



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

# ŘPS – úloha MODBUS MA1S

*Ing. Josef Grosman*

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Tento materiál vznikl v rámci projektu ESF CZ.1.07/2.2.00/07.0247  
**Reflexe požadavků průmyslu na výuku v oblasti automatického řízení a měření,**  
který je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR

# Řídicí počítačové systémy

## Úloha pro samostatná cvičení - MA1S

Implementace protokolu MODBUS ASCII na PC a mikropočítačích řady 51 pro uzly Slave (Server) na PC, Master (Klient) na mikropočítači.

Požadované implementované funkce:

- zápis jediného vnitřního registru (Holding) do uzlu Slave,
- čtení bitového stavu (Coil) z uzlu Slave.

**Rozhraní: RS232, standardní rámec 7,N,2**

1. část: propojení PC – PC (C# MSVS)
2. část: propojení PC – mikropočítač

**Rozhraní: RS485, standardní rámec 7,N,2**

3. část: propojení mikropočítač – mikropočítač

Funkce pro podporu aplikace protokolu MODBUS:

- v souboru Modbus.dll a Modbus.cs pro PC (C#),
- v souboru Modbus.H a Modbus.C pro mikropočítač

## MASTER (klient)

V pravidelných časových intervalech generuje 16 bitovou hodnotu a předává požadavek na zápis do uzlu SLAVE

V pravidelných časových intervalech generuje požadavek na čtení bitové informace z uzlu SLAVE a zobrazuje ji

