ChnModB

JEDNOTKA DEFINUJÍCÍ KOMUNIKAČNÍ PROTOKOL MOD-BUS

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	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 MOD-BUS.

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 ModBus, který se používá při komunikaci se zařízeními EUROTHERM. Obstarává zabezpečení dat, vkládání a vyjímání nadbytečností, tak jak to tento protokol předepisuje. Fyzický přenos dat je zajištěn prostřednictvím nižší komunikační vrstvy.

Knihovna ChnModB definuje komunikační objekt **tChnModB**, který je dědicem od rodičovského komunikačního objektu tChnVirt. Instance objektu tChnModB 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. Protokol ModBus používá pro rozlišení začátku zprávy definované časové prodlevy na komunikační lince, konkrétně na začátku zprávy musí být alespoň taková časová prodleva, která by byla potřebná pro vyslání 3 znaků. Tuto hardwarovou vlastnost musí řešit nižší komunikační vrstva. **Proto, aby protokol ModBus pracoval správně, musí se přilinkovat patřičná komunikační knihovna nižší fyzické vrstvy, která měří časové prodlevy mezi přijatými znaky (např. knihovna ChnComT).**

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

Protože je objekt **tChnModB** 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  = 'MODB';
```

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

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_ErrUnknownCode = $25;`
Chyba neznámého Code - přijatou zprávu nelze akceptovat.

4.2. Kódy jednotlivých typů zpráv

`ReadNBits = $01;`
Konstanta **ReadNBits** definuje kód zprávy pro čtení daného počtu bitů.

`ReadNBits_ = $02;`
Konstanta **ReadNBits_** definuje kód zprávy pro čtení daného počtu bitů.

`ReadNWords = $03;`
Konstanta **ReadNWords** definuje kód zprávy pro čtení daného počtu wordů.

`ReadNWords_ = $04;`
Konstanta **ReadNWords_** definuje kód zprávy pro čtení daného počtu wordů.

`Write1Bit = $05;`
Konstanta **Read1Bit** definuje kód zprávy pro čtení jednoho daného bitu.

`Write1Word = $06;`
Konstanta **Read1Word** definuje kód zprávy pro čtení jednoho daného wordu.

`FastStatus = $07;`
Konstanta **FastStatus** definuje kód zprávy pro čtení stavu.

`WriteNBits = $0f;`
Konstanta **WriteNBits** definuje kód zprávy pro zápis daného počtu bitů.

`WriteNWords = $10;`
Konstanta **WriteNWords** definuje kód zprávy pro zápis daného počtu wordů.

4.3. Konstanty kódů pro metodu ChGetBinParam

`cmd_GetAddrByte = $0202;`
Konstanta **cmd_GetAddrByte** definuje hodnotu Code pro metodu **ChGetBinParam** komunikačního objektu nižší vrstvy. Slouží pro určení stavu přijetí speciálního znaku, který označuje začátek zprávy. Tato konstanta musí mít stejnou hodnotu jako příslušné konstanty z jednotek nižší komunikační vrstvy, které umí určitým způsobem rozlišit normální znaky od adresních (počátečních).

4.4. Konstanty maximálních počtů vysílaných dat

`MaxWordsSize = 250 div 2 {125};`
Konstanta **MaxWordsSize** definuje maximální počet vysílaných a přijímaných dat typu word.

`MaxBitsSize = 250 * 8 {2000};`
Konstanta **MaxBitsSize** definuje maximální počet vysílaných a přijímaných dat typu bit.

`MaxTBuf = 32750;`
Konstanta **MaxTBuff** definuje maximální velikost vysílacího bufferu.

4.5. Struktury přijímacích a vysílacích bufferů

