



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

ŘPS – úloha MODBUS MR4S

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 - MR4S

Implementace protokolu MODBUS RTU 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,
- zápis jediného bitového stavu (Coil) do uzlu Slave.

Rozhraní: RS232, standardní rámec 8,N,2

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

Rozhraní: RS485, standardní rámec 8,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 1 bitovou informaci a předává požadavek na zápis do uzlu SLAVE

kód funkce: 06
+ data

potvrzení

kód funkce: 05
+ data

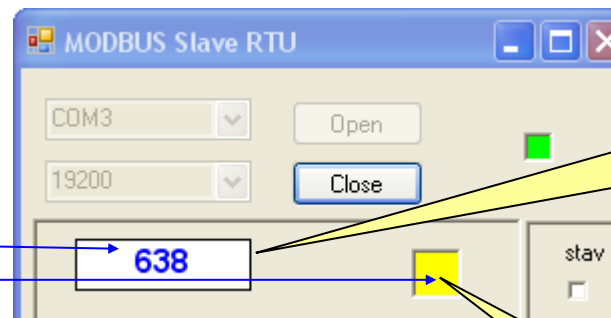
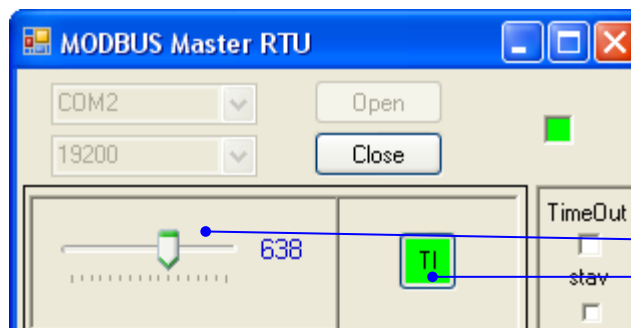
potvrzení

SLAVE (server)

Po příjmu hodnotu zobrazí a odešle potvrzovací odpověď

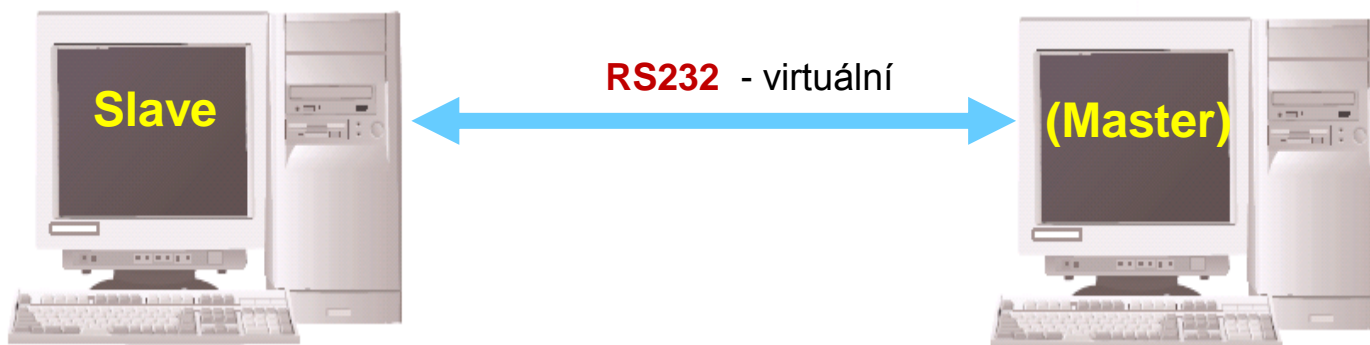
Po příjmu informaci zobrazí a odešle potvrzovací odpověď

1.část : PC-PC (varianta C#)



potvrzení
požadavku
AnsWr

potvrzení
požadavku
AnsWr



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. **Add Reference**
 .NET COM Projects Browse Recent
 Oblast hledání: sharp
 exe
 Modbus.dll
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 ModbusRTU
```

```
// metody pro Modbus RTU
```

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

