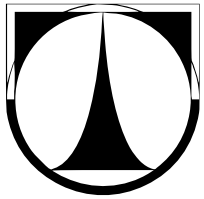


INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

ŘPS – úloha MODBUS MR5M

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 - MR5M

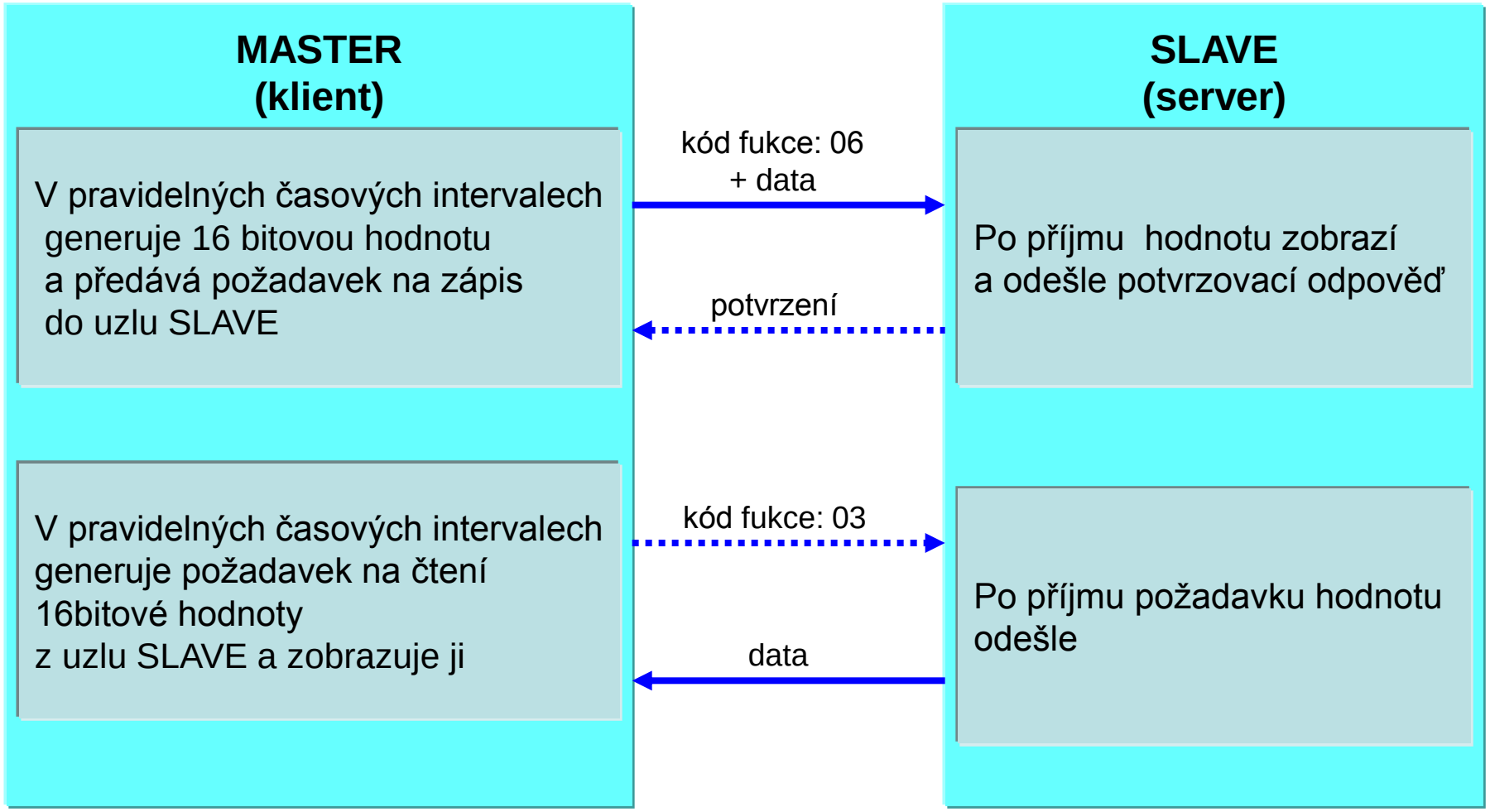
Implementace protokolu MODBUS RTUI na PC a mikropočítačích řady ´51 pro uzly Master (Klient) na PC, Slave (Server) na mikropočítači

Požadované implementované funkce:
- čtení 16bitového vnitřního registru (Holding) z uzlu Slave,
- zápis jediného vnitřního registru (Holding) do uzlu Slave,

