ChnLecom

JEDNOTKA DEFINUJÍCÍ KOMUNIKAČNÍ PROTOKOL LECOM

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. Konstanty a typy 6
4.1. Řídící znaky protokolu 6
4.2. Konstanty výsledků přijímacího automatu 6
4.3. Konstanty velikostí datových typů. 7
4.4. Ostatní konstanty a jednoduché typy 7
4.5. Struktura přijímacích a vysílacích bufferů 8
5. Objekty 8
5.1. tChnLecom 8
5.1.1. Položky 8
5.1.2. Metody 9
5.1.2.1. Init konstruktor 9
5.1.2.2. ChInitParam konstruktor 9
5.1.2.3. Done destruktor 9
5.1.2.4. ChSetOneParam funkce 9
5.1.2.5. ChGetParam funkce 10
5.1.2.6. ChConnect procedura 10
5.1.2.7. ChDisConnect procedura 10
5.1.2.8. ChSend procedura 11
5.1.2.9. ChReceiveReady funkce 11
5.1.2.10. ChReceive procedura 11
5.1.2.11. ChReceiveFlush procedura 11
5.1.2.12. ChGetNode procedura 11
5.1.2.13. ChReceiveTick procedura 11
5.2. tAddChnLecom 12
5.2.1. Metody 12
5.2.1.1. ChInit funkce 12
6. Struktura zpráv protokolu Lecom 12
6.1. Formát zprávy pro zápis parametru 12
6.2. Formát zprávy pro čtení parametru 14

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	Úpravy popisu používání SubCode. Změna číslování chybových konstant res_ErrXxx dle standardu.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky definující komunikační protokol LECOM.

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 ChnLecom definuje komunikační objekt **tChnLecom**, který je dědicem od rodičovského komunikačního objektu tChnVirt. Instance objektu tChnLecom 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 tChnLecom definuje formát protokolu Lecom, který slouží pro programování Lenze měničů, a obstarává zabezpečení dat, vkládání a vyjímání nadbytečností, tak jak to tento protokol předepisuje. Fyzický přenos dat je zajištěn prostřednictvím nižší komunikační vrstvy. Určení, přes jakou fyzickou vrstvu bude komunikace probíhat, je voleno až parametrem nastavovací metody ChSetParam.

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

Protože je objekt **tChnLecom** 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  = 'LECOM';
```

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

4.1. Řídící znaky protokolu

```
EOT    = $04;
STX    = $02;
ENQ    = $05;
ACK    = $06;
NAK    = $15;
ETX    = $03;
```

Konstanty **EOT**, **STX**, **ENQ**, **ACK**, **NAK**, **ETX** definují řídící a zabezpečovací znaky protokolu.

4.2. 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** indikuje chybu rámce přijaté zprávy.

```
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_ErrCode  = $24;
```
Výsledek **res_ErrCode** indikuje chybu kódu.

```
res_ErrEOT   = $25;
```
Výsledek **res_ErrEOT** indikuje předčasně přijatý znak začátku nové zprávy při nedokončení zprávy předchozí.

4.3. Konstanty velikostí datových typů.

```
cSLen = 15;
```
Konstanta **cSLen** definuje maximální délku přenášeného textového řetězce.

```
cRLen = 12;
```
Konstanta **cRLen** definuje maximální délku reálného čísla ve znacích včetně tečky.

```
cWLen =  8;
```
Konstanta **cWLen** definuje maximální délka čísla v hexadecimálním tvaru ve znacích.

```
cOLen = 15;
```
Konstanta **cOLen** definuje maximální délka řetězce v oktalovém tvaru ve znacích.

4.4. Ostatní konstanty a jednoduché typy

```
MaxCode = 6229;
```
Konstanta **MaxCode** definuje maximálně možný přijatý Code zprávy pro staré kódování.

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

```
tRW      = (Rd,Wr);
```
Výčtový typ **tRW** definuje dva základní typy zpráv. Hodnota **Rd** určuje zprávu pro čtení dat a Hodnota **Wr** určuje zprávu pro zápis dat.

```
tCode    = 0..65535;
```
Intervalový typ **tCode** vymezuje interval hodnot pro Code zprávy v novém kódování.

```
tSubCode = 0..255;
```
Intervalový typ **tSubCode** vymezuje interval hodnot pro SubCode zprávy.

