ChnTecom

JEDNOTKA DEFINUJÍCÍ KOMUNIKAČNÍ PROTOKOL TECOMAT

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 1.20

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: 07.04.2004
Datum posledního uložení dokumentu: 07.04.2004
(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. Popis konstant a typů 6
   4.1. Konstanty chybových kódů 6
   4.2. Řídící znaky protokolu 7
   4.3. Kódy služeb protokolu 7
   4.4. Konstanty přístupu k datovým oblastem PLC Tecomatu 9
   4.5. Struktury přijímacích a vysílacích bufferů 9
5. Objekty 11
   5.1. tChnTecom 11
      5.1.1. Položky 11
      5.1.2. Metody 12
         5.1.2.1. Init konstruktor 12
         5.1.2.2. ChInitParam konstruktor 12
         5.1.2.3. Done destruktor 12
         5.1.2.4. ChSetOneParam funkce 12
         5.1.2.5. ChGetParam funkce 13
         5.1.2.6. ChConnect procedura 13
         5.1.2.7. ChDisConnect procedura 13
         5.1.2.8. ChSend procedura 13
         5.1.2.9. ChReceiveReady funkce 14
         5.1.2.10. ChReceive procedura 14
         5.1.2.11. ChReceiveFlush procedura 14
         5.1.2.12. ChGetNode procedura 14
         5.1.2.13. ChReceiveTick procedura 14
   5.2. tAddChnTecom 14
      5.2.1. Metody 15
         5.2.1.1. ChInit funkce 15
6. Struktura zpráv protokolu TECOMAT 15
   6.1. Obecná struktura zpráv 15
   6.2. Struktura konkrétních zpráv 16
      6.2.1. Zpráva Connect - test připojení PLC 16
      6.2.2. Zpráva Ident - identifikace připojeného PLC 16
      6.2.3. Zpráva SetTID - nastavení systémového času 16
      6.2.4. Zpráva SetCW - nastavení řídícího slova 16
      6.2.5. Zpráva GetSW - přečtení stavového slova 17
      6.2.6. Zpráva GetErr - přečtení chybového zásobníku 17
      6.2.7. Zpráva MaskCW - nastavení jednotlivých bitů řídícího slova 17
      6.2.8. Zpráva ReadN - čtení z datové paměti 17
      6.2.9. Zpráva WriteN - zápis do datové paměti 18
      6.2.10. Zpráva WandrN - zápis do a čtení z datové paměti 19
      6.2.11. Zpráva ReadB - čtení bitů z datové paměti 19

6.2.12. Zpráva WriteB - zápis bitů do datové paměti 20
6.2.13. Zpráva ReadBD - destruktivní čtení bitů z datové paměti 21
6.2.14. Zpráva ReadND - destruktivní čtení z datové paměti 21
6.2.15. Zpráva WandrND - zápis do a destruktivní čtení z datové paměti 21
7. Příklad 21

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	22.05.2003	Úprava dokumentu dle ISO9000.
1.20	4.XX	Wil	07.04.2004	Změna číslování chybových konstant res_ErrXxx dle standardu.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky definující komunikační protokol TECOMAT.

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 základní rodičovský prvek pro tvorbu komunikačních objektů a s manuálem „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 definuje formát protokolu TECOMAT používaného při komunikaci se zařízeními PLC TECOMAT firmy Teco a.s. Obstarává zabezpečení dat, vkládání a vyjímání nadbytečností, tak jak to tento protokol předepisuje. Fyzický přenos dat je zajištěn prostřednictvím nižší komunikační vrstvy.

Knihovna ChnTecom definuje komunikační objekt **tChnTecom**, který je dědicem od rodičovského komunikačního objektu tChnVirt. Instance objektu tChnTecom representuje vyšší komunikační vrstvu v komunikačním kanálu. Transformuje předávaná data mezi komunikačními objekty nižších vrstev, které provádějí fyzický přenos, a aplikací nebo případně další vyšší komunikační vrstvou. Určení, přes jakou fyzickou komunikační vrstvu bude komunikace probíhat, je voleno až parametry nastavovací metody ChSetParam.

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

Protože je objekt **tChnTecom** 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. 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 = 'TECOM';
```

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

```
tMessType = (tpSACK, tpShortM, tpLongM);
```

Výčtový typ **tMessType** definuje tři základní typy zpráv od Slave stanice.
Typ **tpSACK** je pro krátké pozitivní potvrzení od Slave stanice na zápis dat.
Typ **tpShortM** je pro krátkou zprávu bez datového pole od Slave stanice.
Typ **tpLongM** je pro dlouhou zprávu s datovým polem od Slave stanice.

```
MaxDataLen  = 245;
```

Konstanta **MaxDataLen** definuje maximální délku přenášeného datového bloku v bytech.

4.1. Konstanty chybových kódů

Následující chyby mohou vracet metody ChReceiveResult, ChSendResult a případně i ChResult.

```
Res_ErrFrame    = $20;
```

Chybný formát zprávy - přijatou zprávu nelze akceptovat.

