# ChnSMS

## JEDNOTKA SÉRIOVÉ KOMUNIKACE POMOCÍ KRÁTKÝCH ZPRÁV (SMS) PŘES GSM MODEM

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.<br>Doplněné konstanty cAtCmdGSM_GetManufID a cAtCmdGSM_GetModelID.<br>Doplněná proměnná CH_Mem.<br>Doplněn typ tSMC_Ctrl. |
| | | | | |

## 1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky sériové komunikace pomocí krátkých zpráv (SMS) přes modem.

## 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 ChnVirt popisujícím rozhraní svých potomků, ChnTypes a uSMS.

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 ChnSMS definuje komunikační objekt **tChnSMS**, který je dědicem od rodičovského komunikačního objektu tChnVirt. Instance objektu tChnSMS representuje vyšší komunikační vrstvu v komunikačním kanálu. Objekt tChnSMS definuje automaty zajišťující inicializaci GSM modemu pro příjem a vysílání krátkých textových zpráv (SMS) a automaty pro vlastní příjem a vysílání SMS zpráv přes GSM modem. Fyzický přenos dat mezi GSM modemem a počítačem je zajištěn prostřednictvím nižší komunikační vrstvy. Určení, přes jakou fyzickou komunikační vrstvu bude komunikace probíhat, je voleno až parametry nastavovací metody ChSetParam.

Knihovna používá pro přenos SMS zpráv PDU formát dat (viz níže). Textový formát přenosu SMS zpráv není zatím zcela naimplementován a odladěn.

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

Protože je objekt **tChnSMS** 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 jednotkách **ChnTypes** a **uSMS**.

# 4. Terminologie a zkratky

SM (SMS) - Short Message (Short Message Service)
: Označení pro krátkou textovou zprávu.

MS, TA - Mobile Station
: Označení pro přenosové zařízení - GSM modem

TE - Terminal
: Označení pro zařízení Master, ke kterému je GSM modem připojen (zpravidla PC).

SMSC (SC) - Short Message Service Center (Service Center).
: Označení pro operátora pro přenos krátkých textových zpráv (SMS).

PDU - Protocol Data Unit
: Označení formátu vysílaných/přijímaných dat mezi TA a TE. Existuje ještě textový formát.

# 5. Popis konstant a typů

```
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  = 'SMS';
```

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

```
  tSMC_Ctrl = (
    SMCS_Beg,          { pocatek modem command cyklu }
    SMCS_Delay1,       { prodleva pred vysilanim commandu }
    SMCS_FlowControl, { ceka se na osetreni HW toku dat }
    SMCS_Send,         { vyslani commandu }
    SMCS_SendDelay,    { test na dokonceni vysilani }
    SMCS_Delay2,       { prodleva po odvysilani }
    SMCS_Rec,          { cteni odpovedi }
    SMCS_RecEnd,       { konec cteni, dekodovani semantiky zpravy }
    SMCS_Ready         { konec modem command cyklu - stabilni stav }
    );
```

Typ tSMC_Ctrl vnitřní udává stav automatu pro vyslání jednoho AT příkazu s čekáním na odpověd.

```
res_Err_WrongPIN = $20;
```

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

## 5.1. 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.

## 5.2. Kódy příkazů pro metodu 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 ResGSM_XXX (viz níže).

## 5.3. 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 výše).

```
ResGSM_Ok           =  0;
```

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

```
ResGSM_ChannelNil   =  2;
```

Konstanta **ResGSM_ChannelNil** definuje chybu, že nelze předat řízení komunikačnímu objektu podřízené vrstvy, který byl zřetězen v komunikačním kanálu za objekt tChnSMS.

```
ResGSM_Error        = 12;
```

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

```
ResGSM_NoAnswer     = 14;
```

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

```
ResGSM_Answer       = 19;
```

Konstanta **ResGSM_Answer** znamená, že modem hlásí jinou odpověď.

```
ResGSM_LongAnswer   = 20;
```

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

```
ResGSM_SHwErr       = 21;
```

Konstanta **ResGSM_SHwErr** znamená, že nastala nízká chyba při vysílání.

```
ResGSM_RHwErr       = 22;
```

Konstanta **ResGSM_RHwErr** znamená, že nastala nízká chyba při příjmu.

```
ResGSM_RTimeOut     = 23;
```

Konstanta **ResGSM_RTimeOut** znamená, že nastal timeout při příjmu.

```
ResGSM_STimeOut     = 24;
```

Konstanta **ResGSM_STimeOut** znamená, že nastal timeout při vysílání.

