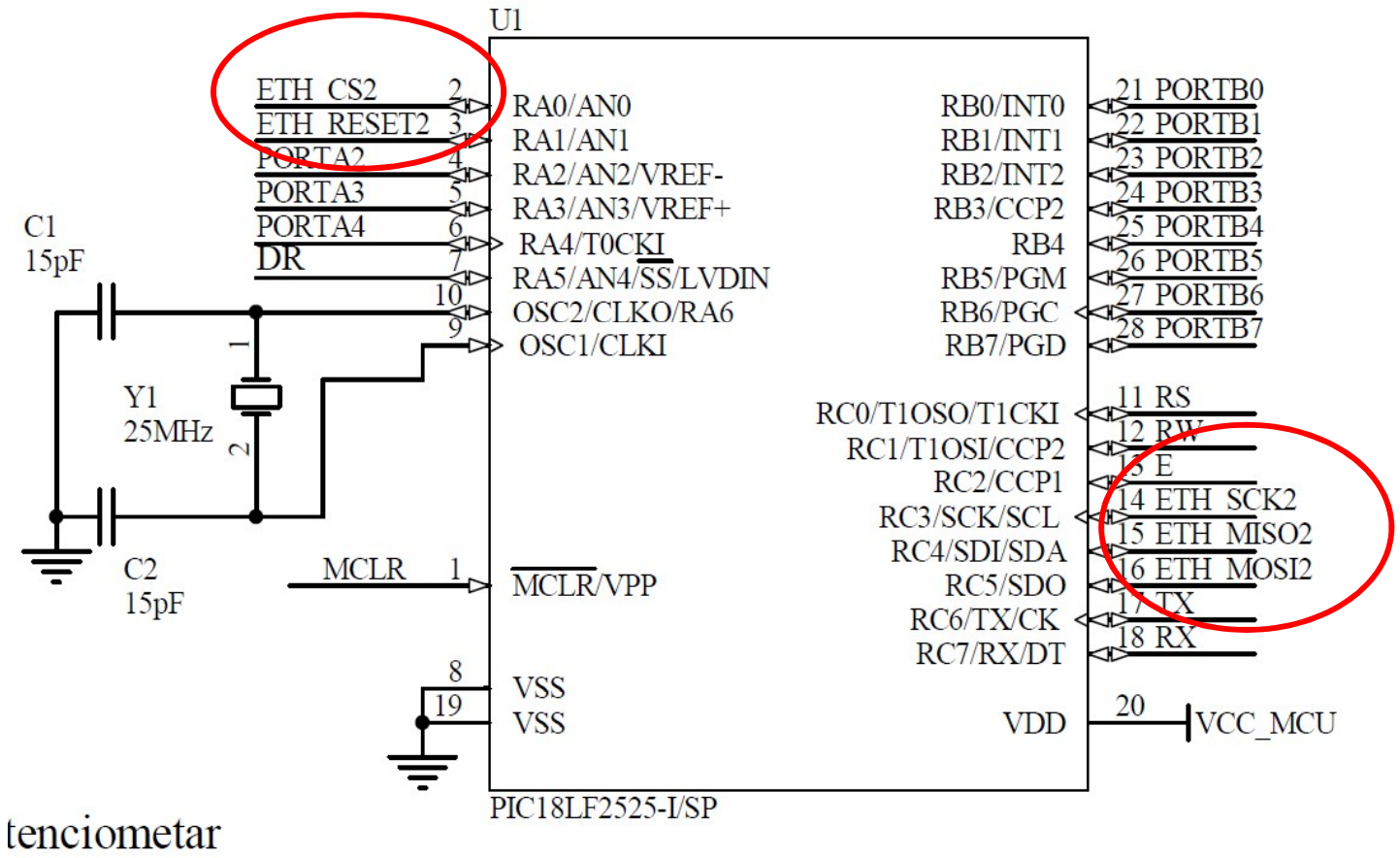


Osnovne komponente sistema su:

