ChnMod

JEDNOTKA OBECNÉ SÉRIOVÉ KOMUNIKACE POMOCÍ MODEMU

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.10

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 16.05.2003
Datum posledního uložení dokumentu: 16.05.2003
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Wi		První vydání.
1.10	4.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Doplněné konstanty CHS_SendBegin, CHS_Send a CHS_ReceiveReConn. Zrušené proměnné Ch_RBuffAt a Ch_MRBuff.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky obecné sériové komunikace pomocí modemu.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem Chnt popisujícím rozhraní svých potomků a ChnTypes.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

Používané termíny a definice jsou popsány v samostatném dokumentu Termíny a definice.

3. Úvod

Knihovna ChnMod byla vytvořena s cílem vytvoření rodičovského objektu **tChnMod** pro tvorbu konkrétních komunikačních objektů sloužících pro obsluhu sériové komunikace prostřednictvím modemů. V tomto objektu jsou definovány virtuální metody obsahující automaty pro navazování a udržování spojení, vysílání a příjem dat a rušení navázaného spojení. Komunikační objekty dědičné od tohoto rodičovského komunikačního objektu vytvoří nebo upraví jen ty metody, které jsou specifické pro daný typ sériového rozhranní pro komunikaci s modemem. Jsou to především metody pro fyzické nastavování a čtení modemových signálů a pro vysílání a příjem jednotlivých zpráv do modemu. Znaky dat přicházející po komunikaci (myslí se data přicházející z protější stanice s níž je navázáno spojení) jsou ukládány do vstupního kruhového vyrovnávacího bufferu, z kterého jsou předávány metodami **ChReceive** a **ChReceiveChar** k dalšímu zpracování. Znaky určené k odeslání jsou zasílány z výstupního bufferu do linky. Určení parametrů komunikace je voleno parametrem nastavovací metody ChSetParam.

Knihovna rovněž definuje objekt **tAddChnMod**, který je dědicem od rodičovského objektu tAddChnVirt. Objekt tAddChnMod zajistí, aby daný komunikační objekt (objekt tChnMod) byl k aplikaci připojen a popřípadě zajistí vytvoření instance tohoto objektu. Po přilinkování této jednotky do aplikace (příkazem "uses ChnMod"), se jméno objektu tChnMod automaticky vloží do seznamu správců komunikačních objektů pro případné použití.

Protože je objekt **tChnMod** dědicem rodičovského komunikačního objektu **tChnVirt**, jsou v této příručce popsány jen odlišnosti a speciality pro tento druh sériové komunikace. Ostatní naleznete v příručce **ChnVirt**. Některé použité konstanty a typy jsou předdefinované v jednotce **ChnTypes**.

4. Konstanty a typy

```
cVerNo = např. $0251; { BCD formát }
cVer   = např. '02.51,07.08.2003';
```

Číslo verze jednotky v BCD tvaru a v textové podobě včetně datumu změny.

```
cName     = 'MOD';
```

Konstanta **cName** definuje jméno komunikačního objektu **tChnMod**.

```
res_Err_WrongPIN = $20;
```

Konstanta **res_ERR_WrongPIN** definuje chybový kód při inicializaci GSM modemu metodou **ChConnect** při zadání špatného SIM PINu.

```
CHS_SendBegin     = $03; počátek vysílání a čekání do stabilního
                         stavu Connect
CHS_Send          = $04; probíhá vysílání
CHS_ReceiveReConn = $04; probíhá nové navázání spojení
```

Konstanty definují vnitřní stav automatů komunikačního objektu.

4.1. Kódy příkazů pro metodu ChSetBinParam

```
cmd_SendCmd  = $10;
```

Konstanta **cmd_SendCmd** definuje kód příkazu pro vyslání příkazu do GSM modemu s doplněním znaku CR na konec příkazu. Předávaným parametrem je ukazatel na buffer dat, který svou strukturou odpovídá datovému typu **tAtCmdBuff** (viz níže), s nastavenými příslušnými hodnotami. Metoda provede pouze první krok automatu pro vyslání požadovaného příkazu, který je třeba dále postrkovat voláním metody **ChGetBinParam** s kódem **cmd_CmdReady** (viz níže) na stabilní stav **byte(SMCS_Ready)**. Totéž platí i pro volání metody **ChSetBinParam** s kódy příkazů **cmd_RecAnsw**, **cmd_CmdAnsw**, **cmd_SendData** a **cmd_RecData**.

```
cmd_RecAnsw  = $11;
```

Konstanta **cmd_RecAnsw** definuje kód příkazu pro přijetí zprávy z modemu s odstraněním koncových znaků CR.

```
cmd_CmdAnsw  = $12;
```

Konstanta **cmd_CmdAnsw** definuje kód příkazu pro vyslání příkazu do modemu s čekáním na odpověď (je to v podstatě sloučení příkazů cmd_SendCmd a cmd_RecAnsw).

```
cmd_SendData = $13;
```

Konstanta **cmd_SendData** definuje kód příkazu pro vyslání dat do GSM modemu bez doplnění koncového znaku CR.

```
cmd_RecData  = $14;
```

Konstanta **cmd_RecData** definuje kód příkazu pro přijetí dat z GSM modemu bez odstranění koncových znaků CR.