```
ushort Crc(byte[] bf,int len);  
int WrCrc(ushort crc,byte[] bf,int n);  
ushort RdCrc(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 nbr,byte[] bf);  
int Wr(byte adr,byte fce,ushort reg,int nbr,byte[] vals,byte[] bf);
```

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

Užité metody třídy ModbusRTU v aplikaci z Modbus.dll	
aplikační	pomocné
WrOne	RdWord
AnsWr	WrWord
AnsErr	Crc
	WrCrc
	RdCrc
Poznámka: v hlavním programu v sekci using přidat Modbus	

Definované a doporučené hodnoty		
význam	symbol	hodnota
Adresa uzlu Slave	ADR_S	1
Funkce zápis registru	FCE_WREG	6
Funkce zápis bitu	FCE_WBIT	5
Adresa zapisovaného registru	REG_WR	0
Adresa zapisovaného bitu	BIT_WR	0

adr. slavu, kód funkce, adr. registrů a bitů, hodnoty, CRC



bfout

→ RS232

bfin

← RS232

```
byte []bfin = new byte[256];  
byte []bfout = new byte[256];
```

bfout[0] **adresa slavu**

bfout[1] **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 ModbusRTU s kódem přijaté funkce

- požadavek na zápis jediného bitového stavu – funkční kód 5, stav indikuje a vrací potvrzení o přijetí požadavku

metoda AnsWr třídy ModbusRTU s kódem přijaté funkce

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

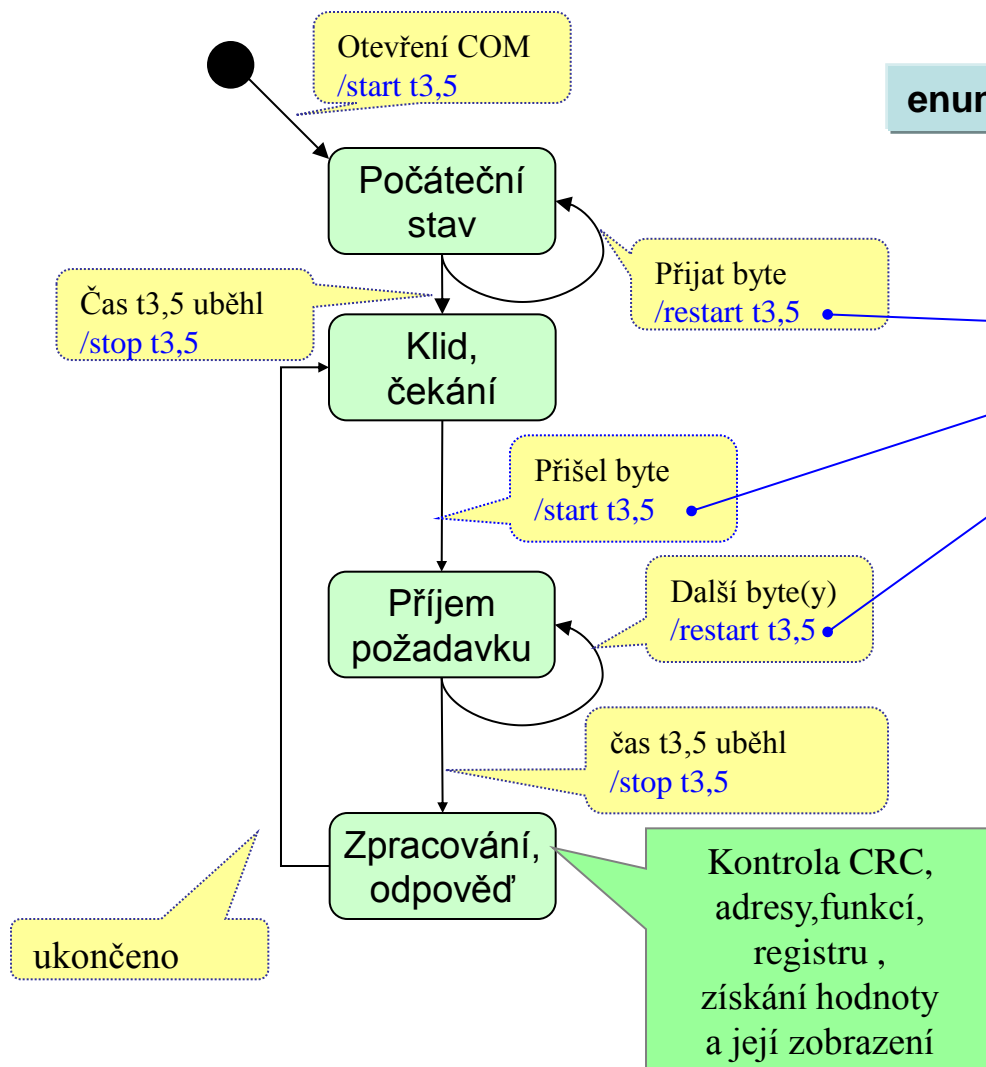
metoda AnsErr třídy ModbusRTU s upraveným kódem funkce a typem chyby

Skupinové vysílání ignorovat .

Implementace generování intervalu 3,5 znaku pro ukončení příjmu zprávy

Omezená (žádná) implementace generování intervalu 1,5 znaku

Slave – zjednodušený stavový diagram