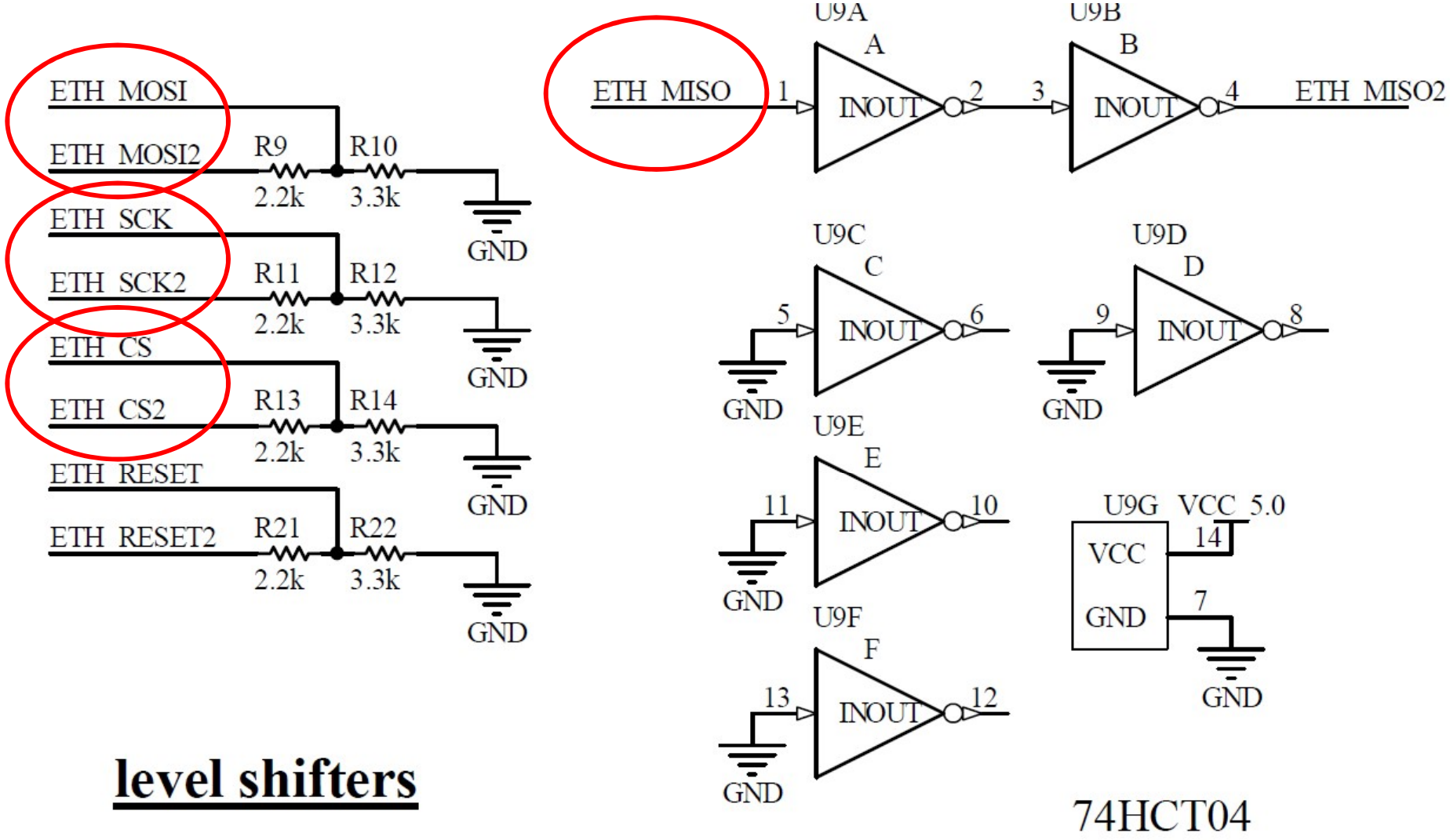
- CRS (Centralni Računarski Sistem)
- Master automat (AM)
- Slejv automati, koje se vezuju za svako ulazno mesto na autoputu (A1-A16).

Master automat AM prikuplja informacije od slejv automata i te informacije prenosi dalje CRS-u. CRS je server računar koji je preko Etherneta povezan sa Master Automatima (AM). AM je povezan preko RS485 veze sa svim slejvovima.