4.2. Kódy příkazů pro metodu ChGetBinParam

```
cmd_GetCTS = $03;
```

Konstanta **cmd_GetCTS** definuje kód příkazu pro čtení hodnoty modemového signálu CTS. Logický stav signálu je navrácen v 0-tém bitu výsledku metody **ChGetBinParam**.

```
cmd_GetDSR = $04;
```

Konstanta **cmd_GetDSR** definuje kód příkazu pro čtení hodnoty modemového signálu DSR. Logický stav signálu je navrácen v 0-tém bitu výsledku metody **ChGetBinParam**.

```
cmd_GetRI  = $05;
```

Konstanta **cmd_GetRI** definuje kód příkazu pro čtení hodnoty modemového signálu RI (Ring Indikator). Logický stav signálu je navrácen v 0-tém bitu výsledku metody **ChGetBinParam**.

```
cmd_GetCD  = $06;
```

Konstanta **cmd_GetDSR** definuje kód příkazu pro čtení hodnoty modemového signálu CD (LRSD). Logický stav signálu je navrácen v 0-tém bitu výsledku metody **ChGetBinParam**.

```
cmd_CmdReady = $15;
```

Konstanta **cmd_CmdReady** definuje kód příkazu pro provedení kroku automatu, který byl odstartován voláním metody **ChSetBinParam** s kódy příkazů **cmd_SendCmd**, **cmd_RecAnsw**, **cmd_CmdAnsw**, **cmd_SendData** nebo **cmd_RecData**, s vrácením aktuálního stavu automatu. Má smysl provádět test pouze na stabilní stav **byte(SMCS_Ready)**, který indikuje ukončení požadovaného příkazu. Po dosažení tohoto stavu je nutno provést test na výsledek úspěšnosti provedení příkazu. Tento test se provede voláním metody **ChGetBinParam** s kódem **cmd_CmdResult** (viz dále).

```
cmd_CmdResult= $16;
```

Konstanta **cmd_CmdResult** definuje kód příkazu pro navrácení výsledku úspěšnosti provedení příkazu, který byl odstartován voláním metody **ChSetBinParam** s kódy **cmd_SendCmd** až **cmd_RecData**. Výsledná hodnota je některá z konstant ResMod_XXX (viz níže).

4.3. Konstanty výsledků po vyslání příkazu do modemu

Následující konstanty jsou navraceny metodou **ChGetBinParm** s kódem příkazu **cmd_CmdResult** po provedení příkazu vyvolaném metodou **ChSetBinParam** (viz víše).

```
ResMod_Ok           =  0;
```

Konstanta **ResMod_Ok** definuje bezchybné provedení příkazu, respektive odezvu "OK" od modemu.

```
ResMod_Connect      =  1;
```

Konstanta **ResMod_Connect** znamená, že modem hlásí zprávu "CONNECT" (spojení navázáno).

```
ResMod_Busy         = 10;
```

Konstanta **ResMod_Busy** znamená, že modem hlásí zprávu "BUSY" (obsazeno).

```
ResMod_Error        = 12;
```

Konstanta **ResMod_Error** znamená, že modem hlásí zprávu "ERROR" (chyba).

```
ResMod_NoCarry      = 13;
```

Konstanta **ResMod_NoCarry** znamená, že modem hlásí zprávu "NO CARRIER" (není nosná).

```
ResMod_NoAnswer     = 14;
```

Konstanta **ResMod_NoAnswer** znamená, že modem hlásí zprávu "NO ANSWER" (žádná odpověď).

```
ResMod_BadConnect   = 15;
```

Konstanta **ResMod_BadConnect** znamená, že modem hlásí zprávu "BAD CONNECT" (nepřípustné navázání spojení).

```
ResMod_Answer       = 19;
```

Konstanta **ResMod_Answer** znamená, že modem hlásí jinou (neznámou) odpověď.

```
ResMod_LongAnswer   = 20;
```

Konstanta **ResMod_LongAnswer** znamená, že modem hlásí příliš dlouhou odpověď.

```
ResMod_SHwErr       = 21;
```

Konstanta **ResMod_SHwErr** znamená, že nastala chyba nízké úrovně při vysílání příkazu do modemu.

```
ResMod_RHwErr       = 22;
```

Konstanta **ResMod_RHwErr** znamená, že nastala chyba nízké úrovně při příjmu odpovědi od modemu.

```
ResMod_RTimeOut      = 23;
```

Konstanta **ResMod_RTimeOut** znamená, že nastal timeout při příjmu odpovědi od modemu.

```
ResMod_STimeOut      = 24;
```

Konstanta **ResMod_STimeOut** znamená, že nastal timeout při vysílání příkazu do modemu.

4.4. Výčtové typy pro určení položek polí a jednoduché typy

```
tDelays = (ESC, HWIni0, HWIni1, HWHangUp0, HWHangUp1,
           AfterI1S, AfterI2S, AfterI3S, Busy);
```

Typ **tDelays** definuje výčet různých časových pauz během komunikace. Obsahuje následující položky:

ESC	- pauza před a po vyslání Escape-řetězce (zpravidla '+++') v SoftWarovém HangUpu (přechod z datového režimu modemu do příkazového)
HWIni0	- doba pro nulování signálu DTR při hardwarové inicializaci modemu
HWIni1	- doba pro nastavení signálu DTR při hardwarové inicializaci modemu
HWHangUp0	- doba pro nulování signálu DTR v hardwarovém zavěšování modemu
HWHangUp1	- doba pro nastavení signálu DTR v hardwarovém zavěšování modemu
AfterI1S	- pauza po vyslání 1. inicializačního řetězce AT příkazů do modemu
AfterI2S	- pauza po vyslání 2. inicializačního řetězce AT příkazů do modemu
AfterI3S	- pauza po vyslání 3. inicializačního řetězce AT příkazů do modemu
Busy	- prodleva mezi vytáčením při BUSY

```
tTimeOuts = (SendAtCmd, CmdAnsw, DialAnsw1, DialAnsw2, DialAnsw3);
```

