

# ChnSofs2

## KNIHOVNA IMPLEMENTUJÍCÍ FIREMNÍ PROTOKOL TYPU DF1 A DF2

Příručka uživatele a programátora



**SofCon<sup>®</sup> spol. s r.o.**  
Střešovická 49  
162 00 Praha 6  
tel/fax: +420 220 180 454  
E-mail: [sofcon@sofcon.cz](mailto: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.Činnost komunikačního objektu	6
4.1. Stavby komunikačního kanálu	7
4.2. Stavby přijímacího a vysílacího automatu	8
5.Konstanty a typy	11
6.Objekty	13
6.1. tChnSofs2	13
6.1.1. Položky	13
6.1.2. Metody	14
6.1.2.1. Init	14
6.1.2.2. ChInitParam	14
6.1.2.3. Done	14
6.1.2.4. ChSetOneParam	15
6.1.2.5. ChGetParam	15
6.1.2.6. ChOpen, ChClose	16
6.1.2.7. ChConnect	16
6.1.2.8. ChDisconnect	16
6.1.2.9. ChSend	17
6.1.2.10. ChSendReady	17
6.1.2.11. ChSendFlush	17
6.1.2.12. ChReceive	18
6.1.2.13. ChReceiveTick	19
6.1.2.14. ChReceiveReady	19
6.1.2.15. ChReceiveFlush	19
6.1.2.16. ChGetTestConnect	19
6.2. tAddChnSofs2	20
6.2.1. Metody	20
6.2.1.1. ChInit funkce	20
7.Příklad	20



## 1. O dokumentu

---

### 1.1. Revize dokumentu

---

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Hv		První vydání
1.01	5.XX	Hv	18.3.2003	Zpřesnění popisu komunikačního objektu a jeho metod
1.10	5.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Doplněné konstanty MaxRBuf a MaxSBuf. Doplněný popis LenMess a typů tTstDF1, tTstDF2, tMessDF1 a tMessDF2.

### 1.2. Účel dokumentu

---

Tento dokument slouží jako popis komunikační knihovny ChnSofs2 pro přenos dat pomocí firemního protokolu typu DF1 a DF2.

### 1.3. Rozsah platnosti

---

Určen pro programátory a uživatele programového vybavení SofCon.

### 1.4. Související dokumenty

---

Pro čtení tohoto dokumentu je nutné se orientovat v koncepci komunikačních objektů, popsána v manuálu ChnVirt.

Komunikační objekt implementovaný v této knihovně může být navázán na objekt typu tUDPPrt (popsán v manuálu UDPPrt) nebo tChnPrt (popsán v manuálu ChnPrt).

Implementovaný protokol je popsán v dokumentu ChnSof.

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 implementuje přenosy dat pomocí firemního protokolu typu DF1 a DF2 popsaném v manuálu ChnSof. Data se přenáší po síti MASTER-SLAVE.

K tomuto účelu knihovna definuje komunikační objekt tChnSofs2, který je potomkem objektu tChnVirt. Instance tohoto objektu představuje nejvyšší komunikační vrstvu, jejímž úkolem je bezpečně přenášet data. Proto tato knihovna už přímo implementuje jednotlivé automaty pro přenos dat a kontrolu navazování spojení. Pokud se tato knihovna použije v běžné aplikaci, chová se jako SLAVE stanice. V současné době je MASTER stanice implementována pouze pro operační systémy typu Windows.

Při používání této knihovny aplikace pouze přijímá a vysílá data. Vlastní knihovna pomocí tikání zajišťuje potvrzování dat a udržování navázaného spojení viz popis protokolu v manuálu ChnSof.

Protože knihovna je odvozena od objektu tChnVirt, je možno pomocí parametrizačního řetězce zadat spodní vrstvy komunikačních objektů. Z definice protokolu typu DF1 a DF2 je řečeno, že vlastní přenos probíhá pomocí protokolu DF0, který v současnosti implementují tyto knihovny:

- tUDPPrt (UDPPrt)      přenos dat probíhá po síti Ethernet pomocí protokolu UDP. Tato knihovna zajišťuje fyzický přenos dat, proto ji nelze zřetěžit s dalšími knihovnami.
- tChnPrt (ChnPrt)      přenos dat probíhá prostřednictvím komunikačního objektu, který je této knihovně přiřazen, např. pomocí RS485, RS232, telefonním, GSM nebo rádiovým modem.

V knihovně je dále implementován objekt tAddChnSofs2, který je potomkem objektu tAddChnVirt. Úkolem tohoto objektu je, aby po přilinkování této jednotky do aplikace (příkazem „**uses ChnSofs2**“), objekt tChnSofs2 zařadil do seznamu správce komunikačních objektů. Později při zpracování parametrizačního řetězce tento správce zajistí automatické vytvoření a propojení instance tohoto objektu s dalšími komunikačními knihovnami. Takto propojené instance objektů se potom nazývají řetězec komunikačních knihoven.

