

ChnGSM

JEDNOTKA SÉRIOVÉ KOMUNIKACE POMOCÍ GSM 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>

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 popsáno. 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)

Upozornění:

V dokumentu použité názvy výrobků, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Obsah :

1.O dokumentu	5
1.1. Revize dokumentu	5
1.2. Účel dokumentu	5
1.3. Rozsah platnosti	5
1.4. Související dokumenty	5
2.Termíny a definice	5
3.Úvod	6
4.Konstanty a typy	6
4.1. Kódy chybových výsledků automatů	6
4.1.1. Kódy chybových výsledků automatů týkající se PINů	7
4.2. Kódy příkazů pro metodu ChSetBinParam	7
4.3. Kódy příkazů pro metodu ChGetBinParam	8
4.4. Konstanty výsledků po vyslání příkazu do modemu	9
4.5. Výčtové typy pro určení položek polí a jednoduché typy	10
4.6. Struktura bufferu pro vyslání AT příkazu do modemu	11
4.7. Konstanty základních AT příkazů a odpovědí pro GSM	12
5.Objekty	12
5.1. tChnGSM	12
5.1.1. Položky	12
5.1.2. Metody	14
5.1.2.1. Init konstruktor	14
5.1.2.2. ChInitParam konstruktor	15
5.1.2.3. Done destruktork	15
5.1.2.4. ChSetOneParam funkce	15
5.1.2.5. ChGetParam funkce	19
5.1.2.6. ChSetBinParam procedura	19
5.1.2.7. ChGetBinParam funkce	19
5.1.2.8. ChOpen procedura	20
5.1.2.9. ChClose procedura	20
5.1.2.10. ChConnect procedura	20
5.1.2.11. ChDisconnect procedura	20
5.1.2.12. ChState funkce	21
5.1.2.13. ChReady funkce	21
5.1.2.14. ChTick procedura	21
5.1.2.15. ChSendTick procedura	21
5.1.2.16. ChReceiveTick procedura	21
5.1.2.17. ChSend procedura	21
5.1.2.18. ChSendReady funkce	22
5.1.2.19. ChSendFlush procedura	22
5.1.2.20. ChReceiveReady funkce	22
5.1.2.21. ChReceiveChar funkce	22
5.1.2.22. ChReceive procedura	22
5.1.2.23. ChReceiveFlush procedura	23
5.2. tAddChnGSM	23
5.2.1. Metody	23
5.2.1.1. ChInit funkce	23
6.Příklad	23

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. Doplněná konstanta cmd_GetChCtrl2. Opravené jméno proměnné DialPPN → DialNum. Doplněné konstanty cAtCmdGSM_ChangePin, cAtAnsGSM_AnswSimPin2, cAtAnsGSM_AnswSimPuk2, cAtAnsGSM_AnswSimBlock, cAtCmdGSM_GetManufID, cAtCmdGSM_GetModelID a cAtCmdGSM_GetSignalQ. Doplněné metody ChState, ChReady. Doplněné konstanty CHS_SendBegin, CHS_Send a CHS_ReceiveReConn.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky sériové komunikace pomocí GSM 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 ChnVirt popisujícím rozhraní svých potomků a ChTypes.

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 ChnGSM definuje komunikační objekt **tChnGSM**, který je dědicem od rodičovského komunikačního objektu tChnVirt. Instance objektu tChnGSM reprezentuje vyšší komunikační vrstvu v komunikačním kanálu. Objekt tChnGSM definuje automaty zajišťující inicializaci modemu, navazování a udržování spojení, vysílání a příjem dat a rušení navázaného spojení. Fyzický přenos dat mezi 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.

Znaky dat přicházející po komunikaci (myslí se data přicházející z protější stanice s níž je navázáno spojení) jsou ukládány do vstupního kruhového vyrovnávacího bufferu nižší komunikační vrstvy, z kterého jsou předávány metodami **ChReceive** a **ChReceiveChar** k dalšímu zpracování. Znaky určené k odeslání jsou zaslány z výstupního bufferu nižší komunikační vrstvy do linky. Určení parametrů komunikace je voleno parametrem nastavovací metody ChSetParam.

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

Protože je objekt **tChnGSM** 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   = 'GSM';
```

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

```
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 chybových výsledků automatů

```
Res_ErrInitStr1 = $A0;
```

Konstanta **Res_ErrInitStr1** definuje chybový kód při inicializaci modemu 1.řetězcem AT příkazů při navazování spojení.

```
Res_ErrInitStr2 = $A1;
```

Konstanta **Res_ErrInitStr2** definuje chybový kód při inicializaci modemu 2.řetězcem AT příkazů při navazování spojení.

