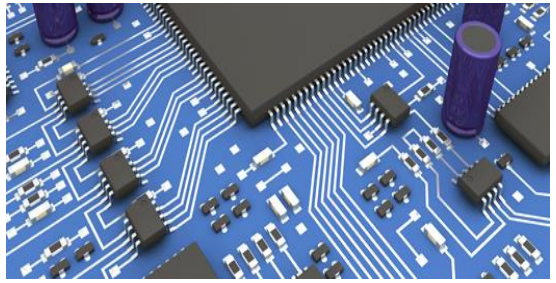
Aplikace Embedded systémů v Mechatronice



MPLAB® X IDE

Michal Bastl
A2/713a

Aplikace Embedded systémů v Mechatronice

Obsah přednášky:

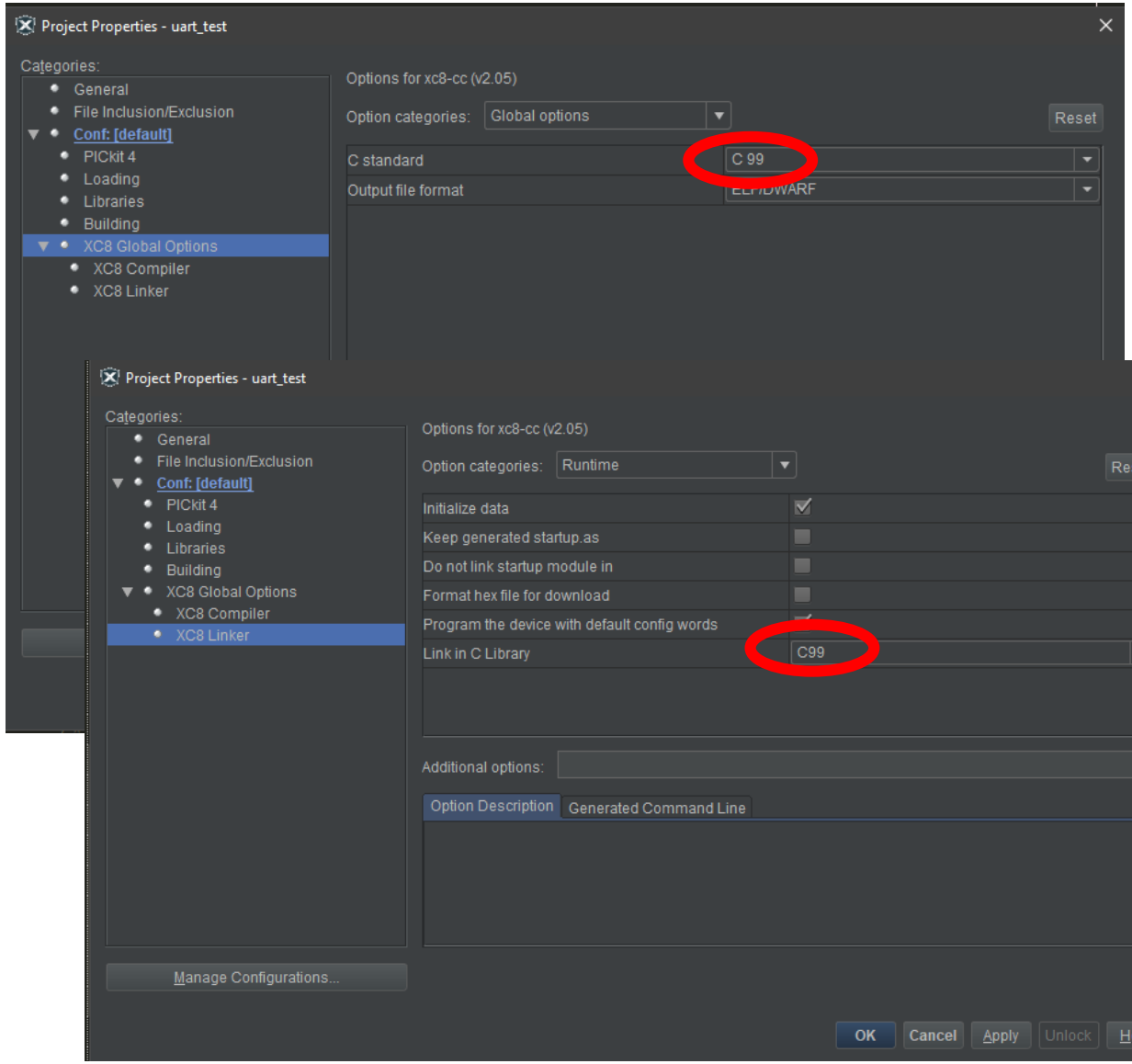
- Opakování
- Nový zápis interruptu XC8 2.02
- UART
- EEPROM
- LCD
- Ukázky použití
- Hardware poznámky
- Zadání DÚ