## 5.4. Výčtové typy pro určení položek polí

```
tDelays = (ESC, AfterI1S, AfterI2S, AfterI3S, BeforPIN, AfterPIN,
           BeforMemSto, BeforSMSData);
```

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

| | |
|---|---|
| ESC | - pauza před a po vyslání řetězce ESC (zpravidla '+++') v SoftWarovém HangUpu (přechod z datového režimu modemu do příkazového) |
| AfterI1S | - pauza po vyslání 1. inicializačního řetězce |
| AfterI2S | - pauza po vyslání 2. inicializačního řetězce |
| AfterI3S | - pauza po vyslání 3. inicializačního řetězce |

BeforPIN - pauza před zadáním SIM PINu do GSM modemu
AfterPIN - pauza po zadání správného SIM PINu do GSM modemu
BeforMemSto - pauza před dotazem na obsazení jednotlivých pamětí pro SMS zprávy
BeforSMSData - pauza před vysláním vlastních dat vysílané SMS zprávy

```
tTimeOuts = (SendAtCmd, CmdAnsw, AnswPIN, AnswSMS, SendSMS);
```

Typ **tTimeOuts** definuje výčet různých timeoutů během komunikace s GSM modemem. Obsahuje následující položky:

SendAtCmd - timeout pro vysílání zprávy do modemu
AnswPIN - timeout pro čekání na odpověď při zadání SIM PINu
AnswSMS - timeout pro čekání na přijetí dat SMS zprávy
SendSMS - timeout pro čekání na odpověď při zadání vlastních dat SMS vysílané zprávy
CmdAnsw - timeout pro odpověď na ostatní příkazy

```
tStrAnsw = (Init1S, Init2S, Init3S, AnswOK, AnswERROR, ESCStr);
```

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

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"
ESCStr - řetězec pro softwarový přechod do příkazového režimu

## 5.5. Řetězcové typy

```
tStrATCmd       = String[40];
```

Typ řetězce pro standardní AT příkazy.

```
tStrSimPin      = String[ 4];
```

Typ řetězce pro SIM PIN.

```
tStrSimPuk      = String[12];
```

Typ řetězce pro SIM PUK.

## 5.6. Struktura bufferu pro vyslání příkazu do modemu

```
pAtCmdBuff = ^tAtCmdBuff;
tAtCmdBuff = record
  Mode    : byte;
  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 **Mode** obsahuje kód způsobu provádění příkazu (shodná s hodnotou parametru Code metody ChSetBinParam). 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 případnou odpověď po pauze Delay2. Položka **RecStr** obsahuje řetězec znaků přijatých dat či odpovědi na vyslaný příkaz.

## 5.7. Konstanty základních AT příkazů a odpovědí pro SMS zprávy