Protože objekt tChnSofs2 je dědicem objektu tChnVirt, jsou v této příručce popsány pouze předefinované metody a některé vlastnosti typické pro tuto komunikaci případně objekt. Nepopsané metody a konstanty najdete v samostatných manuálech používaných jednotek nebo rodičovských objektů.

---

### 4. Činnost komunikačního objektu

---

V další části této kapitoly budou popsány jednotlivé stavy, kterými prochází jednotlivé automaty objektu tChnSofs2. V diagramech bude použita následující symbolika:

- Silnými čarami budou znázorněny přechody vyvolané metodami nad touto šipkou. Poněvadž tyto metody mohou způsobit operace trvající delší dobu případně

skončit chybou, měla by se vždy provádět kontrola chyb a tikání automatu pomocí dále uvedených metod.

- Slabými čarami budou znázorněny přechody po dosažení požadovaných podmínek.
- Kroužek se silnou čarou znamená stabilní stav.
- Kroužek se slabou čarou znamená nestabilní stav tj. stav, kterým se prochází po určité době než jsou splněny požadované podmínky pro přechod do stabilního stavu.

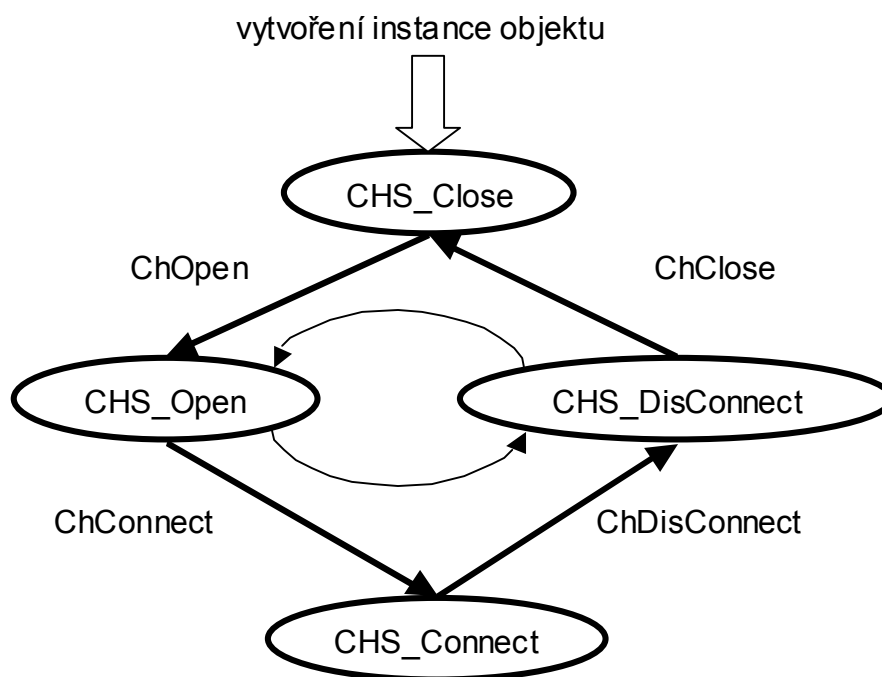
Pokud je metoda zavolána v neplatném stavu, automat zůstává ve stejném stavu a vrací se chyba.

#### 4.1. Stavy komunikačního kanálu

Na následujícím obrázku jsou znázorněny stavy kanálu, které vrací metoda ChReady. Stabilní stavy vrací metody ChState. Interně tyto funkce volají metodu ChTick, která při splnění požadovaných podmínek provede přechod do dalšího stavu tzn. tika automatem komunikačního kanálu. V opačném případě automat zůstane ve stejném stavu.

Dále budou popsány stavy automatu komunikačního kanálu:

CHS_Close	objekty spodních vrstev neexistují a lze nastavit všechny jejich vlastnosti. V tomto stavu nelze vysílat ani přijímat data.
CHS_Open, CHS_DisConnect	objekty spodních vrstev jsou vytvořeny a lze nastavit některé jejich vlastnosti. V tomto stavu nelze vysílat ani přijímat data.
CHS_Connect	objekty spodních vrstev jsou vytvořeny a lze nastavit některé jejich vlastnosti. V tomto stavu lze jak vysílat tak přijímat data.



## 4.2. Stavy přijímacího a vysílacího automatu

V této části budou ukázány jak stavy tak přechody mezi nimi u vysílacích a přijímacích automatů komunikačního objektu. Protože tyto popisy popisují pouze realizaci, je nutné pro úplné pochopení těchto automatů nahlédnout do manuálu definujícím protokoly DF1 a DF2 viz ChnSof. V tomto manuálu také naleznete používanou terminologii.