```
PSendRecord  = ^TSendRecord;
TSendRecord  = record
  case FnCode  : byte of
    ReadNBits,
    ReadNBits_,
    Write1Bit,
    WriteNBits :
      (Dummy3_1: array[1..3] of byte;
       FAddrB  : word;
       CountB  : word;
       BitData : array [0..(MaxBitsSize div 8)-1] of Byte;
      );
    ReadNWords,
    ReadNWords_,
    Write1Word,
    WriteNWords :
      (Dummy3_2: array[1..3] of byte;
       FAddrW  : word;
       CountW  : word;
       WordData: array [0..MaxWordsSize-1] of Word;
      );
    FastStatus :
      (Status  : Byte;
      );
    ReadNBits   or $80,
    ReadNBits_  or $80,
    Write1Bit   or $80,
    WriteNBits  or $80,
    ReadNWords  or $80,
    ReadNWords_ or $80,
    Write1Word  or $80,
    WriteNWords or $80,
    FastStatus  or $80 :
      (ErrCode : Byte;
      );
end;
```

TSendRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu ModBus pro vysílání zpráv, přičemž je zbaven nadbytečností, které jsou při vysílání doplněny. Položka **FnCode** určuje kód (typ) zasílané zprávy. Při čtení nebo zápisu bitů se používá položka **FAddrB**, která určuje absolutní adresu pro čtení či zápis bitů, dále položka **CountB**, která udává počet zapisovaných či čtených údajů (bitů), a nakonec položka **BitData**, která obsahuje hodnoty zapisovaných či přečtených dat (bitů) v podobě celých bytů. Při čtení nebo zápisu wordů se používá položka **FAddrW**, která určuje absolutní adresu pro čtení či zápis wordů, dále položka **CountW**, která udává počet čtených či zapisovaných údajů (wordů), a nakonec položka **WordData**, která obsahuje hodnoty zapisovaných dat (wordů). Položky **Dummy3_1** a **Dummy3_2** nemají zvláštní význam, v záznamu jsou uvedeny pouze pro zarovnání následujících položek na offsetovou adresu dělitelnou čtyřma. Při čtení statusu se používá položka **Status**, která udává stav zařízení. Při přijetí zprávy s chybovým kódem (má nastavený nejvyšší bit) se používá položka **ErrCode**, která udává chybový kód.

```
PRecRecord = ^TRecRecord;
TRecRecord = TSendRecord;
```

TRecRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu ModBus pro příjem zpráv, přičemž je zbaven nadbytečností,

které jsou při příjmu odstraněny. Svou vnitřní strukturou je shodný s typem **TSendRecord**.

5. Popis objektů

5.1. tChnModB

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 : tCrc16;`

Položka **CH_RSum** se používá pro počítání 16-ti bitového kontrolního CRC bloku dat přijímače.

`CH_SSum : tCrc16;`

Položka **CH_SSum** se používá pro počítání 16-ti bitového kontrolního CRC bloku dat 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 nejprve zavolá zděděný konstruktor **Init** (inherited Init) z rodičovského objektu tChnVirt a poté inicializuje položky objektu. Tělo konstruktoru vypadá následovně:

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

5.1.2.2. ChInitParam konstruktor

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

Konstruktor **ChInitParam** slouží ke zkrácenému vytvoření instance komunikačního objektu s definovaným nastavením parametrů kanálu. Ve svém těle nejprve volá konstruktor **Init** a poté metodu **ChSetParam**.

5.1.2.3. ChSetOneParam procedura

```
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 tChnModB 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. Node může nabývat hodnot <1..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..255>. Hodnota 0 je určena všem připojeným stanicím.

5.1.2.4. 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**.

5.1.2.5. 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 ModBus.

5.1.2.6. 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.

5.1.2.7. ChSend procedura

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

Metoda **ChSend** způsobí započetí vysílání zprávy podle protokolu ModBus na podkladu záznamu typu tSendRecord, na který ukazuje parametr Buff. Parametr Len udává délku vysílacího bufferu pro vysílání. Tento parametr není nutno správně naplnit skutečnou délkou zprávy, jelikož protokol umí tuto délku získat automaticky podle vyplněného záznamu vysílacího bufferu. Před voláním této metody se musí záznam správně naplnit daty (viz níže).