Typ **tTimeOuts** definuje výčet různých timeoutů. Obsahuje následující položky:

SendAtCmd	- timeout pro čekání na odpověď na vyslaný AT příkaz do modemu
DialAnsw1	- timeout pro 1. odpověď na PNS (vytáčení čísla)
DialAnsw2	- timeout pro 2. odpověď na PNS (vytáčení čísla)
DialAnsw3	- timeout pro 3. odpověď na PNS (vytáčení čísla)
CmdAnsw	- timeout pro odpověď na ostatní příkazy

```
tStrAnsw = (Init1S, Init2S, Init3S, DialPNS, DialNum,
            AnswOK, AnswERROR, AnswCONNECT, AnswBUSY, AnswNOCARRY,
            ESCStr, ATH);
```

Typ **tStrAnsw** definuje výčet pro různé řídící řetězce posílané do modemu nebo naopak z modemu přijímané. Obsahuje následující položky:

DialPNS	- vytáčecí string (zpravidla „ATDP“ nebo „ATD“ apod.)
DialNum	- vytáčené číslo
Init1S	- 1. inicializační řetězec AT příkazů
Init2S	- 2. inicializační řetězec AT příkazů

Init3S	- 3. inicializační řetězec AT příkazů
AnswOK	- řetězec pro odpověď "OK"
AnswERROR	- řetězec pro odpověď "ERROR"
AnswCONNECT	- řetězec pro odpověď "CONNECT"
AnswBUSY	- řetězec pro odpověď "BUSY"
AnswNOCARRY	- řetězec pro odpověď "NO CARRIER"
ESCStr	- řetězec pro softwarový přechod do příkazového režimu
ATH	- řetězec pro zavěšení linky - HangUp

```
tStrATCmd = string[40];
```

Typ pro omezení délky řetězce pro standardní AT příkazy a jejich odpovědi.

4.5. Struktura bufferu pro vyslání příkazu do modemu

```
pAtCmdBuff = ^tAtCmdBuff;
tAtCmdBuff = record
  Delay1,
  Delay2   : longint;
  TimeOut  : longint;
  CmdStr   : string;
  RecStr   : string;
end;
```

Typ **tAtCmdBuff** svou strukturou definuje formát bufferu dat pro vyslání příkazu do modemu metodou **ChSetBinParam** s kódy příkazů **cmd_SendCmd** až **cmd_RecData**. Položka **CmdStr** obsahuje řetězec znaků (příkazy či data) vysílaných do modemu. Položka **Delay1** určuje pauzu v milisekundách před začátkem vysílání. Položka **Delay2** určuje pauzu v milisekundách po dokončení vysílání. Položka **TimeOut** definuje maximální dobu v milisekundách pro čekání na odpověď po pauze Delay2. Položka **RecStr** obsahuje řetězec znaků přijatých jako odpověď.

5. Objekty

5.1. tChnMod

5.1.1. Položky

```
CH_Cislo    : Byte;
```

Položka **CH_Cislo** určuje číslo sériového kanálu. Jen pro vnitřní použití a dědičné objekty.

```
CH_Master   : Boolean;
```

Položka **CH_Master** určuje, má-li se modem chovat jako Master (volající) nebo jako Slave (volaný). O vlastnostech stanice Master/Slave viz v popisu metody ChSetOneParam.

```
CH_Tick     : Boolean;
```

Položka **CH_Tick** je určena jen pro vnitřní použití, pro určení, zda je vykonávaná činnost automatu kanálu.

```
CH_RTick    : Boolean;
```

Položka **CH_RTick** je určena jen pro vnitřní použití, pro určení, zda je vykonávaná činnost přijímacího automatu.

```
CH_STick    : Boolean;
```

Položka **CH_STick** je určena jen pro vnitřní použití, pro určení, zda je vykonávaná činnost vysílacího automatu.

```
CH_CDShutDown: Boolean;
```

Položka **CH_CDShutDown** je určena jen pro vnitřní použití, pro určení, že nastal výpadek modemového signálu CD (RLSD).

```
CH_ATCmdBuff : pATCmdBuff;
```

Položka **CH_ATCmdBuff** definuje ukazatel na pomocný buffer pro miniautomat vysílání jednotlivých AT příkazů do modemu.

```
CH_FlHwI     : Boolean;
```

Položka **CH_FlHwI** definuje příznak používání hardwarové inicializace modemu při navazování spojení.

```
CH_FlHwH     : Boolean;
```

Položka **CH_FlHwH** definuje příznak používání hardwarového zavěšování (HangUp) modemu při rušení spojení.

```
CH_Delay     : array [tDelays]   of longint;
```

Pole **CH_Delay** definuje seznam různých časových pauz v milisekundách při komunikaci s modemem.

```
CH_TimeOut   : array [tTimeOuts] of longint;
```

Pole **CH_TimeOut** definuje seznam různých timeoutů v milisekundách při komunikaci s modemem.

```
CH_Str       : array [tStrAnsw]  of tStr40;
```