Opakování

K čemu je užitečná Timer periferie?
K čemu slouží přerušení?
Mohu v obsluze přerušení měnit proměnné?
Jak docílím přesné časování pomocí Timeru?
Kolik mohu mít v programu ISR (PIC18)?
Jak se provádí obsluha více zdrojů přerušení?

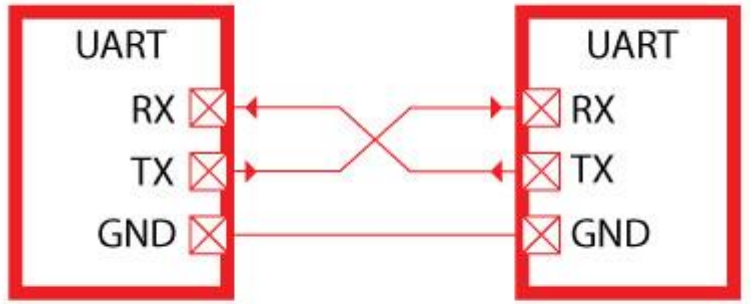
Nový zápis ISR v XC8

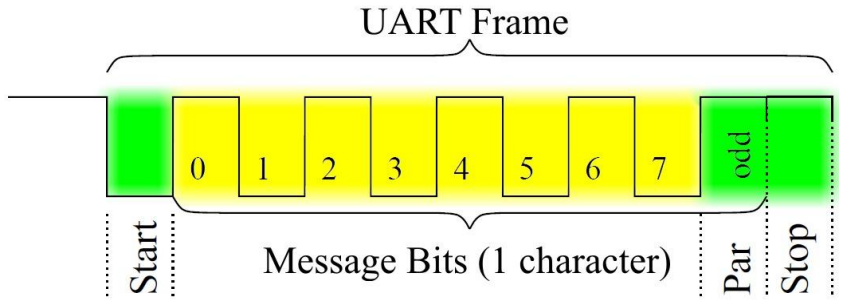
```
void __interrupt() ISR(void){
  if(TMR1IE & TMR1IF){
      TMR1 = 0x8000;
      LED1 ^= 1;
      TMR1IF = 0;
  }

  if(TMR5IE & TMR5IF){
      TMR5 = 0;
      LED2 ^= 1;
      TMR5IF = 0;
  }
}
```