```
cAtCmdGSM_SetPin     = '+CPIN';
```
Řetězec pro dotaz (jaký PIN modem požaduje) či nastavení PINu.
```
cAtAnsGSM_AnswSimPin = 'SIM PIN';
```
Řetězec pro odpověď od GSM modemu pro požadavek zadání SIM PINu.
```
cAtAnsGSM_AnswSimPuk = 'SIM PUK';
```
Řetězec pro odpověď od GSM modemu pro požadavek zadání SIM PUKu.
```
cAtAnsGSM_AnswReady  = 'READY';
```
Řetězec pro odpověď od GSM modemu, že nevyžaduje zadat již žádný PIN.
```
cAtCmdSMS_SetSCNum   = '+CSCA';
```
Řetězec pro dotaz či nastavení čísla operátora (SMSC).
```
cAtCmdSMS_SetMode    = '+CMGF';
```
Řetězec pro dotaz či nastavení textového (TXT) či protokolového (PDU) módu.
```
cAtCmdSMS_GetMemSto  = '+CPMS';
```
Řetězec pro dotaz obsazení pamětí SMS zprávami.
```
cAtCmdSMS_DelMsg     = '+CMGD';
```
Řetězec pro vymazání SMS zprávy z banky paměti.
```
cAtCmdSMS_ReadMsg    = '+CMGR';
```
Řetězec pro čtení SMS zprávy z banky paměti.
```
cAtCmdSMS_WriteMsg   = '+CMGW';
```
Řetězec pro zápis SMS zprávy do banky paměti.
```
cAtCmdSMS_SendMsg    = '+CMGS';
```
Řetězec pro přímé poslání SMS zprávy.
```
cAtCmdSMS_SendMsg2   = '+CMSS';
```
Řetězec pro poslání SMS zprávy z banky paměti.
```
cAtCmdGSM_GetManufID = '+CGMI';
```
Řetězec pro čtení výrobce GSM stanice.
```
cAtCmdGSM_GetModelID = '+CGMM';
```
Řetězec pro čtení modelu GSM stanice.

# 6. Objekty

## 6.1. tChnSMS

### 6.1.1. Položky

```
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_LRMess     : Word;
```
Položka **CH_LRMess** definuje velikost záznamu přijaté SMS zprávy.
```
CH_ATCmdBuff : pATCmdBuff;
```
Položka **CH_ATCmdBuff** definuje ukazatel na pomocný buffer pro miniautomat vysílání jednotlivých AT příkazů do GSM modemu.
```
CH_SimPin     : tStrSimPin;
```
Položka **CH_SimPin** definuje řetězec pro kód SIM PIN.
```
CH_SimPuk     : tStrSimPuk;
```
Položka **CH_SimPuk** definuje řetězec pro kód SIM PUK.
```
CH_SMSCNum    : tPhoneNumStr;
```
Položka **CH_SMSCNum** definuje řetězec čísla operátora SMS zpráv.
```
CH_TextMode   : boolean;
```
Položka **CH_TextMode** definuje příznak použití textového (TXT) či protokolového (PDU) formátu pro SMS zprávy. Pokud není textový mód pro daný GSM modem dostupný, automaticky se použije protokolový.
```
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 tStrAtCmd;
```
Pole **CH_Str** definuje seznam řídících a inicializačních řetězců (příkazů) při komunikaci s modemem.
```
CH_Mem       : array [1..MaxMem] of tMemRec;
```
Pole Chn_Mem obsahuje informace o jednotlivých bankách.
```
CH_FlRecRead: boolean;
```
Položka **CH_FlRecRead** definuje příznak, že kromě nově příchozích SMS zpráv se mají přijímat i případné již přijaté (přečtené) SMS zprávy v bankách paměti.
```
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).