kód funkce: 06  
+ data

potvrzení

kód funkce: 01

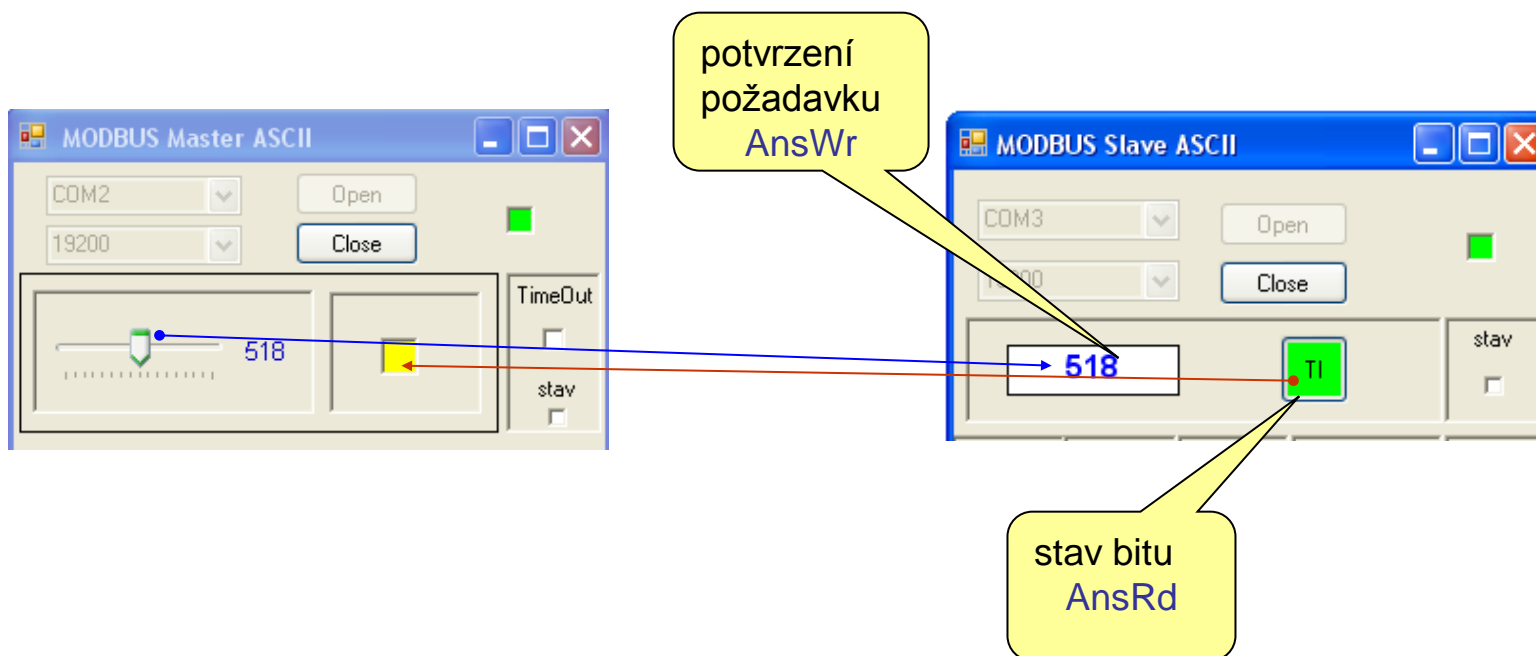
data

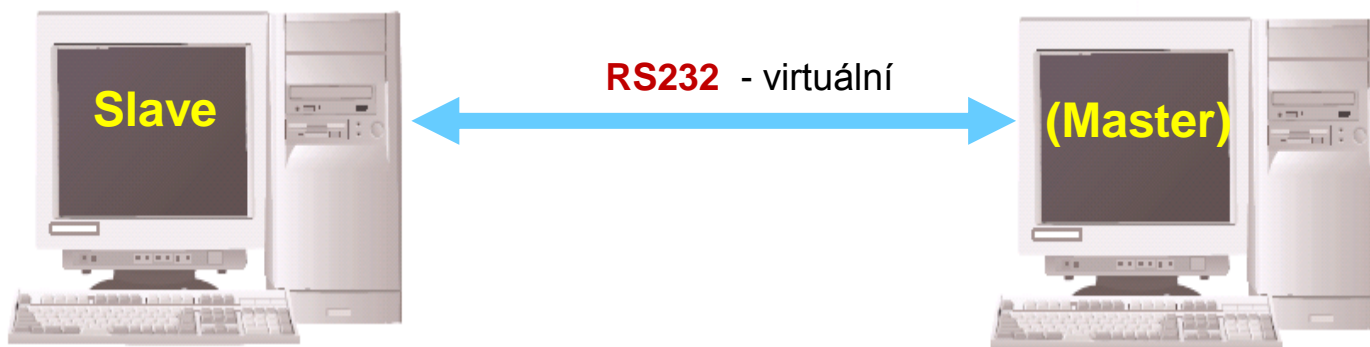
## SLAVE (server)

Po příjmu hodnotu zobrazí a odešle potvrzovací odpověď

Po příjmu požadavku hodnotu bitu odešle

## 1.část : PC-PC (varianta C#)





Podpora pro PC **Modbus.dll** ( zdrojový kód **Modbus.cs**)

C:\PRS\_podklady\modbus\sharp\  
N:\RPS\cviceni\_04\_modbus\sharp\

modbus.dll

Podpora pro testování ModbusMaster.exe a ModbusSlave.exe

C:\PRS\_podklady\modbus\sharp\exe\  
N:\RPS\cviceni\_04\_modbus\sharp\exe\

ModbusMaster.exe  
ModbusSlave.exe

## Zařazení Modbus.dll do aplikace

**1.** pravé tlačítko myši

**2.** Add Reference...

**3.**

**4.**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using Modbus;

namespace projektModbus
{

```

## Podpora pro PC    **Class lib**    **Modbus.dll**    - zdrojový kód **Modbus.cs**

```
namespace Modbus;
```

```
class ModbusASCII
```

```
byte AHex(byte b);  
byte HexAsc(byte b);
```

```
int WrByte(byte b,byte[] bf,int n);  
int WrWord(ushort w,byte[] bf,int n) ;  
int WrEOT(byte[] bf,int n)::;
```

```
int WrOne(byte adr,byte fce,ushort reg,ushort val,byte[] bf);  
int Rd(byte adr,byte fce,ushort reg,ushort val,byte[] bf);
```

```
byte RdByte(byte[] bf,int n);  
ushort RdWord(byte[] bf,int n);
```

```
int AnsRd(byte adr,byte fce,byte bytes,byte[] vals,byte[] bf);  
int Answr(byte adr,byte fce,ushort reg,ushort val,byte[] bf);  
int AnsErr(byte adr,byte fce,byte er,byte[] bf);
```

```
byte Lrc(byte[] bf,int len):byte;
```

Užité metody třídy ModbusASCII v aplikaci z Modbus.dll	
aplikační	pomocné
WrOne	RdByte
Rd	WrByte
AnsWr	Lrc
AnsRd	WrEoT
AnsErr	RdWord
Poznámka: v hlavním programu v sekci <b>using</b> přidat <a href="#">Modbus</a>	

Definované a doporučené hodnoty		
význam	symbol	hodnota
Adresa uzlu Slave	ADR_S	1
Funkce zápis registru	FCE_WREG	6
Funkce čtení bitu	FCE_RBIT	1
Adresa zapisovaného registru	REG_WR	0
Adresa čteného bitu	BIT_RD	0



**:, adr. slavu, kód funkce, adr. registrů a bitů, hodnoty, LRC, CRLF**



**bfout**

RS232

**bfin**

RS232