UART

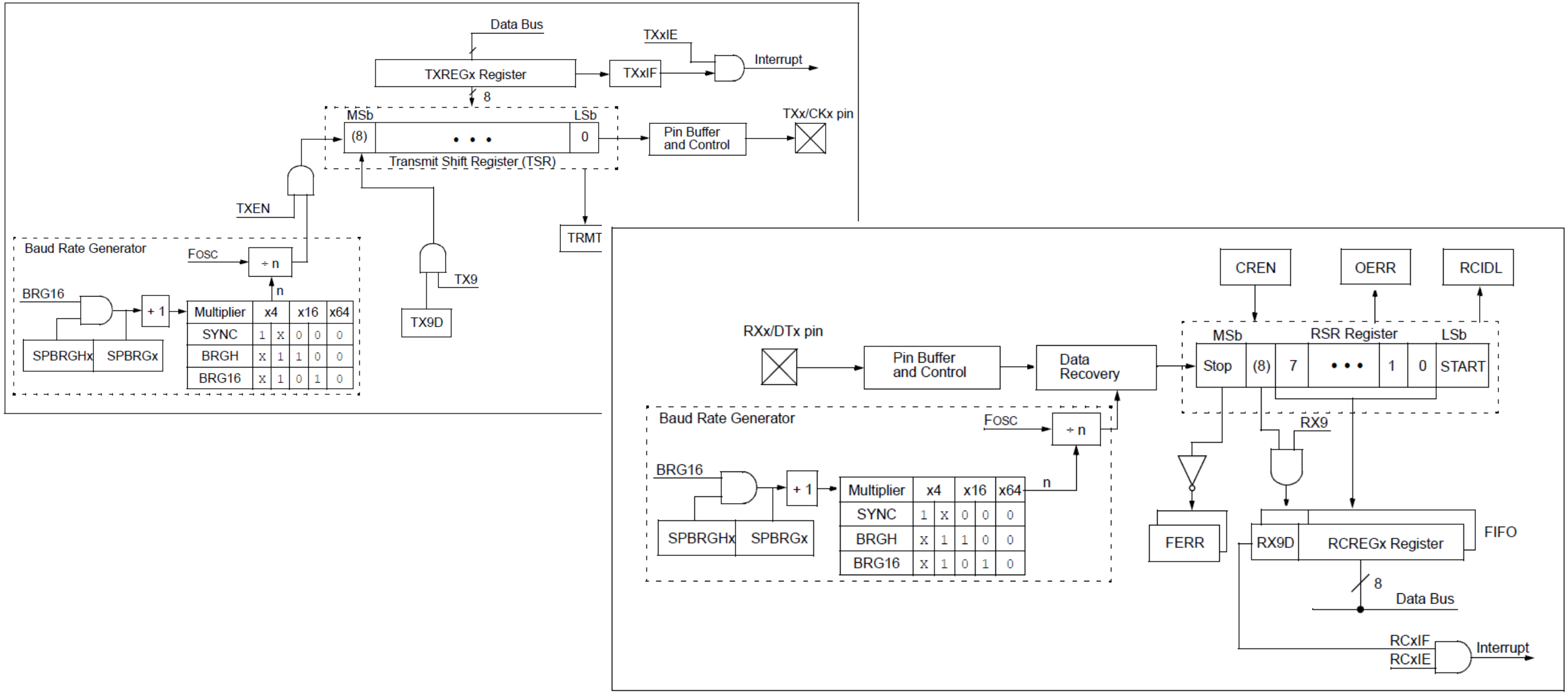
- UART *Universal Asynchronous Receiver and Transmitter*
- Jedná se o sériovou sběrnici
- Asynchronní znamená, že není sdílený CLOCK signál pro komunikující zařízení
- zařízení by však měli mít společnou GND viz obrázek
- režim FULL DUPLEX (vysílaní a příjem v jeden okamžik)
- Standardně 8 bit dat na zprávu





UART

FIGURE 16-1: EUSART TRANSMIT BLOCK DIAGRAM



Inicializace UART periferie

REGISTER 16-1: TxSTAx: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

```
TXSTAbits.TXEN = 1;      // enable TX
```

Inicializace UART periferie

REGISTER 16-2: RCSTAx: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

```
RCSTAbits.SPEN = 1;     // enable UART peripheral
RCSTAbits.CREN = 1;     // enable RX (aka Continuous receive)
```

Inicializace UART periferie

```
TRISCbits.TRISC6 = 0;   // uart TX as output
TRISCbits.TRISC7 = 1;   // uart RX as input

SPBRG = 25;     // (16_000_000 / (64 * 9600)) - 1

// final enable
RCSTAbits.SPEN = 1;     // enable UART peripheral
TXSTAbits.TXEN = 1;     // enable TX
RCSTAbits.CREN = 1;     // enable RX (aka Continuous receive)
```

Piny RX a TX je třeba v TRISx registru nastavit jako I/O

Uart je již „komplexnější“ periferie.
Na PIC18 může fungovat i v synchronním módu
Omezíme se na základní nastavení!!