Pole **CH_Str** definuje seznam řídících a inicializačních řetězců (příkazů) při komunikaci s modemem.

```
CH_Rep       : Byte;
```

Položka **CH_Rep** definuje počet opakovaných pokusů při navazování spojení.

```
CH_Cnt       : Byte;
```

Položka **CH_Cnt** definuje zbývající počet opakování při navazování spojení.

```
CH_FlowCntrl: byte;
```

Položka **CH_FlowCntrl** definuje způsob řízení toku dat. Přípustné hodnoty této položky jsou:

0 - Žádné řízení toku dat.
1 - CTS/RTS - před vysíláním se čeká na signál CTS.
2 - DSR/DTR - před vysíláním se čeká na signál DSR.

```
CH_Char_CR   : char;
```

Položka **CH_Char_CR** definuje kód znaku pro Enter (CR).

```
CH_Char_LF   : char;
```

Položka **CH_Char_LF** definuje kód znaku pro odřádkování (LF).

```
CH_Char_BS   : char;
```

Položka **CH_Char_BS** definuje kód znaku pro BackSpace (BS).

```
CH_Ctrl2     : tCHState;
```

Položka **CH_Ctrl2** je určena jen pro vnitřní použití, pro uložení aktuálního mezistavu vnitřního automatu kanálu.

```
CH_Ctrl2Save : tCHState;
```

Položka **CH_Ctrl2Save** je určena jen pro vnitřní použití, pro uložení návratového mezistavu vnitřního automatu kanálu.

```
CH_SCtrlHW   : tCHState;
```

Položka **CH_SCtrl2** je určena jen pro vnitřní použití, pro uchování aktuálního stavu pomocného vysílacího automatu.

5.1.2. Všeobecně veřejné metody

5.1.2.1. Init konstruktor

```
constructor Init;
```

Konstruktor **Init** slouží k vytvoření a inicializaci instance komunikačního objektu. Ve svém těle nejdříve zavolá zděděný konstruktor **Init** (inherited Init) z rodičovského objektu tChnVirt a poté inicializuje položky objektu. Tělo konstruktoru vypadá následovně:

```
inherited Init;
CH_Type     := cName;
CH_Name     := CH_Type;
CH_NumName  := ChNumName(CH_Type);
CH_Cislo    := 1;
CH_FlHWI    := true;
CH_FlHWH    := true;
CH_Rep      := 3;
CH_Cnt      := 0;
CH_SCtrl    := CHS_SendReady;
CH_SCtrlHW  := CHS_SendReady;
CH_Ctrl2    := CHS_Close;
CH_Ctrl2Save:= CH_Ctrl2;
CH_Master   := False;
CH_RBuffAT  := nil;
CH_MRBuffAT := 0;
CH_Tick     := false;
CH_STick    := false;
CH_RTick    := false;
CH_CDShutDown:=false;
CH_FlowCntrl:= 0;
CH_Char_CR  := #$0D;
CH_Char_LF  := #$0A;
CH_Char_BS  := #$08;
CH_Delay[ESC]          :=02000;
CH_Delay[HWIni0]       :=02000;
CH_Delay[HWIni1]       :=05000;
CH_Delay[HWHangUp0]    :=02000;
CH_Delay[HWHangUp1]    :=05000;
CH_Delay[AfterI1S]     :=02000;
CH_Delay[AfterI2S]     :=00500;
CH_Delay[AfterI3S]     :=00500;
CH_Delay[Busy]         :=60000;
CH_TimeOut[SendAtCmd] :=00500;
CH_TimeOut[CmdAnsw]    :=05000;
CH_TimeOut[DialAnsw1] :=10000;
CH_TimeOut[DialAnsw2] :=05000;
CH_TimeOut[DialAnsw3] :=05000;
CH_Str[DialPNS]        := 'ATDP';
CH_Str[DialPNN]        := '';
CH_Str[Init1S]         := 'ATZ';
CH_Str[Init2S]         := '';
CH_Str[Init3S]         := '';
CH_Str[AnswOK]         := 'OK';
CH_Str[AnswERROR]      := 'ERROR';
CH_Str[AnswCONNECT]    := 'CONNECT';
CH_Str[AnswBUSY]       := 'BUSY';
CH_Str[AnswNOCARRY]    := 'NO CARRIER';
CH_Str[EscStr]         := '+++';
CH_Str[ATH]            := 'ATH';
New(CH_ATCmdBuff);
with CH_ATCmdBuff^ do
begin
  CmdStr :='';
  Delay1 :=0;
  Delay2 :=0;
  TimeOut:=0;
  RecStr :='';
end;
```

5.1.2.2. ChInitParam konstruktor

```
constructor ChInitParam(const S: ParamStr);
```

Konstruktor **ChInitParam** slouží ke zkrácenému vytvoření instance komunikačního objektu s definovaným nastavením parametrů kanálu. Ve svém těle nejprve volá konstruktor **Init** a poté metodu **ChSetParam**.

5.1.2.3. Done destruktor

```
destructor Done;
```

Destruktor **Done** slouží ke zrušení instance komunikačního objektu. Pokud jsou v paměti alokovány přijímací a vysílací buffery, budou odstraněny z paměti a poté je zavolán zděděný destruktor **Done** (inherited Done) z rodičovského objektu tChnVirt.