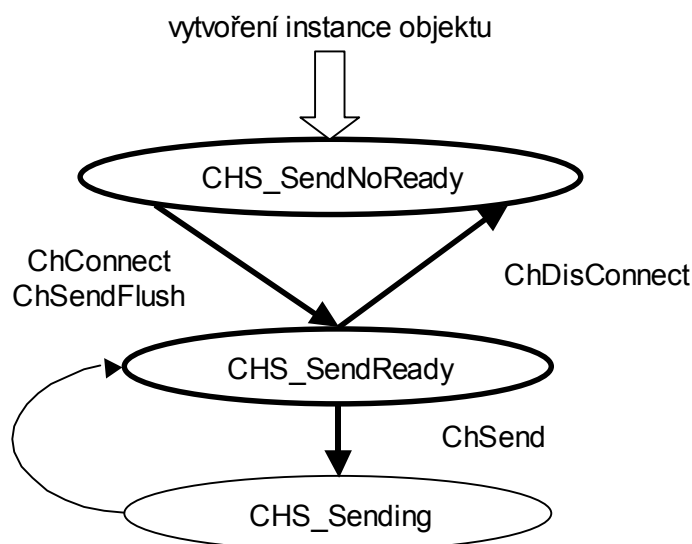
Vysílací automat lze popsat třemi stavy, mezi kterými se přechází dle činnosti přijímacího automatu. Stav vysílacího automatu se zjišťuje pomocí metody ChSendReady. Přechod zobrazený slabou čarou je vyvolán přijímacím automatem na základě příjmu potvrzení odvysílané zprávy.

Pozn. Odvysílání zprávy se vždy provede až po příjmu rámce od MASTER do té doby je tato zpráva uložena v bufferu. Po jejím odvysílání se čeká na rámec potvrzující přijetí této zprávy protistanicí. Pokud v dalším rámci nepřijde potvrzení, zpráva se zopakuje.

Dále budou popsány stavy vysílacího automatu:

CHS_SendNoReady	v tomto stavu je kanál ve stavu různém od CHS_Connect
CHS_SendReady	v tomto stavu je kanál ve stavu CHS_Connect a je připraven odvysílat zadanou zprávu
CHS_Sending	v tomto stavu je kanál ve stavu CHS_Connect a vysílá zadanou zprávu nebo čeká na potvrzení jejího příjmu protistanicí



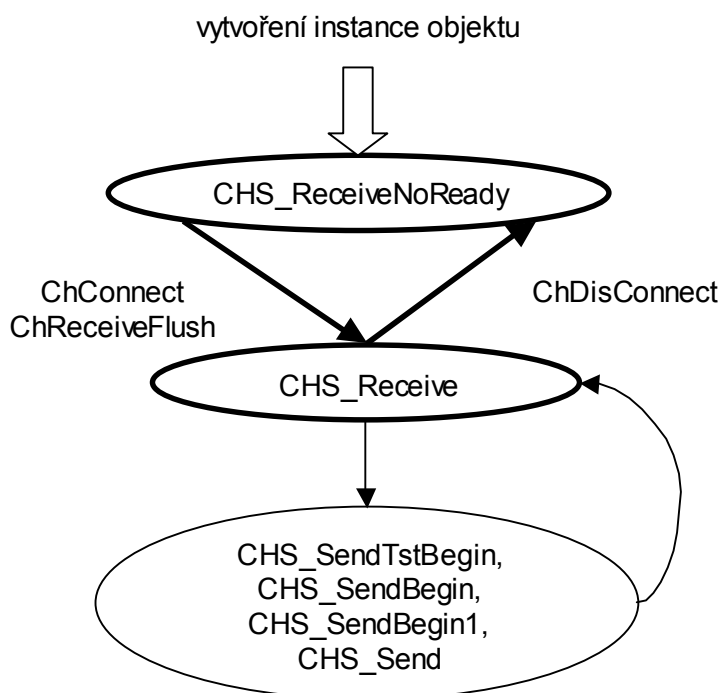


V další části bude popsán přijímací automat, který se skládá z těchto stavů:

CHS_ReceiveNoReady	v tomto stavu je kanál buď ve stavu různém od CHS_Connect nebo není žádná zpráva přijata
CHS_ReceiveReady	tento stav vrací metoda ChReceiveReady, jestliže byla přijata platná zpráva. Přijímací automat se do tohoto stavu nikdy nedostane.
CHS_Receive	v tomto stavu se provádí příjem zprávy
CHS_SendTstBegin	v tomto stavu se připraví rámec, v kterém se určí typ používaného protokolu DF1 a DF2
CHS_SendBegin1	v tomto stavu se připraví rámec, který obsahuje zprávu určenou k odvysílání případně pouze potvrzení přijaté zprávy
CHS_SendBegin	v tomto stavu se kontroluje stav vysílacího automatu spodních vrstev, zda lze vyslat další zprávu
CHS_Send	v tomto stavu proběhne vlastní odvysílání zprávy

Přechody mezi základními stavy přijímacího automatu lze zobrazit následovně. Metoda ChReceive vrací tyto stavy:

- CHS\_ReceiveReady                      pokud byla v přijímacím automatu přijata nějaká zpráva.
- CH\_RCtrl (výše popsané stavy)      pokud žádná zpráva nebyla přijata.