`Res_ErrInitStr3` = \$A2;
Konstanta **Res_ErrInitStr3** definuje chybový kód při inicializaci modemu 3.řetězcem AT příkazů při navazování spojení.

`Res_ErrPhoneNum` = \$A3;
Konstanta **Res_ErrPhoneNum** definuje chybový kód, který oznamuje, že uživatel nezadal žádné telefonní číslo.

`Res_ErrDial` = \$A4;
Konstanta **Res_ErrDial** definuje chybový kód při vytáčení čísla při navazování spojení.

`Res_ErrCDAbsent` = \$A5;
Konstanta **Res_ErrCDAbsent** definuje chybový kód při neaktivním signálu CD (Carrier Detect) - nosná.

`Res_ErrSWHangUp` = \$A6;
Konstanta **Res_ErrSWHangUp** definuje chybový kód při softwarovém zavěšování (Pozn.: Modem může být stále připojen).

4.1.1. Kódy chybových výsledků automatů týkající se PINů

`Res_ErrWrongPIN` = \$20;
Konstanta **Res_ErrWrongPIN** definuje chybový kód při špatné hodnotě Sim PINu.

`Res_ErrNonePIN` = \$30;
Konstanta **Res_ErrNonePIN** definuje chybový kód při nezadaném Sim PINu uživatelem, když ho modem potřebuje.

`Res_ErrWrongPUK` = \$21;
Konstanta **Res_ErrWrongPUK** definuje chybový kód při špatné hodnotě Sim PUKu.

`Res_ErrNonePUK` = \$31;
Konstanta **Res_ErrNonePUK** definuje chybový kód při nezadaném Sim PUKu uživatelem, když ho modem potřebuje.

`Res_ErrWrongPIN2` = \$22;
Konstanta **Res_ErrWrongPIN2** definuje chybový kód při špatné hodnotě Sim PIN 2.

`Res_ErrNonePIN2` = \$32;
Konstanta **Res_ErrNonePIN2** definuje chybový kód při nezadaném Sim PIN2 uživatelem, když ho modem potřebuje.

`Res_ErrWrongPUK2` = \$23;
Konstanta **Res_ErrWrongPUK2** definuje chybový kód při špatné hodnotě Sim PUK 2.

`Res_ErrNonePUK2` = \$33;
Konstanta **Res_ErrNonePUK2** definuje chybový kód při nezadaném Sim PUK2 uživatelem, když ho modem potřebuje.

`Res_ErrSimBlocked` = \$40;
Konstanta **Res_ErrSimBlocked** definuje chybový kód při zablokované Sim kartě.

`Res_ErrNoSimCard` = \$50;
Konstanta **Res_ErrNoSimCard** definuje chybový kód, který sděluje, že není vložena SIM karta.

4.2. Kódy příkazů pro metodu ChSetBinParam

`cmd_SetMCR` = \$0101;

Konstanta definuje kód příkazu pro nastavení všech modemových signálů (Modem Control Registeru) dle nejnižšího byte parametru Param funkce.

```
cmd_SetDTR = $0102;
```

Konstanta definuje kód příkazu pro nastavení modemového signálu DTR dle 0-tého bitu parametru Param funkce.

```
cmd_SetRTS = $0103;
```

Konstanta definuje kód příkazu pro nastavení modemového signálu RTS dle 0-tého bitu parametru Param funkce.

```
cmd_SendCmd = $0201;
```

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 = $0202;
```

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

```
cmd_CmdAnsw = $0203;
```

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 = $0203;
```

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.

4.3. Kódy příkazů pro metodu ChGetBinParam

```
cmd_GetMSR = $0101;
```

Konstanta definuje kód příkazu pro vrácení stavu všech modemových signálů (Modem Status Registeru) v nejnižším bytu výsledku funkce.

```
cmd_GetDCTS = $0102;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu DCTS v 0-tém bitu výsledku funkce.

```
cmd_GetDDSR = $0103;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu DDSR v 0-tém bitu výsledku funkce.

```
cmd_GetTERI = $0104;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu TERI v 0-tém bitu výsledku funkce.

```
cmd_GetDRLSD = $0105;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu DRLSD v 0-tém bitu výsledku funkce.

```
cmd_GetCTS = $0106;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu CTS v 0-tém bitu výsledku funkce.

```
cmd_GetDSR = $0107;
```


Konstanta definuje kód příkazu pro vrácení stavu modemového signálu DSR v 0-tém bitu výsledku funkce.