5.1.2.4. ChSetOneParam funkce

```
function  ChSetOneParam(const S: tWordString; var CmdL: tCmd)
          : tChResult;
```

Metoda **ChSetOneParam** slouží k dekódování a nastavení jednoho konkrétního parametru, který je zadán v parametru S. Tato metoda se volá v aplikaci prostřednictvím metody **ChSetParam**. Metoda **ChSetOneParam** komunikačního objektu tChnMod dekóduje tyto parametry:

MAS=MASTER/SLAVE

Parametrem **MAS** ("Master or Slave") se určí, zda má být modem v postavení Master (volající) nebo Slave (volaný).

Významy postavení Master/Slave:

Stanice Master - při navazování spojení započatém zavoláním metody **ChConnect** a v jeho průběhu se modem patřičně nainicializuje a vytáčí číslo protější stanice. Stav automatu kanálu **CHS_Connect** nastane tehdy, pokud protější stanice zvedne sluchátko a naváže s volajícím modemem spojení (nastaví se modemový signál CD - Carrier Detect). Dále může probíhat přenos dat pomocí metod **ChSend** a **ChReceive** (Master stanice by měla začít vysílat jako první). Při ztrátě spojení (signálu CD) se snaží toto spojení obnovit. Jakékoliv přijaté a do ztráty spojení nevyzvednuté znaky jsou zapomenuty a pokud se před ztrátou spojení vysílala nějaká zpráva, bude opět odvysílána po jeho navázání.

Stanice Slave - při navazování spojení započatém zavolání metody **ChConnect** a v jeho průběhu se modem patřičně nainicializuje, ale číslo protější stanice nevytáčí (sama se nesnaží nikam dovolat, naopak čeká, až ji někdo zavolá). Jedním z kroků inicializace modemu by proto mělo být jeho nastavení tak, aby automaticky zvedl linku na příchozí volání. Stav automatu kanálu **CHS_Connect** nastane však ihned po inicializaci a nastavení modemu, ale skutečný fyzický stav navázání spojení nastane, až někdo modemu zavolá a naváže s ním spojení (toto fyzické spojení ošetřují metody vysílání a příjmu dat sami). Po dosažení stavu automatu kanálu **CHS_Connect** může následovat přenos dat pomocí metod **ChSend** a **ChReceive** (Slave stanice by měla nejdříve něco přijmou a až poté vyslat odpověď). Při ztrátě spojení Slave stanice toto spojení neobnovuje (to musí zajistit stanice Master).

LRB=Size

Parametrem **LRB** ("Length of Receive Buffer") se určí velikost vstupního kruhového vyrovnávacího bufferu pro data. Buffer je alokován na heapu a každá jeho položka zaujímá v paměti prostor o velikosti 2 byte (přijatý znak a status). Platí, čím větší je vstupní buffer, tím více znaků dokáže udržet, aniž by byly znaky metodami **ChReceive** a **ChReceiveChar** odebírány. Velikost bufferu je shora omezena na 32750. Je proto nutno volit kompromis mezi velikostí bufferu a periodou, kterou přijaté znaky odebíráme.

ARB=Size

Parametrem **ARB** ("Length of AT Commands Receive Buffer") se v této verzi knihovny již nepoužívá a je zde pouze pro kompatibilitu se staršími aplikacemi

HWI=ON/OFF

Parametrem **HWI** ("Flag of HardWare Init") se nastaví příznak použití hardwarové inicializace modemu při navazování spojení.

HWH=ON/OFF

Parametrem **HWH** ("Flag of HardWare HangUp") se nastaví příznak použití hardwarového zavěšení (HangUp) modemu při ukončování navázaného spojení.

REP=Count

Parametrem **REP** ("Number of Repetitions") se nastaví počet pokusů při navázování spojení. Toto nastavení má smysl zejména pro zapojení modemu jako Master.

SFC=aaa

Parametrem **SFC** („SoftWare Flow Control“) se nastaví způsob řízení toku dat (položka **CH_FlowCntrl**). Parametr aaa může nabývat těchto hodnot:
0 - Žádné řízení toku dat.
1 - CTS/RTS - před vysíláním se čeká na signál CTS.
2 - DSR/DTR - před vysíláním se čeká na signál DSR.

I1S='sss'

Parametrem **I1S** ("First Init Command") se nastaví první inicializační řetězec AT příkazů **CH_Str[Init1S]** vysílaný do modemu při navazování spojení.

I2S='sss'

Parametrem **I2S** ("Second Init Command") se nastaví druhý inicializační řetězec AT příkazů **CH_Str[Init2S]** vysílaný do modemu při navazování spojení.

I3S='sss'

Parametrem **I3S** ("Third Init Command") se nastaví třetí inicializační řetězec AT příkazů **CH_Str[Init3S]** vysílaný do modemu při navazování spojení.

PNS='sss'

Parametrem **PNS** ("Phone Number String") se nastaví řetězec **CH_Str[DialPNS]**, který obsahuje příkaz pro vytáčení čísla protější stanice při navazování spojení. Toto nastavení má smysl zejména pro zapojení modemu jako Master.

PNN='sss'

Parametrem **PNN** ("Phone Number") se nastaví řetězec **CH_Str[DialPNN]**, který obsahuje volané telefonní číslo cílové stanice při navazování spojení. Toto nastavení má smysl zejména pro zapojení modemu jako Master. Pokud se volá z pobočkové ústředny (např. musí se volat přes 0), je nutné vložit za vytáčenou první číslici pauzu, která se do řetězce zadává jako čárka např: PNN='0,023112424'.