```
res_ErrSum      = $21;
```
Chyba kontrolního součtu - přijatou zprávu nelze akceptovat.

```
res_ErrLen      = $22;
```
Chyba délky zprávy - přijatou zprávu nelze akceptovat.

```
res_ErrVal      = $23;
```
Chyba při převodu čísla na textový řetězec a naopak.

```
Res_ErrUnknownCode = $25;
```
Chyba indikující neznámý Frame Control.

4.2. Řídící znaky protokolu

```
SD1   = $10;
SD2   = $68;
ED    = $16;
SACK  = $E5;
```
Konstanta **SD1** definuje kód pro začátek krátké zprávy bez datového pole.
Konstanta **SD2** definuje kód pro začátek dlouhé zprávy s datovým polem.
Konstanta **ED** definuje kód pro konec zprávy s datovým i nedatovým polem.
Konstanta **SACK** definuje kód pro krátké potvrzení od Slave stanice na zápis dat.

4.3. Kódy služeb protokolu

Nasledující konstanty znaků určují kód služby (FC - Frame Control) dané zprávy:

```
FCM_Connect       = $69;
FCM_Ident         = $6E;
FCM_Write         = $63;
FCM_Read          = $6C;
FC2_SetTID        = $08;
FC2_SetCW         = $09;
FC2_GetSW         = $0A;
FC2_GetErr        = $0E;
FC2_MaskCW        = $11;
FC2_ReadN         = $0B;
FC2_WriteN        = $0C;
FC2_WandrN        = $0D;
FC2_ReadB         = $0F;
FC2_WriteB        = $10;
FC2_ReadBD        = $90;
FC2_ReadND        = $91;
FC2_WandrND       = $93;
FCS_ConnectRep    = $00;
FCS_IdentRep      = $00;
FCS_Reply         = $08;
FCS_UnknownMes    = $02;
FCS_DataNotReady  = $09;
FCS_BadMesParam   = $0C;
```
Konstanta **FCM_Connect** definuje kód pro krátkou zprávu Connect od Master stanice, která se používá k otestování připojení daného PLC Tecomatu.
Konstanta **FCM_Ident** definuje kód pro krátkou zprávu Ident od Master stanice, která se používá pro identifikaci připojeného PLC Tecomatu.
Konstanta **FCM_Write** definuje hlavní kód pro dlouhou zprávu pro zápis dat od Master stanice. Podle vedlejšího kódu (FC2_SetTID, FC2_SetCW, FC2_MaskCW, FC2_WriteN či FC2_WriteB) se zprávy dále dělí na různé způsoby zápisů.

Konstanta **FCM_Read** definuje hlavní kód pro dlouhou zprávu pro čtení dat od Master stanice. Podle vedlejšího kódu (FC2_GetSW, FC2_GetErr, FC2_ReadN, FC2_WandrN, FC2_ReadB, FC2_ReadBD, FC2_ReadND či FC2_WandrND) se zprávy dále dělí na různé způsoby čtení.
Konstanta **FC2_SetTID** definuje vedlejší kód pro dlouhou zprávu SetTID od Master stanice, která se používá pro nastavení systémového času PLC Tecomatu.
Konstanta **FC2_SetCW** definuje vedlejší kód pro dlouhou zprávu SetCW od Master stanice, která se používá pro nastavení řídícího slova PLC Tecomatu.
Konstanta **FC2_GetSW** definuje vedlejší kód pro dlouhou zprávu GetSW od Master stanice, která se používá pro přečtení stavového slova PLC Tecomatu.
Konstanta **FC2_GetErr** definuje vedlejší kód pro dlouhou zprávu GetErr od Master stanice, která se používá pro přečtení chybového zásobníku PLC Tecomatu.
Konstanta **FC2_MaskCW** definuje vedlejší kód pro dlouhou zprávu MaskCW od Master stanice, která se používá pro nastavení jednotlivých bitů řídícího slova PLC Tecomatu.
Konstanta **FC2_ReadN** definuje vedlejší kód pro dlouhou zprávu ReadN od Master stanice, která se používá pro čtení z datové paměti PLC Tecomatu.
Konstanta **FC2_WriteN** definuje vedlejší kód pro dlouhou zprávu WriteN od Master stanice, která se používá pro zápis do datové paměti PLC Tecomatu.
Konstanta **FC2_WandrN** definuje vedlejší kód pro dlouhou zprávu WandrN od Master stanice, která se používá pro zápis do a čtení z datové paměti PLC Tecomatu.
Konstanta **FC2_ReadB** definuje vedlejší kód pro dlouhou zprávu ReadB od Master stanice, která se používá pro čtení bitů z datové paměti PLC Tecomatu.
Konstanta **FC2_WriteB** definuje vedlejší kód pro dlouhou zprávu WriteB od Master stanice, která se používá pro zápis bitů do datové paměti PLC Tecomatu.
Konstanta **FC2_ReadBD** definuje vedlejší kód pro dlouhou zprávu ReadBD od Master stanice, která se používá pro destruktivní čtení bitů z datové paměti PLC Tecomatu.
Konstanta **FC2_ReadND** definuje vedlejší kód pro dlouhou zprávu ReadND od Master stanice, která se používá pro destruktivní čtení z datové paměti PLC Tecomatu.
Konstanta **FC2_WandrND** definuje vedlejší kód pro dlouhou zprávu WandrND od Master stanice, která se používá pro zápis do a destruktivní čtení z datové paměti PLC Tecomatu.
Konstanta **FCS_ConnectRep** definuje kód pro krátkou pozitivní odpověď od Slave stanice na zprávu Connect.
Konstanta **FCS_IdentRep** definuje kód pro dlouhou pozitivní odpověď od Slave stanice na zprávu Ident.
Konstanta **FCS_Reply** definuje kód pro dlouhou pozitivní odpověď od Slave stanice na zprávu pro čtení dat.
Konstanta **FCS_UnknownMes** definuje kód pro krátkou negativní odpověď od Slave stanice, že tuto službu daná Slave stanice nezná.
Konstanta **FCS_DataNotReady** definuje kód pro krátkou negativní odpověď od Slave stanice, že dosud nemá připravená data, ale očekává, že je brzy připravená mít bude.

Konstanta **FCS_BadMesParam** definuje kód pro dlouhou negativní odpověď od Slave stanice, při zadání chybných parametrů komunikační služby.

4.4. Konstanty přístupu k datovým oblastem PLC Tecomatu

Paměť PLC Tecomatu se skládá ze zápisníkové paměti, kde jsou uloženy registry, a přídavné paměti DataBox.