5.1.2.8. ChReceiveReady funkce

```
function  ChReceiveReady: tChState;
```

Metoda **ChReceiveReady** způsobí provedení jednoho či více kroků 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.

5.1.2.9. ChReceive procedura

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

Metoda **ChReceive** provede přijetí 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, případné 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**.

5.1.2.10. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

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

5.1.2.11. 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**.

5.1.2.12. 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. tAddChnModB

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

6. Popis protokolu ModBus

Jednotka aplikuje protokol ModBus podle ANSI X3,28-2,5-A4. Tento protokol používá firma Eurotherm pro komunikaci v řídících systémech Eurotherm řady 800, 900, 2200, 2400 a 3000.

6.1. Struktura jednotlivých zpráv

Pozn.: U položek velikosti 2 byte (word) se vysílá nejdříve vyšší byte a poté nižší.

6.1.1. Čtení n bitů – FnCode = 01h nebo 02h

Master posílá typ požadavku na čtení ze zařízení.

DNode	FnCode	FAddrB	CountB	CRC16

DNode je adresa Slave stanice, kam je zpráva zasílána (byte).
FnCode je funkční kód pro zápis n bitů, je to hodnota 01h nebo 02h (byte).
FAddrB je počáteční adresa čtených parametrů - bitů (2 byte).
CountB je počet čtených parametrů - bitů (2 byte).
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, FAddrB a CountB se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Pokud zařízení (**Slave**) dekódovalo zprávu správně, odpovídá na požadavek na čtení dat touto zprávou:

Node	FnCode	CountB*	BitData[0..CountB*-1]	CRC16

Node je adresa Slave stanice, které byla zpráva zasílána (byte).
FnCode je funkční kód pro čtení n bitů, je to hodnota 01h nebo 02h (byte).
CountB* je počet bytů potřebný k vyslání CountB čtených bitů (2 byte).
BitData[..] jsou položky dat typu byte obsahující jednotlivé bity (CountB* bytů)

CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, CountB a BitData se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Příklad:

Master posílá:

DNode	FnCode	FAddrB		CountB		Crc16	
13h	01h	00h	02h	00h	0Eh	1Fh	7Ch

Slave odpovídá:

Node	FnCode	CountB*	BitData[0]	BitData[1]	Crc16	
13h	01h	02h	01000000b	00000010b	B1h	FEh

první vysílaný bit (bit určený ve FAddrB)

tyto dva bity se neposílají a proto jsou v nich 0

6.1.2. Čtení n wordů – FnCode = 03h nebo 04h

Master posílá typ požadavku na čtení ze zařízení.

DNode	FnCode	FAddrW	CountW	CRC16

DNode je adresa Slave stanice, kam je zpráva zasílána (byte).
FnCode je funkční kód pro čtení n wordů, je to hodnota 03h nebo 04h (byte).
FAddrW je počáteční adresa čtených parametrů - wordů (2 byte).
CountW je počet čtených parametrů - wordů (2 byte).
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, FAddrW a CountW se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Pokud zařízení (**Slave**) dekódovalo zprávu správně, odpovídá na požadavek na čtení dat touto zprávou:

Node	FnCode	CountW * 2	WordData[0..CountW-1]	CRC16

Node je adresa Slave stanice, které byla zpráva zasílána (byte).
FnCode je funkční kód pro čtení n wordů, je to hodnota 03h nebo 04h (byte).
CountW je počet čtených parametrů - wordů (2 byte).
WordData[..] jsou položky dat typu word (CountW wordů)
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, CountW a WordData se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Příklad:

Master posílá:

DNode	FnCode	FAddrW		CountW		Crc16	
01h	03h	00h	08h	00h	02h	45h	FAh

Slave odpovídá:

Node	FnCode	CountW *2	WordData[0]		WordData[1]		Crc16	
01h	03h	04h	00h	64h	00h	32h	09h	39h

6.1.3. Zápis jednoho bitu - FnCode = 05h

Master posílá typ zápisového požadavku a data na zápis do zařízení.