```
cmd_GetRI = $0108;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu RI v 0-tém bitu výsledku funkce.

```
cmd_GetRLSD = $0109;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu RLSD v 0-tém bitu výsledku funkce.

```
cmd_GetCD = cmd_GetRLSD;
```

Konstanta má stejnou hodnotu jako výše zmíněná konstanta `cmd_GetRLSD`.

```
cmd_GetChCtrl2 = $0201;
```

Konstanta definuje kód příkazu pro požadavek na čtení stavu vnitřního automatu při navazování nebo rušení spojení.

```
cmd_CmdReady = $0202;
```

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 = $0203;
```

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).

4.4. 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_Connect = 1;
```

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

```
ResGSM_Busy = 10;
```

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

```
ResGSM_Error = 12;
```

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

```
ResGSM_NoCarry = 13;
```

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

```
ResGSM_Answer = 19;
```

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

```
ResGSM_LongAnswer = 20;
```

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

ResGSM_SHwErr = 21;

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

ResGSM_RHwErr = 22;

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

ResGSM_RTimeOut = 23;

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

ResGSM_STimeOut = 24;

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

4.5. Výčtové typy pro určení položek polí a jednoduché typy

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

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
- BeforPIN - prodleva před zadáním PINu.
- AfterPIN - prodleva po správně zadaném PINu pro nalezení sítě operátora.

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

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

- SendAtCmd - timeout pro čekání na odpověď na vyslaný AT příkaz do modemu
- AnswPIN - timeout pro čekání na odpověď při zadávání PINu
- 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 AT příkazy

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

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

- DialPNS - vytáčení 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.

```
tStrSimPin = String[ 8];
```

Typ pro omezení délky řetězce pro kódy SIM PIN a SIM PIN2.

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

Typ pro omezení délky řetězce pro kódy SIM PUK a SIM PUK2.

4.6. Struktura bufferu pro vyslání AT příkazu do modemu

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

Typ **tAtCmdBuff** svou strukturou definuje formát bufferu dat pro vyslání AT příkazu do modemu metodou **ChSetBinParam** s kódy příkazů **cmd_SendCmd** až **cmd_RecData**. Položka **Mode** určuje způsob vyslání zprávy do modemu a způsob čekání na odpověď. Tuto položku vyplní metoda **ChSetBinParam** sama dle daného kódu příkazu. 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 jako odpověď.

4.7. Konstanty základních AT příkazů a odpovědí pro GSM

```

cAtCmdGSM_SetPin      = '+CPIN';
    Řetězec pro dotaz (jaký PIN modem požaduje) či nastavení PINu.
cAtCmdGSM_ChangePin   = '+CPDW';
    Řetězec pro změnu SIM 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_AnswSimPin2 = 'SIM PIN2';
    Řetězec pro odpověď od GSM modemu pro požadavek zadání SIM PINu 2.
cAtAnsGSM_AnswSimPuk2 = 'SIM PUK2';
    Řetězec pro odpověď od GSM modemu pro požadavek zadání SIM PUKu 2.
cAtAnsGSM_AnswSimBlock = 'BLOCKED';
    Řetězec pro odpověď od GSM modemu že karta je zablokovaná.
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) SMS zpráv.
cAtCmdSMS_SetMode     = '+CMGF';
    Řetězec pro dotaz či nastavení textového (TXT) či protokolového (PDU)
    módu pro SMS zprávy.
cAtCmdSMS_GetMemSto   = '+CPMS';
    Řetězec pro dotaz obsazení paměti 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_GetManufID  = '+CGMI';
    Řetězec pro čtení výrobce GSM stanice.
cAtCmdSMS_GetModelID  = '+CGMM';
    Řetězec pro čtení modelu GSM stanice.
cAtCmdSMS_GetSignalIQ = '+CSQ';
    Řetězec pro čtení kvality signálu.

```

5. Objekty

5.1. tChnGSM

5.1.1. Položky

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_CDS ShutDown: Boolean;
Položka **CH_CDS ShutDown** je určena jen pro vnitřní použití, pro určení, že nastal výpadek modemového signálu CD (RLSD).

CH_FlTestCD : Boolean;
Položka **CH_FITestCD** je určena jen pro vnitřní použití, pro určení, zda se má testovat modemový signál 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_FIHwI** definuje příznak používání hardwarové inicializace modemu při navazování spojení.

CH_FlHwH : Boolean;
Položka **CH_FIHwH** 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 řídicích 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_RepAT : Byte;
Položka **CH_RepAT** definuje počet opakovaných pokusů při neúspěšné komunikaci s modemem prostřednictvím AT příkazů.

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 vysláním se čeká na signál CTS.
2 - DSR/DTR - před vysláním 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_StationID: string[50];
```