Vlastní činnost přijímacího automatu lze symbolicky popsat takto:

Pozn.

1. CH\_RCtrl se používá pro uložení aktuálního stavu automatu přijímače.

#### ▪ CHS\_Receive

```

if (rámeček není typu DF1, DF2 ani TST) then
begin
  CH_RCtrl=CHS_SendTstBegin
  goto CHS_SendTstBegin
end;
if (rámeček typu TST) then
begin
  CH_RCtrl=CHS_SendTstBegin
  goto CHS_SendTstBegin
end;
if (rámeček obsahuje DATA) then
begin
  if (přiját nový rámeček) then
  begin
    if (přijatý rámeček se musí potvrdit) then
      <uložení potvrzení paketu>
    end
  end;
if (čeká se na potvrzení odvysílané zprávy) then
begin
  if (data jsou potvrzena) then
  begin
    CH_RCtrl=CHS_SendBegin1
    exit
  end
  else
  begin
    CH_RCtrl=CHS_SendBegin1
    goto CHS_SendBegin1
  end
end
end

```

```

else
begin
  CH_RCtrl=CHS_SendBegin1
  exit
end

```

- **CHS\_SendTstBegin**

```

čekání na uvolnění vysílače spodních vrstev
příprava rámce
odvysílání rámce
CH_RCtrl=CHS_Send
goto CHS_Send

```

- **CHS\_SendBegin1**

```

if (bude se vysílat nová zpráva) then
begin
  příprava rámce
  CH_RCtrl=CHS_SendBegin
  goto CHS_SendBegin
end
else
begin
  if (bude se vysílat nějaká zpráva) then
  begin
    if (bude se vysílat pouze potvrzení přijatého rámce) then
      příprava zprávy s potvrzením přijatého rámce
    else
      zopakování nepotvrzené zprávy
    CH_RCtrl=CHS_SendBegin
    goto CHS_SendBegin
  end
  else
  begin
    CH_RCtrl=CHS_Receive
  end;
end

```

- **CHS\_SendBegin**

```

čekání na uvolnění vysílače spodních vrstev
odvysílání rámce
CH_RCtrl=CHS_Send
goto CHS_Send

```

- **CHS\_Send**

```

čekání na uvolnění vysílače spodních vrstev
if (spodní vrstvy odvysílaly rámeček) then
begin
  CH_RCtrl=CHS_Receive
  CH_SCtrl=CHS_SendReady
end

```

## 5. Konstanty a typy

---

```

cVerNo = např $0500;
cVer   = např. '05.00,11.12.2002';

```

Konstanty udávají verzi a poslední změnu jednotky ve standardním formátu definovaném v knihovně LibVer.

```

cName = 'SOFS2';

```

Konstanta obsahuje název jednotky a zároveň určuje jméno objektu v parametrizačním řetězci, pomocí kterého lze nastavit parametry této jednotky.

MaxRBuf = 32750;

konstanta definuje maximální velikost přijímacího bufferu.

MaxTBuf = 65500;

konstanta definuje maximální velikost vysílacího bufferu.

### Chybové kódy

Res\_ErrLen = \$00C3;

Tato chyba se vrací, pokud se při příjmu zprávy metodou ChReceive zjistí, že přijatá zpráva je delší než buffer nastavený metodou ChReceiveBuffer.

Další chybové kódy najdete v popisu použitých komunikačních objektů – např. UDPPrt, ChnPrt, ChnCom, ChnSoft a ChnVirt.

LenMess = 2300

Konstanta definuje maximální velikost datové části. Je definovaná v jednotce ChnSoft.

### Deklarace typů zpráv

pTstDF1 = ^tTstDF1;

tTstDF1 = record

Cmd : Byte; { identifikace zprávy }

end;

pTstDF2 = ^tTstDF2;

tTstDF2 = record

Len1 : Word; { délka tMessSof - (mění se dle Rec) }

Dummy : Byte; { řádek }

Dest : tAdr; { adresa cíle }

Sour : tAdr; { adresa zdroje }

{ \$ifdef MnAn }

Mno : Byte; { číslo zprávy }

Ano : Byte; { číslo poslední zprávy }

{ \$endif }

Cmd : Byte; { identifikace zprávy }

end;

Typy **tTstDF1** a **tTstDF2** deklarují typy interních zpráv přenášených protokolem SofCon typu DF1 nebo DF2 pro testování připojení jednotky Slave. Tyto typy jsou definované v jednotce ChnSoft.

