EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

ŘPS – úloha MODBUS MA3M

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 - MA3M

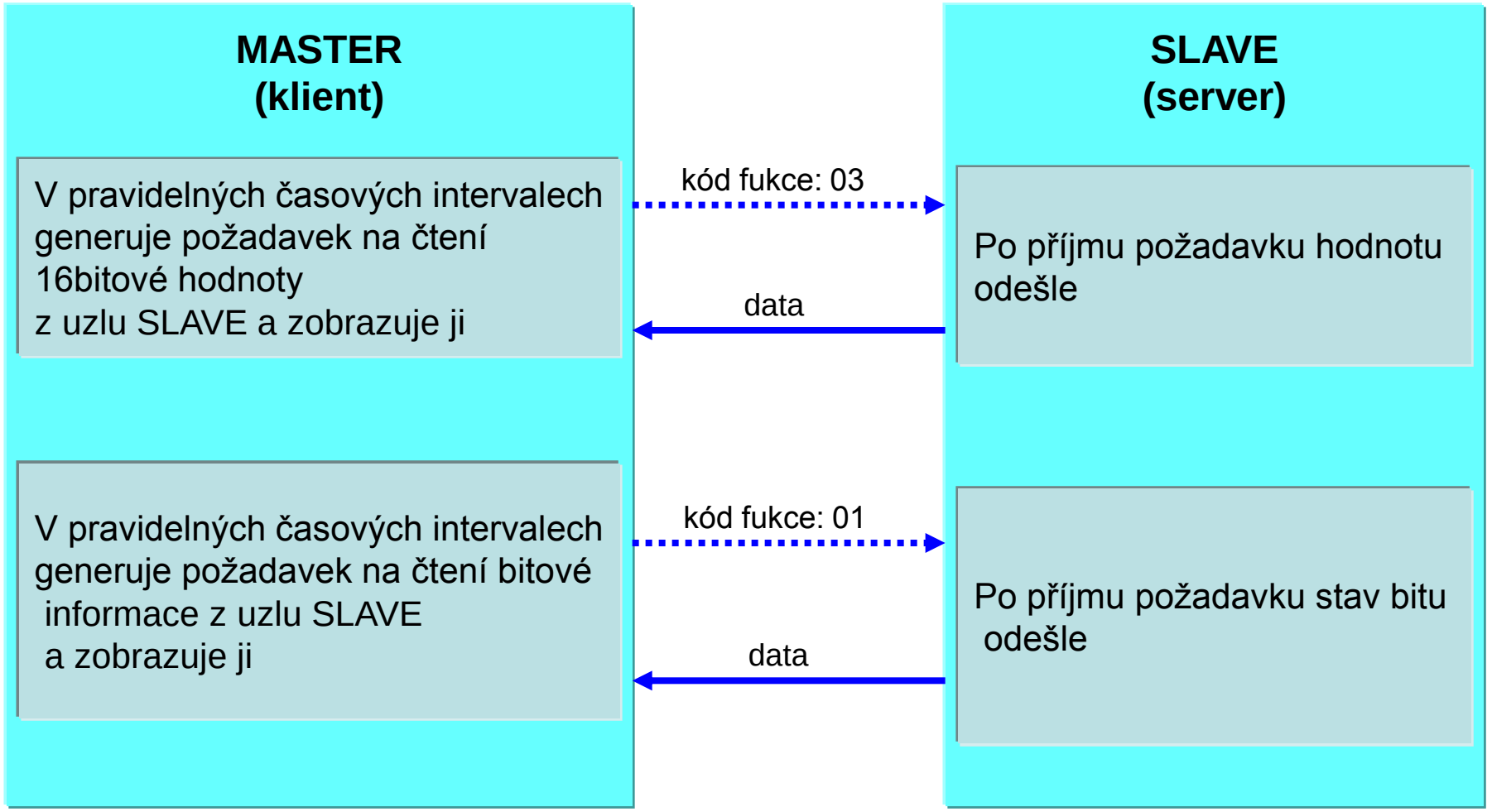
Implementace protokolu MODBUS ASCII na PC a mikropočítačích řady ´51 pro uzly Master (Klient) na PC, Slave (Server) na mikropočítači

Požadované implementované funkce:
- čtení 16bitového vnitřního registru (Holding) z uzlu Slave,
- čtení bitového stavu (Coil) z uzlu Slave.