```
byte []bfin = new byte[512];  
byte []bfout = new byte[512];
```

bfout[0] :  
bfout[1],bfout[2] **adresa slavu**  
bfout[3],bfout[4] **kód funkce**

.

## Slave – implementace na PC (server)

### Konfigurace:

Přijímá požadavky od klienta a vrací odpovědi

- požadavek na zápis jediného vnitřního registru (hodnota 0 až 1023) – funkční kód 6, hodnotu zobrazí a vrací potvrzení o přijetí požadavku

**metoda AnsWr třídy Modbus ASCII s kódem přijaté funkce**

- požadavek na čtení bitové hodnoty – funkční kód 1 a vrací stav požadovaného bitu

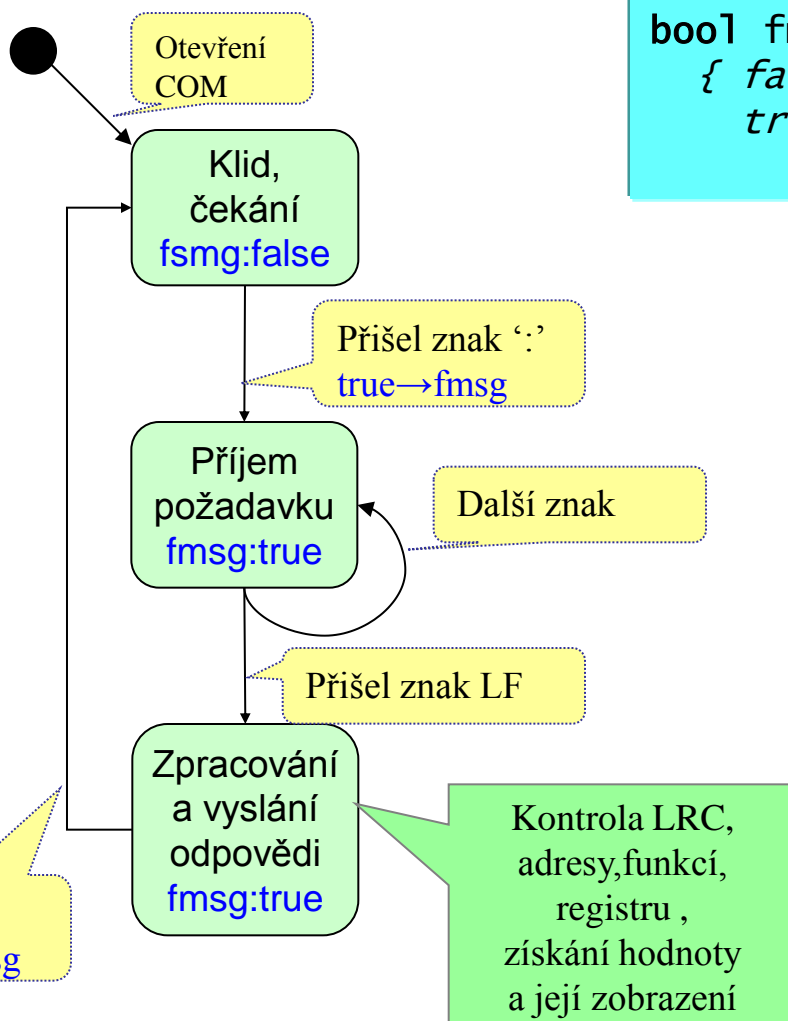
**metoda AnsRd třídy Modbus ASCII s kódem přijaté funkce**

Kontrolovat přijatý požadavek a vracet chybovou odpověď v případě neimplementované funkce, neexistující adresy registru nebo bitu a hodnoty mimo rozsah

**metoda AnsErr třídy Modbus ASCII s upraveným kódem funkce a typem chyby**

Skupinové vysílání ignorovat .

## Slave – zjednodušený stavový diagram



```

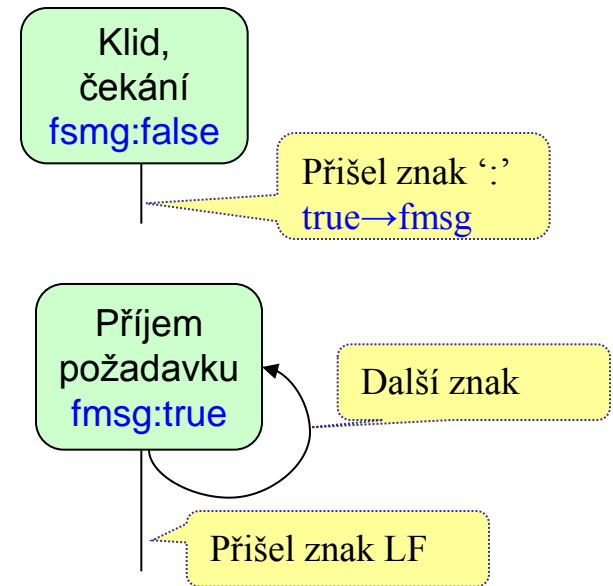
bool fsmg;
{ false – čekání na požadavek
  true – příjem, zpracování požadavku
    a vyslání odpovědi }
  
```

## Slave – příjem požadavku

DataReceived