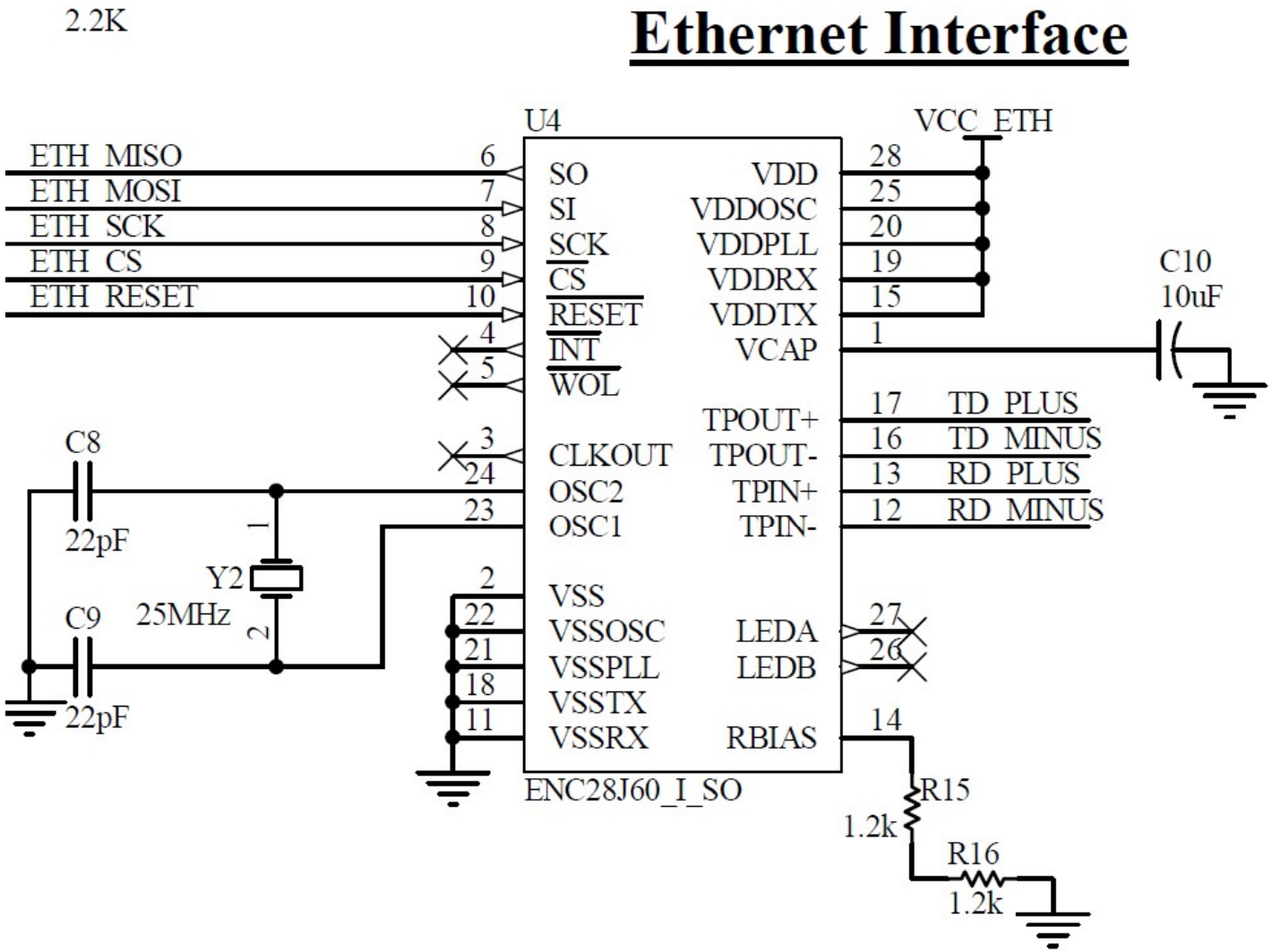
Rad sa Ethernet modulom.