EXAMPLE 16-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$Desired\ Baud\ Rate = \frac{FOSC}{64([SPBRGHx:SPBRGx] + 1)}$$

Solving for SPBRGHx:SPBRGx:

$$X = \frac{\frac{FOSC}{Desired\ Baud\ Rate}}{64} - 1$$

$$= \frac{\frac{16000000}{9600}}{64} - 1$$

$$= [25.042] = 25$$

$$Calculated\ Baud\ Rate = \frac{16000000}{64(25 + 1)}$$

$$= 9615$$

$$Error = \frac{Calc.\ Baud\ Rate - Desired\ Baud\ Rate}{Desired\ Baud\ Rate}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

UART použití

Zápis dat na sběrnici:

```
while(1){
    while (TMR1 < DELAY);      // busy waiting for TMR1

    LATD2 = ~LATD2;
    TMR1 = 0;

    TXREG = '0' + i;          // char '0' + index
    i = (i == 9 ? 0 : i + 1);    // cycle 0...9
  }  // end of main loop
```

čtení dat a zápis dat na sběrnici:

```
while(1){
    while(!PIR1bits.RCIF);  // waiting for data available flag
    LATB0 = ~LATB0;        // LED signal
    TXREG = RCREG;         // read byte and send it back
  }   // end of main loop
```

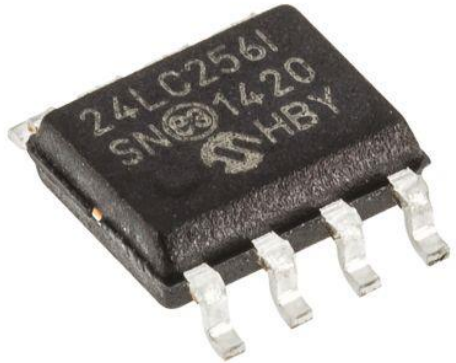
Pomocí přerušení:

```
RCIE = 1;          //RX interrupt enable
PEIE = 1;          // global interrupt enable
GIE = 1;           // peripheral interrupt enable

 void __interrupt() handle(void){

    // check both interrupt enable & interrupt flag
    if (PIE1bits.RC1IE && PIR1bits.RCIF){
       TXREG = RCREG; // Transmit back one character
       LATD2 = ~LATD2;
       PIR1bits.RCIF = 0;
    }

    return;
 }
```

EEPROM

Electrically Erasable Programmable Read-Only Memory

- Je typ paměti, která uchovává data i po odpojení napájení
- práce s EEPROM je pomocí registru procesoru je komplikovanější
- Kompilátor XC8 poskytneme vám funkce pro zápis/čtení
- eeprom procesoru PIC18 na EduKitu má velikost 1024 bajtu



EEPROM

Funkce pro zápis/čtení EEPROM:

```
void DATAEE_WriteByte(int bAdd, char bData)
{
   char GIEBitValue = INTCONbits.GIE;

   EEADRH = ((bAdd >> 8) & 0x03);
   EEADR = (bAdd & 0xFF);
   EEDATA = bData;
   EECON1bits.EEPGD = 0;
   EECON1bits.CFGS = 0;
   EECON1bits.WREN = 1;
   INTCONbits.GIE = 0;     // Disable interrupts
   EECON2 = 0x55;
   EECON2 = 0xAA;
   EECON1bits.WR = 1;
   // Wait for write to complete
   while (EECON1bits.WR)
   {
   }

   EECON1bits.WREN = 0;
   INTCONbits.GIE = GIEBitValue;   // restore interrupt enable
}
```

```
char DATAEE_ReadByte(int bAdd)
{
   EEADRH = ((bAdd >> 8) & 0x03);
   EEADR = (bAdd & 0xFF);
   EECON1bits.CFGS = 0;
   EECON1bits.EEPGD = 0;
   EECON1bits.RD = 1;
   NOP();  // NOPs may be required for latency at high frequencies
   NOP();

   return (EEDATA);
}
```

Programátor EEPROM maže!!!