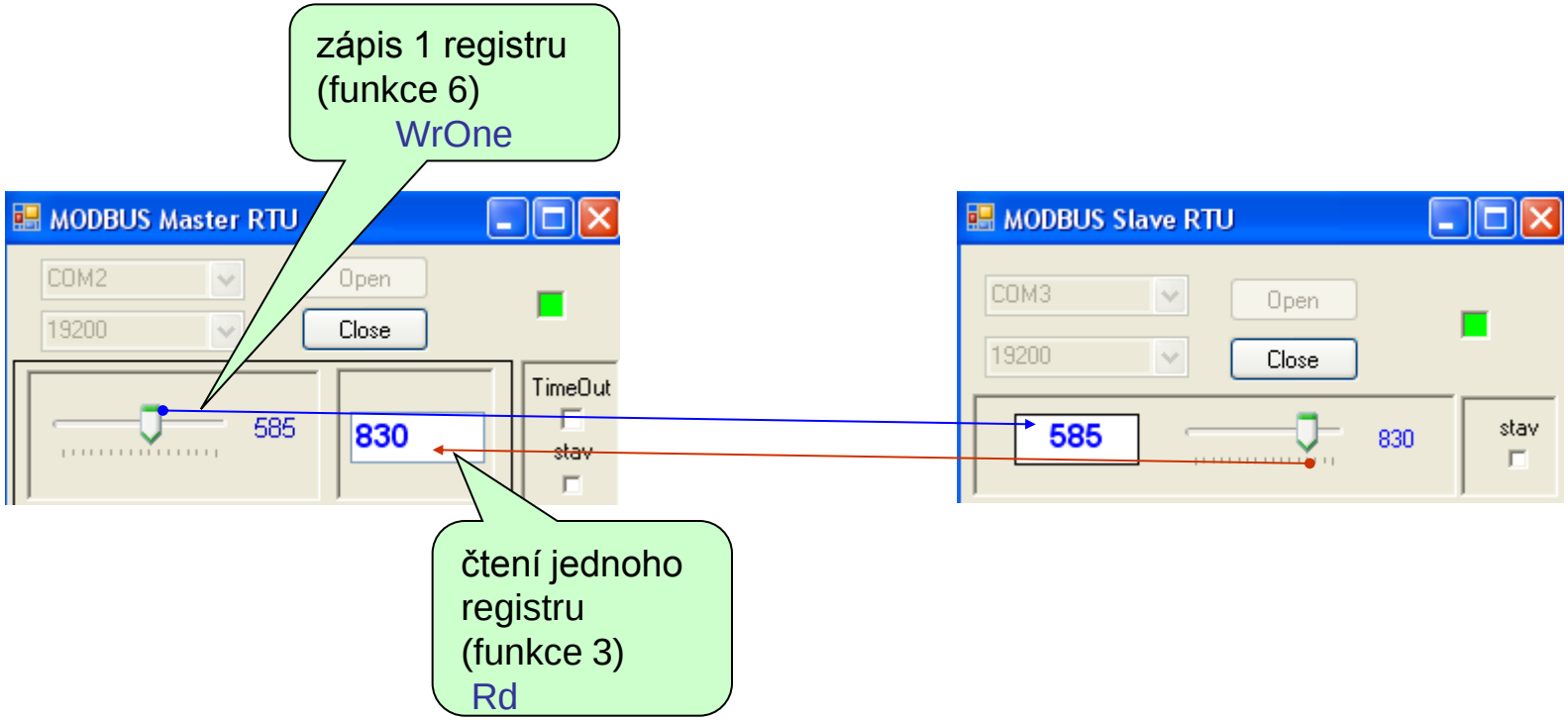
Rozhraní: RS232, standardní rámec 8,N,2
1. část: propojení PC – PC (C# MSVS)
2. část: propojení PC – mikropočítač

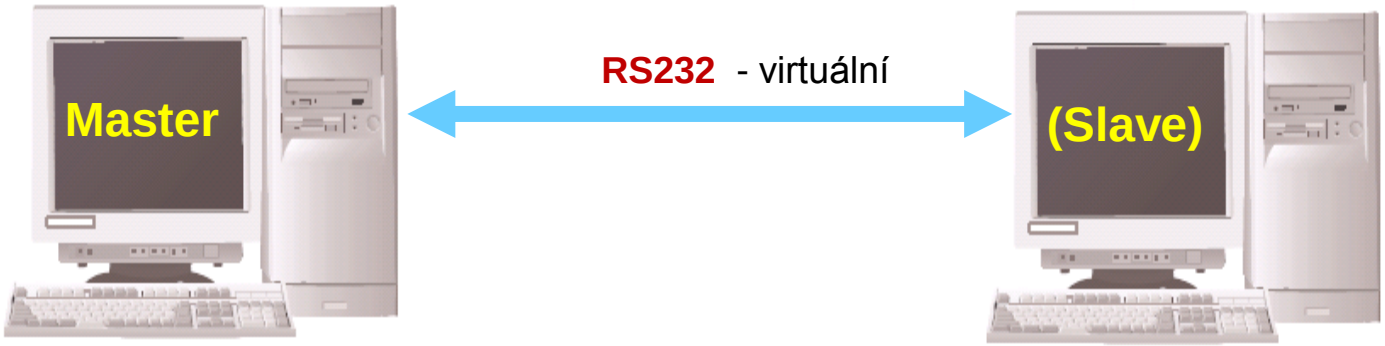
Rozhraní: RS485, standardní rámec 8,N,2
3. část: propojení mikropočítač – mikropočítač

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)
SLAVE
(server)
V pravidelných časových intervalech
generuje 16 bitovou hodnotu
a předává požadavek na zápis
do uzlu SLAVE
kód fukce: 06
+ data
Po příjmu hodnotu zobrazí
a odešle potvrzovací odpověď
potvrzení
V pravidelných časových intervalech
generuje požadavek na čtení
16bitové hodnoty
z uzlu SLAVE a zobrazuje ji
kód fukce: 03
Po příjmu požadavku hodnotu
odešle
data