pMessDF1 = ^tMessDF1;

tMessDF1 = record

Cmd : Byte; { identifikace zprávy }

Rec : tMess; { vlastní přenášená data nebo hlavička }

end;

pMessDF2 = ^tMessDF2;

tMessDF2 = record

Len1 : Word; { WIN tProcMess }

Dummy : Byte; { délka tMessSof - (mění se dle Rec) }

Dest : tAdr; { Radek }

Sour : tAdr; { adresa cíle }

{ \$ifdef MnAn }

Mno : Byte; { číslo zprávy }

Ano : Byte; { číslo poslední zprávy }

{ \$endif }

Cmd : Byte; { identifikace zprávy }

Rec : tMess; { vlastní přenášená data }

end;

Typy **tMessDF1** a **tMessDF2** deklarují interní typy zpráv přenášených protokolem SofCon typu DF1 nebo DF2 pro přenos dat mezi stanicemi Slave a Master. Tyto typy uvnitř metod objektu používají pro přetypování. Jsou definované v jednotce ChnSoft.

## 6. Objekty

---

### 6.1. tChnSofs2

---

#### 6.1.1. Položky

Dále uvedené položky slouží pro vnitřní použití a neměly by být proto nastavovány přímo, jestliže nebude uvedeno jinak.

CH\_LRBuff : word;  
Položka obsahuje velikost přijaté zprávy v byte včetně hlavičky DF1 .

CH\_LRMess : Word;  
Položka obsahuje velikost přijaté zprávy v byte bez hlavičky DF1.

CH\_SBuff : Pointer;  
Položka obsahuje ukazatel na vysílací buffer, do kterého se přesune vysílaná zpráva rozšířená o hlavičku DF1.

CH\_MSBuff : Word;  
Položka obsahuje délku vysílacího bufferu v bytech.

CH\_LSBuff : Word;  
Položka obsahuje celkovou délku vysílaného rámce včetně hlavičky.

CH\_SMess : Pointer;  
Položka obsahuje ukazatel na buffer obsahující vysílanou zprávu.

CH\_LSMess : Word;  
Položka obsahuje velikost bufferu obsahující vysílanou zprávu v bytech.

CH\_SCODE : Byte;  
Pomocná položka pro sestavení vysílaného rámce. Odpovídá položce CODE v definici protokolu.

CH\_SPacket : Byte;  
Položka obsahuje číslo naposledy odvysílaného rámce.

CH\_RCODE : Byte;  
Pomocná položka pro dekódování přijatého rámce. Odpovídá položce CODE v definici protokolu.

CH\_RPacket : Byte;  
Položka obsahuje číslo naposledy přijatého rámce.

CH\_UseDF2 : Boolean;  
Příznak, který určuje typ používaného rámce. Implicitně je nastaven na TRUE – tzn. používá se DF2.

CH\_RNewBuff : Boolean;  
Položka obsahuje příznak, že v přijímacím bufferu je nová zpráva.

CH\_SNewBuff : Boolean;  
Položka obsahuje příznak, že v vysílacím bufferu je uložena zpráva čekající na potvrzení příjmu od protistanice.

CH\_RTick : Boolean;  
Položka se používá jako příznak o vykonávání činnosti přijímacího automatu.

CH\_TstConnect : Boolean;  
Položka se používá jako příznak o navázaném spojení s MASTER stanicí.

```
CH_SMessTst      : tTstCMD;
```

V položce je instance bufferu pro posílání rámce typu TST.

## 6.1.2. Metody

Neuvedené metody jsou zděděny od předka tChnVirt a jejich popis najdete v samostatném manuálu ChnVirt. Pokud je komunikační objekt zřetězen s dalšími komunikačními objekty, způsobí volání metody předka ve většině případů volání stejné metody u všech komunikačních objektů na spodních vrstvách.

### 6.1.2.1. Init

```
constructor Init;
```

Konstruktor slouží k vytvoření a inicializaci instance komunikačního objektu. Ve svém těle volá nejdříve konstruktor předka tChnVirt a poté inicializuje své položky.

Po ukončení volání jsou položky nastaveny na následující hodnoty:

```
CH_Type      := 'Sofs2';
CH_Name      := 'Sofs2';
CH_NumName   := ChNumName(CH_Type);
CH_LRBuff   := 0;
CH_LRMess    := 0;
CH_SBuff     := nil;
CH_MSBuff    := 0;
CH_LSBuff    := 0;
CH_SMess     := nil;
CH_LSMess    := 0;
CH_SCODE     := 0;
CH_SPacket   := 1;
CH_RCODE     := 0;
CH_RPacket   := 0;
CH_RNewBuff  := False;
CH_SNewBuff  := False;
CH_RTick     := False;
CH_TstConnect := False;
{$ifdef DF1}
CH_UseDF2    := False; { komunikace probiha pomoci DF1 }
{$endif}
{$ifdef DF2}
CH_UseDF2    := True;  { komunikace probiha pomoci DF2 }
{$endif}
```

Další položky jsou nastaveny podle použitých komunikačních objektů na nižších vrstvách.