DNode	FnCode	FAddrB	BitData[1], BitData[0]	CRC16

DNode je adresa slave stanice, které se zpráva posílá, pokud je rovna 0, je zpráva zasílána všem připojeným slave stanicím a není na ni žádná odpověď (byte)
FnCode je kód pro zápis jednoho wordu 05h (byte)
FAddrB je adresa pro zápis parametrů - bitů (2 byte)
BitData[1] je pevná hodnota 00h
BitData[0] je hodnota zapisovaného bitu rozšířená na byte. Pokud má zapisovaný bit hodnotu nula má i BitData[0]hodnotu 00h. V opačném případě má BitData[0] hodnotu FFh.
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, FAddrB a BitData[0] se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Zařízení (**Slave**), pokud zprávu správně dekódovalo, posílá jako odpověď na požadavek zápisu master stanici naprosto stejnou zprávu.

6.1.4. Zápis jednoho wordu - FnCode = 06h

Master posílá typ zápisového požadavku a data na zápis do zařízení.

DNode	FnCode	FAddrW	WordData[0]	CRC16

DNode je adresa slave stanice, které se zpráva posílá, pokud je rovna 0, je zpráva zasílána všem připojeným slave stanicím a není na ni žádná odpověď (byte)
FnCode je kód pro zápis jednoho wordu 06h (byte)
FAddrW je adresa pro zápis parametrů - wordu (2 byte)
WordData[0] je hodnota posílaného wordu (2 byte)
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, FAddrW a WordData se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Zařízení (**Slave**), pokud zprávu správně dekódovalo, posílá jako odpověď na požadavek zápisu master stanici naprosto stejnou zprávu.

6.1.5. Zápis n wordů - FnCode = 10h

Master posílá typ zápisového požadavku a data na zápis do zařízení.

DNode	FnCode	FAddrW	CountW	CountW*2	WordData[0..CountW-1]	CRC16

DNode je adresa slave stanice, které se zpráva posílá, pokud je rovna 0, je zpráva zasílána všem připojeným slave stanicím a není na ni žádná odpověď (byte)
FnCode je kód pro zápis n wordů 10h (byte)
FAddrW je adresa pro zápis parametrů - wordů (2 byte)
CountW je počet zapisovaných parametrů - wordů (2 byte)
CountW*2 je počet zapisovaných bytů (byte)
WordData[..] je pole hodnot posílaných wordů (CountW*2 byte)
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, FAddrW, CountW a WordData se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Zařízení (**Slave**), pokud zprávu správně dekódovalo, posílá jako odpověď na požadavek zápisu master stanici tuto zprávu:

Node	FnCode	FAddrW	CountW	CRC16

Node je adresa slave stanice, které se zpráva posílala (byte)
FnCode je kód pro zápis n wordů 10h (byte)
FAddrW je adresa pro zápis parametrů - wordů (2 byte)
CountW je počet zapsaných parametrů - wordů (2 byte)
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, FAddrW a CountW se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

6.1.6. Zápis n bitů - FnCode = 0Fh

Master posílá typ zápisového požadavku a data na zápis do zařízení.