## 6.1.2. Metody

### 6.1.2.1. 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_SCtrl    := CHS_SendReady;
  CH_Tick     := false;
  CH_STick    := false;
  CH_RTick    := false;
  CH_LRMess   := 0;
  CH_SimPin   := '1234';
  CH_SimPuk   := '1234';
  CH_SMSCNum  := '420603052000';
  CH_TextMode := False;
  CH_FlRecRead:= False;
  CH_FlowCntrl:= 0;
  CH_Char_CR  := #$0D;
  CH_Char_LF  := #$0A;
  CH_Char_BS  := #$08;
  CH_Delay[ESC]          := 02000;
  CH_Delay[AfterI1S]     := 01500;
  CH_Delay[AfterI2S]     := 00500;
  CH_Delay[AfterI3S]     := 00500;
  CH_Delay[AfterPin]     := 03000;
  CH_Delay[BeforPIN]     := 00250;
  CH_Delay[BeforMemSto]  := 00500;
  CH_Delay[BeforSMSData]:= 00500;
  CH_TimeOut[SendAtCmd]  := 00500;
  CH_TimeOut[CmdAnsw]    := 02000;
  CH_TimeOut[AnswPIN]    := 03000;
  CH_TimeOut[AnswSMS]    := 04000;
  CH_TimeOut[SendSMS]    := 05000;
  CH_Str[Init1S]         := 'ATZ';
  CH_Str[Init2S]         := '';
  CH_Str[Init3S]         := '';
  CH_Str[AnswOK]         := 'OK';
  CH_Str[AnswERROR]      := 'ERROR';
  CH_Str[EscStr]         := '+++';
  New(CH_ATCmdBuff);
  with CH_ATCmdBuff^ do
  begin
    CmdStr :='';
    Delay1 :=0;
    Delay2 :=0;
    TimeOut:=0;
    RecStr :='';
  end;
```

### 6.1.2.2. 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**.

### 6.1.2.3. 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.

### 6.1.2.4. 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 tChnSMS dekóduje tyto parametry:

**PIN**='xxxx'
: Parametrem **PIN** se nastaví kód pro SIM PIN (položka **CH_SimPin**).

**PUK**='yyyyyy'
: Parametrem **PUK** se nastaví kód pro SIM PUK (položka **CH_SimPuk**).

**SCN**=PhoneNumber
: Parametrem **SCN** ("Service Center Number") se nastaví číslo operátora pro SMS zprávy (položka **CH_SMSCNum**).

**TXT**=ON/OFF
: Parametrem **TXT** („TXT or PDU“) se nastaví příznak používání textového (TXT) či protokolového (PDU) formátu SMS zpráv (položka **CH_FlTextMode**).

**RRM**=ON/OFF
: Parametrem **RRM** („Receive Readed Messages“) se nastaví příznak (položka **CH_FlRecRead**), že kromě nově příchozích SMS zpráv se mají přijímat i případné již přijaté (přečtené) SMS zprávy v bankách paměti.

**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 (položka **CH_Str[Init1S]**) vysílaný do modemu při navazování spojení.

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

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

**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".

**SHS**='sss'

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

**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.

**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.

**DBP**=llll

Parametrem **DBP** („Delay Before PIN“) se nastaví pauza (položka **CH_Delay[BeforPIN]**) před zadáním SIM PINu.

**DAP**=llll

Parametrem **DAP** („Delay After PIN“) se nastaví pauza (položka **CH_Delay[AfterPIN]**) po správném zadání SIM PINu.

**DBM**=llll

Parametrem **DBM** („Delay Before Read Memory“) se nastaví pauza (položka **CH_Delay[BeforMemSto]**) před dotazem na obsazení paměti pro SMS zprávy.

**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.

**QAS**=llll

Parametrem **QAS** ("TimeOut for Answer SMS Data") se nastaví maximální doba **CH_TimeOut[AnswSMS]** pro čekání příjem dat příchozí SMS zprávy. Parametr llll se zadává v milisekundách.

**QSS**=llll

Parametrem **QSS** ("TimeOut for Send SMS Data") se nastaví maximální doba **CH_TimeOut[SendSMS]** pro čekání odpověď z modemu při vyslání dat vysílané SMS zprávy. Parametr llll se zadává v milisekundách.

### 6.1.2.5. 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**.

### 6.1.2.6. 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).

### 6.1.2.7. 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 vyslat AT příkaz do modemu s čekáním na odpověď.

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

### 6.1.2.8. ChOpen procedura

```
procedure ChOpen;
```

Metoda **ChOpen** nastaví technické vybavení komunikačního kanálu prostřednictvím metod **ChOpen** a **ChConnect** podřízeného komunikačního objektu, který provede fyzickou inicializaci, a pokud nastavení proběhlo v pořádku, způsobí přechod kanálu do stavu **CHS_Open**.

### 6.1.2.9. ChClose procedura