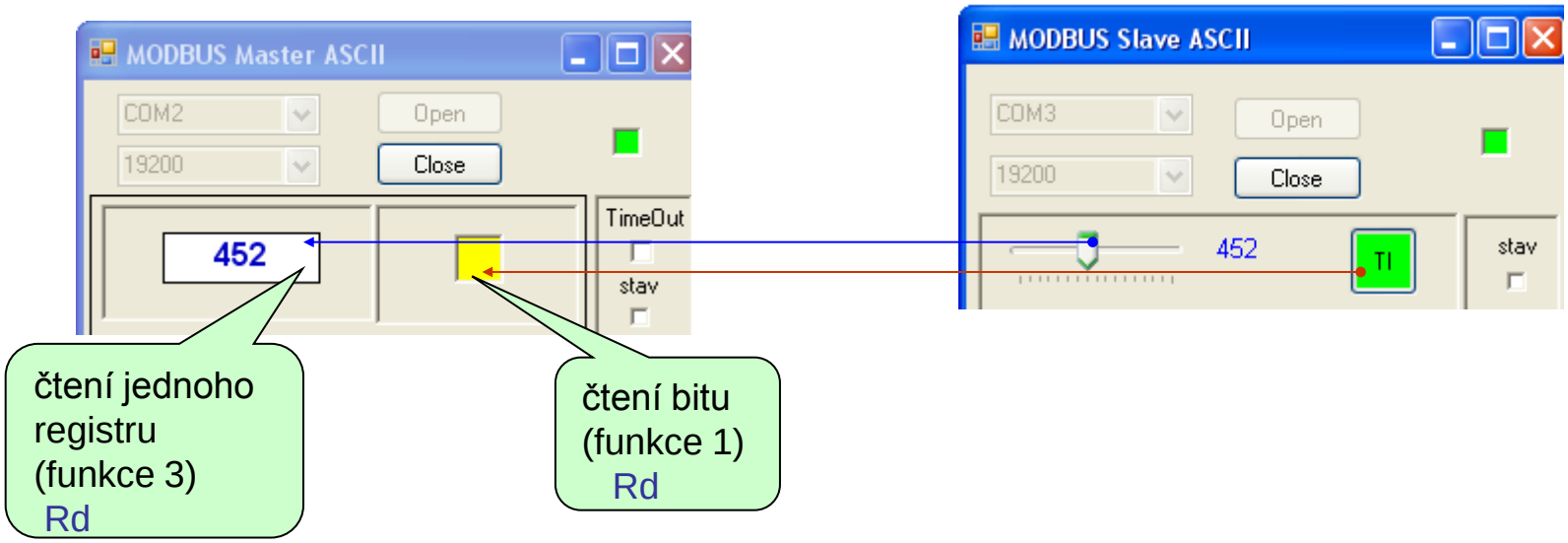
Rozhraní: RS232, standardní rámec 7,N,2
1. část: propojení PC – PC (C# MSVS)
2. část: propojení PC – mikropočítač

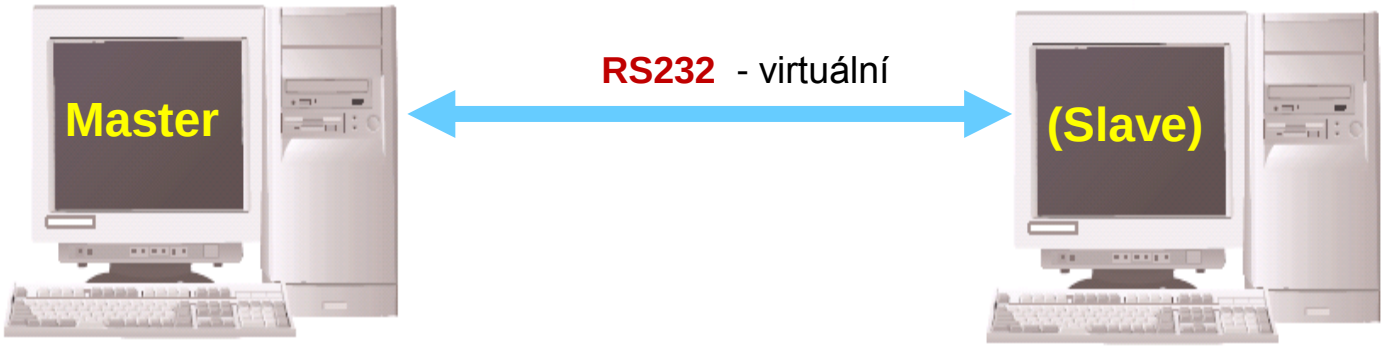
Rozhraní: RS485, standardní rámec 7,N,2
3. část: propojení mikropočítač – mikropočítač

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)
SLAVE
(server)
V pravidelných časových intervalech generuje požadavek na čtení 16bitové hodnoty z uzlu SLAVE a zobrazuje ji
kód fukce: 03
data
Po příjmu požadavku hodnotu odešle
V pravidelných časových intervalech generuje požadavek na čtení bitové informace z uzlu SLAVE a zobrazuje ji
kód fukce: 01
data
Po příjmu požadavku stav 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\