```
enum Tstav{stPocatek,stKlid,stVysilani,stPrijem};
```

Po posledním byte ve frontě

Upozornění: použít delegáta
(*BeginInvoke(metoda...)*)

v samostatné metodě

```
timer3_5.Enabled=true;
```

Slave– počáteční stav

Click
(Open)

```
stav = Tstav.stPocatek;  
Timer3_5.Enabled=true;
```

DataReceived

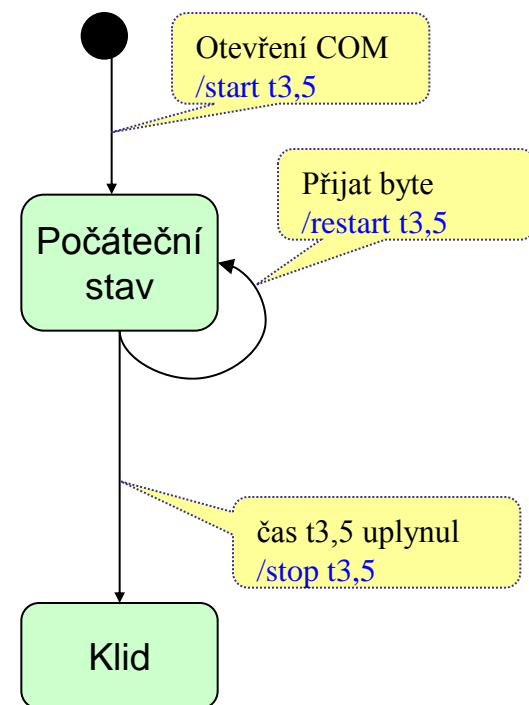
```
while(comPort.BytesToRead > 0){  
    byte b = (byte)comPort.ReadByte();  
    switch(stav){  
        case Tstav.stPocatek:break;  
        case Tstav.stKlid:  
            .  
        case Tstav.stPrijem:  
            .  
    }  
}  
BeginInvoke(metoda...);
```

timer3_5.Enabled=true;

Tick
3_5

```
Timer3_5.Enabled=false;  
switch(stav){  
    case Tstav.stPocatek: stav=Tstav.stKlid;  
                        break;  
    case Tstav.stPrijem:  
        . // zpracování odpovědi
```

Tstav stav;



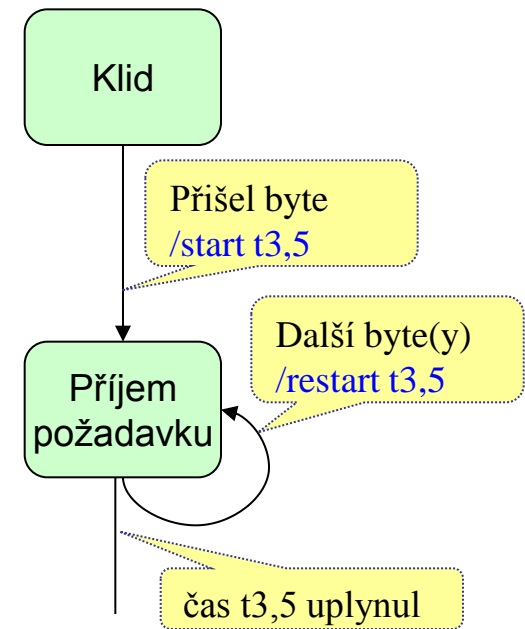
Slave – příjem požadavku

DataReceived

```
Tstav.stKlid:  
    stav=Tstav.stPrijem;  
    bfin[0]=b;  
    ix=0;  
    break;  
Tstav.stPrijem:  
    bfin[++ix]=b;  
    break;
```

Tick
3_5

```
case Tstav.stPrijem:  
    . // zpracování požadavku
```



Slave – zpracování požadavku, kontrola položek, příprava odpovědi

1. CRC