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\

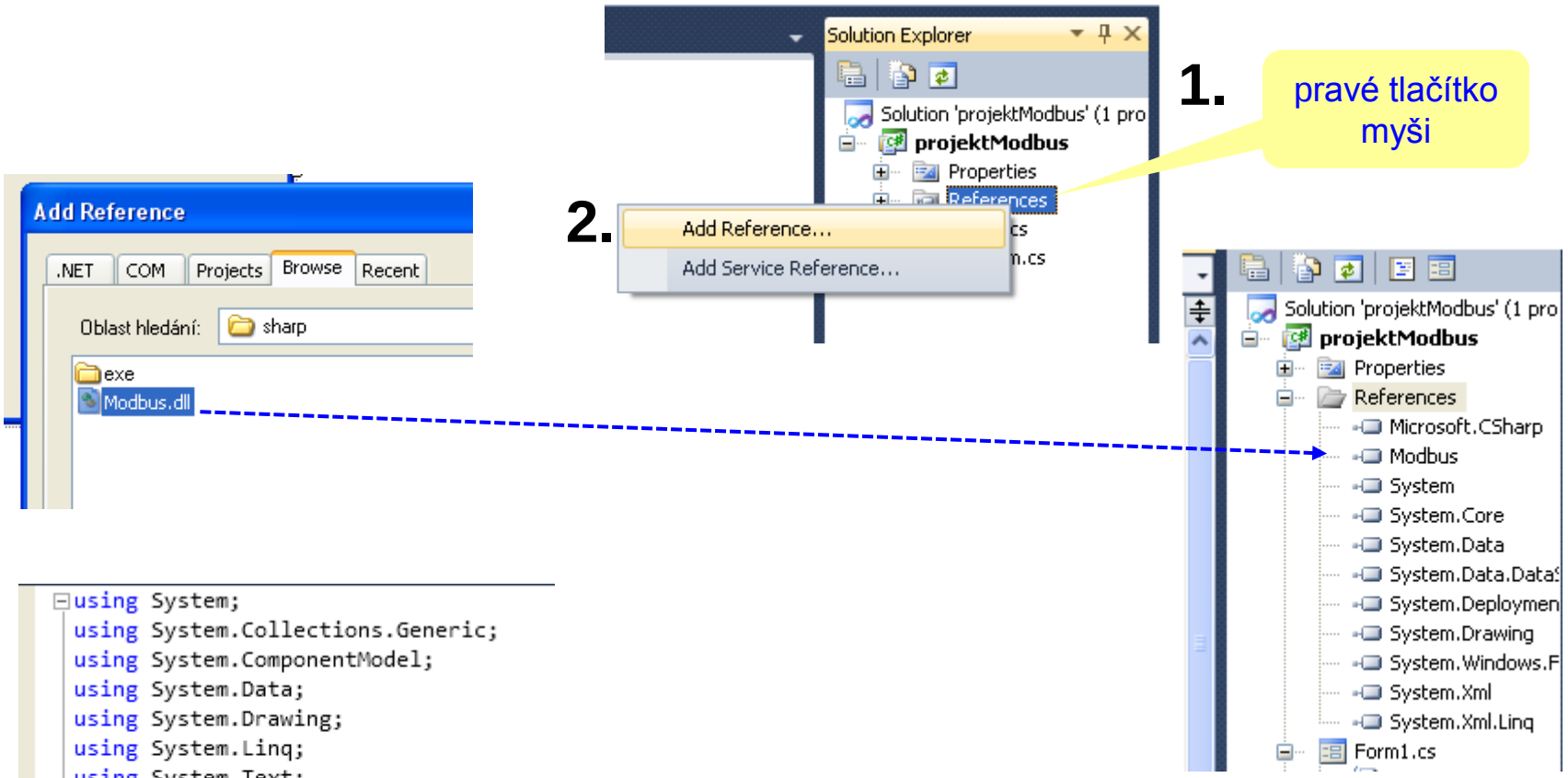
modbus.dll

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



4.

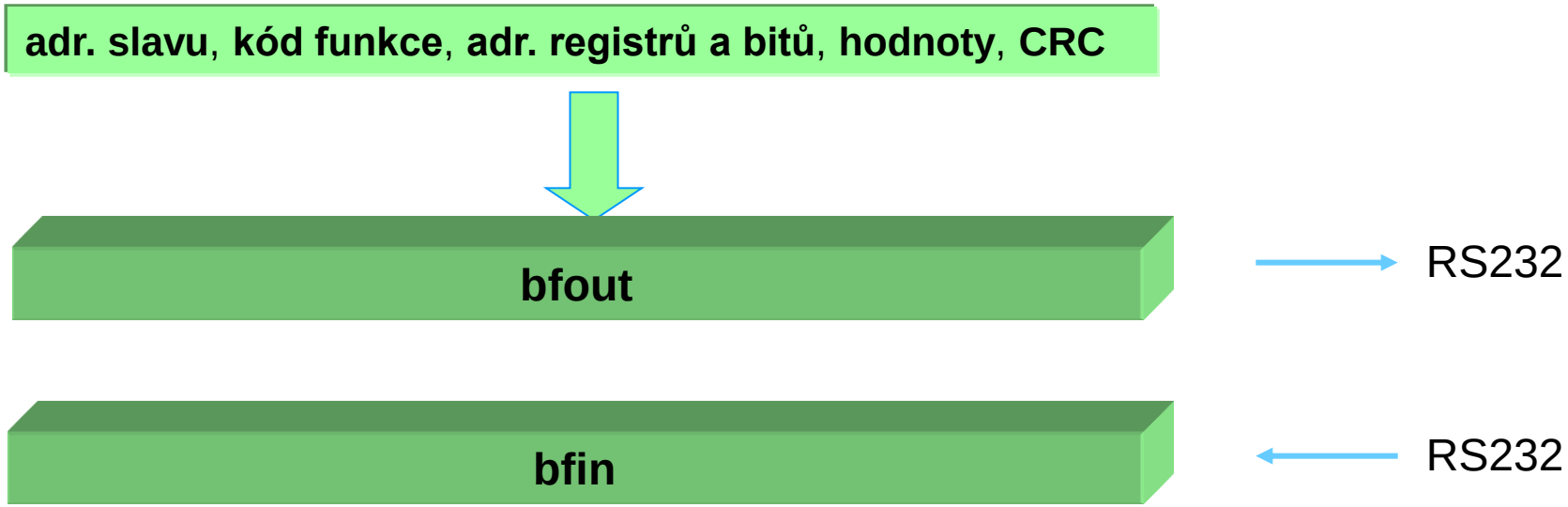
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using Modbus;

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
Rd	WrWord
AnsWr	Crc
AnsRd	WrCrc
AnsErr	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 čtení registru	FCE_RREG	3
Adresa zapisovaného registru	REG_WR	0
Adresa čteného bitu	REG_RD	0




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

bfout[0] **adresa slavu**
bfout[1] **kód funkce**
.
.

Master – implementace na PC (klient)

Konfigurace:

Realizuje funkce (požadavky na server)

- požadavek na zápis jediného vnitřního registru (hodnota 0 až 1023) – funkční kód 6
  metoda WrOne třídy ModbusRTU s kódem funkce 6 (FCE_WREG)
- požadavek na čtení 16 bitové hodnoty – funkční kód 3
  metoda Rd třídy ModbusRTU s kodem funkce 3 (FCE_RREG)