OKS='sss'

Parametrem **OKS** ("OK String") se nastaví řetězec **CH_Str[AnswOK]** pro odpověď "OK".

ERS='sss'

Parametrem **ERS** ("ERROR String") se nastaví řetězec **CH_Str[AnswERROR]** pro odpověď "ERROR".

COS='sss'

Parametrem **COS** ("CONNECT String") se nastaví řetězec **CH_Str[AnswCONNECT]** pro odpověď "CONNECT".

BUS='sss'

Parametrem **BUS** ("BUSY String") se nastaví řetězec **CH_Str[BUSY]** pro odpověď "BUSY".

NCS='sss'

Parametrem **NCS** ("NO CARRIER String") se nastaví řetězec **CH_Str[NOCARRY]** pro odpověď "NO CARRIER".

SHS='sss'

Parametrem **SHS** ("SoftWare HangUp String") se nastaví řetězec **CH_Str[EscStr]** pro softwarový přechod do příkazového režimu modemu.

HUS='sss'

Parametrem **HUS** ("HANGUP String") se nastaví řetězec **CH_Str[ATH]** pro zavěšení linky.

DES=llll

Parametrem **DES** ("Delay for Esc") se nastaví pauza **CH_Delay[ESC]** před a po vyslání řetězce pro softwarový přechod do příkazového režimu modemu při zavěšování. Parametr llll se zadává v milisekundách.

DIL=llll

Parametrem **DIL** ("Delay for Init0") se nastaví pauza **CH_Delay[HWIni0]** pro nulování modemového signálu DTR při hardwarové inicializaci modemu. Parametr llll se zadává v milisekundách.

DIH=llll

Parametrem **DIH** ("Delay for Init1") se nastaví pauza **CH_Delay[HWIni1]** pro nastavení modemového signálu DTR při hardwarové inicializaci modemu. Parametr llll se zadává v milisekundách.

DHL=llll

Parametrem **DHL** ("Delay for HangUp0") se nastaví pauza **CH_Delay[HWHangUp0]** pro nulování modemového signálu DTR při hardwarovém zavěšování linky při rušení spojení. Parametr llll se zadává v milisekundách.

DHH=llll

Parametrem **DHH** ("Delay for HangUp1") se nastaví pauza **CH_Delay[HWHangUp1]** pro nastavení modemového signálu DTR při hardwarovém zavěšování linky při rušení spojení. Parametr llll se zadává v milisekundách.

DI1=llll

Parametrem **DI1** ("Delay After First Init") se nastaví pauza **CH_Delay[AfterI1S]** po vyslání prvního inicializačního řetězce do modemu. Parametr llll se zadává v milisekundách.

DI2=llll

Parametrem **DI2** ("Delay After Second Init") se nastaví pauza **CH_Delay[AfterI2S]** po vyslání druhého inicializačního řetězce do modemu. Parametr llll se zadává v milisekundách.

DI3=llll

Parametrem **DI3** ("Delay After Third Init") se nastaví pauza **CH_Delay[AfterI3S]** po vyslání třetího inicializačního řetězce do modemu. Parametr llll se zadává v milisekundách.

DBU=llll

Parametrem **DBU** ("Delay If Busy") se nastaví pauza **CH_Delay[Busy]** před opětovným vytáčením čísla při obsazené lince. Parametr llll se zadává v milisekundách.

QAT=llll

Parametrem **QAT** ("TimeOut for Sending AT Command") se nastaví maximální doba **CH_TimeOut[SendATCmd]** pro vysílání AT příkazu do modemu. Parametr llll se zadává v milisekundách.

QCA=llll

Parametrem **QCA** ("TimeOut for Answer After Command") se nastaví maximální doba **CH_TimeOut[CmdAnsw]** pro čekání na odpověď z modemu po vyslání AT příkazu do modemu. Parametr llll se zadává v milisekundách.

QD1=llll

Parametrem **QD1** ("TimeOut for First Answer After Dial") se nastaví maximální doba **CH_TimeOut[DialAnsw1]** pro čekání na první odpověď z modemu při vyslání příkazu pro vytáčení čísla. Parametr llll se zadává v milisekundách.

QD2=llll

Parametrem **QD2** ("TimeOut for Second Answer After Dial") se nastaví maximální doba **CH_TimeOut[DialAnsw2]** pro čekání na druhou odpověď z modemu při vyslání příkazu pro vytáčení čísla. Parametr llll se zadává v milisekundách.

QD3=llll

Parametrem **QD3** ("TimeOut for Third Answer After Dial") se nastaví maximální doba **CH_TimeOut[DialAnsw3]** pro čekání na třetí odpověď z modemu při vyslání příkazu pro vytáčení čísla. Parametr llll se zadává v milisekundách.

5.1.2.5. ChGetParam funkce

```
function  ChGetParam(const S: TParamStr): TParamStr;
```

Metoda **ChGetParam** navrací nastavené hodnoty parametrů komunikačního objektu. Nejprve vrátí nastavení parametrů rodičovského komunikačního objektu tChnVirt a poté k nim připojí seznam svých parametrů. Seznam parametrů je uveden výše u popisu metody **ChSetOneParam**.

5.1.2.6. ChSetBinParam procedura

```
procedure ChSetBinParam (NumName: Word; Code: Word; Param: longint);
```

Metoda **ChSetBinParam** provede nastavení parametrů v binárním tvaru, popřípadě provedení určitých akcí. Parametrem Code je kód pro určení nastavovaného parametru či prováděné akce (viz. Kódy příkazů pro metodu ChSetBinParam).

