ChnICS

JEDNOTKA DEFINUJÍCÍ KOMUNIKAČNÍ PROTOKOL ICSLINK

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	1.XX	Tu	22.05.2003	Úprava dokumentu dle ISO9000.
1.20	1.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 ICSLink.

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 ICSlink používaného při komunikaci se zařízeními PLC firmy DACS 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 ChnICS definuje komunikační objekt **tChnICS**, který je dědicem od rodičovského komunikačního objektu tChnVirt. Instance objektu tChnICS 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 **tAddChnICS**, který je dědicem od rodičovského objektu tAddChnVirt. Objekt tAddChnICS zajistí, aby daný komunikační objekt (objekt tChnICS) byl k aplikaci připojen a popřípadě zajistí vytvoření instance tohoto objektu. Po přilinkování této jednotky do aplikace (příkazem "uses ChnICS"), se jméno objektu tChnICS automaticky vloží do seznamu správců komunikačních objektů pro případné použití.

Protože je objekt **tChnICS** 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 = 'ICS';
```

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

```
tData = integer;
```

Typ **tData** definuje typ přenášených číselných dat.

```
MaxDataLen  = 32;
```

Konstanta **MaxDataLen** definuje maximální počet přenášených číselných dat.

```
MaxTextLen  = 200;
```

Konstanta **MaxTextLen** definuje maximální délku přenášeného textu včetně krajních hraničních znaků.

```
MaxTBuf = 65500;
```

Konstanta **MaxTBuff** definuje maximální velikost vysílacího bufferu.

4.1. Řídící znaky protokolu

```
SOH   = $01;
STX   = $02;
ETX   = $03;
EOT   = $04;
```

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

Konstanta **STX** definuje kód znaku pro oddělení adresy příjemce a vysílajícího.
Konstanta **ETX** definuje kód znaku pro konec číselných a textových dat.
Konstanta **EOT** definuje kód znaku pro konec zprávy.

4.2. Kódy služeb protokolu

Následující konstanty znaků určují kód služby (OpCode - Operation Code) dané zprávy.

Pro Master stanici:

```
OCM_ReadData      = 'RD';
OCM_ReadParams    = 'RL';
OCM_WriteData     = 'WD';
```

Pro Slave stanici:

```
OCS_ReadData      = 'DB';
OCS_ReadParams    = 'LB';
OCS_WriteData     = 'OK';
```

Konstanta **OPM_ReadData** definuje mnemonický kód zprávy pro požadavek na čtení dat od Master stanice.
Konstanta **OPM_ReadParams** definuje mnemonický kód zprávy pro požadavek na čtení parametrů od Master stanice.
Konstanta **OPM_WriteData** definuje mnemonický kód zprávy pro požadavek na zápis dat od Master stanice.
Konstanta **OCS_ReadData** definuje mnemonický kód odpovědi od Slave stanice na zprávu pro čtení dat.
Konstanta **OCS_ReadParams** definuje mnemonický kód odpovědi od Slave stanice na zprávu pro čtení parametrů.
Konstanta **OCS_WriteData** definuje mnemonický kód odpovědi od Slave stanice na zprávu pro zápis dat.

4.3. Konstanty výsledků přijímacího automatu

V této kapitole jsou popsány definice chybových kódů, které může vracet metoda **ChReceiveResult** v průběhu přijímacího automatu.

```
Res_ErrFrame    = $20;
```

Výsledek **res_ErrFrame** indukuje chybný formát zprávy - přijatou zprávu nelze akceptovat.

```
res_ErrSum   = $21;
```

Výsledek **res_ErrSum** indikuje chybu kontrolního součtu při přijmu zprávy.

```
res_ErrLen   = $22;
```

Výsledek **res_ErrLen** indikuje chybnou délku bloku dat.

```
res_ErrVal   = $23;
```

Výsledek **res_ErrVal** indikuje chybu při převodu čísla na textový řetězec a naopak.

```
res_ErrETX   = $24;
```

Výsledek **res_ErrETX** indikuje chybu nepřijatého ETX znaku na konci zprávy.

```
res_ErrEOT   = $25;
```

Výsledek **res_ErrEOT** indikuje chybu nepřijatého EOT znaku na konci zprávy.

4.4. Struktury přijímacích a vysílacích bufferů

```
pSendRecord = ^tSendRecord;
tSendRecord = record
  OpCode  : string[2];
  BlockNum: 0..9;
  Data    : array [0..MaxDataLen-1] of tData;
  Text    : string[MaxTextLen];
end;
```

TSendRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu ICS posílaným Master i Slave stanicí ven z jednotky, přičemž je zbaven nadbytečností, které jsou při vysílání doplněny. Položka **OpCode** definuje mnemonický kód požadované služby, položka **BlockNum** definuje číslo přenášeného bloku, položka **Data** definuje pole přenášených dat a položka **Text** definuje případný posílaný textový řetězec.