```
while(comPort.BytesToRead > 0)
{
    byte b=(byte)comPort.ReadByte();
    if(b==(byte)':')
    {
        ix=0;
        fmsg=true;
    }
    else if(fmsg) ix++;
    bfin[ix]=b;
}
```

```
if(b==(byte)'\n' && fmsg)
{
    .
    .
    fmsg=false;
}
```



## Slave – zpracování požadavku, kontrola položek, příprava odpovědi

### 1. LRC

```
if(Ma.Lrc(bfin,ix-4)!=Ma.RdByte(bfin,ix-3)
{
    .. možná informace o chybné LRC
}
else {
```

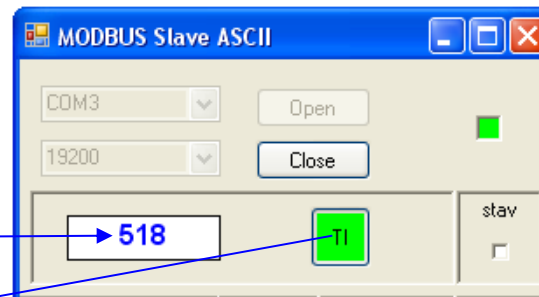
ModbusASCII Ma;

### 2. adresa

```
adr_r=Ma.RdByte(bfin,1);
if(adr_r == ADR_S)
{
```

### 3. kód funkce

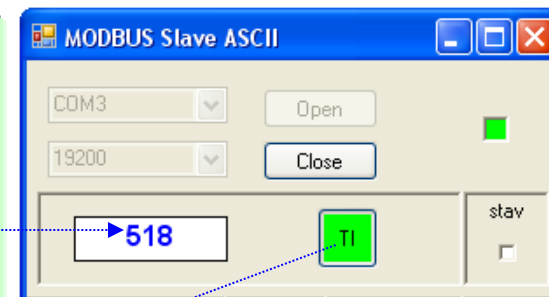
```
kod_r=Ma.RdByte(bfin,3);
er=0;
switch(kod_r) {
    case FCE_WREG:
        :
    case FCE_RBIT:
        :
    default er=1;
}
```



## Slave – zpracování požadavku, kontrola položek, příprava odpovědi

### FCE\_WREG:

```
reg=Ma.RdWord(bfin,5);
val=Ma.RdWord(bfin,9);
if(reg!=REG_WR) er=2;
else if(val>1023) er=3;
else .. zobrazení hodnoty
if(er==0) n= Ma.Answr(ADR_S,kod_r,reg,val,bfout);
```



### FCE\_RBIT:

```
reg=Ma.RdWord(bfin,5);
pocet=Ma.RdWord(bfin,9);
if(reg!=BIT_RD || pocet!=1) er=2;
else {
    tlačitko -> vals
    n= Ma.Answr(ADR_S,kod_r,1,vals,bfout);
}
```

byte []vals = byte[1];

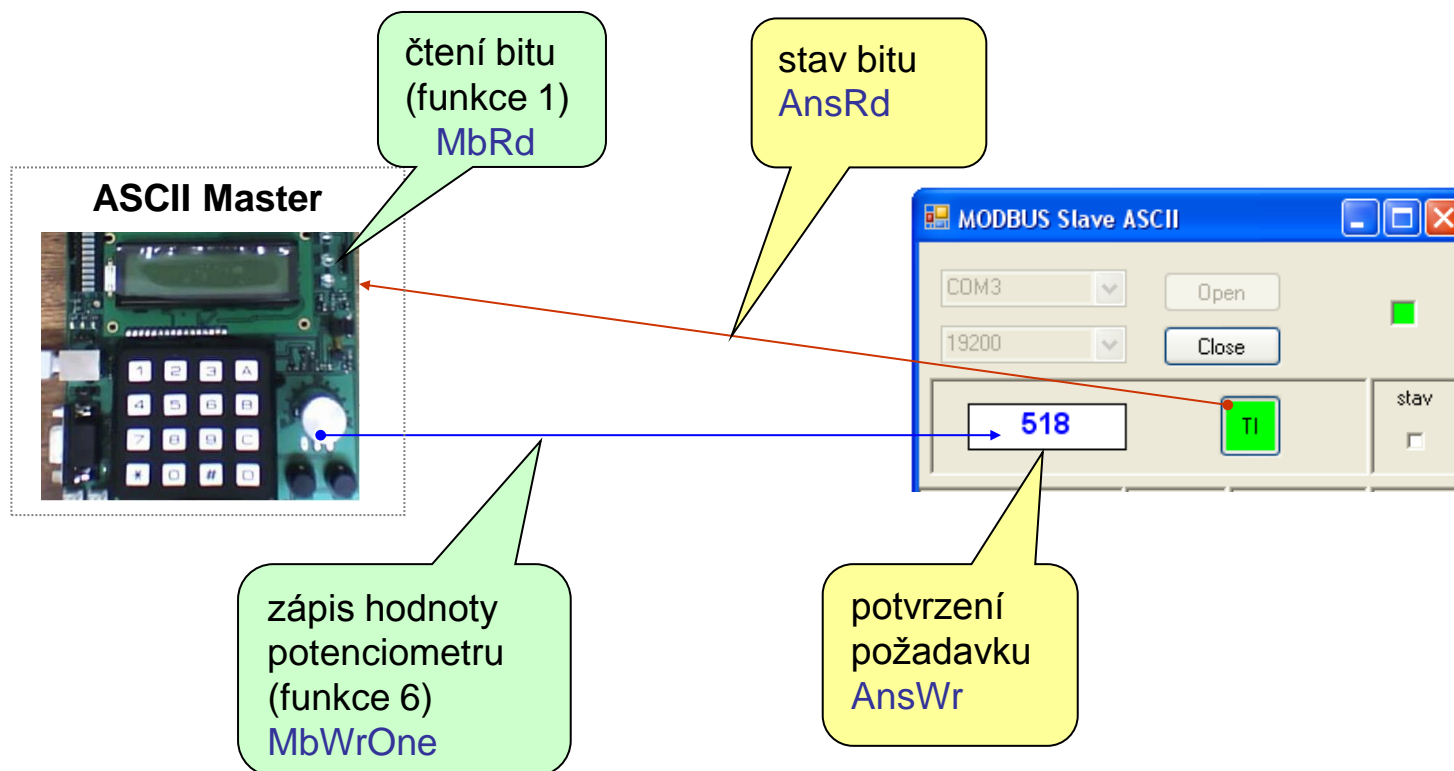
### 4. chyba

```
if(er>0) n=Ma.Answr(adr_r,(byte)(kod_r|0x80),er,bfout);
```

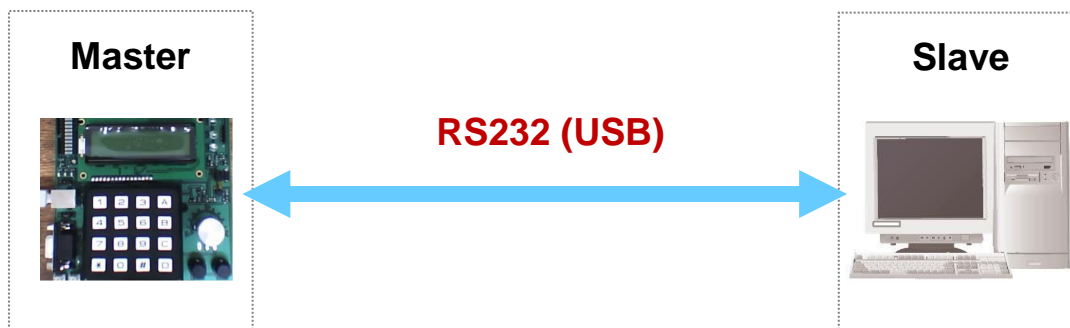
### 5.odeslání odpovědi

```
n=Ma.WrByte(Ma.Lrc(bfout,n-1),bfout,n);
n=Ma.WrEOT(bfout,n);
comPort.Write(bfout, 0, n);
```

## 2.část : PC-mikropočítač



## pro mikropočítač (2. část)



Podpora pro mikropočítač **Modbus.c, Modbus.h**

C:\RPS\_podklady\modbus\C\  
N:\RPS\cviceni\_04\_modbus\C\

MODBUS.C  
MODBUS.H  
MAIN.C  
ADC.C  
LCD.C  
LEDBAR.C  
TYPY.H



## Podpora pro mikropočítač      prototypy funkcí **Modbus.H** – zdrojový kód **Modbus.C**