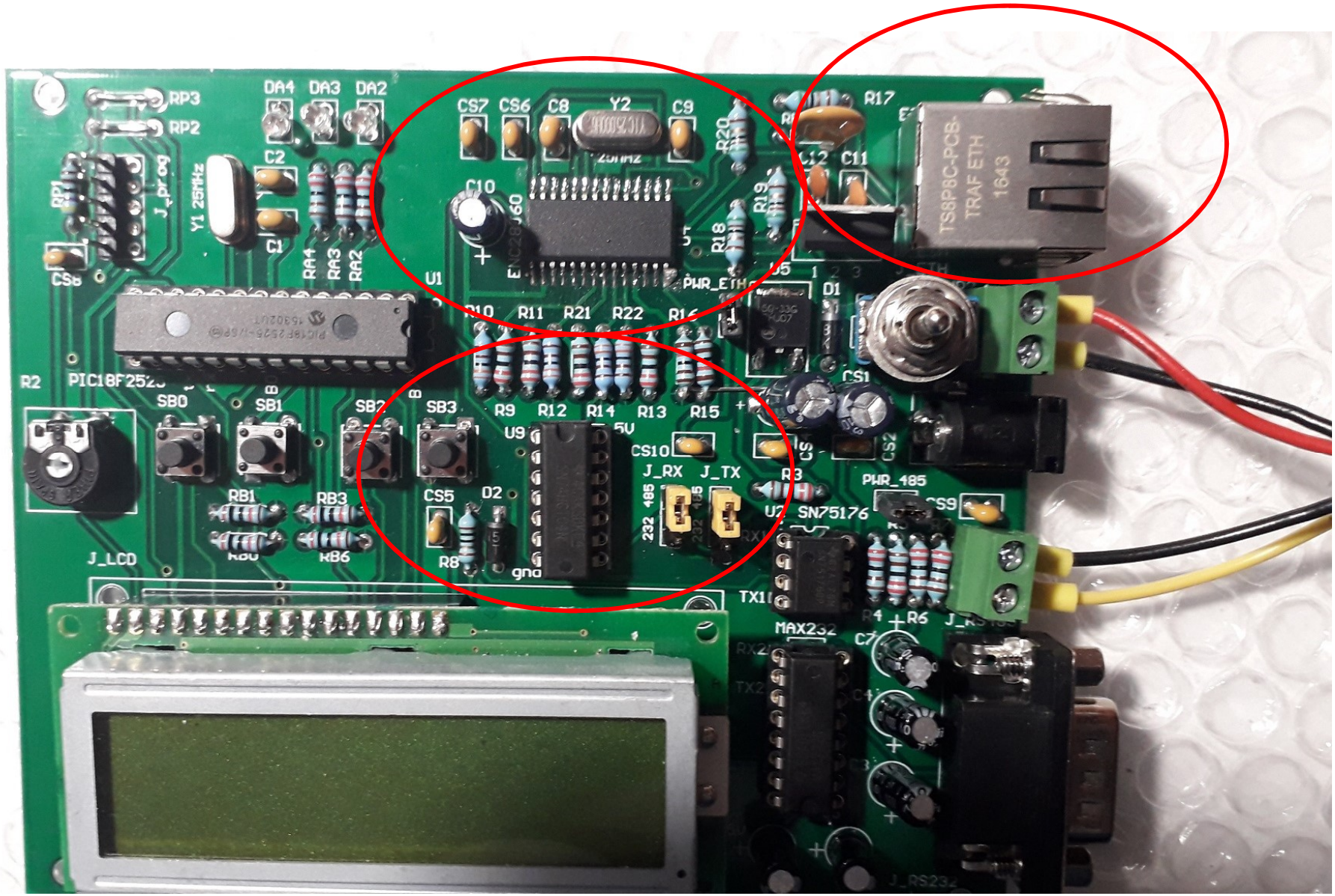


Rad sa Ethernet modulom.



Rad sa Ethernet modulom.