Položka **CH_StationID** definuje řetězec s identifikací připojené GSM stanice, který se hodnotou naplní během navazování spojení započatém voláním metody ChConnect.

```
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_SimPin2 : tStrSimPuk;
```

Položka **CH_SimPin2** definuje řetězec pro kód SIM PIN2.

```
CH_SimPuk2 : tStrSimPuk;
```

Položka **CH_SimPuk2** definuje řetězec pro kód SIM PUK2.

5.1.2. 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_FlHWI     := true;
CH_FlHWH     := true;
CH_Rep       := 3;
CH_Cnt       := 0;
CH_SCtrl     := 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_FlTestCD  := false;
CH_SimPin    := '';
CH_SimPuk    := '';
CH_SimPin2   := '';
CH_SimPuk2   := '';
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_Delay[BeforPIN] :=00250;
CH_Delay[AfterPIN] :=03000;
CH_TimeOut[SendAtCmd] :=00500;
CH_TimeOut[CmdAnsw] :=05000;
CH_TimeOut[AnswPIN] :=03000;
CH_TimeOut[DialAnsw1] :=10000;
CH_TimeOut[DialAnsw2] :=05000;
CH_TimeOut[DialAnsw3] :=05000;
CH_Str[DialPNS] := 'ATD';
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';
CH_RepAT := 1;
New(CH_ATCmdBuff);
with CH_ATCmdBuff^ do
begin
  CmdStr := '';
  Delay1 :=0;
  Delay2 :=0;
  TimeOut:=0;
  RecStr := '';
end;
CH_StationID:= '';

```

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 destruktork

```
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 dekodování a nastavení jednoho konkrétního parametru, který je zadán v parametru S. Tato metoda se volá v aplikaci prostřednictvím metody **ChSetParam**. Metoda **ChSetOneParam** komunikačního objektu **tChnGSM** dekoduje tyto parametry:

PIN='sss'

Parametrem **PIN** se nastaví řetězec pro kód SIM PIN.

PUK='sss'

Parametrem **PUK** se nastaví řetězec pro kód SIM PUK.

PI2='sss'

Parametrem **PI2** se nastaví řetězec pro kód SIM PIN2.

PU2='sss'

Parametrem **PU2** se nastaví řetězec pro kód SIM PUK2.

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 Master stanice snaží toto spojení obnovit. Jakékoliv přijaté a do ztráty spojení nevyzvednuté znaky jsou zapomenuty a pokud se před ztrátou spojení vysílala nějaká zpráva, bude uložena do pomocného bufferu a opět odvysílána po jeho navázání.

Stanice Slave - při navazování spojení započatém zavoláním 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 příchozím volání (modemový registr S0 nastavit na hodnotu různou od nuly např. příkazem AT S0=2). Stav automatu kanálu **CHS_Connect** nastane ihned po inicializaci a nastavení modemu, skutečný fyzický stav navázání spojení s protější stanicí ale 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).

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 navazování 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_FlowCtrl**). 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.

DBP=llll

Parametrem **DBP** ("Delay Before PIN") se nastaví pauza **CH_Delay[BeforPIN]** před zadáním PINu.

DAP=llll

Parametrem **DAP** ("Delay After PIN") se nastaví pauza **CH_Delay[AfterPIN]** po správně zadaném PINu.

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.

QAP=llll

Parametrem **QAP** ("TimeOut for Answer After PIN") se nastaví maximální doba **CH_TimeOut[AnswPIN]** pro čekání na odpověď z modemu po vyslání příkazu pro zadání PINu. 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** navrácí 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 vyslat AT příkaz do modemu s čekáním na odpověď.

```

write('Test komunikace Modem-Computer: ');
ModNumName:=ChNumName('GSM');
with AtCmdBuff do
begin
  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 metod **ChOpen** a **ChConnect** podřízené fyzické vrstvy 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 deinitializace technického vybavení prostřednictvím metod **ChDisconnect** a **ChClose** podřízené fyzické vrstvy 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**), by měl kanál přejít 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 došlo k takovému nainicializování modemu, aby byl připraven k fyzické navázání spojení od volající stanice. Ve stavu **CHS_Connect** se mohou již volat metody pro příjem a vysílání zpráv, které fyzické navázání spojení s protější stanicí sami ošetří. Během navazování spojení mohou samozřejmě nastat určité chyby (viz. 4.1 Kódy chybových výsledků automatů), které aplikace musí dekodovat a patřičně ošetřit.

Algoritmus správného volání metody **ChConnect** s testem na konečný stav automatu **CHS_Connect** a případné ošetření chyb během navazování spojení je uveden v popisu knihovny **ChnVirt**.

5.1.2.11. ChDisconnect procedura