```
procedure ChClose;
```

Metoda **ChClose** uzavře komunikační kanál provedením deinicializace technického vybavení prostřednictvím metod **ChDisConnect** a **ChClose** podřízeného komunikačního objektu, který provede fyzické uzavření, a způsobí přechod do stavu **CHS_Close**. Lze opětovně volat metodu **ChOpen**.

### 6.1.2.10. ChConnect procedura

```
procedure ChConnect;
```

Před voláním této metody musí být kanál ve stavu **CHS_Open**. Metoda **ChConnect** provede započetí nastavení GSM modemu, aby se mohl zapojit do telekomunikační sítě. Na základě následovném opakovaném volání metody **ChTick** (například prostřednictvím metod **ChReady** nebo **ChState**), přejde kanál po čase do stavu **CHS_Connect** (samozřejmě, že byl správně zadán PIN). V tomto stavu je možno přijímat a vysílat SMS zprávy.

### 6.1.2.11. 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 SMS zpráv a lze opětovně volat metodu **ChConnect**.

### 6.1.2.12. ChTick procedura

```
procedure ChTick;
```

Metoda **ChTick** způsobí provedení jednoho či více kroků kanálových automatů. Je rovněž automaticky volána v metodách **ChReady** a **ChState**.

### 6.1.2.13. 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**.

### 6.1.2.14. 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**.

### 6.1.2.15. ChState funkce

```
function  ChState: TChState;
```

Metoda **ChState** provede krok automatu komunikačního kanálu na základě volání metody **ChTick** a navrátí naposled dosažený stabilní stav komunikačního kanálu uložený v položce **CH_State**. Pomocí metody **ChState** je možno provádět test na dosažení základních stabilních stavů automatu komunikačního kanálu.

### 6.1.2.16. ChReady funkce

```
function  ChReady: TChState;
```

Metoda **ChReady** provede krok automatu komunikačního kanálu na základě volání metody **ChTick** a navrátí aktuální stav komunikačního kanálu uložený v položce **CH_Ctrl**. Také pomocí metody **ChReady** je možno provádět test na dosažení základních stabilních stavů automatu komunikačního kanálu.

### 6.1.2.17. ChSend procedura

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

Metoda **ChSend** způsobí započetí vysílání SMS zprávy na podkladu záznamu typu tSendRecord, na který ukazuje parametr Buff. Parametr Len udává délku vysílacího bufferu pro vysílání. Tento parametr není nutno správně naplnit skutečnou délkou zprávy, jelikož tuto délku lze získat automaticky podle vyplněného záznamu vysílacího bufferu. Před voláním této metody se musí záznam správně naplnit daty. Po volání této metody by mělo následovat volání metody **ChSendReady** s testem na **CHS_SendReady** (čekací smyčka do odvysílání zprávy).

### 6.1.2.18. 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**.

### 6.1.2.19. ChSendFlush procedura

```
procedure ChSendFlush;
```

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

### 6.1.2.20. 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**.

### 6.1.2.21. ChReceive procedura

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

Pokud byla přijata nějaká SMS zpráva (ChReceiveReady = CHS_ReceiveReady), provede metoda **ChReceive** přijetí celé zprávy a její uložení do přijímacího bufferu, který svou vnitřní strukturou odpovídá datům záznamu typu tRecRecord a byl definován metodou **ChReceiveBuffer**. Metoda naplní buffer pouze patřičnými položkami typu tRecRecord. Odpověď na zprávu generuje uživatel sám pomocí metody **ChSend**.

#### 6.1.2.22. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

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

## 6.2. tAddChnSMS

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

### 6.2.1. Metody

#### 6.2.1.1. ChInit funkce

```
function ChInit: pChnVirt;
```

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

# 7. Příklad

Následující příklad ukazuje způsob vyslání SMS zprávy a příjem SMS zprávy. Fyzický přenos se realizuje prostřednictvím knihovny ChnCom.