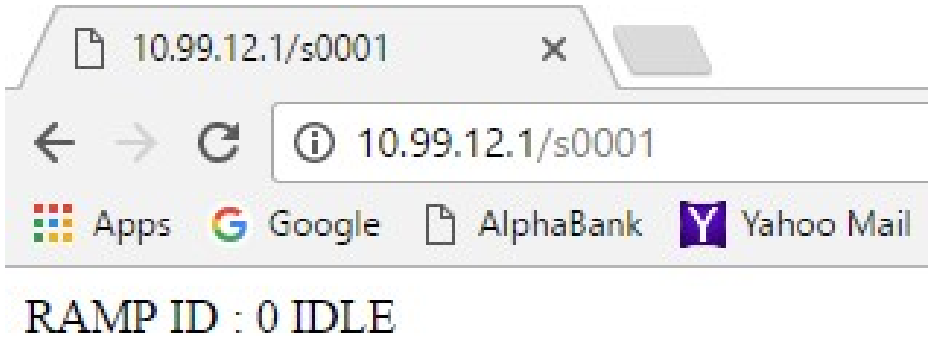

RP3
RP2
DA4
DA3
DA2
CS7
CS6
C8
Y2
C9
R17
C12
C11
TS8P8C-PCB-
TRAF ETH
1643
C2
C1
Y1 25MHz
J_prog
RP1
CS8
RA4
RA3
RA2
U1
C10
R20
R19
R18
U5
PWR_ETH
D1
R10
R11
R21
R22
R16
CS1
R2
PIC18F252
SB0
SB1
SB2
SB3
R9
R12
R14
R13
R15
5V
U9
CS10
J_RX
J_TX
R3
PWR_485
CS9
U2
SN75176
RB1
RB3
CS5
D2
R8
gnd
J_LCD
TX1
MAX232
R4
R6
C7
C4
J_RS232

```
const unsigned char httpHeader[] = "HTTP/1.1 200 OK\nContent-type: ";
const unsigned char httpMimeTypeHTML[] = "text/html\n\n";
const unsigned char httpMimeTypeScript[] = "text/plain\n\n";
unsigned char httpMethod[] = "GET /";

// mE ethernet NIC pinout
sfr sbit SPI_Ethernet_Rst at RA1_bit; //SPI_ETH_RST
sfr sbit SPI_Ethernet_CS at RA0_bit;  //SPI_ETH_CS2
sfr sbit SPI_Ethernet_Rst_Direction at TRISA1_bit;
sfr sbit SPI_Ethernet_CS_Direction at TRISA0_bit;

// end ethernet NIC definitions
unsigned char myMacAddr[6] = {0x00, 0x14, 0xA5, 0x76, 0x19, 0x3f};
// jedinstvena MAC adresa uredaja
unsigned char myIpAddr[4] = {10, 99, 12, 1};
// IP adresa uredaja
unsigned char getRequest[15];  // HTTP request buffer
```

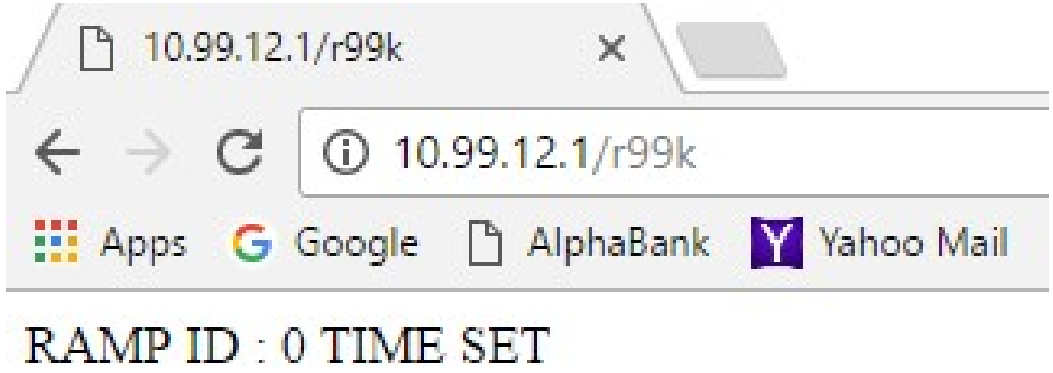
```
// inicijalizujemo SPI

SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64,
   _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW,
   _SPI_LOW_2_HIGH);

// inicijalizujemo Ethernet
 SPI_Ethernet_Init(myMacAddr, myIpAddr,
   SPI_Ethernet_FULLDUPLEX);
```

```
SPI_Ethernet_doPacket();

unsigned int SPI_Ethernet_UserUDP(unsigned char
   *remoteHost, unsigned int remotePort, unsigned int
   destPort, unsigned int reqLength, TEthPktFlags *flags) {
    return 0;
}

unsigned int SPI_Ethernet_UserTCP(unsigned char
   *remoteHost, unsigned int remotePort, unsigned int
   localPort, unsigned int reqLength, char *canClose)
{
}
```