5.1.2.7. ChGetBinParam funkce

```
function  ChGetBinParam (NumName: Word; Code: Word): longint;
```

Metoda **ChGetBinParam** vrátí nastavení parametrů v binárním tvaru, popřípadě provedení určité akce s vrácením stavu či výsledku dané akce. Parametrem Code je kód pro určení čteného parametru či statusu prováděné akce (viz. Kódy příkazů pro metodu ChGetBinParam):

Příklad použití metod **ChSetBinParam** a **ChGetBinParam**:
Ukázkový příklad jak vyslad AT příkaz do modemu s čekáním na odpověď.

```
write('Test komunikace Modem-Computer: ');
ModNumName:=ChNumName('MOD');
                    {ModNumName je pomocná prom. typu tChName}
with AtCmdBuff do   {AtCmdBuff je pomocná prom. typu tAtCmdBuff}
  begin             {nastavení parametrů}
    Str:='AT';
    Delay1:=0;
    Delay2:=0;
    TimeOut:=Ch_TimeOut[CmdAnsw];
  end;
ChSetBinParam(ModNumName, cmd_CmdAnsw, longint(@AtCmdBuff));
repeat
until ChGetBinParam(ModNumName, cmd_CmdReady) = byte(SMCS_Ready);
writeln('Vysledek: ',ChGetBinParam(ModNumName, cmd_CmdResult));
```

5.1.2.8. ChOpen procedura

```
procedure ChOpen;
```

Metoda **ChOpen** nastaví technické vybavení komunikačního kanálu prostřednictvím metody **ChInitHW** (viz. níže), a pokud nastavení proběhlo v pořádku, způsobí přechod kanálu do stavu **CHS_Open**.

5.1.2.9. ChClose procedura

```
procedure ChClose;
```

Metoda **ChClose** uzavře komunikační kanál provedením deinicializace technického vybavení prostřednictvím metody **ChDoneHW** (viz. níže) a způsobí přechod do stavu **CHS_Close**. Lze opětovně volat metodu **ChOpen**.

5.1.2.10. ChConnect procedura

```
procedure ChConnect;
```

Před voláním této metody musí být kanál ve stavu **CHS_Open**. Metoda **ChConnect** provede započetí navázání spojení. Na základě následovném opakovaném volání metody **ChTick** (například prostřednictvím metod **ChReady** nebo **ChState**), přejde kanál po čase do stavu **CHS_Connect**. To znamená, že v případě Master

stanice došlo k navázání fyzického spojení s volanou stanicí a v případě Slave stanice se čeká na fyzické navázání spojení od volající stanice, ale mohou se již volat metody pro příjem a vysílání zpráv, které fyzické navázání spojení sami ošetří. V tomto stavu je možno přijímat data z komunikační linky, naopak je možno požadovaná data odvysílat.

5.1.2.11. ChDisConnect procedura

```
procedure ChDisConnect;
```

Metoda **ChDisConnect** provede započetí ukončí navázaného komunikační spojení. Na základě následovném opakovaném volání metody **ChTick** (například prostřednictvím metod **ChReady** nebo **ChState**), přejde kanál po čase do stavu **CHS_DisConnect**. Je přerušen příjem a vysílání datových zpráv a lze opětovně volat metodu **ChConnect**.

5.1.2.12. ChTick procedura

```
procedure ChTick;
```

Metoda **ChTick** způsobí provedení jednoho či více kroků kanálových automatů. Zabezpečuje udržování navázaného spojení s protější stanicí, a proto ji je nutné periodicky volat. Je rovněž automaticky volána v metodách **ChReady, ChState, ChSendReady** a **ChReceiveReady**.

5.1.2.13. ChSendTick procedura

```
procedure ChSendTick;
```

Metoda **ChSendTick** způsobí provedení jednoho či více kroků vysílacích automatů. Je nutné ji periodicky volat během vysílání. Je rovněž automaticky volána v metodě **ChSendReady**. Pokud se v průběhu vysílání přeruší spojení s protější stanicí, metoda zajistí nové navázání spojení a, v případě nastavení stanice jako Master, odvysílání celé zprávy znova. Pokud je stanice nastavena jako Slave, zpráva se znova vysílat nebude.

5.1.2.14. ChReceiveTick procedura

```
procedure ChReceiveTick;
```

Metoda **ChReceiveTick** způsobí provedení jednoho či více kroků přijímacích automatů. Je nutné ji periodicky volat během příjmu. Je rovněž automaticky volána v metodě **ChReceiveReady**. Pokud se v průběhu vysílání přeruší spojení s protější stanicí, metoda zajistí nové navázání spojení.

5.1.2.15. ChSend procedura

```
procedure ChSend(Buff: Pointer; Len: Word);
```

Pokud je kanál ve stavu **CHS_Connect**, způsobí metoda **ChSend** započetí vysílání zprávy délky `Len` uložené na adrese určené ukazatelem `Buff`. Pokud je parametr Len = 0, nebude se vysílat žádná zpráva. Po volání této metody by mělo následovat volání metody **ChSendReady** s testem na **CHS_SendReady** (čekací smyčka do odvysílání zprávy). Pokud je modem nastaven jako Slave a není ve fyzickém spojení s protější stanicí, žádné vysílání se neprovede (metoda

ChSendReady vrátí stav CHS_SendReady a metoda ChSendResult vrátí výsledek res_Ok).