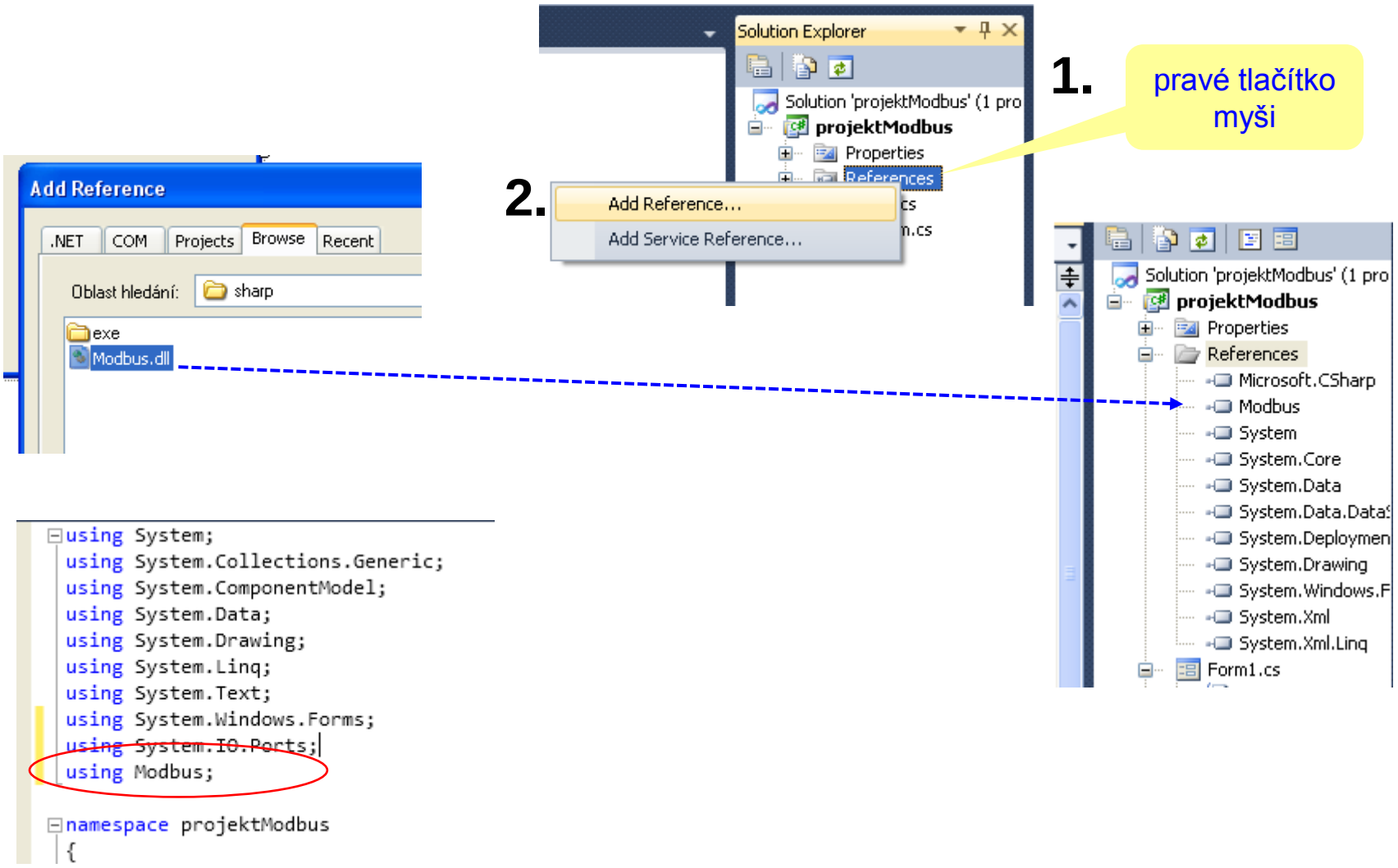
modbus.dll

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

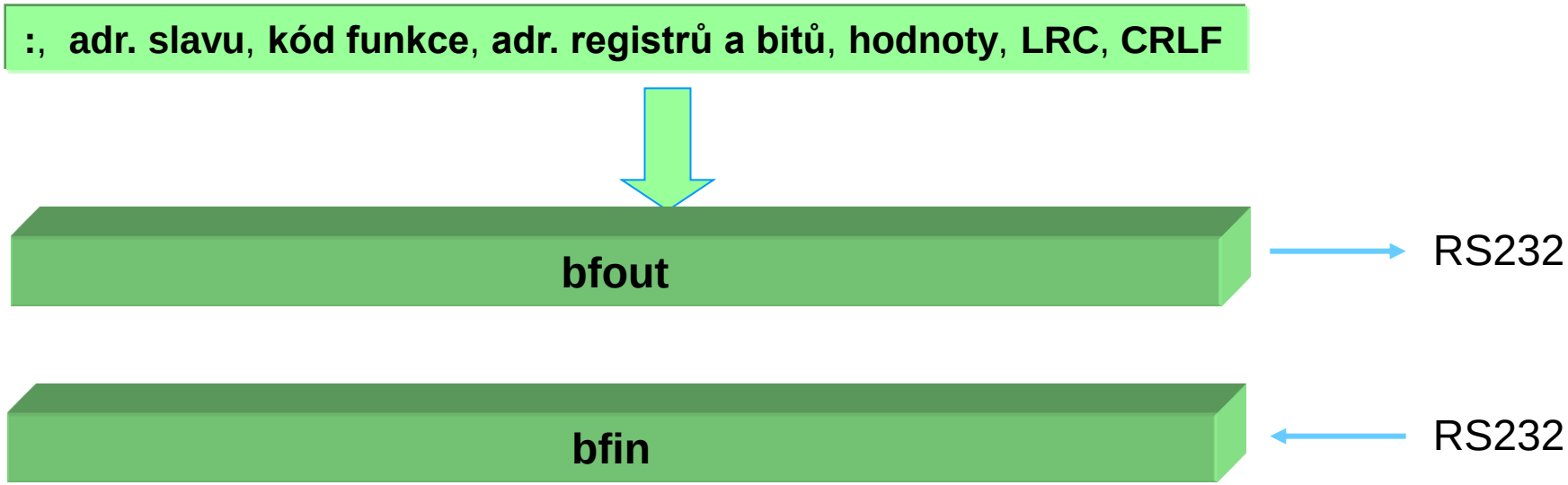


Podpora pro PC Class lib **Modbus.dll** - zdrojový kód **Modbus.cs**

```
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
Rd	RdByte
AnsRd	WrByte
AnsErr	Lrc
	WrEoT
	RdWord
Poznámka: v hlavním programu v sekci **using** přidat Modbus	

Definované a doporučené hodnoty		
význam	symbol	hodnota
Adresa uzlu Slave	ADR_S	1
Funkce čtení registru	FCE_RREG	3
Funkce čtení bitu	FCE_RBIT	1
Adresa čteného registru	REG_RD	0
Adresa čteného bitu	BIT_RD	0




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

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

.

.

Master – implementace na PC (klient)

Konfigurace:

Realizuje funkce (požadavky na server)

- požadavek na čtení 16bitové hodnoty vnitřního registru – funkční kód 3
  metoda Rd třídy Modbus ASCII s kódem funkce 3 (FCE_RREG)