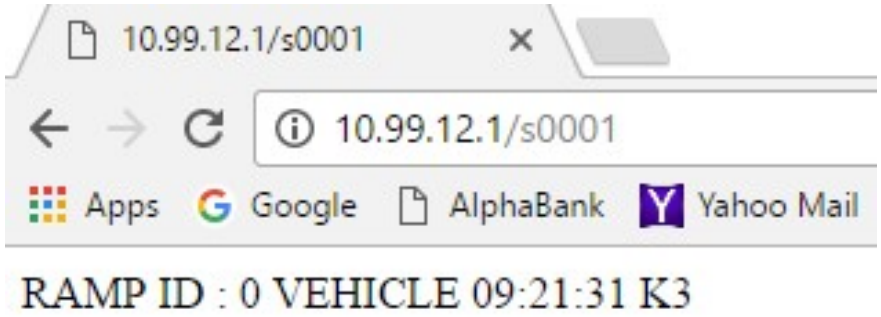
```
unsigned int SPI_Ethernet_UserTCP(unsigned char *remoteHost,
   unsigned int remotePort, unsigned int localPort, unsigned int
   reqLength, char *canClose)
{

   unsigned int len = 0;
   unsigned int i;
   if (localPort != 80) return (0);
   // get 10 first bytes only of the request, the rest does not
   metter here
   for (i = 0; i < 10; i++)
   getRequest[i] = SPI_Ethernet_getByte();
   getRequest[i] = 0;
   if (memcmp(getRequest, httpMethod, 5))  return (0);   // only
   GET method is supported here
```




```
if (getRequest[5] == 's'){  // primio komandu za prozivanje slejvova "s"
    // s 0x3? 0x3? 0x3? 0x3?
    // 0x3? predstavlja ASCII karakter, (?  -  0, 1, 2, ... , 9, A, B, C, D, E i F)
    if (((getRequest[6] & 0xF0) == 0x30) && ((getRequest[7] & 0xF0) == 0x30) &&
       ((getRequest[8] & 0xF0) == 0x30) && ((getRequest[9] & 0xF0) == 0x30))
    {
      ...
    }
}
```




```
if (getRequest[5] == 'r‘){
    // primio komandu za podesavanje RTC  "r"
    Flag2 = 0x01;   // podize FLAG 2 za postavljanje sata realnog vremena
    seconds = getRequest[6]; //nova vrednost za sekunde
    minutes = getRequest[7]; // nova vrednost za minute
    hours = getRequest[8];   // nova vrednost za sate
}
```




```
 if (len == 0)
   {
      FormirajNiz();
      len = putConstString(httpHeader); // HTTP header
      len += putConstString(httpMimeTypeHTML);
      len += putString(niz); // with HTML MIME type
}             //  if(len == 0)
   return (len); // return to the library with the number of
   bytes to transmit
}
```

```
unsigned int   putConstString(const char *s)   {
      unsigned int ctr = 0 ;
      while(*s) {
            SPI_Ethernet_putByte(*s++) ;
            ctr++ ;
       }
      return(ctr) ;
}

unsigned int    putString (char *s) {
      unsigned int ctr = 0 ;
      while(*s){
            SPI_Ethernet_putByte(*s++) ;
            ctr++ ;
      }
      return(ctr) ;
}
```