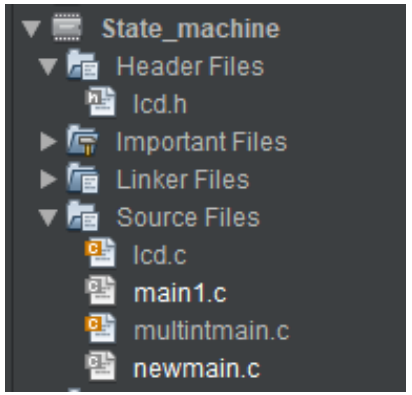
LCD

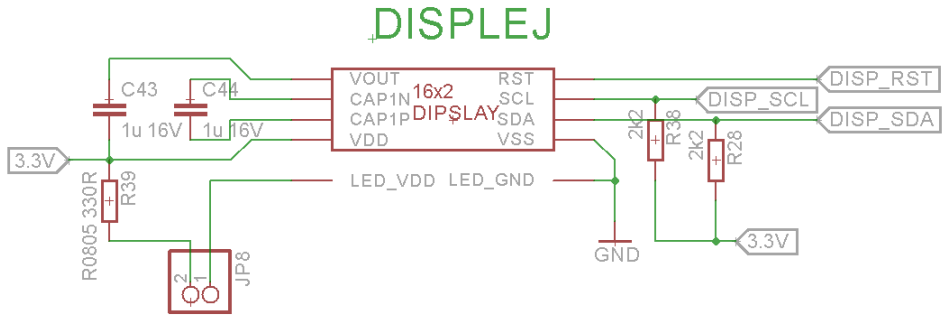
RAYSTAR OPTRONICS RX1602A3-BIW-TS:

řadič: ST7032
sběrnice: I2C

Použití:

```
#include <stdio.h>
#include "lcd.h„

 LCD_Init();
 char text[17];
 sprintf(text,"Mechlab je bozi!"); //funkce z stdio.h
 LCD_ShowString(1,text);

 if(BTN1){
      LCD_Clear();    //smazání
```