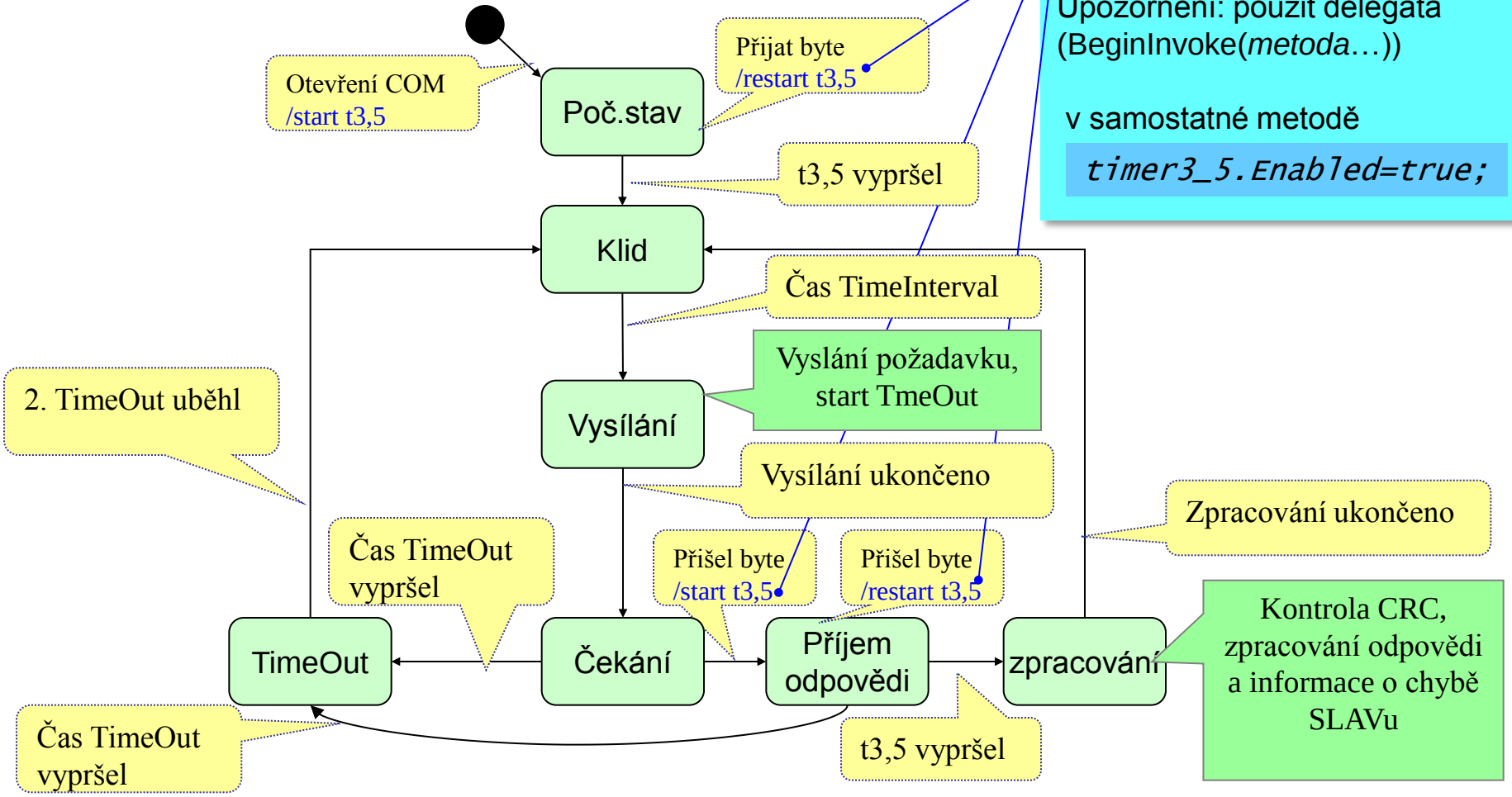
Požadavky odesílat střídavě v pravidelných časových intervalech 200 ms,
jen když je sériový kanál otevřen a Master je ve stavu **klidu**
realizace časovačem intervalu

Implementovat generování čekacího TimeOut intervalu 500 ms na odpvěď od Slave,
po vypršení TimeOutu vyčkat 500 ms a vátit se do stavu **klidu**

Zjednodušený příjem odpovědi
příchozí adresu Slave není nutno testovat, pouze správnost CRC
zpracovat jen odpověď na požadavek čtení registru (FCE_RREG)
informovat o chybové odpovědi od Slave

Implementace generování intervalu 3,5 znaku pro ukončení příjmu zprávy
Omezená (žádná) implementace generování intervalu 1,5 znaku

Master – zjednodušený stavový diagram



Po posledním bytu ve frontě

Upozornění: použit delegáta (BeginInvoke(*metoda*…))

v samostatné metodě

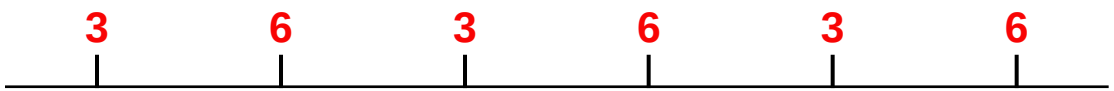
```
timer3_5.Enabled=true;
```

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

Master – vyslání požadavku

Časovač **Sample**

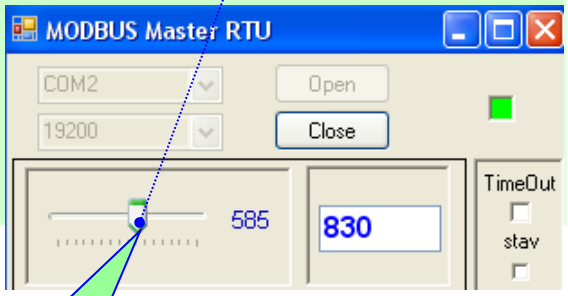
střídavě každých cca 200 ms vysílá rámec s funcí **3** (čtení bitu) a **6** (zápis registru)




```
ModbusRTU Mr;
bool prep;
Tstav stav;
```

Tick
Sample

```
if (comPort.IsOpen && stav == Tstav.stKlid)
{
      stav=Tstav.stVysilani;
      prep = !prep;
      if (prep) n=Mr.WrOne(ADR_S,FCE_WREG,REG_WR,pot,bfout)
      else n=Mr.Rd(ADR_S,FCE_RREG,REG_RD,1,bfout);
      n=Mr.WrCrc(Mr.Crc(bfout,n),bfout,n);
      comPort.Write(bfout,0,n);
      TimerOut.Interval=500;
      TimerOut.Enabled=true;
      stav=Tstav.stCekani;
}
```