```
RegX  = $00;
RegY  = $01;
RegS  = $02;
RegR  = $03;
DBox0 = $80;
DBox1 = $81;
DBox2 = $82;
DBox3 = $83;
DBox4 = $84;
DBox5 = $85;
DBox6 = $86;
DBox7 = $87;
```

Konstanta **RegX** je pro přístup k registrům X (obrazy vstupů).
Konstanta **RegY** je pro přístup k registrům Y (obrazy výstupů).
Konstanta **RegS** je pro přístup k registrům S (systémové registry).
Konstanta **RegR** je pro přístup k registrům R (uživatelské registry).
Konstanta **DBox0** je pro přístup k oblasti přídavné paměti DataBox adresy $00000 - $0FFFF.
Konstanta **DBox1** je pro přístup k oblasti přídavné paměti DataBox adresy $10000 - $1FFFF.
Konstanta **DBox2** je pro přístup k oblasti přídavné paměti DataBox adresy $20000 - $2FFFF.
Konstanta **DBox3** je pro přístup k oblasti přídavné paměti DataBox adresy $30000 - $3FFFF.
Konstanta **DBox4** je pro přístup k oblasti přídavné paměti DataBox adresy $40000 - $4FFFF.
Konstanta **DBox5** je pro přístup k oblasti přídavné paměti DataBox adresy $50000 - $5FFFF.
Konstanta **DBox6** je pro přístup k oblasti přídavné paměti DataBox adresy $60000 - $6FFFF.
Konstanta **DBox7** je pro přístup k oblasti přídavné paměti DataBox adresy $70000 - $7FFFF.

4.5. Struktury přijímacích a vysílacích bufferů

```
pMaSendRecord = ^tMaSendRecord;
tMaSendRecord = record
  case FC : byte of
    FCM_Connect:
      ();
    FCM_Ident  :
      ();
    FCM_Read   ,
    FCM_Write  :
      (case FC2 : byte of
         FC2_SetTID :
           (Year      : 0..99;
            Month     : 1..12;
```

```
            Day       : 1..31;
            Hour      : 0..23;
            Minute    : 0..59;
            Second    : 0..59;
            DayWeek   : 1..07;);
          FC2_SetCW  :
            (CWL      : byte;
             CWH      : byte;);
          FC2_GetSW  :
            ();
          FC2_GetErr :
            ();
          FC2_MaskCW :
            (C0L      : byte;
             C0H      : byte;
             C1L      : byte;
             C1H      : byte;);
          FC2_ReadN  ,
          FC2_ReadND ,
          FC2_WriteN ,
          FC2_WandrN ,
          FC2_WandrND,
          FC2_ReadB  ,
          FC2_ReadBD ,
          FC2_WriteB :
            (Dummy1_2  : array[1..2]of byte;
             Data      : array[0..MaxDataLen-1]of byte;);
      );
  end;
```

TMaSendRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu Tecomat posílaným Master stanicí do zařízení PLC TECOMAT, přičemž je zbaven nadbytečností, které jsou při vysílání doplněny. Položka **FC** udává kód požadované služby. Při čtení či zápisu dat se dále používá položka **FC2**, která určuje vedlejší kód služby. Při zápisu systémového času se používají položky **Year** (poslední dvojčíslí roku), **Month** (měsíc), **Day** (den), **Hour** (hodina), **Minute** (minuty), **Second** (sekundy) a **DayWeek** (číslo dne v týdnu - 1=pondělí, 7=neděle). Při nastavování kontrolního slova se používají položky **CWL** (dolní byte nového kontrolního slova) a **CWH** (horní byte nového kontrolního slova). Při nastavování jednotlivých bitů kontrolního slova se používají položky **C0L** (nulovací maska dolního byte), **C0H** (nulová maska horního byte), **C1L** (jedničková maska dolního byte) a **C1H** (jedničková maska horního byte). Při čtení či zápisu registrů nebo jejich bitů se používá položka **Data**, která definuje pole dat posílaných do nebo čtených z PLC Tecomatu. Položka **Dummy1_2** nemá žádný zvláštní význam, v záznamu je definována pouze pro zarovnání začátku položky Data na adresu dělitelnou čtyřma.