```
procedure ChDisconnect;
```

Metoda **ChDisconnect** provede započetí ukončení navázaného spojení s protější stanicí. 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** pro navázání nového spojení.

Algoritmus správného volání metody **ChDisconnect** s testem na konečný stav automatu **CHS_DisConnect** a případné ošetření chyb během rušení spojení je uveden v popisu knihovny **ChnVirt**.

5.1.2.12. ChState funkce

```
function ChState;
```

Metoda **ChState** provede krok automatu (volá metodu **ChTick**). Jako návratovou hodnotu vrací naposledy dosažený stabilní stav komunikačního kanálu.

5.1.2.13. ChReady funkce

```
function ChReady;
```

Metoda **ChReady** provede krok automatu (volá metodu **ChTick**). Jako návratovou hodnotu vrací aktuální stav komunikačního kanálu.

5.1.2.14. ChTick procedura

```
procedure ChTick;
```

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

5.1.2.15. 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 protějščí 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.

5.1.2.16. 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 protějščí stanicí, metoda zajistí nové navázání spojení.

5.1.2.17. 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`).

5.1.2.18. 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**.

5.1.2.19. ChSendFlush procedura

```
procedure ChSendFlush;
```

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

5.1.2.20. 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**.

5.1.2.21. 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.

5.1.2.22. 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**.

5.1.2.23. ChReceiveFlush procedura

procedure ChReceiveFlush;

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

5.2. tAddChnGSM

Typ **tAddChnGSM** je typem objektu, který slouží k definování prvku v seznamu správců komunikačních objektů (tzv. správce komunikačního objektu **tChnGSM** v seznamu správců). Objekt **tAddChnGSM** 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 **tChnGSM** a ukazatel na instanci tohoto objektu vrací jako svoji funkční hodnotu.

6. Příklad

Příklad ukazuje použití komunikační jednotky **ChnGSM** s podřízenou fyzickou vrstvou definovanou komunikační jednotkou **ChnCom**.

```
uses
  uString,
  ChnVirt,
  ChnGSM,
  ChnCom;
...
const
  ParamStr : tParamStr =
    'NAM=GSM MAS=MASTER PIN=''1234'' I2S=''ATL1M3'' '+'
    'PNN=''02242351'' REP=1 '+'
    'NAM=COM COM=1 IRQ=4 PAR=N BD=9600 BIT=8 STO=2 LRB=500';
  LMess     = 40;

type
  tMess     = array [0..LMess+10] of Byte;

var
  Chn       : pChnVirt;
  SMess     : ^tMess;
  RMess     : ^tMess;
  LSMess    : Word;
  LRMess    : Word;

begin
  ...
  New(SMess);
  New(RMess);
  ...
  { inicializace Chn }
  Chn:=ChnCollection^.ChNewInit(ChnGSM.cName);
  with Chn^ do
```

```
begin
  { nastavení parametrů komunikace }
  ChSetParam(ParamStr);
  if ChResult<>res_Ok then WriteLn('Chyba');
  { otevření hardwarového kanálu }
  ChOpen;
  repeat
    if ChResult<>res_Ok then WriteLn('Chyba');
  until ChReady=CHS_Open;
  { definování místa, kam se má přijatá zpráva uložit }
  ChReceiveBuffer(RMess, SizeOf(RMess^));
  if ChReceiveResult<>res_Ok then WriteLn('Chyba');
  { navázání spojení s volanou stanicí }
  ChConnect;
  repeat
    if ChResult<>res_Ok then WriteLn('Chyba');
  until ChReady=CHS_Connect;
  ...
  { naplnění zprávy daty }
  ...
  { vyslání 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;
    ...
  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');
  ...
  { ukončení }
  ChDisconnect;
  repeat
    if ChResult<>res_Ok then WriteLn('Chyba');
  until ChReady=CHS_DisConnect;
  ChClose;
  repeat
    if ChResult<>res_Ok then WriteLn('Chyba');
  until ChReady=CHS_Close;
  end;
  { zrušení instance objektu }
  Dispose(Chn, Done);
  ...
end.
```