DNode	FnCode	FAddrB	CountB	CountB*	BitData[0..CountB*-1]	CRC16

DNode je adresa slave stanice, které se zpráva posílá, pokud je rovna 0, je zpráva zasílána všem připojeným slave stanicím a není na ni žádná odpověď (byte)
FnCode je kód pro zápis n bitů 0Fh (byte)
FAddrB je adresa pro zápis parametrů - bitů (2 byte)
CountB je počet zapisovaných bitů (2 byte)
CountB* je počet bytů potřebný k vyslání CountB zapisovaných bitů (2 byte).
BitData[..] jsou položky dat typu byte obsahující jednotlivé bity (CountB* bytů)
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, FAddrB, CountB a BitData se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Zařízení (**Slave**), pokud zprávu správně dekódovalo, posílá jako odpověď na požadavek zápisu master stanici tuto zprávu:

Node	FnCode	FAddrB	CountB	CRC16

Node je adresa slave stanice, které se zpráva posílala (byte)
FnCode je kód pro zápis n wordů 10h (byte)
FAddrB je adresa pro zápis parametrů - bitů (2 byte)
CountB je počet zapsaných parametrů - bitů (2 byte)
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode, FAddrB a CountB se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

6.1.7. Rychlé čtení statusu - FnCode = 07h

Master posílá typ požadavku na čtení parametrů do zařízení.

DNode	FnCode	CRC16

DNode je adresa slave stanice, které se zpráva posílá (byte)
FnCode je kód pro rychlé čtení statusu 07h (byte)
CRC16 je cyklický součet zprávy (2 byte).

Položka FnCode se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

Zařízení (**Slave**), pokud zprávu správně dekódovalo, posílá jako odpověď na požadavek čtení master stanici tuto zprávu:

Node	FnCode	Status	CRC16

Node je adresa slave stanice, které se zpráva posílala (byte)
FnCode je kód pro rychlé čtení statusu 07h (byte)
Status je hodnota statusu (byte)
CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode a Status se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

6.1.8. Odpověď na chybný příkaz

Pokud zařízení (**Slave**) zprávu správně přijalo a dekódovalo, ale nemohlo z nějakých důvodů zadaný příkaz provést, odpovídá master stanici touto zprávou:

Node	FnCode	ErrCode	CRC16

Node je adresa slave stanice, které se zpráva posílala (byte)
FnCode je kód přijatý od Master stanice s nastaveným nejvyšším bitem - or 80h

(byte)

ErrCode je hodnota chyby (byte)

CRC16 je cyklický součet zprávy (2 byte).

Položky FnCode a ErrCode se musí patřičně nastavit ve struktuře variantního záznamu vysílacího bufferu.

7. Příklad

Následující příklad ukazuje způsob vyslání zprávy pro čtení statusu do PLC Eurotherm pro otestování připojení tohoto zařízení. Fyzický přenos se realizuje prostřednictvím knihovny ChnComT s navolenou pauzou mezi znaky na 3,5 ms, která pro správný chod vyžaduje periodické volání procedur IncRecTime v obsluze přerušení od časovače.

```
uses
  uString,
  ChnVirt,
  ChnComT,
  ChnModB,
  Tick;

const
  ParamStr : tParamStr =
             'NAM=MODB MAS=MASTER LSB=500 NOD=1 DNO=2 ' +
             'NAM=COMT COM=1 IRQ=4 BD=9600 BIT=8 STOP=1 ' +
             'PAR=E LRB=1000 RTM=35';

var
  Chn   : pChnVirt;
  SMess : pSendRecord;
  RMess : pRecRecord;
  LSMess: word;
  LRMess: word;

procedure ComTUserTick1; far;
  { procedura pro zrychlený časovač pro COM ChnComT }
var I:ChnComT.tCtCom;
begin
  for I:=Low(ChnComT.tCtCom) to High(ChnComT.tCtCom) do
    ChnComT.IncRecTime(I);
end;

begin
  ...
  { inicializace systémového časovače }
  UserTick1:= ComTUserTick1;
  InitTickI;
  ...
  New(SMess);
  New(RMess);
  ...
  { vytvoření instance Chn }
  Chn:=ChnCollection^.ChNewInit(ChnModB.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 ReadStatus)}
    with SMess^ do
    begin
      FnCode:= FastStatus;
      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 ReadStatus)}
      with RMess^ do
      begin
        if FnCode = FastStatus then
          writeln('Přijat status ',Status)
        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.
```