### 6.1.2.2. ChInitParam

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

Konstruktor slouží ke zkrácenému vytvoření instance komunikačního objektu s nastavením parametrů komunikace. V těle se nejdříve volá konstruktor Init a poté metoda ChSetParam.

### 6.1.2.3. Done

```
destructor Done; virtual;
```

Destruktor Done slouží ke zrušení instance objektu. V těle metody se nejdříve provede uvolnění alokované paměti pro vysílací a přijímací buffer. Nakonec se zavolá destruktory předka.

### 6.1.2.4. ChSetOneParam

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

Metoda slouží k dekodování a nastavení jednoho konkrétního parametru, který je zadán v parametru S. Tato metoda je volána při zpracování konfiguračního řetězce metodou ChSetParam. V případě chyby se jeho zpracování přeruší a funkce se ukončí.

Po jejím ukončení by se vždy měla volat funkce ChResult, která vrací Res\_Ok, pokud nedošlo k žádné chybě. V opačném případě vrací Res\_ErrParamStr.

V objektu tChnSofs2 se touto metodou mohou nastavit následující parametry:

#### **LSB=Size**

Zadáním parametru LSB (Length of Send Buffer) dojde k alokování nového vysílacího bufferu CH\_SBuff dané velikosti Size. Pokud předtím existoval nějaký buffer, je nejdříve uvolněn. Alokovaný buffer slouží pro transformaci vysílané zprávy, která je poté předána spodním vrstvám k odvysílání. Při transformaci je ke zprávě přidána hlavička protokolu DF1. Velikost bufferu může nabývat hodnot 1 až 65500 byte.

#### **LRB=Size**

Zadáním parametru LRB (Length of Receive Buffer) dojde k alokování nového přijímacího bufferu CH\_RBuff dané velikosti Size. Pokud předtím existoval nějaký jiný buffer, je nejdříve uvolněn. Alokovaný buffer slouží pro příjem přijaté zprávy ze spodních vrstev. Z tohoto bufferu se po volání metody ChReceive překopíruje položka DATA protokolu DF1 do připraveného bufferu CH\_RMess zadaném metodu ChReceiveBuffer. Velikost bufferu může nabývat hodnot 1 až 32750 byte.

#### **DF2=Flag**

Tímto parametrem se určuje typ používaného protokolu. Parametr může nabývat hodnot 0 a 1. Pokud je zadána 0, používá se protokol typu DF1. V opačném případě se používá protokol typu DF2. Tento parametr lze měnit pouze, pokud je kanál v uzavřeném stavu, CHS\_Close. Implicitně se používá protokol typu DF2.

Příklad:

Příklad ukazuje, jak je možné nastavit parametry komunikačního objektu tChnSofs2 - LSB na hodnotu 1000, LRB na hodnotu 2000 a objektu tChnPRT - NODE na hodnotu 20 a DNODE na hodnotu 30. Pro jejich nastavení se musí volat metoda ChSetParam, která při zpracování jednotlivých parametrů volá metodu ChSetOneParam příslušného objektu.

```
ChSetParam('NAM=SOFS2 LSB=1000 LRB=2000 NAM=PRT NOD=20 DNO=30');
```

### 6.1.2.5. ChGetParam

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

Tato metoda vrací nastavené parametry řetězce komunikačních objektů od objektu tChnSofs2 po parametry komunikačních objektů na spodních vrstvách. Seznam parametrů objektu tChnSofs2 je uveden v popisu metody ChSetOneParam. Další parametry jsou závislé na použitých komunikačních objektech. Popis jejich parametrů najdete v příslušných manuálech.

### 6.1.2.6. ChOpen, ChClose

```
procedure ChOpen; virtual;
procedure ChClose; virtual;
```

Metody ChOpen a ChClose jsou zděděny od předka tChnVirt a slouží pro inicializaci a deinicializaci komunikačního kanálu. Po úspěšném dokončení funkce ChOpen nelze přijímat ani vysílat data.

Pozn. Prováděné inicializace a deinicializace závisí na komunikačních objektech ve spodních vrstvách.

Doporučené volání těchto funkcí:

```
ChClose; {ChOpen}
if ChResult <> res_Ok then Exit;
theCommTimer.SaveTime; { objekt tTimer - TIMEOUT operace }
while ChReady<>CHS_Close {CHS_Open} do
begin
  if ChResult <> res_Ok then Exit;
  if theCommTimer.TstTime(wReadyTime) then exit;
  Wait(1); { pouze v pripade pouziti ReTOS }
end;
```

### 6.1.2.7. ChConnect

```
procedure ChConnect; virtual;
```

Metoda slouží pro nastavení komunikačního kanálu do stavu, ve kterém lze přijímat a vysílat data.

V těle metody se nejdříve zavolá metoda předka tChnVirt. Pokud nedošlo k žádné chybě tj. ChResult vrací Res\_Ok, přiřadí se komunikačnímu objektu buffer pro příjem zpráv. Dále se provede nastavení automatu přijímače do stavu CHS\_Receive a automatu vysílače do stavu CHS\_SendReady.