```
pMaRecRecord  = ^tMaRecRecord;
tMaRecRecord  = record
  case MessType : tMessType of
    tpSACK  :
      ();
    tpShortM:
      ();
    tpLongM :
      (case FC : byte of
         FCS_IdentRep    :
           (IdentStr : string[10];
            ProfiBus : char;
            VerHW    : string[3];
```

```
            VerSW    : string[3];);
          FCS_Reply       :
            (Dummy1_2  : array[1..2]of byte;
             Data : array[0..MaxDataLen-1]of byte;);
          FCS_BadMesParam :
            (Err : word;);
      );
  end;
```

TMaRecRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu Tecomat přijímaným Master stanicí ze zařízení PLC TECOMAT, přičemž je zbaven nadbytečností, které jsou při příjmu odstraněny. Položka **MessType** udává typ přijaté zprávy. Při příjmu krátké či dlouhé zprávy (tpShortM nebo tpLongM) se dále používá položka **FC**, která určuje kód služby. Při příjmu odpovědi na zprávu "Ident" se používají položky **IdentStr** (identifikační řetězec připojeného PLC zařízení), **ProfiBus** (znak implementace profibusu), **VerHW** (řetězec s verzí hardwaru) a **VerSW** (řetězec s verzí softwaru). Při odpovědi na čtení registrů nebo jejich bitů se používá položka **Data**, která definuje pole dat čtených z PLC Tecomatu. Položka **Dummy1_2** nemá žádný zvláštní význam, v záznamu je definována pouze pro zarovnání začátku položky Data na adresu dělitelnou čtyřma.

```
tSlSendRecord = tMaRecRecord;
pSlSendRecord = ^tSlSendRecord;
```

TSlSendRecord je typ, který svou strukturou odpovídá datům protokolu Tecomat vysílaným Slave stanicí do nadřízeného systému (Master stanice). Svou strukturou je shodný s typem pro strukturu přijímacího bufferu Master stanice.

```
tSlRecRecord  = tMaSendRecord;
pSlRecRecord  = ^tSlRecRecord;
```

TSlRecRecord je typ, který svou strukturou odpovídá datům protokolu Tecomat přijímaným Slave stanicí z nadřízeného systému (Master stanice). Svou strukturou je shodný s typem pro strukturu vysílacího bufferu Master stanice.

5. Objekty

5.1. tChnTecom

5.1.1. Položky

```
CH_RTick   : Boolean;
```

Položka **CH_RTick** označuje, že je vykonávaná činnost přijímacího automatu. Tato položka se používá pro ladění.

```
CH_SBuff   : Pointer;
```

Položka **CH_SBuff** definuje ukazatel na vysílací buffer.

```
CH_MSBuff  : Word;
```

Položka **CH_MSBuff** definuje délku vysílacího bufferu.

```
CH_LRMess  : Word;
```

Položka **CH_LRMess** definuje délku přijímané zprávy.

```
CH_RSum    : tSum8;
```

Položka **CH_RSum** se používá pro počítání kontrolního součtu přijímače.

```
CH_SSum    : tSum8;
```

Položka **CH_SSum** se používá pro počítání kontrolního součtu vysílače.

```
CH_Master  : Boolean;
```

Položka **CH_Master** definuje, je-li stanice zapojena v síti jako nadřazená jednotka (Master) nebo jako podřízená jednotka (Slave).

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 zavolá zděděný konstruktor **Init** (inherited Init) od rodičovského objektu tChnVirt a inicializuje položky objektu. Tělo konstruktoru vypadá následovně:

```
inherited Init;
CH_Type    := cName;
CH_Name    := CH_Type;
CH_NumName := ChNumName(CH_Type);
CH_RTick   := false;
CH_SBuff   := nil;
CH_MSBuff  := 0;
CH_LRMess  := 0;
CH_Master  := true;
IData      := 0;
```

5.1.2.2. ChInitParam konstruktor

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

Konstruktor **ChInitParam** je sloučením konstruktoru **Init** a metody **ChSetParam**. Slouží ke zkrácenému vytvoření instance komunikačního objektu s nastavením parametrů komunikace.

5.1.2.3. Done destruktor