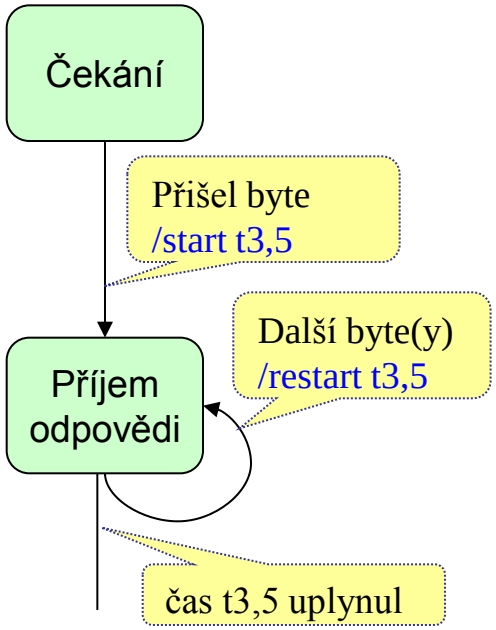


0 až 1023

Master – příjem odpovědi

DataReceived

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

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



Tick
3_5

```
switch(stav){
          .
   case Tstav.stPrijem:
          . // zpracování odpovědi
```

Master – zpracování odpovědi

1. CRC

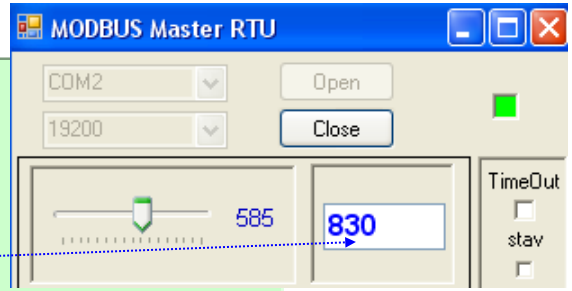
```
if(Mr.Crc(bfin,ix-1)!=Mr.RdCrc(bfin,ix-1)
{
     .. možná informace o chybné CRC
}
else {
```

2. adresa a kód funkce

```
adr_r=bfin[0];
kod_r=bfin[1];
```

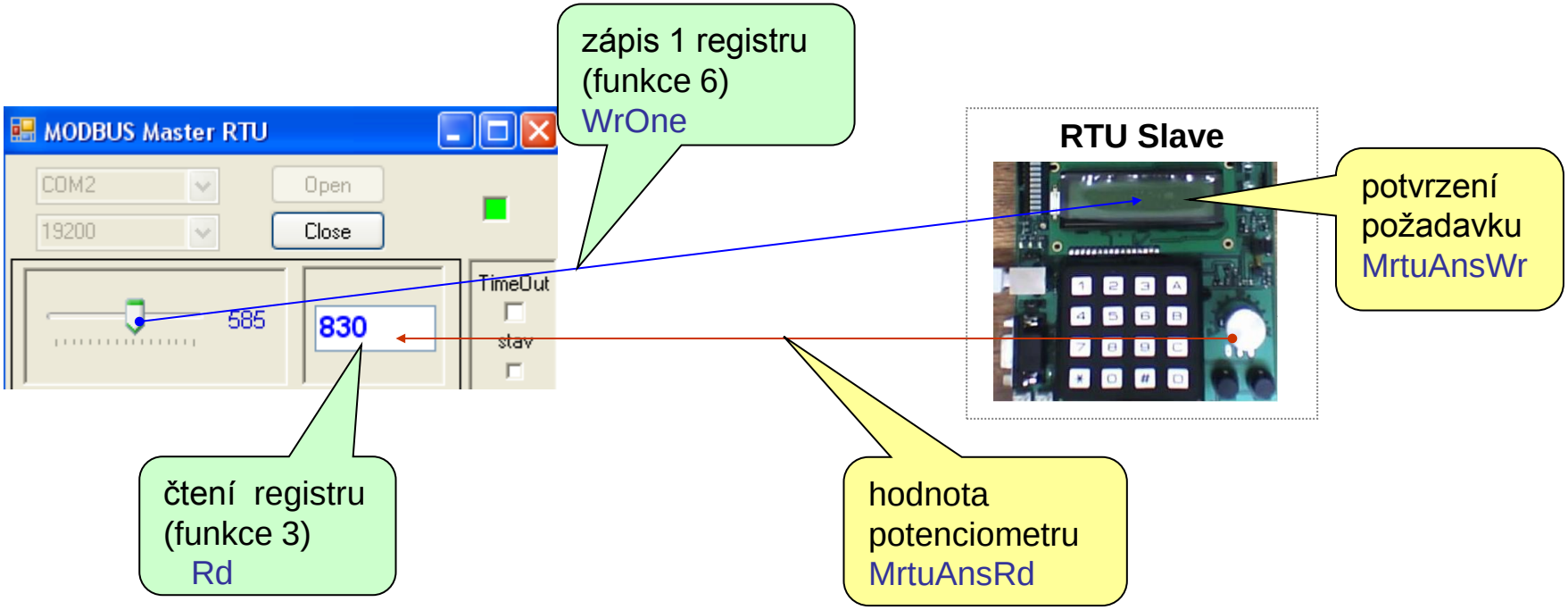
3. reakce na odpověď

```
if(kod_r== FCE_RREG)
{
   pocet=bfin[2];
   val=Mr.RdWord(bfin,3);
         ..
 else if (kod_r>=0x80)
 {
       er= bfin[2];
       switch(er) {
             informace o chybě slavu
       }
 }
 stav=Tstav.stKlid;
```