Pozn. Nastavení kanálu závisí na komunikačních objektech ve spodních vrstvách.

Doporučené volání této funkce:

```
{ pripojeni komunikacniho kanalu }
ChConnect;
if ChResult <> res_Ok then Exit;
theCommTimer.SaveTime; { objekt tTimer - TIMEOUT operace }
while ChReady<>CHS_Connect do
begin
  if ChResult <> res_Ok then Exit;
  if theCommTimer.TstTime(wReadyTime) then exit;
  Wait(1); { pouze v pripade pouziti ReTOS }
end;
```

### 6.1.2.8. ChDisconnect

```
procedure ChDisconnect; virtual;
```

Metoda slouží pro nastavení komunikačního kanálu do stavu, ve kterém nelze přijímat a vysílat data.

V těle metody se nejdříve zavolá metoda předka tChnVirt. Poté se nastaví automat přijímače do stavu CHS\_ReceiveNoReady a automat vysílače do stavu CHS\_SendNoReady.

Pozn. Nastavení kanálu závisí na komunikačních objektech ve spodních vrstvách.



Doporučené volání této funkce:

```

ChDisconnect;
if ChResult<>res_Ok then exit;
theCommTimer.SaveTime; { objekt tTimer - TIMEOUT operace }
while ChReady<>CHS_DisConnect do
begin
  if ChResult<>res_Ok then exit;
  if theCommTimer.TstTime(wReadyTime) then exit;
  Wait(1); { pouze v pripade pouziti ReTOS }
end;

```

### 6.1.2.9. ChSend

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

Metoda slouží pro určení zprávy, která se má odvíšlat ve vhodném stavu komunikačních automatů určeném definicí protokolu viz ChnSof.

Před voláním metody musí být kanál připojen např. ChState vrací CHS\_Connect a automat vysílače musí být ve stavu CHS\_SendReady. Pokud jsou tyto podmínky splněny, zapamatuje se ukazatel a velikost vysílané zprávy. Automat vysílače přejde do stavu CHS\_Sending.

Pozn. Zpráva je úspěšně odvíšlána tj. ChSendReady vrací CHS\_SendReady, až se obdrží potvrzovací rámec od MASTER stanice viz definice protokolu v manuálu ChnSof.

Doporučené volání této funkce je uvedeno u metody ChReceive.

### 6.1.2.10. ChSendReady

```
function ChSendReady:TChState; virtual;
```

Metoda vrací stav automatu vysílače, který je závislý na stavu automatu přijímače viz definice protokolu ChnSof. Pokud kanál není připojen, např. ChState nevrací CHS\_Connect, vrací se vždy CHS\_SendNoReady. V opačném případě se vrací stav automatu vysílače.

Funkce se používá pro zjištění, jestliže už byla zpráva odvíšlána, po odvíšláni vrací CHS\_SendReady.

### 6.1.2.11. ChSendFlush

```
procedure ChSendFlush; virtual;
```

Před voláním této metody musí být kanál připojen, např. ChState vrací CHS\_Connect. Pokud je tato podmínka splněna, provede se volání metody ChSendFlush předka tChnVirt. Poté se stav automatu vysílače nastaví na CHS\_SendReady a chybový stav vysílače na Res\_Ok.

Funkce se používá pro ukončení vysílání a nastavení vysílacího automatu do výchozího stavu, tzn. do stavu po volání metody ChConnect.

### 6.1.2.12. ChReceive

procedure ChReceive(var Len: Word); virtual;

Před voláním této metody musí být kanál připojen, např. ChState vrací CHS\_Connect. Pokud je v přijímacím bufferu uložena nějaká zpráva, je tato zpráva překopírována do bufferu nastaveném metodou ChReceiveBuffer. Do tohoto bufferu se nastaví DATA dle protokolu DF1. (Pozn. tato data obsahují hlavičku protokolu typu DF2, kterou by měla zpracovat vlastní aplikace.) Jestliže je přijatá zpráva delší než nastavený buffer, vrací se chyba Res\_ErrLen. V případě že v přijímacím bufferu není uložena žádná zpráva, vrací se Len=0. Protože při příjmu rámce může dojít k chybě, musí se po metodě ChReceiveReady volat metoda ChReceiveResult, která může vracet chybové kódy dle použitých komunikačních objektů.