```
destructor Done;
```

Destruktor **Done** slouží ke zrušení instance komunikačního objektu. Pokud je alokován vysílací buffer, je odstraněn z paměti. Na konci destruktoru je volána zděděná metoda **Done** od přímého rodičovského objektu pro uzavření podřízené komunikační vrstvy.

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 tChnTecom dekóduje tyto parametry:

MAS=MASTER / SLAVE

Parametrem **MAS** ("Master or Slave") se určuje, zda je jednotka v komunikační síti jako Master (nadřízená) nebo jako Slave (podřízená).

LSB=Size

Parametrem **LSB** ("Length of Send Buffer") je alokován nový vysílací buffer **CH_MSBuff** dané velikosti Size.

NOD=Node

Parametrem **NOD** ("Node") se určuje číslo (adresa) stanice **CH_Node** v komunikační síti. Pro nadřízenou jednotku (Master) může Node nabývat hodnot <0..126>. Pro podřízenou jednotku PLC (Slave) může nabývat hodnot <0..99>.

DNO=DNode

Parametrem **DNO** ("Destination Node") se určuje číslo (adresa) stanice **CH_DNode** v komunikační síti, které budou zprávy určeny. Tuto položku je možno také definovat prostřednictvím metody **ChDestNode**. Pro nadřízenou jednotku (Master) může DNode nabývat hodnot <0..99>. Pro podřízenou jednotku PLC (Slave) může nabývat hodnot <0..126>.

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. ChConnect procedura

```
procedure ChConnect;
```

Metoda **ChConnect** zavolá zděděnou metodu **ChConnect** od přímého rodičovského objektu a pokud nenastala žádná chyba, nastaví automat přijímače **CH_RCtrl** do počátečního stavu pro příjem zprávy v protokolu Tecomat.

5.1.2.7. ChDisConnect procedura

```
procedure ChDisConnect;
```

Metoda **ChDisConnect** zavolá zděděnou metodu **ChDisConnect** od přímého rodičovského objektu a pokud nenastala žádná chyba, nastaví automat přijímače **CH_RCtrl** do neaktivního stavu, aby se nepřijímaly žádné zprávy.

5.1.2.8. ChSend procedura

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

Metoda **ChSend** způsobí započetí vysílání zprávy podle protokolu Tecomat na podkladu záznamu typu tMaSendRecord nebo tSlSendRecord, na který ukazuje parametr Buff. Parametr Len udává délku vysílacího bufferu pro vysílání. Tento parametr je nutno správně zadat v případě dlouhé zprávy s daty a to následovně: Len = počet vysílaných bytů v poli Data + 3. V případě krátké zprávy bez datového pole či krátkého potvrzení se tento parametr ignoruje. Před voláním této metody se musí záznam správně naplnit daty (viz níže).

5.1.2.9. ChReceiveReady funkce

```
function  ChReceiveReady: tChState;
```

Metoda **ChReceiveReady** způsobí provedení kroku přijímacího automatu na základě volání metody **ChReceiveTick**. Jako svoji funkční hodnotu vrací aktuální stav automatu přijímače komunikačního kanálu, který je uložen v položce **CH_RCtrl**. Zpravidla se provádí test pouze na stabilní stav **CHS_ReceiveReady** (který znamená, že byla přijata nějaká zpráva), protože ostatní stavy jsou stavy probíhajícího příjmu.

5.1.2.10. ChReceive procedura

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

Metoda **ChReceive** provede přijmutí celé zprávy a její uložení do přijímacího bufferu, který svou vnitřní strukturou odpovídá datům záznamu typu tMaRecRecord nebo tSlRecRecord a byl definován metodou **ChReceiveBuffer**. Metoda naplní buffer pouze patřičnými položkami typu tMaRecRecord nebo tSlRecRecord, úvodní a zakončovací řídicí znaky a kontrolní součet ze zprávy metoda vyhodnotí a pro uživatele odstraní. Odpověď na zprávu, ať došla v pořádku nebo porušená, generuje uživatel sám pomocí metody **ChSend**.

5.1.2.11. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

Metoda **ChReceiveFlush** způsobí vyprázdnění přijímacích bufferů a nastavení stavu automatu přijímače na počátek příjmu zpráv v protokolu Tecomat.

5.1.2.12. ChGetNode procedura

```
procedure ChGetNode(var SNode, DNode : tNode);
```

Po volání metody **ChGetNode** je do proměnné **SNode** uloženo číslo (adresa) stanice, která zprávu odeslala, a do proměnné **DNode** číslo (adresa) stanice, pro kterou byla zpráva určena. Tuto metodu má smysl volat po přijetí zprávy metodou **ChReceive**.

5.1.2.13. ChReceiveTick procedura

