ChnPrt

JEDNOTKA DEFINUJÍCÍ KOMUNIKAČNÍ PROTOKOL

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

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	19.05.2003	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.

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 ChnPrt definuje komunikační objekt **tChnPrt**, který je dědicem od rodičovského objektu tChnvirt. Instance objektu tChnPrt 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. Objekt tChnPrt definuje formát protokolu, který zajišťuje zabezpečení dat, vkládání a vyjímání nadbytečností. Přenášený protokol je binární a obsahuje adresu adresáta i odesílatele. Umožňuje zaslat zprávu všem připojeným stanicím najednou. Je vhodný pro vytváření sítí s rozhraním typu RS485, RS232, telefonní modem, radiový modem apod. Fyzický přenos dat je zajištěn prostřednictvím nižší komunikační vrstvy. Určení, přes jakou fyzickou komunikační vrstvu bude komunikace probíhat, je voleno až parametry nastavovací metody ChSetParam.

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

Protože je objekt **tChnPrt** dědicem rodičovského komunikačního objektu **tChnVirt**, jsou v této příručce popsány jen odlišnosti a speciality pro tento druh sériové komunikace. Ostatní naleznete v příručce popisující objekt **tChnVirt**. 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  = 'PRT';
```

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

```
SOH    = $01;
DLE    = $10;
ETX    = $03;
```

Konstanta **SOH** definuje kód znaku pro začátek zprávy.

Konstanta **DLE** definuje kód znaku, který slouží pro rozlišení řídících znaků zprávy od ostatních.

Konstanta **ETX** definuje kód znaku pro konec zprávy.

```
MaxTBuf = 32750;
```

Maximální velikost vysílacího bufferu.

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_ErrCrc = $21;`
Chyba kontrolního součtu CRC16 - přijatou zprávu nelze akceptovat.

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

`res_ErrSOH = $25;`
Chyba - přijat SOH v průběhu zprávy - přijatou zprávu lze akceptovat, ale předchozí se ztratila.

`res_ErrETX = $26;`
Chyba - nepřijat ETX na konci zprávy - přijatou zprávu nelze akceptovat.

5. Definice přenášeného protokolu

1	1	1	2	LEN	2	1	velikost v byte
SOH	DNODE	NODE	LEN	DATA	CRC	ETX	

Význam položek:
SOH - začátek zprávy (sekvence znaků DLE, SOH)
DNODE - adresa cílové stanice (destination node)
NODE - adresa zdrojové stanice (node)
LEN - délka vysílaného pole DATA
DATA - pole dat
CRC - zbytek po dělení cyklickým polynomem CRC16, $x^{16}+x^{15}+x^2+1$, který je generován z položek zprávy SOH až poslední byte pole DATA
ETX - konec zprávy (sekvence znaků DLE, ETX)

U položek **LEN** a **CRC** je zasílán nižší byte jako první.

Velikost přenášeného pole dat **DATA** je omezena na 32734 byte.

Pokud některý z bytů položek **DNODE** až poslední byte **CRC** je roven hodnotě znaku DLE, je tento znak vyslán dvakrát (DLE, DLE). Tím je zajištěno rozlišení, jedná-li se o řídící znak protokolu (SOH nebo ETX) či nikoliv.

6. Objekty

6.1. tChnPrt

6.1.1. Položky

`CH_RTick : Boolean;`

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

`CH_DLE : Boolean;`

Položka **CH_DLE** slouží pro vnitřní použití pro uložení příznaku přijmutí znaku DLE.

`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_vLRMess : Word;`

Položka **CH_vLRMess** definuje pomocnou délku přijímané zprávy (kolikátý byte zprávy je již přijat). Položka slouží jen pro vnitřní použití objektu.

`CH_RCrc : tCrc16;`

Položka **CH_RCrc** definuje objekt pro generaci cyklického polynomu přijímače.

`CH_SCrc : tCrc16;`

Položka **CH_SCrc** definuje objekt pro generaci cyklického polynomu vysílače.

6.1.2. Metody

6.1.2.1. Init konstruktor

`constructor Init;`

Konstruktor **Init** slouží k vytvoření a inicializaci instance komunikačního objektu. Ve svém těle zavolá zděděný konstruktor **Init** (inherited Init) od 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_RTick   := false;
CH_DLE     := false;
CH_SBuff   := nil;
CH_MSBuff  := 0;
CH_LRMess  := 0;
CH_vLRMess := 0;
```

6.1.2.2. 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.