```
if(Mr.Crc(bfin,ix-1)!=Mr.RdCrc(bfin,ix-1)
{
    .. možná informace o chybné CRC
}
else {
```

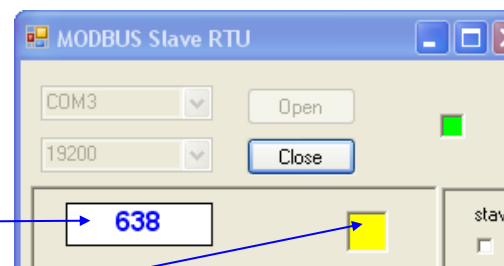
ModbusRTU Mr;

2. adresa

```
adr_r=bfin[0];
if(adr_r == ADR_S)
{
```

3. kód funkce

```
kod_r=bfin[1];
er=0;
switch(kod_r) {
    case FCE_WREG:
        .
        .
    case FCE_WBIT:
        .
        .
    default: er=1;
}
```



Slave – zpracování požadavku, kontrola položek, příprava odpovědi

FCE_WREG:

```
reg=Mr.RdWord(bfin,2);
val=Mr.RdWord(bfin,4);
if(reg!=REG_WR) er=2;
else if(val>1023) er=3;
else .. zobrazení hodnoty
if(er==0) n= Mr.Answr(ADR_S,kod_r,reg,val,bfout);
```



FCE_WBIT:

```
reg=Mr.RdWord(bfin,2);
val=Mr.RdWord(bfin,4);
if (reg!=BIT_WR) er=2;
else switch (val) {
    case 0xFF00: .. žlutá
    case 0x0000: .. bílá
    default: er=3;
}
if(er==0) n= Mr.Answr(ADR_S,kod_r,reg,val,bfout);
```

4. chyba

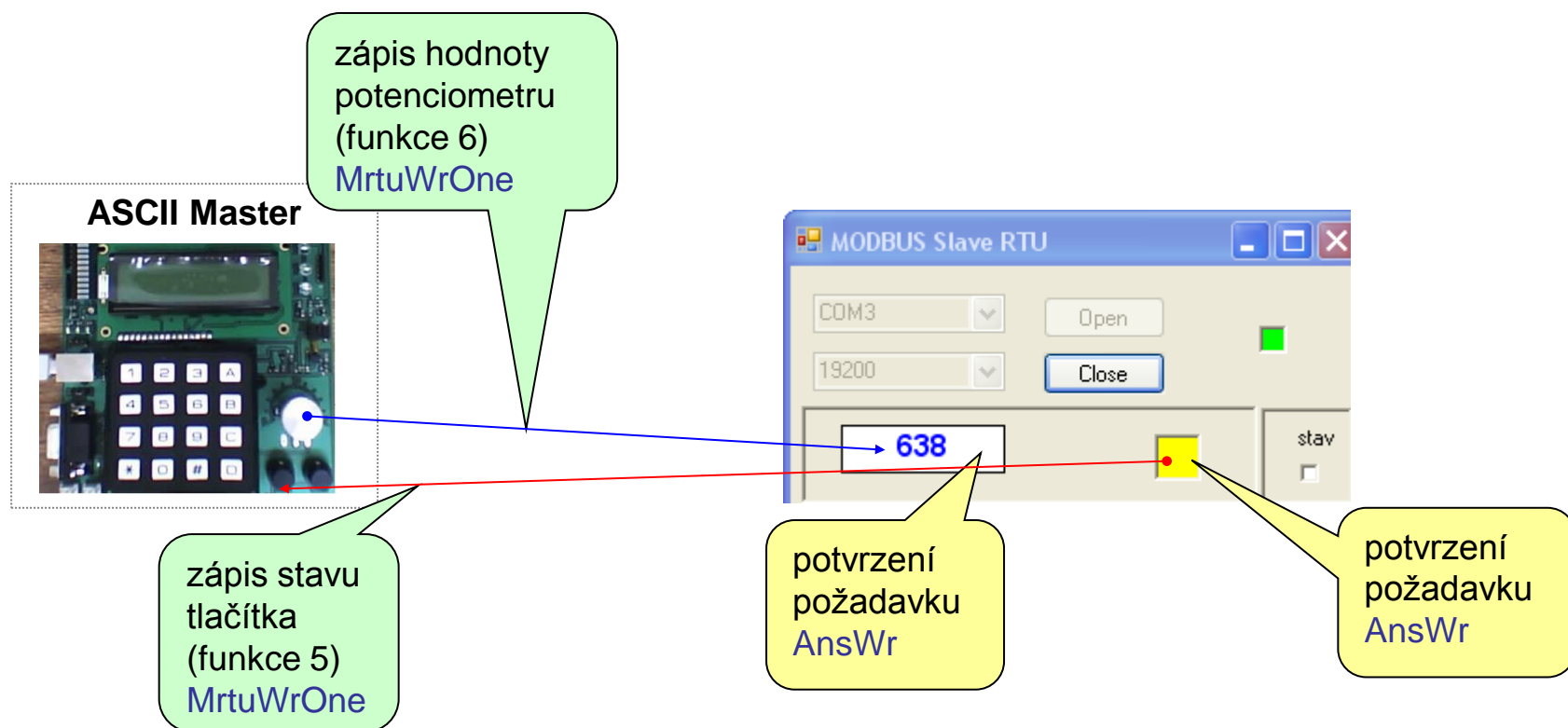
```
if(er>0) n=Mr.Answr(adr_r,(byte)(kod_r|0x80),er,bfout);
```

odeslání
odpovědi