- požadavek na čtení bitové hodnoty – funkční kód 1
  metoda Rd třídy Modbus ASCII s kodem funkce 1 (FCE_RBIT)

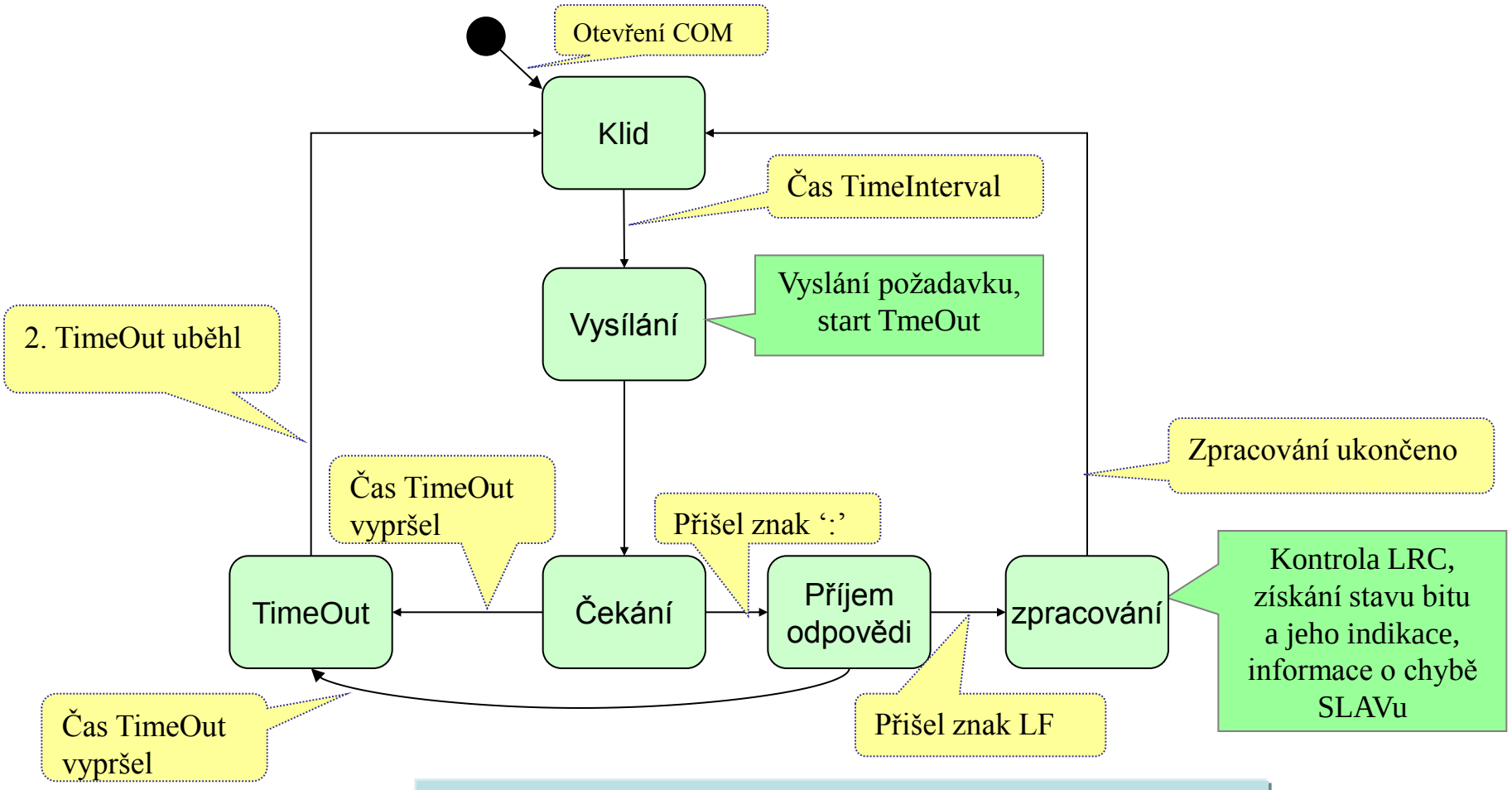
Požadavky odesílat střídavě v pravidelných časových intervalech 200 ms,
jen když je sériový kanál otevřen a Master je ve stavu **klidu**
realizace časovačem intervalu

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 LRC
zpracovat odpovědi na požadavky čtení registru (FCE_RREG)
a čtení bitu (FCE_RBIT)
informovat o chybové odpovědi od Slave

Master – zjednodušený stavový diagram

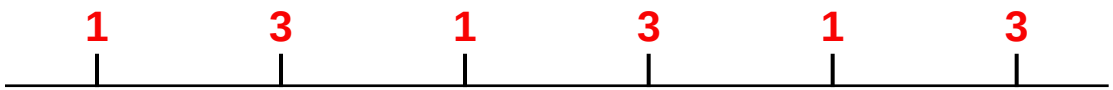



```
enum Tstav{stKlid,stVysilani,stCekani,stPrijem,stTimeOut};
```

Master – vyslání požadavku

Časovač **Sample**

střídavě každých cca 200 ms vysílá rámec s funcí **1** (čtení bitu) a **3** (čtení registru)




```
ModbusASCII Ma;
bool prep;
Tstav stav;
```

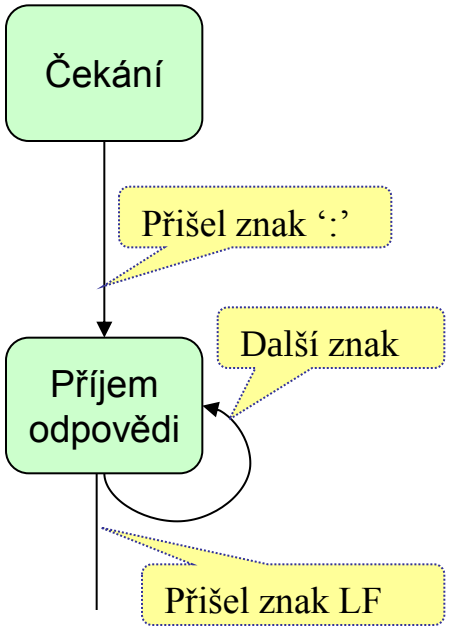
Tick
Sample