5.1.2.16. ChSendReady funkce

```
function  ChSendReady: TCHState;
```

Metoda **ChSendReady** způsobí provedení kroku vysílacího automatu na základě volání metody **ChSendTick**. Jako svoji funkční hodnotu vrátí aktuální stav automatu vysílače komunikačního kanálu, který je uložen v položce **CH_SCtrl**. Pokud kanál není ve stavu **CHS_Connect**, vrací metoda stav **CHS_SendNoReady**.

5.1.2.17. ChSendFlush procedura

```
procedure ChSendFlush;
```

Pokud je kanál ve stavu **CHS_Connect** způsobí metoda **ChSendFlush** ukončení vysílání a přechod automatu vysílače do stavu **CHS_SendReady**.

5.1.2.18. ChReceiveReady funkce

```
function  ChReceiveReady: TCHState;
```

Metoda **ChReceiveReady** způsobí provedení kroku automatu přijímače na základě volání metody **ChReceiveTick**. Jako svoji funkční hodnotu vrátí aktuální stav přijímacího automatu, který je uložen v položce **CH_RCtrl**. Pokud kanál není ve stavu **CHS_Connect**, vrátí metoda stav **CHS_ReceiveNoReady**.

5.1.2.19. ChReceiveChar funkce

```
function  ChReceiveChar: Byte;
```

Pokud je kanál ve stavu **CHS_Connect** a v přijímacím bufferu jsou přijata nějaká data, navrací metoda **ChReceiveChar** jeden přijatý znak z přijímacího bufferu a nastaví výsledek operace přijímače na status tohoto přijatého znaku. Metodu **ChReceiveChar** je možno volat pouze ve stavu automatu přijímače **CHS_ReceiveReady**, proto před voláním této metody musí předcházet volání metody **ChReceiveReady** s testem na stav **CHS_ReceiveReady**, jinak v případě nepřijetí žádného znaku skončí volání metody **ChReceiveChar** chybou.

5.1.2.20. ChReceive procedura

```
procedure ChReceive(var Len: Word);
```

Pokud je kanál ve stavu **CHS_Connect** a je nadefinován buffer pro uložení přijaté zprávy (metodou **ChReceiveBuffer**), provede metoda **ChReceive** přijetí zprávy a její uložení do přijímacího bufferu. V proměnné `Len` navrací délku přijaté zprávy. Ve svém těle volá metodu **ChReceiveChar** (pro přijetí jednoho znaku zprávy) tak dlouho, dokud je přijímač ve stavu **CHS_ReceiveReady**.

5.1.2.21. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

Metoda **ChReceiveFlush** způsobí vyprázdnění přijímacích bufferů a nastaví stav přijímače kanálu **CH_RCtlr** na stabilní stav **CHS_ReceiveNoReady**.

5.1.3. Metody veřejné pouze pro dědičné objekty

Následující metody musí dědičné objekty vždy předefinovat na základě použitého hardware. Tyto metody by se v aplikaci používat neměly.

5.1.3.1. ChInitHW procedura

```
procedure ChInitHW;
```

Metoda **ChInitHW** je určena pro hardwarovou inicializaci (nastavení) komunikačního kanálu.

5.1.3.2. ChDoneHW procedura

```
procedure ChDoneHW;
```

Metoda **ChDoneHW** je určena pro hardwarové uzavření komunikačního kanálu.

5.1.3.3. ChSendHW procedura

```
procedure ChSendHW(Buff: Pointer; Len: Word);
```

Metoda **ChSendHW** je určena pro fyzické vysílání zprávy do komunikační linky.

5.1.3.4. ChSendReadyHW funkce

```
function  ChSendReadyHW: TChState;
```

Metoda **ChSendReadyHW** vrátí stav (je vysláno, dosud se vysílá apod.) fyzického vysílání zprávy.

5.1.3.5. ChSendFlushHW procedura

```
procedure ChSendFlushHW;
```

Metoda **ChSendFlushHW** je určena pro fyzické ukončení vysílání a vyprázdnění vysílacího bufferu.

5.1.3.6. ChSetModem procedura

```
procedure ChSetModem(S,R: Byte);
```

Metoda **ChSetModem** je určena pro nastavení modemových signálů. Parametr S určuje masku signálů, které se mají nastavit, a parametr R určuje masku signálů, které se mají vynulovat.

5.1.3.7. ChGetModem funkce

```
function  ChGetModem: Byte;
```

Metoda **ChGetModem** je určena pro navrácení nastavení modemových signálů. Výsledek je byte, kde jednotlivé bity určují jednotlivé modemové signály.

5.2. tAddChnMod

Typ **tAddChnMod** je typem objektu, který slouží k definování prvku v seznamu správců komunikačních objektů (tzv. správce komunikačního objektu tChnMod v seznamu správců). Objekt tAddChnMod je dědicem od rodičovského objektu **tAddChnVirt**.

5.2.1. Metody

5.2.1.1. ChInit funkce

```
function ChInit: pChnVirt;
```

Metoda **ChInit** slouží k vytvoření instance komunikačního objektu tChnMod a ukazatel na instanci tohoto objektu vrací jako svoji funkční hodnotu.

6. Příklad

Příklad použití komunikační jednotky ChnMod je uveden v příručce **ChnComM** nebo **ChnV40M**.