```
n=Mr.WrCrc(Mr.Crc(bfout,n),bfout,n);
comPort.Write(bfout, 0, n);
```

```
stav=Tstav.stklid;
```

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 WrWord(word val,byte *bf);  
word RdWord(byte *bf);  
word MrtuRdCrc(byte *bf);  
byte MrtuWrCrc(word crc,byte *bf );
```

```
byte MrtuWr(byte adr,byte fce,word reg,word nbr,byte *vals,byte *bf);  
byte MrtuWrOne(byte adr,byte fce,word reg,word val,byte *bf);  
byte MrtuRd(byte adr,byte fce,word reg,word val,byte *bf);
```

```
byte MrtuAnsErr(byte adr,byte fce,byte er,byte *bf);  
byte MrtuAnsRd(byte adr,byte fce,byte reg,byte *vals,byte *bf);  
byte MrtuAnsWr(byte adr,byte fce,word reg,word val,byte *bf);
```

```
word MrtuCrc(byte *bf, byte len);
```

Užité funkce v aplikaci ze souboru Modbus.C	
aplikační	pomocné
MrtuWrOne	RdWord
MrtuAnsWr	WrWord
MrtuAnsErr	MrtuCrc
	MrtuWrCrc
	MrtuRdCrc
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 zápis bitu	FCE_WBIT	5
Adresa zapisovaného registru	REG_WR	0
Adresa zapisovaného bitu	BIT_WR	0

adr. slavu, kód funkce, adr. registrů a bitů, hodnoty, CRC



bfout

→ RS232

bfin

← RS232

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

bfout[0] **adresa slavu**

bfout[1] **kód funkce**

.

.

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

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

Časový interval 3,5 znaku – generování časovačem T1 v režimu 1

Formát UART: 8,N,2 \rightarrow 11 bitů, $f_{\text{bit}} = 19200 \text{ bit/s} \rightarrow t_{\text{bit}} = 1/f_{\text{bit}}$

$$t_{3,5} = 3,5 \cdot 11 \cdot t_{\text{bit}} \approx 2 \text{ ms tik}$$

pro časovač T1: $t_{3,5} = N3_5 \cdot 12 / f_{\text{osc}}$

pro sériový kanál řízený časovačem T2 je $t_{\text{bit}} = 32 \cdot \text{NBIT} / f_{\text{osc}}$

$$t_{3,5} = 3,5 \cdot 11 \cdot 32 \cdot \text{NBIT} / f_{\text{osc}} = N3_5 \cdot 12 / f_{\text{osc}}$$

$$N3_5 = \text{NBIT} \cdot 109 \rightarrow \text{\#define N3_5 } 109 * \text{NBIT}$$

(re)start t3,5

```
TH1=(word)(-N3_5) >> 8;  
TL1=(byte)(-N3_5);  
TF1=0;  
TR1=1;
```

čas uplynul: přerušení nebo
1 \rightarrow TF1

Poznámka: oba časovače T0 a T1 budou nastaveny v režimu 1: TMOD = 0x11;

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 MrtuWrOne s kódem funkce 6 (FCE_WREG)
- požadavek na zápis jediného bitového stavu – funkční kód 5
aplikační funkce MrtuWrOne s kódem funkce 5 (FCE_WBIT)

Požadavky odesílat střídavě v pravidelných časových intervalech cca 200 ms,
jen když je Master ve stavu **klidu**
realizace časovačem T0