```
byte AHex(byte c);  
byte HexAsc(byte b);  
  
byte WrWord(word val,byte *bf);  
word RdWord(byte *bf);  
byte MbRdByte(byte *bf);  
word MbRdWord(byte *bf);  
byte MbWrByte(byte b,byte *bf);  
byte MbWrWord(word w,byte *bf);  
  
byte MbRd(byte adr,byte fce,word reg,word val,byte *bf);  
byte MbWrOne(byte adr,byte fce,word reg,word val,byte *bf);  
byte MbWr(byte adr,byte fce,word reg,word nbr,byte *vals,byte *bf);  
  
byte MbAnsWr(byte adr,byte fce,word reg,word val,byte *bf);  
byte MbAnsRd(byte adr, byte fce, byte bytes, byte *vals,byte *bf);  
byte MbAnsErr(byte adr,byte fce,byte er,byte *bf);  
  
byte MbLrc(byte *bf,byte len);  
byte MbWrEoT(byte *bf);
```

Užité funkce v aplikaci ze souboru <b>Modbus.C</b>	
aplikační	pomocné
MbWrOne	MbRdByte
MbRd	MbWrByte
MbAnsWr	MbLrc
MbAnsRd	MbWrEoT
MbAnsErr	MbRdWord
Poznámka: v hlavním programu <code>#include "Modbus.H"</code>	

Definované a doporučené hodnoty		
význam	symbol	hodnota
Adresa uzlu Slave	ADR_S	1
Funkce zápis registru	FCE_WREG	6
Funkce čtení bitu	FCE_RBIT	1
Adresa zapisovaného registru	REG_WR	0
Adresa čteného bitu	BIT_RD	0

**:, adr. slavu, kód funkce, adr. registrů a bitů, hodnoty, LRC, CRLF**



**bfout**

RS232

**bfin**

RS232

```
xbyte bfin[256],bfout[256];
```

```
bfout[0]      :  
bfout[1],bfout[2]  adresa slavu  
bfout[3],bfout[4]  kód funkce  
.  
.
```

funkce pro vyslání zprávy:  
- bf: pointer na pole znaků  
- len: počet bytů k vyslání

```
void SendBuf(byte *bf,byte len)  
{  
    while(len--)  
    {  
        SBUF=*bf++ | 0x80;  
        while(!TI);  
        TI=0;  
    }  
}
```

## Master – implementace na mikropočítači (klient)

### Konfigurace:

Realizuje funkce (požadavky na server)

- požadavek na zápis jediného vnitřního registru (hodnota 0 až 1023) – funkční kód 6  
aplikační funkce MbWrOne s kódem funkce 6 (FCE\_WREG)
- požadavek na čtení bitové hodnoty – funkční kód 1  
aplikační funkce MbRd s kódem funkce 1 (FCE\_RBIT)

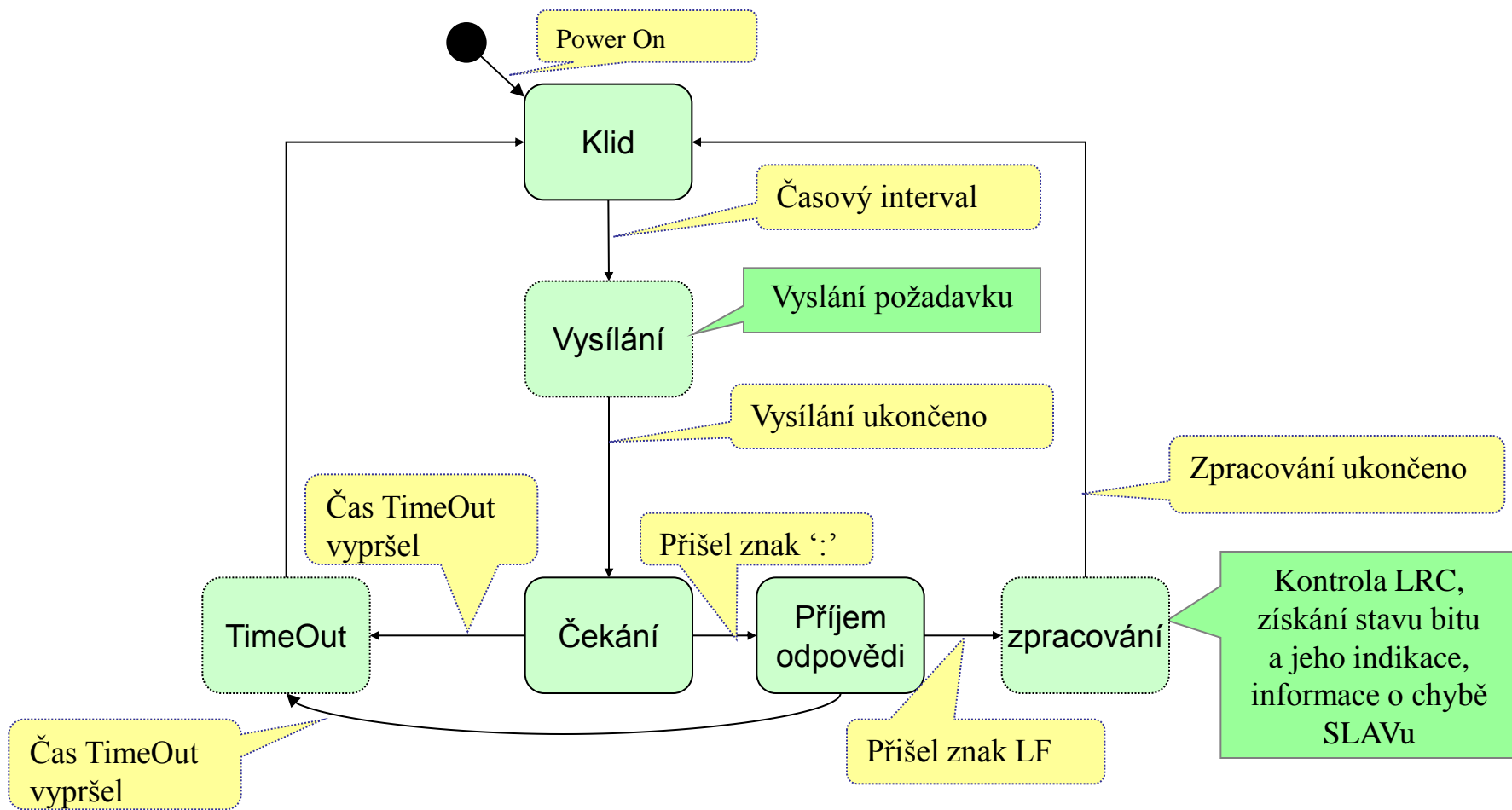
Požadavky odesílat střídavě v pravidelných časových intervalech cca 200 ms,  
jen když je sériový kanál otevřen a Master je ve stavu **klidu**  
realizace časovačem T0 se základními tiky 30 ms ( $30 * 7 = 210$ )

Implementovat generování čekacího Timeout intervalu 500 ms na odpověď od Slave  
( $30 * 17 = 510$ )

Zjednodušený příjem odpovědi

příchozí adresu Slave není nutno testovat, pouze správnost LRC  
zpracovat jen odpověď na požadavek čtení bitu (FCE\_RBIT)  
informace o chybě Slave: jen omezeně, nebo vůbec

## Master – zjednodušený stavový diagram