6.1.2.3. 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.

6.1.2.4. ChSetOneParam funkce

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

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

LSB=Size

Parametrem **LSB** ("Length of Send Buffer") je alokován nový vysílací buffer **CH_MSBuff** dané velikosti Size. Do tohoto bufferu je transformována vysílaná zpráva, která je předána objektu nižší komunikační vrstvy k odeslání. Size může nabývat hodnot 17 až 32750 byte.

NOD=Node

Parametrem **NOD** ("Node") se určuje číslo (adresa) stanice **CH_Node** v komunikační síti. Node může nabývat hodnot 0 až 255.

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**. DNode může nabývat hodnot 0 až 255, přičemž hodnota 0 znamená, že zpráva je určena všem připojeným stanicím.

Příklad:

Příklad ukazuje, jak je možné nastavit v komunikačním objektu se jménem PRT parametry LSB na hodnotu 1000, NODE na hodnotu 20, DNODE na hodnotu 30 a v komunikačním objektu se jménem COM, který byl zřetězen v komunikačním kanálu za objekt se jménem PRT, parametr BD na hodnotu 9600.

```
ChSetParam('NAM=PRT LSB=1000 NOD=20 DNO=30 NAM=COM BD=9600');
```

6.1.2.5. ChGetParam funkce

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

Metoda **ChGetParam** navrací nastavené hodnoty parametrů komunikačního objektu. Nejprve vrátí nastavení parametrů rodičovského komunikačního objektu

tChnVirt a poté k nim připojí seznam svých parametrů. Seznam parametrů je uveden výše u popisu metody **ChSetOneParam**.

6.1.2.6. 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.

6.1.2.7. 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.

6.1.2.8. ChSend procedura

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

Metoda **ChSend** způsobí započetí vysílání zprávy s datovým polem, na které ukazuje parametr Buff, podle výše definovaného protokolu. Parametr Len udává délku vysílacího bufferu pro vysílání (pro tento protokol délku vysílaných dat).

6.1.2.9. 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.

6.1.2.10. ChReceive procedura

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

Metoda **ChReceive** provede přijmutí celé zprávy a její uložení do přijímacího bufferu, který byl definován metodou **ChReceiveBuffer**. Metoda naplní buffer pouze patřičnými daty, 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**.

6.1.2.11. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

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

6.1.2.12. 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**.

6.1.2.13. 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**.

6.2. tAddChnPrt

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

6.2.1. Metody

6.2.1.1. ChInit funkce

```
function ChInit: pChnVirt;
```

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

7. Příklad

Příklad ukazuje použití komunikačních jednotek ChnPrt a ChnCom. Je vytvořen komunikační kanál definovaných vlastností, po kterém je zasílána zpráva a z kterého je poté očekáván příjem zpráv.

```
uses
  uString,
  ChnVirt,
  ChnCom,
  ChnPrt,
  ...
const
  ParamStr : tParamStr =
             'NAM=PRT LSB=500 NOD=1 DNO=2 '+
             'NAM=COM COM=1 IRQ=4 BD=9600 BIT=8 STO=1 PAR=E LRB=1000';

type
  tMess    = array [0..65500] of Byte;

var
  Chn      : pChnVirt;
  SMess    : ^tMess;
```

```
  RMess    : ^tMess;
  LSMess   : Word;
  LRMess   : Word;

begin
  ...
  New(SMess);
  New(RMess);
  ...
  { vytvoření instance Chn }
  Chn:=ChnCollection^.ChNewInit(ChnPrt.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 přjatá 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 a naplnění délky vysílané zprávy LSMess }
    ...
    { 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
      { zpráva se přijala }
    ...
    { 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.
```