```
procedure ChReceiveTick;
```

Metoda **ChReceiveTick** způsobí provedení jednoho či více kroků automatu přijímače. Je nutné ji periodicky volat při přijímání. Metoda **ChReceiveTick** je rovněž automaticky volána v metodě **ChReceiveReady**.

5.2. tAddChnTecom

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

6. Struktura zpráv protokolu TECOMAT

6.1. Obecná struktura zpráv

Krátká zpráva bez datového pole vysílaná a přijímaná stanicí Master i Slave.

SD1	DNO	SNO	FC	SUM	ED

Dlouhá zpráva s datovým polem vysílaná stanicí Master a přijímaná stanicí Slave.

SD2	LEN	LEN	SD2	DNO	SNO	FC	FC2	DATA...	SUM	ED

Dlouhá zpráva s datovým polem vysílaná stanicí Slave a přijímaná stanicí Master.

SD2	LEN	LEN	SD2	DNO	SNO	FC	DATA...	SUM	ED

Krátké pozitivní potvrzení od stanice Slave.

SACK

Význam položek:

SD1 - úvodní znak pro krátkou zprávu bez datového pole (start delimiter 1)
pevná hodnota $10

SD2 - úvodní znak pro dlouhou zprávu s datovým polem (start delimiter 2)
pevná hodnota $68

LEN - délka zprávy (počet bytů položek DNode+SNode+FC+(FC2)+DATA)

DNO - adresa cílové stanice (destination address)
hodnota pro PLC 0..99
hodnota pro nadřízený systém 0..126

SNO - adresa zdrojové stanice (source address)
hodnota pro nadřízený systém 0..126
hodnota pro PLC 0..99

FC - kód služby (frame control byte)
konkrétní hodnoty viz. výše

FC2 - vedlejší kód služby (second frame control byte)
konkrétní hodnoty viz. výše

DATA - vlastní tělo datové zprávy
maximálně 245 bytů

SUM - kontrolní součet (frame check sum)
bytový součet všech bytů položek DNode, SNode, FC, (FC2) a DATA se zanedbáním vyšších řádu vzniklých přenosem.

ED - koncový znak (end delimiter)
pevná hodnota $16

SACK - krátké potvrzení (short acknowledge)
pevná hodnota $E5

6.2. Struktura konkrétních zpráv

V následujících odstavcích jsou popsány způsoby nastavování jednotlivých položek ve strukturách vysílacích bufferů pro konkrétní příklady zpráv protokolu Tecomat.

6.2.1. Zpráva Connect - test připojení PLC

Master stanice vysílá krátkou zprávu s nastavenými položkami:

```
FC       := FCM_Connect;
```

Slave stanice odpovídá opět krátkou zprávou s nastavenými položkami:

```
MessType := tpShortM;
FC       := FCS_ConnectRep;
```

6.2.2. Zpráva Ident - identifikace připojeného PLC

Master stanice vysílá krátkou zprávu s nastavenými položkami:

```
FC       := FCM_Ident;
```

Slave stanice odpovídá dlouhou zprávou s nastavenými položkami:

```
MessType := tpLongM;
FC       := FCS_IdentRep;
IdentStr := ... např. 'CPM-1D'
ProfiBus := ... např. 'B'
VerHW    := ... např. '1.0'
VerSW    := ... např. '1.2'
```

6.2.3. Zpráva SetTID - nastavení systémového času

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Write;
FC2      := FC2_SetTID;
Year     := ... např. 98
Month    := ... např. 11
Day      := ... např. 3
Hour     := ... např. 10
Minute   := ... např. 45
Second   := ... např. 00
DayWeek  := ... např. 2
```

Slave stanice odpovídá krátkým potvrzením s nastavenými položkami:

```
MessType := tpSACK;
```

6.2.4. Zpráva SetCW - nastavení řídícího slova

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Write;
FC2      := FC2_SetCW;
CWL      := ... např. 00000000bin
```

```
CWH      := ... např. 10000001bin
```

Slave stanice odpovídá krátkým potvrzením s nastavenými položkami:

```
MessType := tpSACK;
```

6.2.5. Zpráva GetSW - přečtení stavového slova

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Read;
FC2      := FC2_GetSW;
```

Slave stanice odpovídá opět dlouhou zprávou s nastavenými položkami:

```
MessType := tpLongM;
FC       := FCS_Reply;
Data[0]  := ... např. 00000000bin
Data[1]  := ... např. 10000001bin
```

Délka vysílané zprávy se nastaví na 6 ($4_{hlavička}+2_{data}$).

Výnam položek pole Data:

Data[0] - dolní byte stavového slova

Data[1] - horní byte stavového slova

6.2.6. Zpráva GetErr - přečtení chybového zásobníku

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Read;
FC2      := FC2_GetErr;
```

Slave stanice odpovídá opět dlouhou zprávou s nastavenými položkami:

```
MessType := tpLongM;
FC       := FCS_Reply;
Data[0..31]:= ...
```

Délka vysílané zprávy se nastaví na 36 (4+32).

Pole Data se naplní daty chybového zásobníku, který obsahuje 8 chybových hlášení v pořadí od nejstaršího k nejnovějšímu, každé o velikosti 4 byte, celková délka pole je 32 byte.

6.2.7. Zpráva MaskCW - nastavení jednotlivých bitů řídícího slova

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Write;
FC2      := FC2_MaskCW;
C0L      := ... např. 00000000bin
C0H      := ... např. 00000000bin
C1L      := ... např. 00000000bin
C1H      := ... např. 10000000bin
```