```
enum {stKlid,stCekani,stPrijem} stav;
```

## Master – vyslání požadavku

střídavě každých cca 210 ms vysílá rámec s funcí **1** (čtení bitu) a **6** (zápis registru)



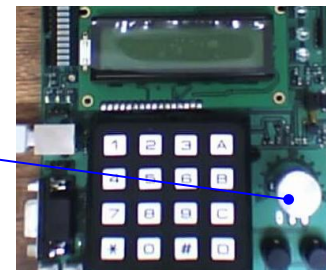
časovač T0

```
if(++cnt_ticks>=N_TICKS && stav==stKlid){
    cnt_ticks=0;
    DIR485=1;      /* na vysílání – pro RS485*/
    prep=!prep;
    if(prepare){ val= ... ;
        itx=MbWrOne(ADR_S,FCE_WREG,REG_WR,val,bfout);}
    else itx=MbRd(ADR_S,FCE_RBIT,BIT_RD,1,bfout);
    itx+=MbWrByte(MbLrc(bfout+1,itx-1),bfout+itx);
    itx+=MbWrEoT(bfout+itx);
    SendBuf(bfout,itx);
    stav=stCekani;
    DIR485=0;      /* zpět na příjem – pro RS485 */
}
```

```
#define N_TICKS 7
#define TIMEOUT 17
bit prep;
```

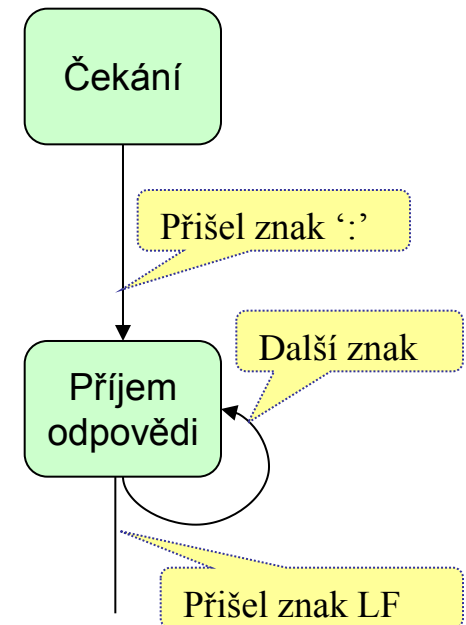
TimeOut