Implementovat generování čekacího TimeOut intervalu 500 ms na odpověď od Slave

Zjednodušený příjem odpovědi

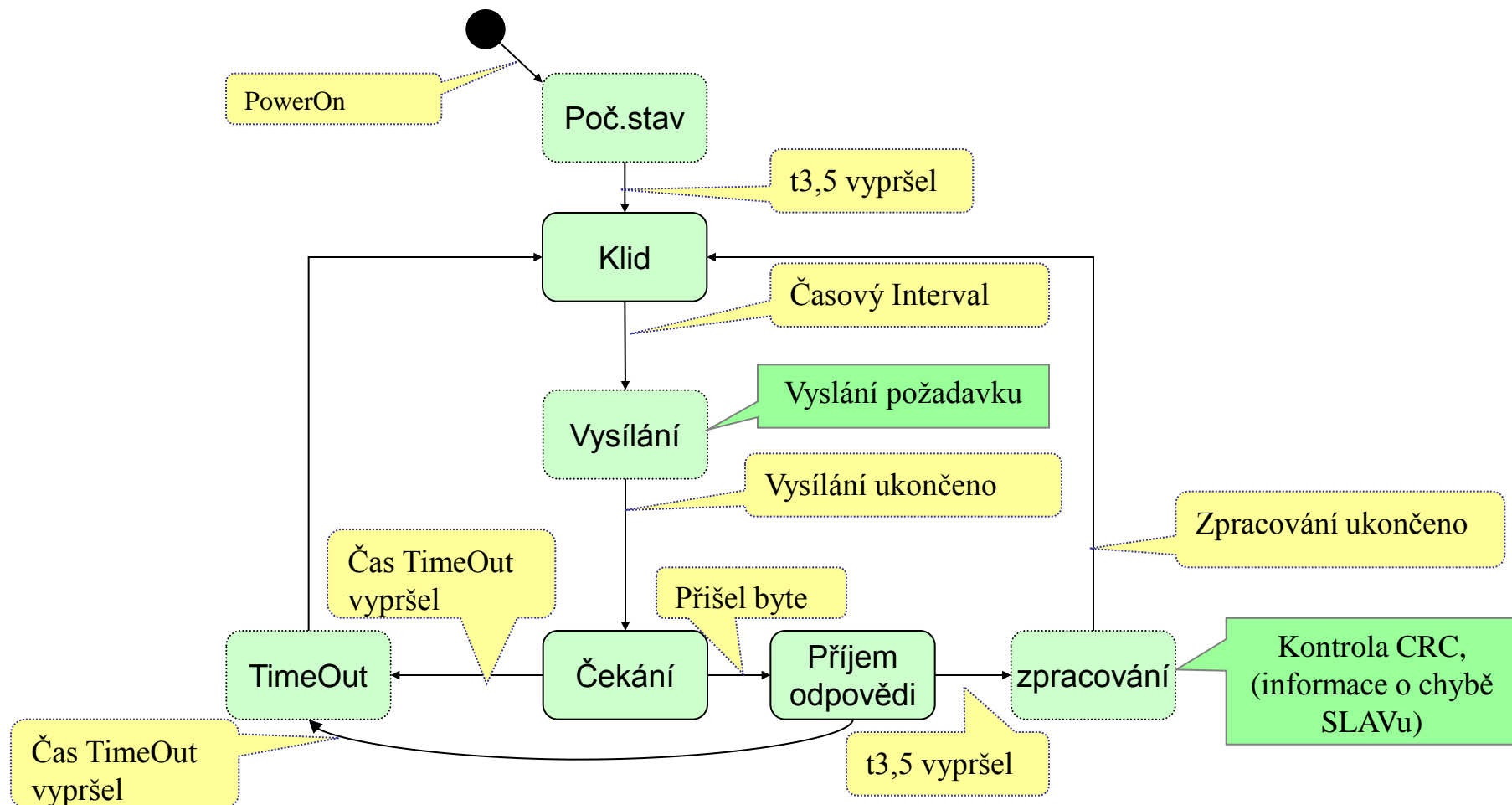
příchozí adresu Slave není nutno testovat, pouze správnost CRC

zpracovat jen informace o chybě Slave: jen omezeně (žlutá LED), nebo vůbec

Implementace generování intervalu 3,5 znaku pro ukončení příjmu zprávy časovačem T1