```
uses
  ChnVirt,
  ChnCom,
  uSMS,
  ChnSMS;

const
  ParamStr : tParamStr =
             'NAM=SMS I1S=''ATZ'' I2S=''ATE0'' PIN=''1234'' ' +
             'SCN=''420603052000'' SFC=CTS TXT=OFF RRM=ON DAP=3000 '+
             'NAM=COM COM=1 IRQ=4 BD=19200 BIT=8 STO=1 PAR=N '+
             'LRB=2000 DTR=ON RTS=ON';

var
  Chn   : pChnVirt;
  SMess : pSendRecord;
  RMess : pRecRecord;
  LSMess: word;
  LRMess: word;

begin
  ...
  New(SMess);
  New(RMess);
  ...
  { vytvoření instance Chn }
  Chn:=ChnCollection^.ChNewInit(ChnSMS.cName);
  with Chn^ do
  begin
    { nastavení parametrů komunikace }
    ChSetParam(ParamStr);
    if ChResult<>res_Ok then WriteLn('Chyba');
```

```
ChOpen;
repeat
  if ChResult<>res_Ok then WriteLn('Chyba');
until ChReady=CHS_Open;
if ChResult<>res_Ok then WriteLn('Chyba');
{ definování místa, kam se má přijatá zpráva uložit }
ChReceiveBuffer(RMess,SizeOf(RMess^));
if ChReceiveResult<>res_Ok then WriteLn('Chyba');
ChConnect;
repeat
  if ChResult<>res_Ok then WriteLn('Chyba');
until ChReady=CHS_Connect;
if ChResult<>res_Ok then WriteLn('Chyba');
...
{ naplnění zprávy daty }
with SMess^ do
begin
  MsgStat      := tsStoUnSent;
  PDU_Type     := $01;
  MsgReference := $00;
  PhoneNumType := cPNT_Internat;
  PhoneNum     := '420603987654';
  Protocol_ID  := cPID_SMS;
  DataCodScheme:= cDCS_Impl7Bit;
  SMS_Message  := 'Toto je text SMS zpravy';
  TypTime      := tmNoTime;
  LSMess := 0;
end;
{ vyslání SMS zprávy }
if ChSendReady=CHS_SendReady then
begin
  ChSend(SMess, LSMess);
  { čekání na odvysílání zprávy }
  repeat
    if ChSendResult<>res_Ok then WriteLn('Chyba');
  until ChSendReady=CHS_SendReady;
  if ChSendResult<>res_Ok then WriteLn('Chyba');
  ...
end;
...
{ čekání na příjem zprávy }
while not ChReceiveReady=CHS_ReceiveReady do
begin
  if ChReceiveResult<>res_Ok then WriteLn('Chyba');
end;
{ příjem zprávy }
ChReceive(LRMess);
if ChReceiveResult<>res_Ok then WriteLn('Chyba')
else
  { dekódování přijetí SMS zprávy}
  with RMess^ do
  begin
    write('Byla prijata ');
    case MsgStat of
      tsRecUnRead:writeln('nova zprava:');
      tsRecRead  :writeln('jiz drive ctena zprava:');
    end;
    writeln('Cislo odesilatele: ',PhoneNum);
    case TypTime of
      tmAbsTime:
        write  ('Abs Time  : ',DateToStr(Time,'DD.MM.YYYY'));
        writeln('  ',TimeToStr(Time,'HH:MM:SS'));
      tmRelTime:
        writeln('Rel Time  : ',Days,' dni, ',Mins,' minut');
      tmNoTime :
        writeln('No Time');
    end;
```

```
        writeln('Text zpravy : ',SMS_Message);
      end;
    ...
    { ukončení }
    ChDisConnect;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_DisConnect;
    if ChResult<>res_Ok then WriteLn('Chyba');
    ChClose;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_Close;
    if ChResult<>res_Ok then WriteLn('Chyba');
  end;
  { zrušení instance Chn }
  Dispose(Chn,Done);
  ...
end.
```