```
if(cnt_ticks>=TIMEOUT){
    cnt_ticks=0;
    LED_R=!LED_R; // signalizace TimeOutu
    stav=stKlid;
}
```



## Master – příjem odpovědi

```
if (RI)
{
    byteIn=SBUF&0x7F;
    RI=0;
    if(stav==stCekani && byteIn==':')
    {
        stav=stPrijem;
        bfin[ix=0]=byteIn;
    }
    else if(stav==stPrijem)
    {
        if(byteIn==':')ix=0;
        else ix++;
        bfin[ix]=byteIn;
        if(byteIn=='\n')
        {
            . // zpracování odpovědi
            .
            stav=stKlid;
        }
    }
}
```



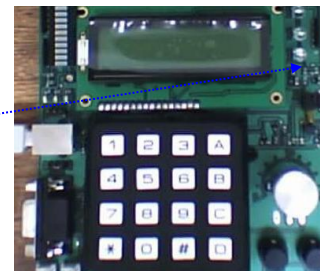
## Master – zpracování odpovědi

### 1. LRC

```
if(MbLrc(bfin+1,ix-4)==(lrc=MbRdByte(bfin+ix-3)))  
{
```

### 2. kód funkce a reakce na odpověď

```
if( (kod_r=MbRdByte(bfin+3))==FCE_RBIT)  
{  
    pocet=MbRdByte(bfin+5);  
    val=MbRdByte(bfin+7);  
    if(val & 1) .. ; // LED svítí  
    else .. ; // LED nesvítí  
}
```





## 3.část : mikropočítač – mikropočítač

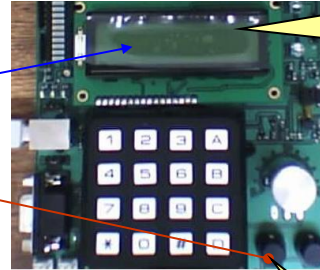
ASCII Master



čtení bitu  
(funkce 1)  
MbRd

zápis hodnoty  
potenciometru  
(funkce 6)  
MbWrOne

ASCII Slave



potvrzení  
požadavku  
MbAnsWr

stav tlačítka  
MbAnsRd

## **Pro 3.část : mikropočítač – mikropočítač**

**je nezbytné**

- 1. správně nastavit propojky pro modul UART  
bud' přenos konektorem USB  
nebo přenos konektory RS232/485**
- 2. správně přepínat budič RS485  
pro příjem  
nebo pro vysílání**

**Propojky volby  
pro modul UART**

**USB x RS**

Pro nahrávání  
programu : **USB**  
Aplikace : **RS**

**Propojky volby RS**

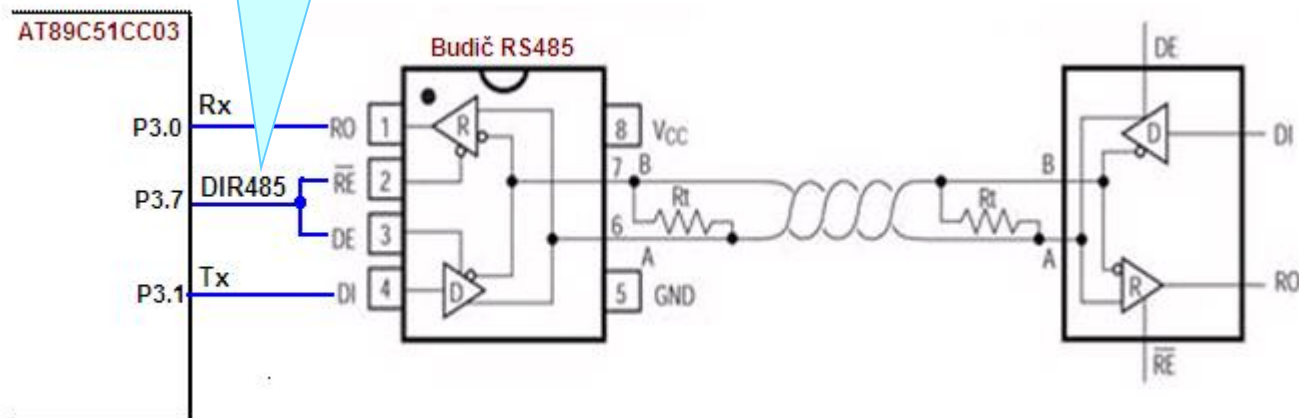
**RS232 x RS485**

Aplikace : **RS85**

**RS485 konektory**



#define DIR485 P3\_7



DIR485	směr
0	Rx (příjem)
1	Tx (vysílání)

1. Nastavit na příjem (0)
2. Před vysláním zprávy nastavit na vysílání (1) a po vyslání zprávy zpět na příjem (0)