```
if (comPort.IsOpen && stav == Tstav.stKlid)
{
      stav=Tstav.stVysilani;
      prep = !prep;
      if (prep) n=Ma.Rd(ADR_S,FCE_RREG,REG_RD,1,bfout)
      else n=Ma.Rd(ADR_S,FCE_RBIT,BIT_RD,1,bfout);
      n=Ma.WrByte(Ma.Lrc(bfout,n-1),bfout,n);
      n=Ma.WrEoT(bfout,n);
      comPort.Write(bfout,0,n);
      TimerOut.Interval=500;
      TimerOut.Enabled=true;
      stav=Tstav.stCekani;
}
```

Master – příjímání odpovědi

DataReceived

```
while(comPort.BytesToRead > 0) {
   byte b = (byte)comPort.ReadByte();
   switch (stav) {
     case Tstav.stCekani:
        if(b==(byte)':‘)
        {
           stav=Ttav.stPrijem;
           bfin[ix=0]=b;
         } break;
     case Tstav.stPrijem:
         {
           if(b==(byte)':‘)ix=0 else ix++;
           bfin[ix]=b;
           if (b==(byte)’\n’)
           {
                .
                .
```



Master – zpracování odpovědi

1. LRC

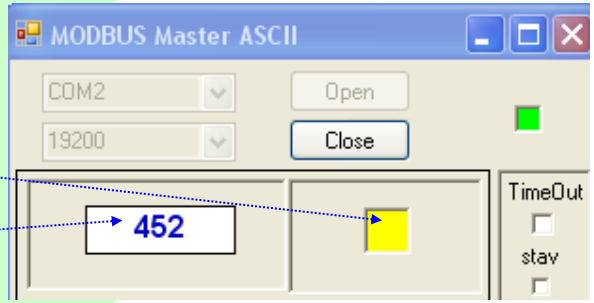
```
if(Ma.Lrc(bfin,ix-4)!=Ma.RdByte(bfin,ix-3)
{
      .. informace o chybné LRC
}
```

2. adresa a kód funkce