```
pRecRecord  = ^tRecRecord;
tRecRecord  = tSendRecord;
```

TRecRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu ICS přijímaným Master i Slave stanicí, přičemž je zbaven nadbytečností, které jsou při příjmu odstraněny. Svou vnitřní strukturou je shodný s typem **TSendRecord**.

5. Objekty

5.1. tChnICS

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    : tXor8;
```

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

```
CH_SSum    : tXor8;
```

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;
CH_DataI   := 0;
CH_PomS    := '';
```

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 tChnICS 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 jednotku Master i Slave může Node nabývat hodnot <0..207>.

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 jednotku Master i Slave může DNode nabývat hodnot <0..207>.

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 ICSlink.

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 ICSlink na podkladu záznamu typu tSendRecord, na který ukazuje parametr Buff. Parametr Len udává délku vysílacího bufferu pro vysílání. Tento parametr je nutno správně zadat a to následovně: Len = počet vysílaných bytů v poli Data + 4. 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 tRecRecord a byl definován metodou **ChReceiveBuffer**. Metoda naplní buffer pouze patřičnými položkami typu tRecRecord, úvodní a zakončovací řídicí znaky a kontrolní součet ze zprávy metoda vyhodnotí a pro uživatele odstraní. V parametru Len vrátí velikost přijaté zprávy a to následovně: Len = počet přijatých bytů v poli Data + 4. 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 ICSlink.

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. tAddChnICS

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

6. Struktura zpráv protokolu ICSlink

6.1. Obecná struktura zpráv

Zpráva s datovým i textovým polem vysílaná a přijímaná stanicí Master i Slave.

SOH	DNO	STX	SNO	OpCode	CBN	,	Data	Text	ETX	SUM	EOT

Význam položek:

SOH - úvodní znak zprávy
- pevná hodnota $01
- obvykle se vysílají tři tyto znaky pro jasné označení začátku zprávy

STX - oddělovací znak pro položky DNO a SNO
- pevná hodnota $02

DNO - adresa cílové stanice (destination address)
- hodnota 0..207 převedená na ASCII znaky ‘0’..#$FF

SNO - adresa Slave stanice (source address), které nebo od které je zpráva
- hodnota 0..207 převedená na ASCII znaky ‘0’..#$FF

OpCode - mnemonický kód služby (Operation Code)
- 2 ASCII znaky ‘ ‘..#$FF

CBN - číslo přenášeného datového bloku (Communication Block Number)
- hodnota 0..9 převedená na ASCII znaky ‘0’..’9’

, - oddělovací znak CBN a pole Data či Text
- pevná hodnota v ASCII ‘,’

DATA - nepovinné pole dat typu integer převedené na ASCII znaky s dekadickou representací
- jednotlivá čísla musí být navzájem oddělena alespoň jedním ASCII znakem ‘,’
- maximálně 32 položek (čísel)

TEXT - nepovinný řetězec ASCII znaků ohraničený dvěma navzájem shodnými hraničními znaky
- hraničními znaky nesmí být znaky: #$00..#$1F, ’+’, ’-’ , ’,’, ‘.’, ‘0’..’9’
- ostatními znaky nesmí být znaky: #$00..#$1F

ETX - oddělovací znak pro pole Data nebo Text a kontrolní součet SUM
- pevná hodnota $03

SUM - kontrolní součet
- bytový XOR všech bytů položek od posledního SOH až ETX.

EOT - koncový znak zprávy
pevná hodnota $04

Pozn.: Pokud uživatel nebude chtít vysílat textový řetězec Text, musí nastavit položku Text ve struktuře vysílacího bufferu TSendRecord na hodnotu ‘’. Naopak pokud položka Text ve struktuře přijímacího bufferu TRecRecord má hodnotu ‘’, nepřijal se žádný textový řetězec.

7. Příklad

Následující příklad ukazuje způsob vyslání zprávy s datovým i textovým polem do PLC. Fyzický přenos se realizuje prostřednictvím knihovny ChnCom.

```
uses
  uString,
  ChnVirt,
  ChnCom,
  ChnICS;

const
  ParamStr : tParamStr =
             'NAM=ICS 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 : pSendRecord;
  RMess : pRecRecord;
  LSMess: word;
  LRMess: word;

begin
  ...
  New(SMess);
  New(RMess);
  ...
  { vytvoření instance Chn }
  Chn:=ChnCollection^.ChNewInit(ChnICS.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');
    ...
    { příklad naplnění zprávy daty }
    with SMess^ do
    begin
      OpCode  :=OCM_ReadData;
      BlockNum:=0;
      Text    :='"Požadavek na čtení dat"';
      Data[0] :=0;
      Data[1] :=10;
      Data[2] :=15;
      LSMess  :=4+3*SizeOf(tData);
    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 ReadData)}
      with RMess^ do
      begin
        if (OpCode = OCS_ReadData) and
           (LRMess = 4+3*SizeOf(tData)) then
          writeln('Ok data přijata v poli Data')
        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.
```