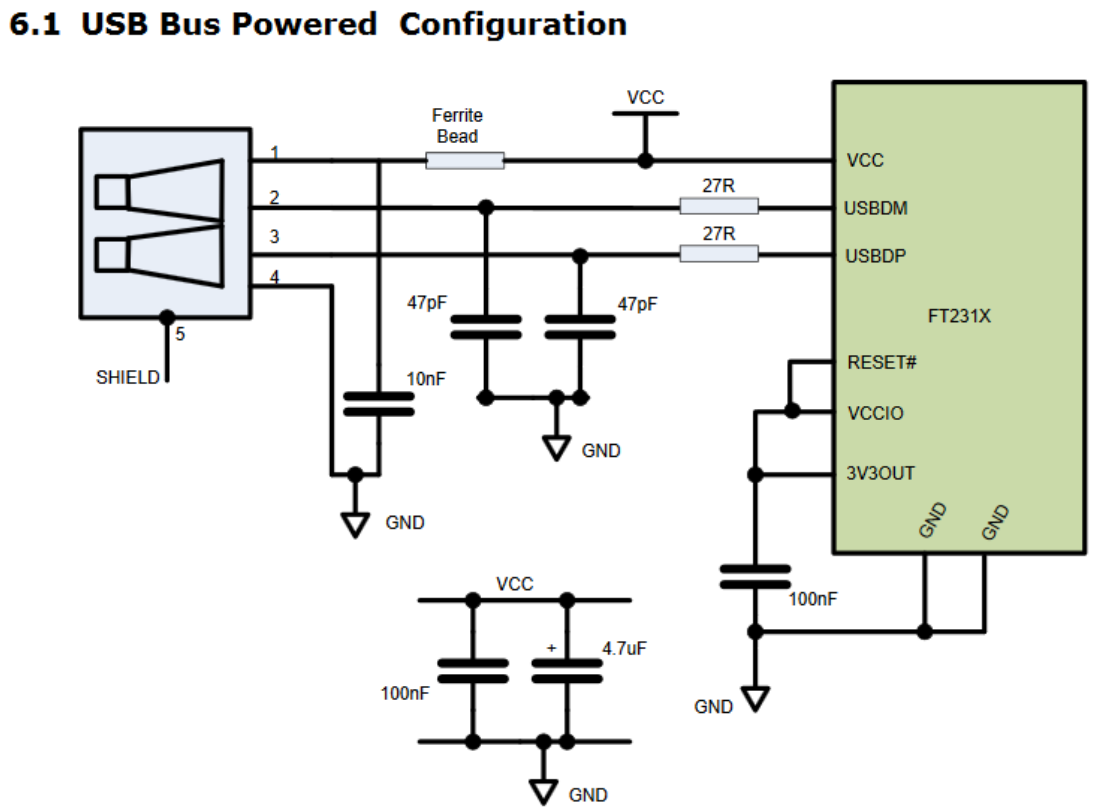




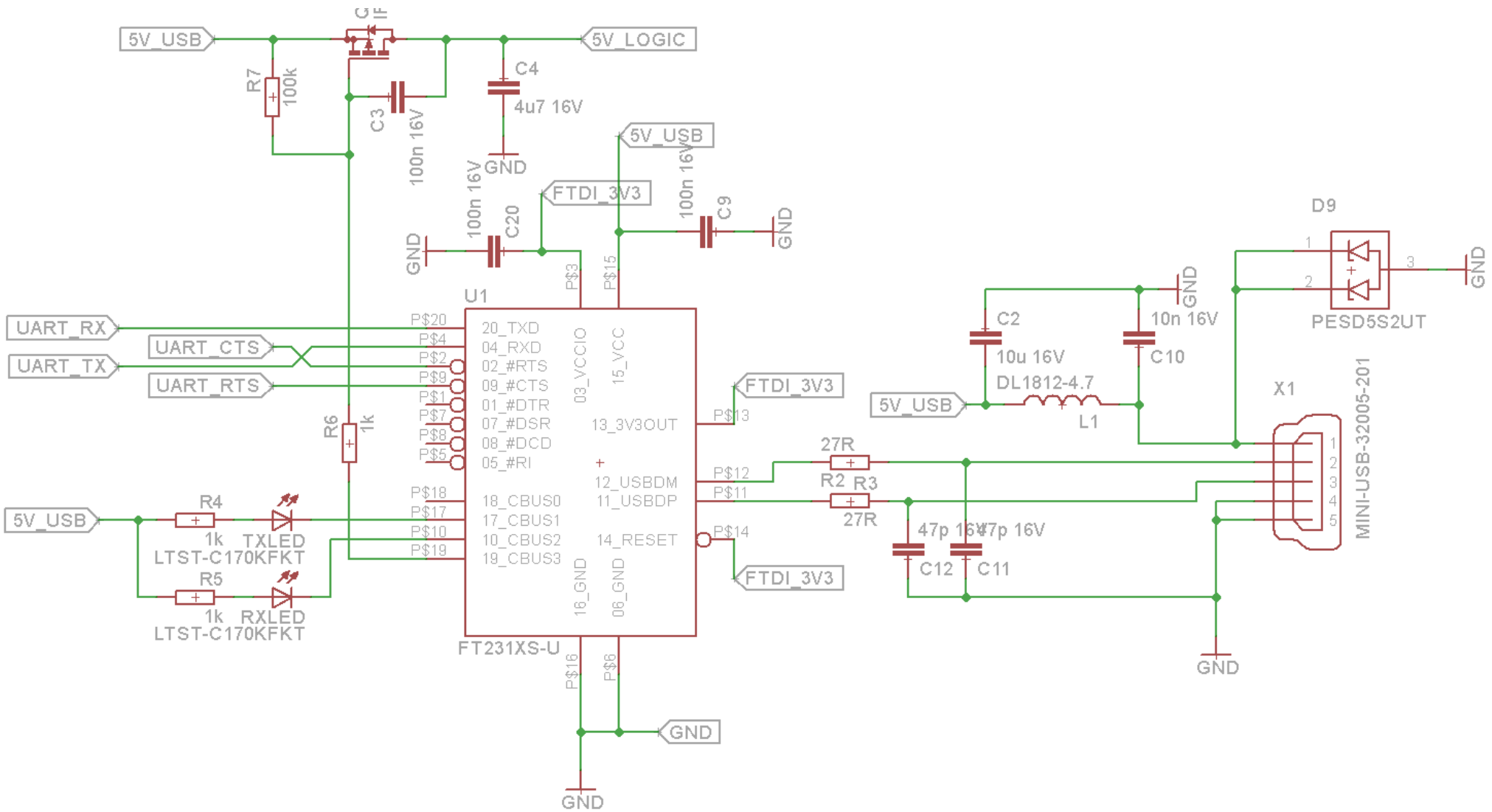
Hardware

Zákaznické obvody pro UART → USB:

- sběrnice uart lze pomocí specializovaných obvodů převést na usb
- na EduKitu je k tomuto účelu použit obvod FTDI 231
- tento obvod obsahuje vlastní mikrokontroler a lze jej pomocí obslužné aplikace nastavovat
- lze mu například přiřadit jméno, zvolit funkce pro vlastní GPIO apod



Hardware


5V_USB
5V_LOGIC
R7
100k
C3
100n 16V
C4
4u7 16V
GND
FTDI_3V3
C20
C9
U1
UART_RX
UART_CTS
UART_TX
UART_RTS
R6
1k
20_TXD
04_RXD
02_#RTS
09_#CTS
01_#DTR
07_#DSR
08_#DCD
05_#RI
03_VCCIO
15_VCC
13_3V3OUT
12_USBDM
11_USBDP
18_CBUS0
17_CBUS1
10_CBUS2
19_CBUS3
14_RESET
16_GND
06_GND
FT231XS-U
R4
R5
1k
TXLED
RXLED
LTST-C170KFKT
27R
R2
R3
C2
10u 16V
C10
10n 16V
DL1812-4.7
L1
C12
C11
47p 16V
D9
PESD5S2UT
X1
MINI-USB-32005-201

Zadání 2. DÚ

Zadání:

Vytvořte jednoduchý model digitálně řízeného signalizačního majáku:

maják má tři základní režimy: svítí trvale, bliká, nesvítí
maják má nastavitelnou rychlost blikání v rozsahu 1-20 Hz
maják je řízen prostřednictvím rozhraní UART pomocí sady příkazů, z nichž každý začíná písmenem A a končí tečkou '.'.

při odpojení a znovuzapojení si maják pamatuje poslední nastaveni (využijte EEPROM)….za 5 bodů, jinak 4

Maják je obsluhován pomocí této sady příkazů:

AC. (continuous) – nastaví režim kontinálního svícení
AB. (blink) – nastaví režim blikání
AON. – start svícení, dle nastaveného režimu
AOFF. – vypne svícení, blikání
ASnn. (sequence) – maják nn-krát zabliká a poté se vypne
AFnn. (frequency) – nastavení frekvence blikání
AF?. – vypíše nastavenou hodnotu frekvence
A?. – vypíše stav majáku (zapnuto/vypnuto + nastavený režim)

Pozn:
Přepnutí režimu v zapnutém stavu způsobí okamžitou změnu operace.
blikání emulujte na LEDkách na kitu