Omezená (žádná) implementace generování intervalu 1,5 znaku

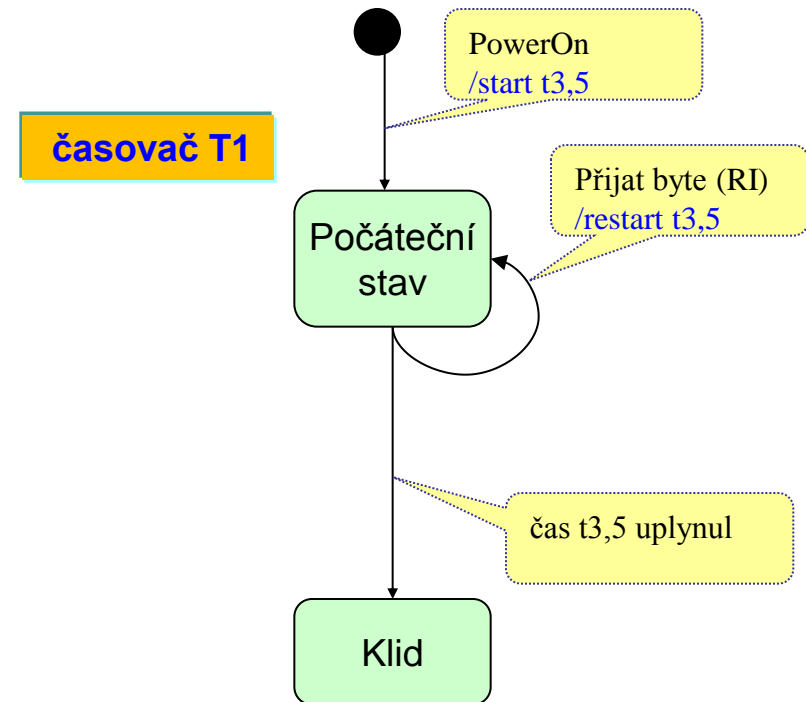
Master – zjednodušený stavový diagram



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

Master – počáteční stav

```
void main(void)
{
    .. // inicializace
    do
    {
        RI=0;
        TH1=(word)(-N3_5) >> 8;
        TL1=(byte)(-N3_5) ;
        TR1=1;
        while(!TF1);
        TF1=0;
        TR1=0;
    } while(RI);
    stav=stKlid;
    while(1)
    {
        ..
    }
}
```



Master – vyslání požadavku

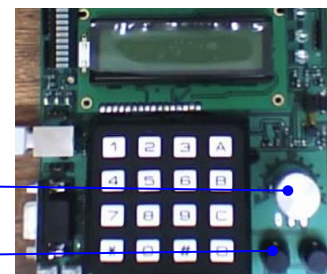
střídavě každých cca 200 ms vysílá rámec s funcí **5** (zápis bitu) a **6** (zápis registru)

5 6 5 6 5 6

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

časovač T0

```
if(++cnt_ticks>=N_TICKS && stav==stKlid){
    cnt_ticks=0;
    DIR485=1; /* na vysílání */
    prep=!prep;
    if(prepare) {val= ...;
        itx=MrtuWrOne(ADR_S,FCE_WREG,REG_WR,val,bfout);}
    else { val= ...;
        itx=MrtuWrOne(ADR_S,FCE_WBIT,BIT_WR,val,bfout);}
    itx+=MrtuWrCrc(MrtuCrc(bfout,itx),bfout+itx);
    SendBuf(bfout,itx);
    stav=stCekani;
    DIR485=0; /* zpět na příjem */
}
```