```
adr_r=Ma.RdByte(bfin,1);
kod_r=Ma.RdByte(bfin,3);
```

3. reakce na odpověď

```
switch (kod_r) {
   case FCE_RBIT:
     .
     .
   case FCE_RREG:
     .
     .
   default: if (kod_r>=0x80){
       er= Ma.RdByte(bfin,5);
       switch (er) {
             informace o chybě slavu
       }
   }
 }
 stav=Tstav.stKlid;
```



4. informace o chybě Slavu

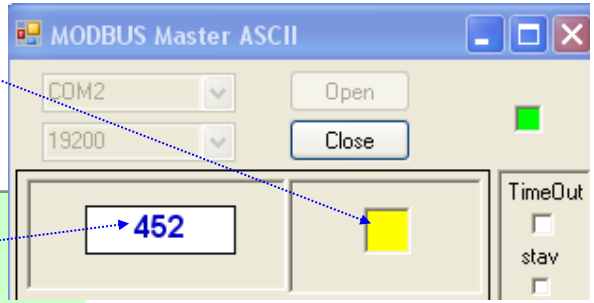
Master – zpracování odpovědi

FCE_RBIT:

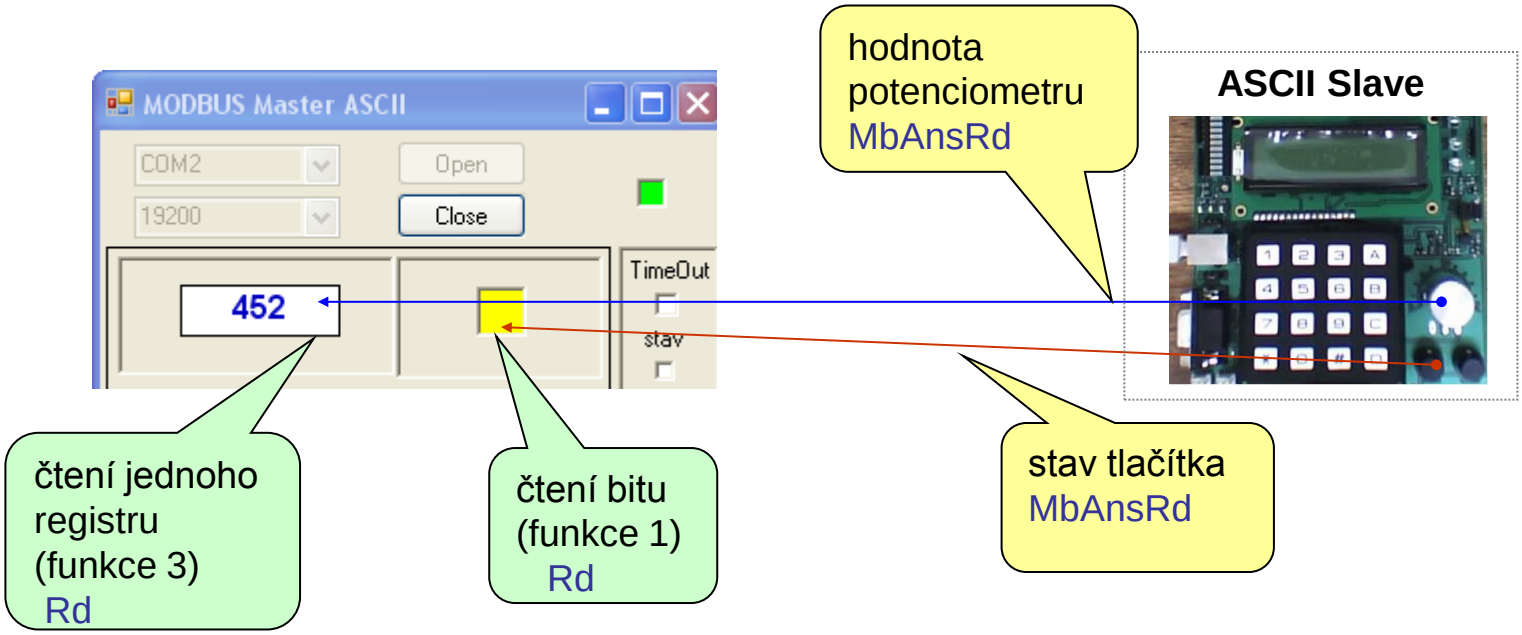
```
    pocet=Ma.RdByte(bfin,5);
    val=Ma.RdByte(bfin,7);
    if (val&1 ==1) žlutá ;
    else bílá ;
break;
```

FCE_RREG:

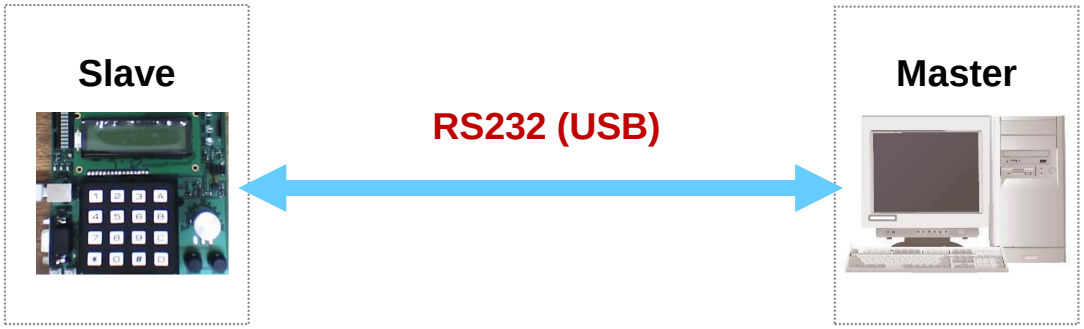
```
    pocet=Ma.RdByte(bfin,5);
    val=Ma.RdWord(bfin,7);
          ..
break;
```



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

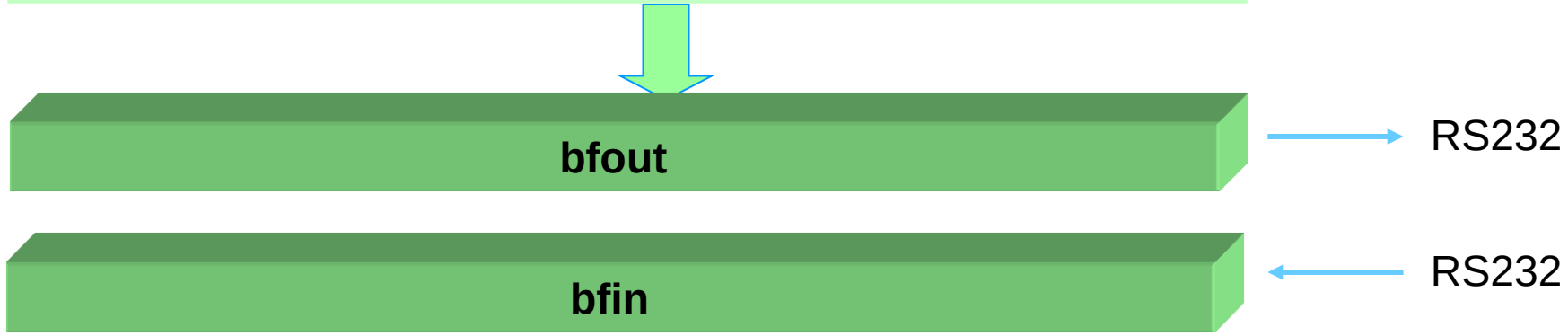
Podpora pro mikropočítač prototypy fukcí **Modbus.H** – zdrojový kód **Modbus.C**

```
byte 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 Modbus.C	
aplikační	pomocné
MbRd	MbRdByte
MbAnsRd	MbWrByte
MbAnsErr	MbLrc
	MbWrEoT
	MbRdWord
Poznámka: v hlavním programu #include “Modbus.H”	

Definované a doporučené hodnoty		
význam	symbol	hodnota
Adresa uzlu Slave	ADR_S	1
Funkce čtení registru	FCE_RREG	3
Funkce čtení bitu	FCE_RBIT	1
Adresa čteného registru	REG_RD	0
Adresa čteného bitu	BIT_RD	0

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




```
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;
  }
}
```

Slave – implementace na mikropočítači (server)

Konfigurace:

Přijímá požadavky od klienta a vrací odpovědi

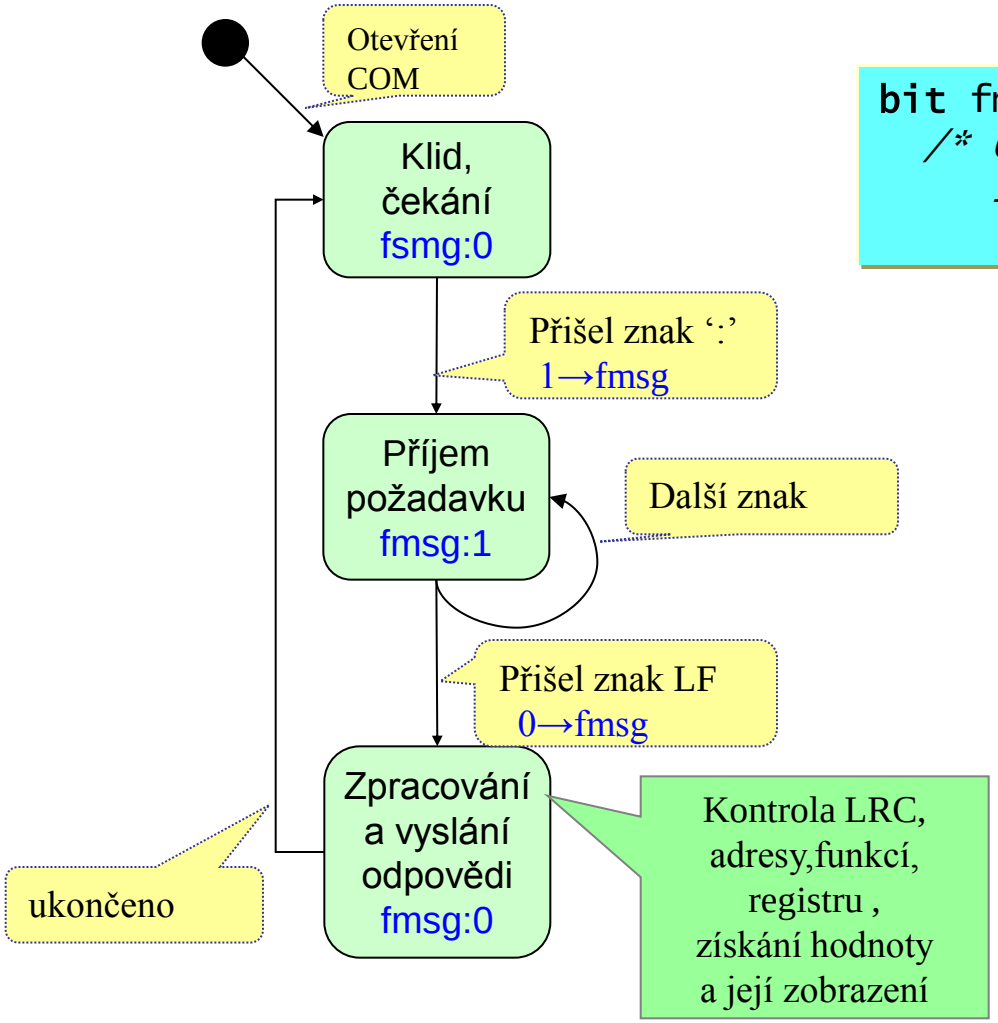
- požadavek na čtení 16bitové hodnoty – funkční kód 3 a vrací požadovanou hodnotu,
  aplikační funkce MbAnsRd s kódem přijaté funkce
- požadavek na čtení bitové hodnoty – funkční kód 1 a vrací stav tlačítka
  aplikační funkce MbAnsRd s kódem přijaté funkce

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

aplikační funkce MbAnsErr s upraveným kódem funkce a typem chyby

Skupinové vysílání ignorovat .

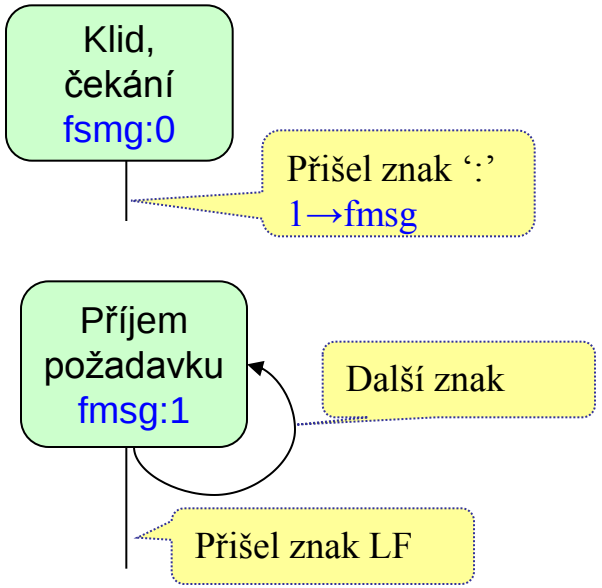
Slave – zjednodušený stavový diagram




```
bit fmsg;
  /* 0 - čekaní na požadavek
     1 - příjem,zpracování požadavku
           a vyslaní odpovědi */
```

Slave – příjem požadavku

```
if(RI)
{
   if((byteIn=SBUF&0x7F)==':')
   {
      ix=0;
      fmsg=1;
   }
   else if(fmsg) ix++;
   RI=0;
   bfin[ix]=byteIn;
   if(fmsg && byteIn=='\n')
   {
      fmsg=0;
        .
        .
```



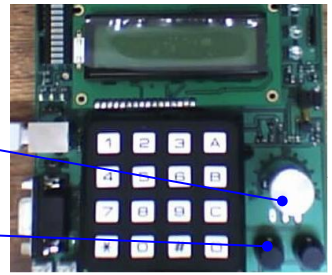
Slave – zpracování požadavku, kontrola položek, příprava odpovědi

1. LRC a adresa

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

2. kód funkce

```
  kod_r=MbRdByte(bfin+3);
  er=0;
  switch(kod_r)
  {
    case FCE_RREG:
       .
       .
    case FCE_RBIT:
       .
       .
    default: er=1;
  }
```

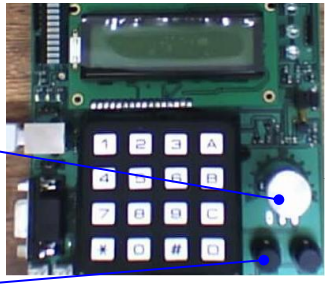


Slave – zpracování požadavku, kontrola položek, příprava odpovědi

FCE_RREG:

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

```
if((reg=MbRdWord(bfin+5))==REG_RD&&(pocet=MbRdWord(bfin+9))==1){
   val=... ;
   vals[0]=val>>8; vals[1]=val;
   itx=MbAnsRd(ADR_S,kod_r,2,vals,bfout);
}
else er=2;
break;
```