Slave stanice odpovídá krátkým potvrzením s nastavenými položkami:

```
MessType := tpSACK;
```

6.2.8. Zpráva ReadN - čtení z datové paměti

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Read;
FC2      := FC2_ReadN;
Data[0]  := ... např. RegR
Data[1]  := ... např. $00
Data[2]  := ... např. $00
Data[3]  := ... např. 10
 :
Data[n+0]:= ...
```

```
Data[n+1]:= ...
Data[n+2]:= ...
Data[n+3]:= ...
```

Délka vysílané zprávy se nastaví na 4+4n.
Význam položek pole Data:

Data[0] - oblast zápisníku ze které se čte (viz. 4.4. Konstanty přístupu k datovým oblastem PLC Tecomatu)
Data[1] - dolní byte adresy prvního čteného registru
Data[2] - horní byte adresy prvního čteného registru
Data[3] - počet čtených registrů

Může se číst až "n" bloků dat. Celkový počet čtených registrů by neměl přesáhnout maximální možnou délku pole dat (MaxDataLen).

Slave stanice odpovídá opět dlouhou zprávou s nastavenými položkami:

```
MessType := tpLongM;
FC       := FCS_Reply;
Data[1..m]:= ...
```

Délka vysílané zprávy se nastaví na 4+m, kde "m" je celkový počet čtených registrů.
Pole Data se naplní daty příslušných registrů.

6.2.9. Zpráva WriteN - zápis do datové paměti

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Write;
FC2      := FC2_WriteN;
Data[0]  := ... např. RegR
Data[1]  := ... např. $00
Data[2]  := ... např. $00
Data[3]  := ... např. 5
Data[4..4+Data[3]] := ... např. $10,$20,$30,$40,$50
 :
Data[n+0]:= ...
Data[n+1]:= ...
Data[n+2]:= ...
Data[n+3]:= ...
Data[n+4..n+4+Data[3]] := ...
```

Délka vysílané zprávy se nastaví na 4+(4+počet dat)n.
Význam položek pole Data:

Data[0] - oblast zápisníku do které se zapisuje (viz. 4.4. Konstanty přístupu k datovým oblastem PLC Tecomatu)
Data[1] - dolní byte adresy prvního zapisovaného registru
Data[2] - horní byte adresy prvního zapisovaného registru
Data[3] - počet zapisovaných registrů
Data[4] až Data[4+Data[3]] - příslušná zapisovaná data

Může se zapisovat až "n" bloků dat. Celková délka pole Data nesmí přesáhnout maximální možnou délku pole dat (MaxDataLen).

Slave stanice odpovídá krátkým potvrzením s nastavenými položkami:

```
MessType := tpSACK;
```

6.2.10. Zpráva WandrN - zápis do a čtení z datové paměti

Pozn.: Tato zpráva je sloučením zpráv ReadN a WriteN s tím rozdílem, že se čte a zapisuje pouze jeden blok dat.

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Read;
FC2      := FC2_WandrN;
Data[0]  := ... např. RegR
Data[1]  := ... např. $00
Data[2]  := ... např. $00
Data[3]  := ... např. 10
Data[4]  := ... např. RegR
Data[5]  := ... např. $00
Data[6]  := ... např. $00
Data[7]  := ... např. 5
Data[8..Data[7]] := ... např. $01,$02,$03,$04,$05
```

Délka vysílané zprávy se nastaví na 4+4+4+Data[7].
Význam položek pole Data:

Data[0] - oblast zápisníku ze které se čte (viz. 4.4. Konstanty přístupu k datovým oblastem PLC Tecomatu)
Data[1] - dolní byte adresy prvního čteného registru
Data[2] - horní byte adresy prvního čteného registru
Data[3] - počet čtených registrů
Data[4] - oblast zápisníku do které se zapisuje (viz. 4.4. Konstanty přístupu k datovým oblastem PLC Tecomatu)
Data[5] - dolní byte adresy prvního zapisovaného registru
Data[6] - horní byte adresy prvního zapisovaného registru
Data[7] - počet zapisovaných registrů
Data[8] až Data[8+Data[7]] - příslušná zapisovaná data

Celkový počet čtených registrů by neměl přesáhnout maximální možnou délku pole dat (MaxDataLen) a celkový počet zapisovaných registrů nesmí přesáhnout hodnotu MaxDataLen-8.

Slave stanice odpovídá opět dlouhou zprávou s nastavenými položkami:

```
MessType := tpLongM;
FC       := FCS_Reply;
Data[1..m]:= ...
```

Délka vysílané zprávy se nastaví na 4+m, kde "m" je celkový počet čtených registrů.
Pole Data se naplní daty příslušných registrů.

6.2.11. Zpráva ReadB - čtení bitů z datové paměti

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Read;
FC2      := FC2_ReadB;
Data[0]  := ... např. RegR
Data[1]  := ... např. $00
Data[2]  := ... např. $00
Data[3]  := ... např. 2
 :
Data[n+0]:= ...
Data[n+1]:= ...
Data[n+2]:= ...
Data[n+3]:= ...
```