Doporučené volání metody ChReceive a metody ChSend:

```
{ cekani dokud neni vysilani a prijem READY - viz manual ChnSof}
if (ChReceiveReady=CHS_ReceiveReady)and
  (ChSendReady =CHS_SendReady )then
begin
  { příjem zpravy }
  ChReceive(Len);
  if ChReceiveResult<>res_Ok then
  begin
    { zpracovani chyby pri prijmu - ovlivnuje vysilac }
    ChReceiveFlush; if ChReceiveResult<>res_Ok then ;
    ChSendFlush;   if ChSendResult  <>res_Ok then ;
    Len:=0;
  end;
  if Len<>0 then
  begin
    { prijata zprava se musi dekodovat a osetrit }
    { priprava parametru zpravy pro vysilani }
    if m_fUseDF2 then { zpracovani hlavicky DF2 }
    with pMessDF2(pSMess)^ do
    begin
      Dummy:=pMessDF2(pRMess)^.Dummy;
      { nastaveni identifikatoru procesu, instance procesu,
        cil - jedinecna logicka adresa }
      Dest :=pMessDF2(pRMess)^.Sour;
      { nastaveni identifikatoru procesu, instance procesu,
        zdroj - jedinecna logicka adresa }
      Sour :=pMessDF2(pRMess)^.Dest;
      {$ifdef MnAn}
      Mno :=pMessDF2(pRMess)^.Mno;   { cislo zpravy }
      Ano :=pMessDF2(pRMess)^.Ano;   { cislo posledni zpravy }
      {$endif}
      Len1 :=0;
    end;
    {
      zpracovani zpravy a priprava odpovedi
    }
    { vyslani odpovedi }
    ChSend(pSMess, _MessLen);
    if ChSendResult<>res_Ok then
    begin
      { zpracovani chyby pri vysilani }
      ChSendFlush; if ChSendResult<>res_Ok then ;
    end
  end
  else
  begin
    if ChReceiveResult<>res_Ok then
    begin
      ChReceiveFlush; if ChReceiveResult<>res_Ok then ;
    end
  end
end
```

```

end;
if ChSendResult<>res_Ok then
begin
  ChSendFlush;    if ChSendResult  <>res_Ok then ;
end;
end;
{ je pripojena MASTER stanice }
if ChGetTestConnect then
begin
  { nastaveni priznaku pripojene MASTER stanice }
end
else
begin
  { neuplynul timeout odposledniho pripojeni k MASTER stanici}
  if theCommTimer.TstTime(wPCConnectTime) then { objekt tTimer}
  begin
    { nastaveni priznaku odpojene MASTER stanice }
  end;
end;
end;
end;

```

### 6.1.2.13. ChReceiveTick

procedure ChReceiveTick; virtual;

V metodě je implementován automat pro příjem a vysílání dat dle definice protokolu viz manuál ChnSof. Metoda je určena pro interní použití a je volána z metody ChReceiveReady.

### 6.1.2.14. ChReceiveReady

function ChReceiveReady:TChState; virtual;

Metoda způsobí provedení jednoho kroku přijímacího automatu voláním metody ChReceiveTick. Pokud je přijata nějaká zpráva, vrací se CHS\_ReceiveReady. V opačném případě se vrací stav automatu CH\_RCtrl.

Funkce se používá pro zjištění, jestliže už byla nějaká zpráva přijata, v tomto případě vrací CHS\_ReceiveReady.

### 6.1.2.15. ChReceiveFlush

procedure ChReceiveFlush; virtual;

Před voláním této metody musí být kanál připojen, např. ChState vrací CHS\_Connect. Pokud je tato podmínka splněna, provede se volání metody ChReceiveFlush předka tChnVirt. Poté se stav automatu přijímače nastaví na CHS\_Receive a chybový stav přijímače na Res\_Ok.

Funkce se používá pro ukončení příjmu a nastavení přijímacího automatu do výchozího stavu, tzn. do stavu po volání metody ChConnect.

### 6.1.2.16. ChGetTestConnect

function ChGetTestConnect:Boolean;

Touto metodou lze zjistit, zda je navázáno spojení s MASTER stanicí. Protože spojení se zjišťuje na základě příjmu rámce od MASTER stanice a po volání této funkce se stav spojení nastaví na FALSE, může nastat tato situace. Při prvním volání této funkce se vrátí TRUE, ale při druhém volání už FALSE, protože mezi voláními nepřišel žádný rámec od MASTER stanice. Proto je doporučováno používat tuto funkci ve spojení s měřením času pro určení TIMEOUT.

## 6.2. tAddChnSofs2

---

Typ tAddChnSofs2 je typem objektu, který slouží k definování prvku v seznamu správce komunikačních objektů. Objekt tAddChnSofs2 je potomkem objektu tAddChnVirt.

### 6.2.1. Metody

#### 6.2.1.1. ChInit funkce

```
function ChInit: pChnVirt;
```

Tato metoda slouží k vytvoření instance komunikačního objektu tChnSofs2 a vrácení jeho ukazatele pro pozdější zpracování.

## 7. Příklad

---

---

Ideový příklad volání jednotlivých metod komunikačního objektu je uveden v knihovně ChnVirt. Protože v knihovně ChnSofs2 je implementován automat potvrzování, měl by se pro příjem a vysílání zpráv použít kód uvedený v metodě ChReceive knihovny ChnSofs2.