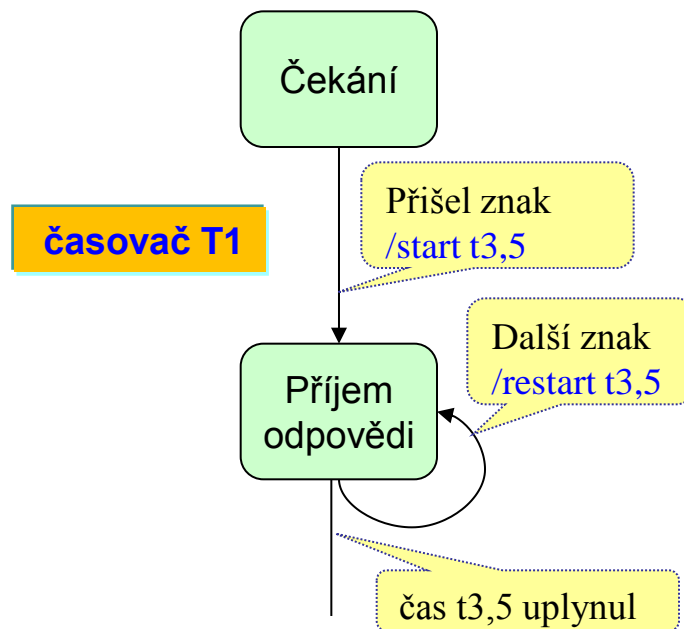
TimeOut

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

Master – příjem odpovědi

```
if (RI)
{
    byteIn=SBUF;
    RI=0;
    switch(stav)
    {
        case stCekani:
            ix=0;
            bfin[ix++]=byteIn;
            stav=stPrijem;
            TH1=(word)(-N3_5) >> 8;
            TL1=(byte)(-N3_5) ;
            TF1=0;
            TR1=1;
            break;
        case stPrijem:
            bfin[ix++]=byteIn;
            TH1=(word)(-N3_5) >> 8;
            TL1=(byte)(-N3_5) ;
            TF1=0;
            break;
    }
}
```

```
if(TF1){
```



Master – zpracování odpovědi

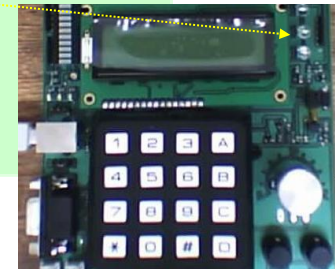
časovač T1

CRC

kód funkce

chybná odpověď

```
if(TF1)
{
    TR1=0;
    if(stav==stPrijem)
    {
        if(MrtuCrc(bfin,ix-2)==MrtuRdCrc(bfin+ix-2))
        {
            if ((kod_r=bfin[1]) >= 0x80)
                LED_Y=0;
            else LED_Y=1;
        }
        stav=stKlid;
    }
}
```



3.část : mikropočítač – mikropočítač

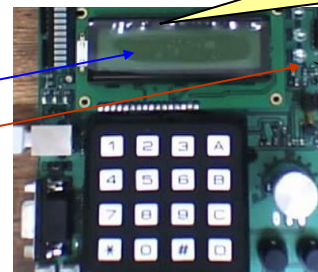
RTU Master



zápis hodnoty
potenciometru
(funkce 6)
MrtuWrOne

zápis stavu
tlačítka
(funkce 5)
MrtuWrOne

RTU Slave



potvrzení
požadavku
MrtuAnsWr

potvrzení
požadavku
MrtuAnsWr

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

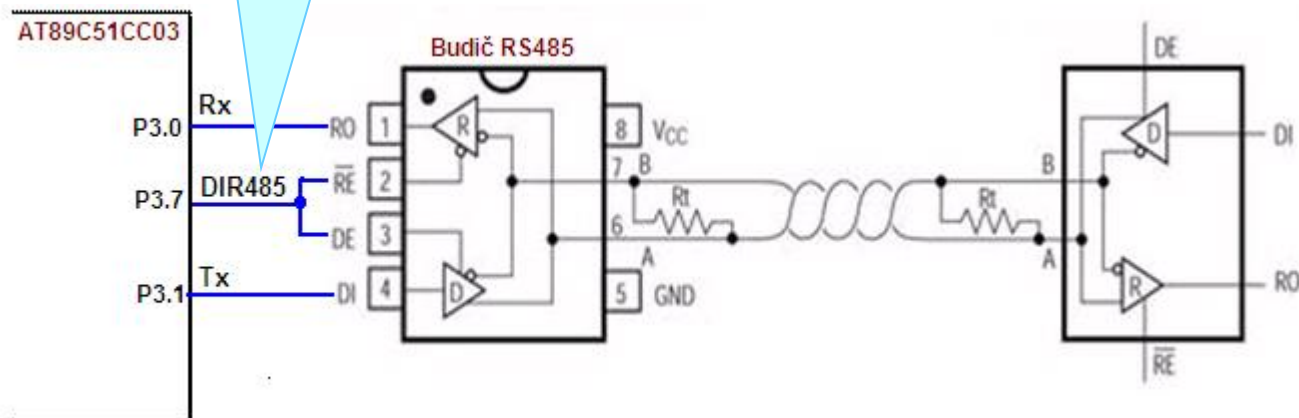
RS232 x RS485

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)