Délka vysílané zprávy se nastaví na 4+4n.
Význam položek pole Data:

Data[0] - oblast zápisníku ze které se čte (viz. 4.4. Konstanty přístupu k datovým oblastem PLC Tecomatu, tato služba však nedovoluje číst bity z přídavné paměti DataBox)
Data[1] - dolní byte adresy registru, ze kterého se čte požadovaný bit
Data[2] - horní byte adresy registru, ze kterého se čte požadovaný bit
Data[3] - index čteného bitu

Může se číst až "n" bitů. Celkový počet čtených bitů by neměl přesáhnout maximální možnou délku pole dat (MaxDataLen).

Slave stanice odpovídá opět dlouhou zprávou s nastavenými položkami:

```
MessType := tpLongM;
FC       := FCS_Reply;
Data[1..m]:= ...
```

Délka vysílané zprávy se nastaví na 4+m, kde "m" je celkový počet čtených registrů.
Pole Data se naplní hodnotami čtených bitů tak, že pokud čtený bit má hodnotu 0, nastaví se položka v poli Data na hodnotu $00, a pokud má čtený bit hodnotu 1, nastaví se položka v poli Data na hodnotu $FF.

6.2.12. Zpráva WriteB - zápis bitů do datové paměti

Master stanice vysílá dlouhou zprávu s nastavenými položkami:

```
FC       := FCM_Write;
FC2      := FC2_WriteB;
Data[0]  := ... např. RegR
Data[1]  := ... např. $00
Data[2]  := ... např. $00
Data[3]  := ... např. $82
 :
Data[n+0]:= ...
Data[n+1]:= ...
Data[n+2]:= ...
Data[n+3]:= ...
```

Délka vysílané zprávy se nastaví na 4+4n.
Význam položek pole Data:

Data[0] - oblast zápisníku do které se zapisuje (viz. 4.4. Konstanty přístupu k datovým oblastem PLC Tecomatu, tato služba však nedovoluje zapisovat bity do přídavné paměti DataBox)
Data[1] - dolní byte adresy registru, do kterého se bude bit zapisovat
Data[2] - horní byte adresy registru, do kterého se bude bit zapisovat
Data[3] - bity 0 až 6 určují index zapisovaného bitů (hodnota 0 až 7)
- bit 7 (nejvrchnější bit) určuje hodnotu zapisovaného bitu

Může se zapisovat až "n" bitů. Celková délka pole Data nesmí přesáhnout maximální možnou délku pole dat (MaxDataLen).

Slave stanice odpovídá krátkým potvrzením s nastavenými položkami:

```
MessType := tpSACK;
```

6.2.13. Zpráva ReadBD - destruktivní čtení bitů z datové paměti

Formát této zprávy je stejný se zprávou "ReadB" s tím rozdílem, že Master stanice nastavuje do položky FC2 konstantu ReadBD (místo konstanty ReadB).

6.2.14. Zpráva ReadND - destruktivní čtení z datové paměti

Formát této zprávy je stejný se zprávou "ReadN" s tím rozdílem, že Master stanice nastavuje do položky FC2 konstantu ReadND (místo konstanty ReadN).

6.2.15. Zpráva WandrND - zápis do a destruktivní čtení z datové paměti

Formát této zprávy je stejný se zprávou "WandrN" s tím rozdílem, že Master stanice nastavuje do položky FC2 konstantu WandrND (místo konstanty WandrN).

7. Příklad

Následující příklad ukazuje způsob vyslání zprávy Connect do PLC Tecomatu pro otestování připojení tohoto zařízení. Fyzický přenos se realizuje prostřednictvím knihovny ChnCom.

```
uses
  uString,
  ChnVirt,
  ChnCom,
  ChnTecom;

const
  ParamStr : tParamStr =
             'NAM=TECOM LSB=500 NOD=1 DNO=2 ' +
             'NAM=COM COM=1 IRQ=4 BD=9600 BIT=8 STOP=1 ' +
             'PAR=E LRB=1000';

var
  Chn   : pChnVirt;
  SMess : pMaSendRecord;
  RMess : pMaRecRecord;
  LSMess: word;
  LRMess: word;

begin
  ...
  New(SMess);
  New(RMess);
  ...
  { vytvoření instance Chn }
  Chn:=ChnCollection^.ChNewInit(ChnTecom.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 (např. zpráva Connect)}
    with SMess^ do
    begin
      FC:=FCM_Connect;
      LSMess:=1;
    end;
    { 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;
      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 správné odpovědi (např. odpověď na Connect)}
      with RMess^ do
      begin
        if (MessType = tpShortM) and
           (FC = FCS_ConnectRep) then
          writeln('Ok')
        else
          writeln('Chyba');
      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.
```