FCE_RBIT:

```
if((reg=MbRdWord(bfin+5))==BIT_RD&&(pocet=MbRdWord(bfin+9))==1){
  vals[0]= ...;
  itx=MbAnsRd(ADR_S,kod_r,1,vals,bfout);
}
else er=2;
break;
```

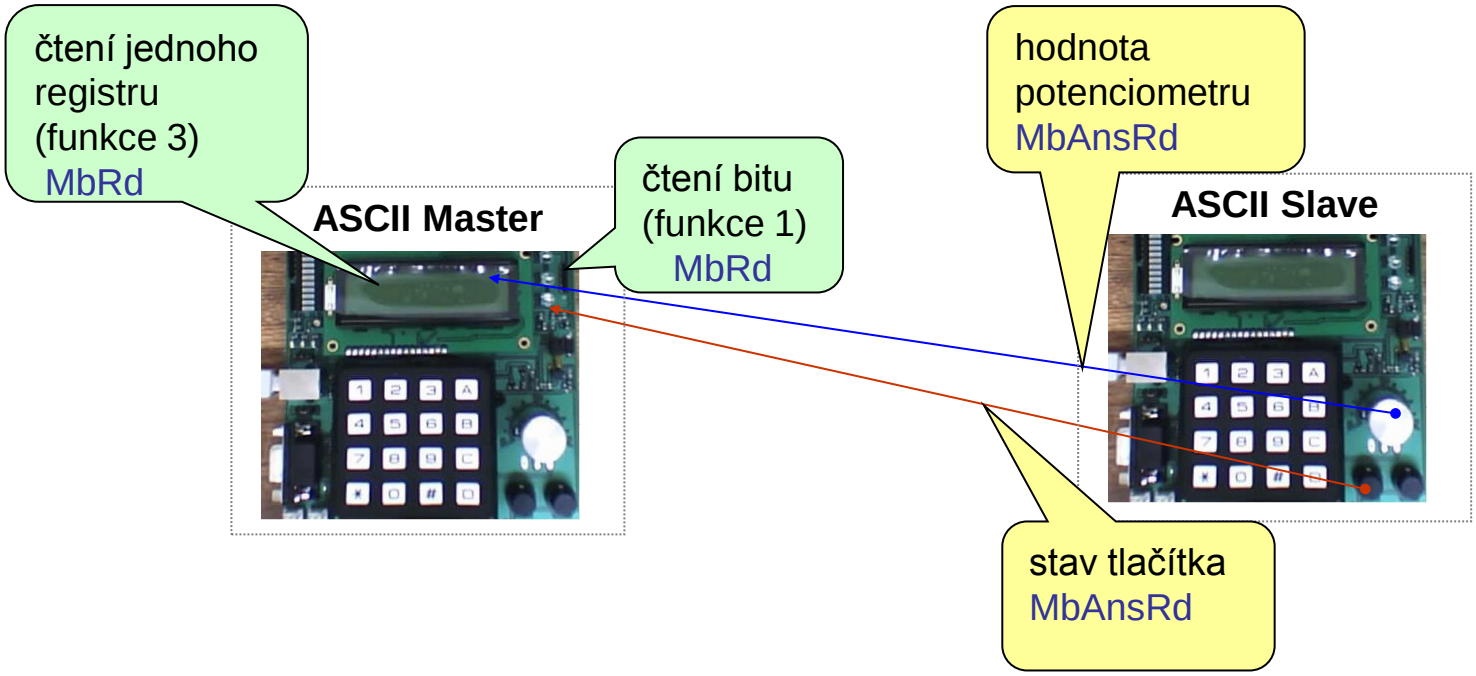
4. chyba

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

odeslání odpovědi

```
DIR485=1;  /* na vysílání */
itx+=MbWrByte(MbLrc(bfout+1,itx-1),bfout+itx);
itx+=MbWrEoT(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

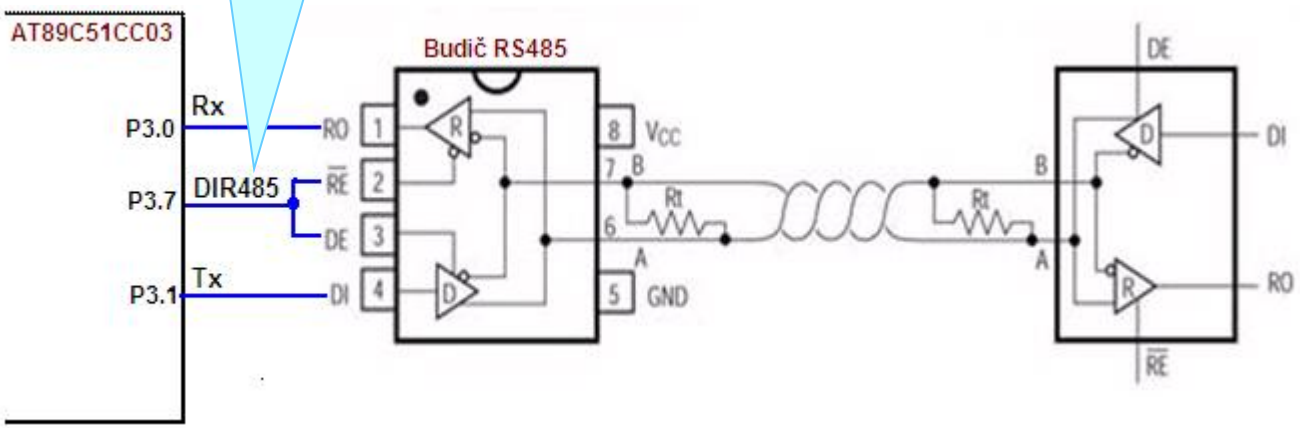
2. správně přepínat budič RS485
   pro příjem
   nebo pro vysílání

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)