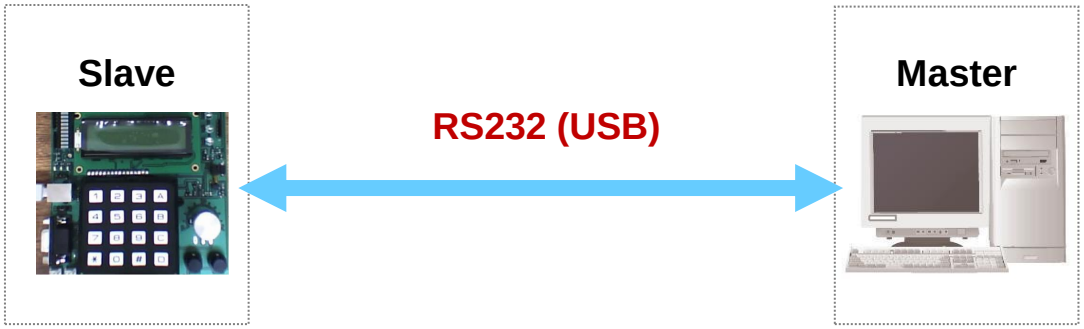


4. informace o chybě Slavu

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\

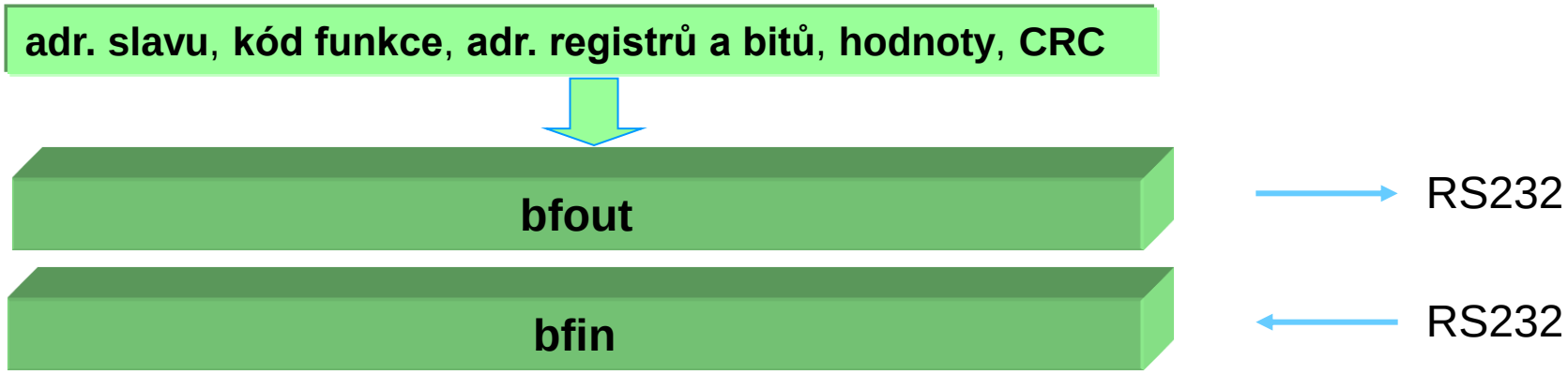
MODBUS.C
MODBUS.H
MAIN.C
ADC.C
LCD.C
LEDBAR.C
TYPY.H

Podpora pro mikropočítač prototypy fukcí **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
MrtuRd	WrWord
MrtuAnsWr	MrtuCrc
MrtuAnsRd	MrtuWrCrc
MrtuAnsErr	MrtuRdCrc
Poznámka: v hlavním programu	#include “Modbus.H”

Definované a doporučené hodnoty		
význam	symbol	hodnota
Adresa uzlu Slave	ADR_S	1
Funkce zápis registru	FCE_WREG	6
Funkce čtení bitu	FCE_RREG	3
Adresa zapisovaného registru	REG_WR	0
Adresa čteného bitu	REG_RD	0




```
                                    //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 → 11 bitů , f_{bit} = 19200 bit/s → $t_{bit} = 1/f_{bit}$

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

pro časovač T1 : $t_{3,5} = N3\_5 \cdot 12/f_{osc}$

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

t3,5 = 3,5·11·32·NBIT/fosc = N3_5·12/fosc

N3_5 = NBIT·109 → #define N3_5 109*NBIT

(re)start t3,5

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

čas uplynul: přerušení nebo 1 → TF1

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

Slave – implementace na mikropočítači (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
  aplikační funkce MrtuAnsWr s kódem přijaté funkce
- požadavek na čtení 16bitové hodnoty – funkční kód 3 a vrací požadovanou hodnotu,
  aplikační funkce MrtuAnsRd s kódem přijaté funkce

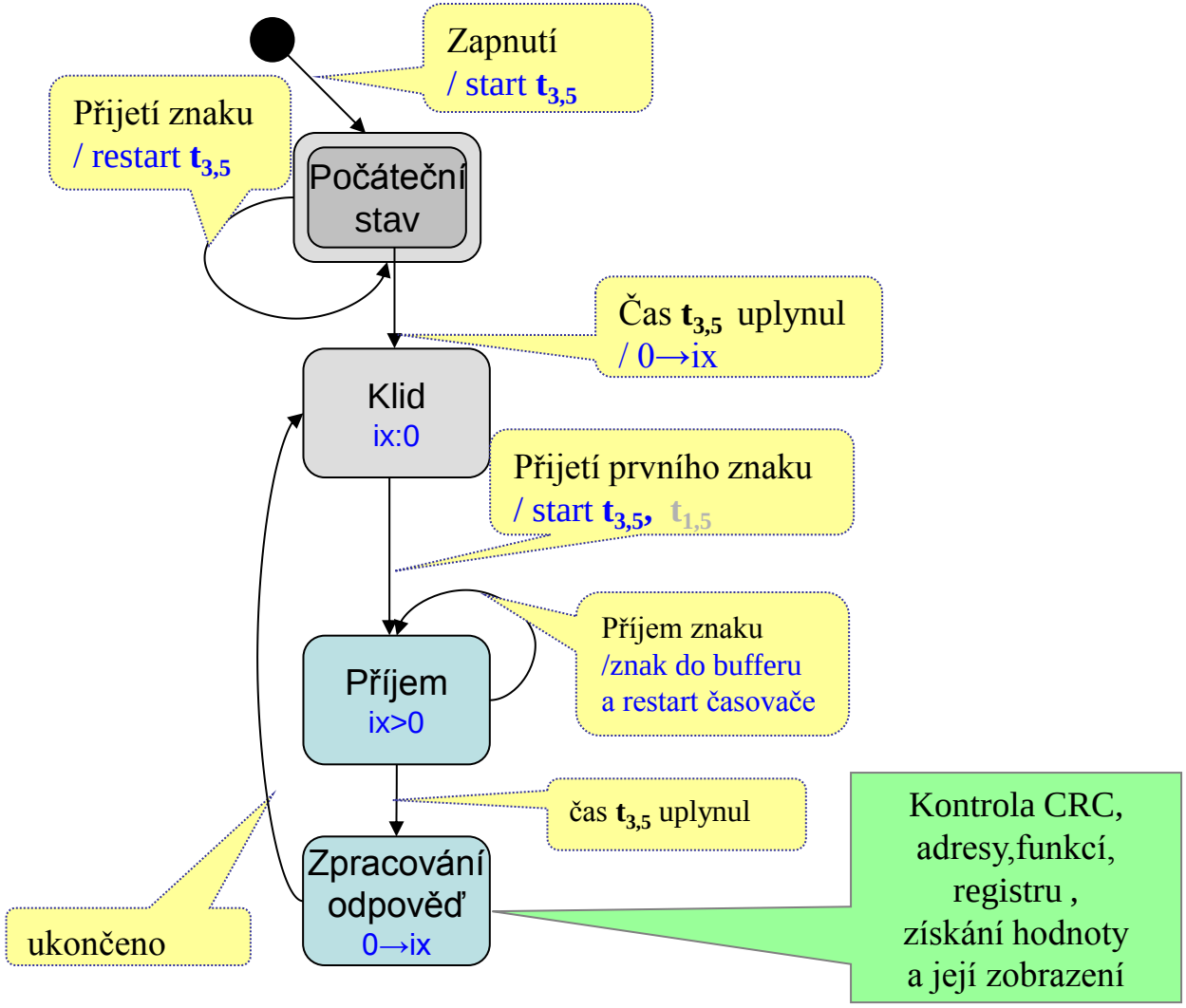
Kontrolovat přijatý požadavek a vracet chybovou odpověď v případě neimplementované funkce, neexistující adresy registru nebo bitu a hodnoty mimo rozsah
aplikační funkce MrtuAnsErr s upraveným kódem funkce a typem chyby

Skupinové vysílání ignorovat .

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

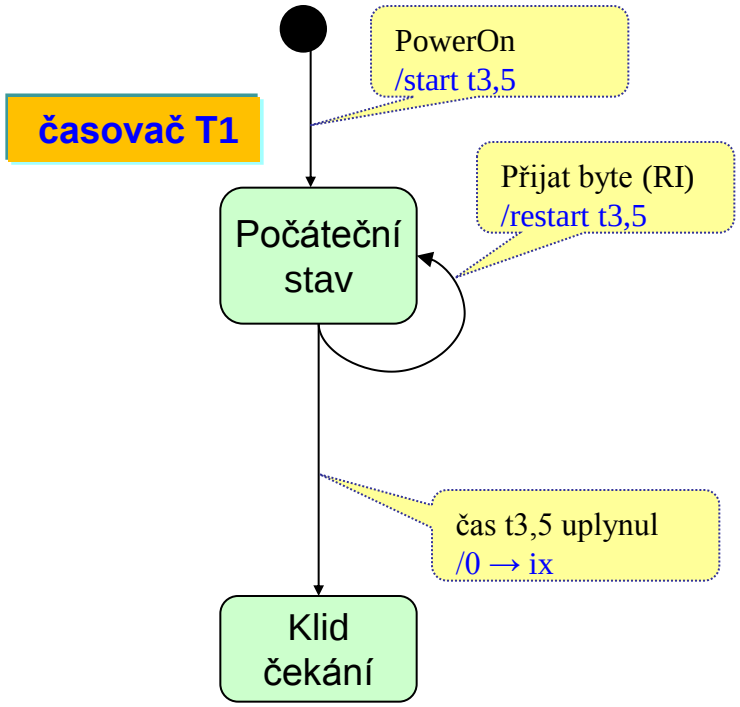
Omezená (žádná) implementace generování intervalu 1,5 znaku

Slave – zjednodušený stavový diagram



Slave– počáteční stav

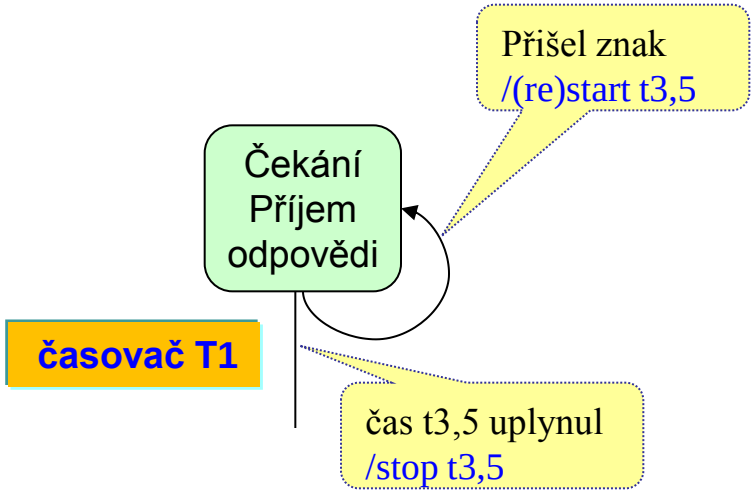
```
void main(void)
{
   .. // inicializace
  do
  {
     RI=0;
     TH1=(word)(-N3_5) >> 8;
     TL1=(byte)(-N3_5) ;
     TR1=1;
     while(!TF1);
     TF1=0;
     TR1=0;
  } while(RI);
  ix=0;
  while(1)
  {
    ..
  }
}
```



Slave – příjem požadavku

```
if (RI)
{
   bfin[ix++]=SBUF;
   RI=0;
   TH1=(word)(-N3_5) >> 8;
   TL1=(byte)(-N3_5) ;
   TF1=0;
   TR1=1;
}
```

```
if(TF1)
{
  TR1=0;
    .  //zpracování požadavku

    .
  ix=0;
}
```



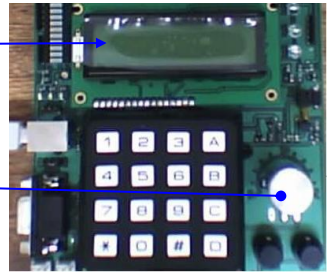
Slave – zpracování požadavku, kontrola položek, příprava odpovědi

CRC a adresa

```
if((bfin[0]==ADR_S)&&(MrtuCrc(bfin,ix-2)==MrtuRdCrc(bfin+ix-2)))
{
```

kód funkce

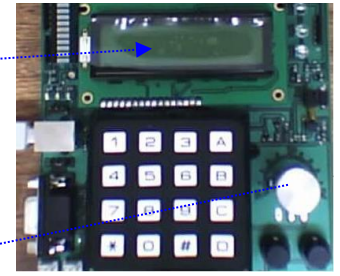
```
er=0;
switch(kod_r=bfin[1])
{
   case FCE_WREG:
          .
          .
   case FCE_RREG:
          .
          .
   default: er=1;
}
```



Slave – zpracování požadavku, kontrola položek, příprava odpovědi

FCE_WREG:

```
if((reg=RdWord(bfin+2))!=REG_WR) er=2;
else if((val=RdWord(bfin+4))>1023) er=3;
else printf(...);
if(er==0) itx=MrtuAnsWr(ADR_S,kod_r,reg,val,bfout);
break;
```



FCE_RREG:

```
if((reg=RdWord(bfin+2))!=REG_RD||(pocet=RdWord(bfin+4))!=1)er=2;
else {
   val= ... ;
   vals[0]=val>>8;
   vals[1]=val;
   itx=MrtuAnsRd(ADR_S,kod_r,2,vals,bfout);
}
break;
```

```
byte vals[2];
```

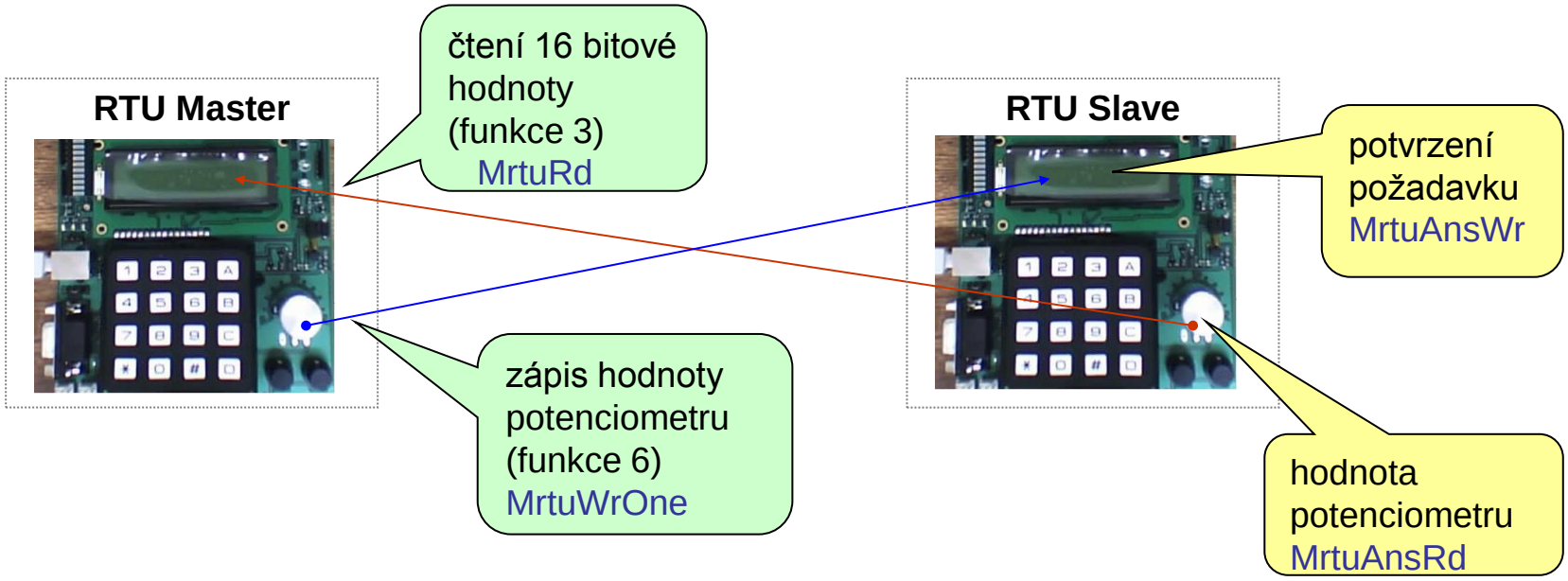
chyba

```
if(er)itx=MrtuAnsErr(adr_r,kod_r|0x80,er,bfout);
```

odeslání odpovědi

```
DIR485=1;  /* na vysílání */
itx+=MrtuWrCrc(MrtuCrc(bfout,itx),bfout+itx);
SendBuf(bfout,itx);
DIR485=0;  /* zpět na příjem */
```

3.část : mikropočítač – mikropočítač



Pro 3.část :
mikropočítač – mikropočítač

je nezbytné

1. správně nastavit propojky pro modul UART
   buď přenos konektorem USB
   nebo přenos konektory RS232/485

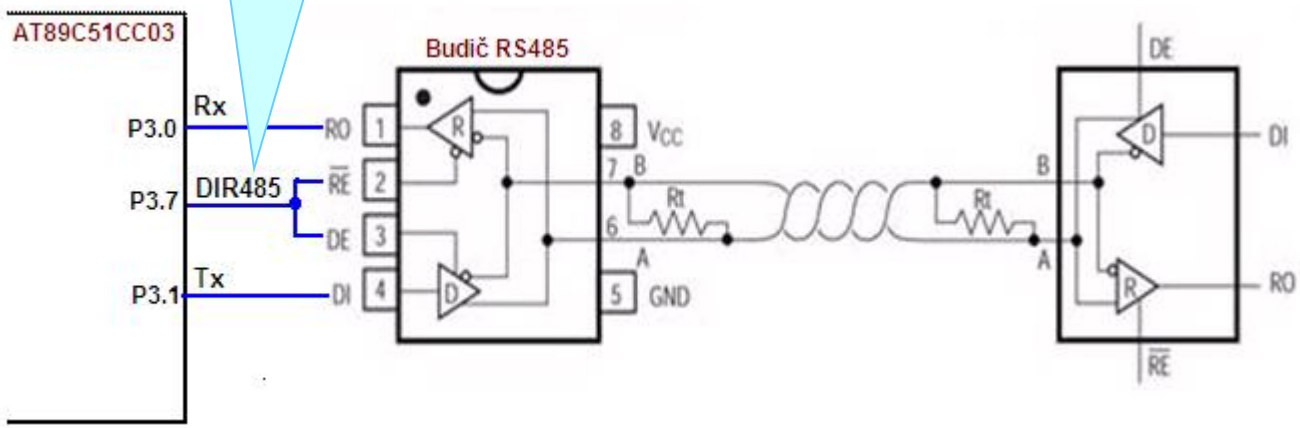
2. správně přepínat budič RS485
   pro příjem
   nebo pro vysílání

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)