```
tParam   = (tpReal, tpByte, tpWord, tpLong, tpString, tpOktStr);
```
Výčtový typ **tParam** definuje základní datové typy.

4.5. Struktura přijímacích a vysílacích bufferů

```
pSendRecord = ^tSendRecord;
tSendRecord = record
  Code    : tCode;
  SubCode : tSubCode;
  case RW: tRW of
    Wr : (case Par: tParam of
            tpReal   : (R : Real);
            tpByte   : (B : Byte);
            tpWord   : (W : Word);
            tpLong   : (L : Longint);
            tpString : (S : String[cSLen]);
            tpOktStr : (O : Longint);
         );
    Rd : ();
end;
```

TSendRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu Lecom pro vysílání zpráv. Položka **Code** udává kód příkazu Lecom protokolu. Položka **SubCode** udává pomocný kód příkazu Lecom protokolu v novém kódování. Položka **RW** určuje typ zprávy, jedná-li se o zprávu pro čtení či zápis dat. Položka **Par** určuje datový formát posílaných či čtených dat. Položka **R** obsahuje vlastní data - reálné číslo, položka **B** obsahuje vlastní data - číslo v hexadecimálním tvaru o velikosti jeden byte, položka **W** obsahuje vlastní data - číslo v hexadecimálním tvaru o velikosti dva byte, položka **L** obsahuje vlastní data - číslo v hexadecimálním tvaru o velikosti čtyři byte, položka **S** obsahuje vlastní data - textový řetězec a položka **O** obsahuje vlastní data - řetězec v oktetovém tvaru.

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

TRecRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu Lecom pro příjem zpráv. Svou strukturou odpovídá záznamu **TSendRecord**.

5. Objekty

5.1. tChnLecom

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_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).

```
CH_UseSCD  : Boolean;
```
Položka **CH_UseSCD** definuje příznak způsobu kódování. Je-li její hodnota True, používá se nové kódování s využitím SubCode, jinak se používá kódování kompatibilní se starým.

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 se nejprve zavolá zděděná metoda **Init** (inherited Init) od rodičovského objektu tChnVirt a poté jsou inicializovány 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_RSNode  := 0;
CH_RDNode  := 0;
CH_Master  := false;
CH_UseSCD  := 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 komunikačního kanálu. Ve svém těle nejprve volá konstruktor **Init** a poté metodu **ChSetParam**.

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 tChnLecom 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. Tento parametr je povinný zadat ještě před voláním metody ChSend.

NOD=Node

Parametrem **NOD** ("Node") se určuje číslo (adresa) stanice **CH_Node** v komunikační síti. Node 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 také možno definovat prostřednictvím metody **ChDestNode**. DNode může nabývat hodnot <0..99>.

SCD=ON/OFF

Parametrem **SCD** ("Use SubCode") se určí způsob kódování zpráv. Pokud je hodnota parametru ON, bude se používat nové kódování s využitím SubCode. Pokud je hodnota parametru OFF, bude se používat staré kódování. Slave stanice by měla tento parametr nastavit pokaždé po příjmu zprávy od Master stanice podle toho, byla-li od Master stanice přijata zpráva využívající SubCode či ne. Tím se zajistí zpětná kompatibilita se staršími Master stanicemi, které kódování se SubCode nepoužívají.

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 Lecom.

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 Lecom 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).

5.1.2.9. 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. 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 Lecom.

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. tAddChnLecom

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

6. Struktura zpráv protokolu Lecom

Jednotka aplikuje komunikační protokol Lecom podle DIN 66019, ISO 1745, ANSI X3,28. Tento protokol používá firma Lenze pro komunikaci s měniči motorů.

6.1. Formát zprávy pro zápis parametru

MASTER posílá zprávu pro zápis parametru a data do zařízení.

Formát zprávy podle starého kódování (CH_UseSCD = False):

EOT	DNode	STX	C1	C2	V_1 ... V_n	ETX	BCC

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až MaxCode)
RW      := Wr;
Par     := ... (prvek z výčtového typy tParam)
R,W,S,O := ... (podle Par naplnit příslušnou položku)
```

Formát zprávy podle nového kódování s využitím SubCode (CH_UseSCD = True):

EOT	DNode	STX	!	CH1..CH4	SbC1	SbC2	V_1 ... V_n	ETX	BCC

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až 65535)
SubCode := ... (číslo 0 až 255)
RW      := Wr;
Par     := ... (prvek z výčtového typy tParam)
R,W,S,O := ... (podle Par naplnit příslušnou položku)
```

SLAVE odpovídá některou z následujících zpráv:

1. Požadovaná data jsem v pořádku zapsal:

ACK

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
RW      := Rd;
Par     := tpString;
S       := Chr(ACK);
```

2. Nemohl jsem zapsat požadovaná data:

NAK

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
RW      := Rd;
Par     := tpString;
S       := Chr(NAK);
```

3. Příkaz jsem dekódoval, ale takový kód parametru neznám:

Formát zprávy podle starého kódování:

STX	C1	C2	EOT

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až MaxCode shodné s Code od Mastera)
RW      := Rd;
Par     := tpString;
S       := Chr(EOT);
```

Formát zprávy podle nového kódování s využitím SubCode:

STX	!	CH1..CH4	SbC1	SbC2	EOT

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až 65535 shodné s Code od Mastera)
SubCode := ... (číslo 0 až 255 shodné se SubCode od Mastera)
RW      := Rd;
Par     := tpString;
S       := Chr(EOT);
```

4. Příkaz jsem nemohl provést, protože byl špatný kontrolní součet:

Formát zprávy podle starého kódování:

STX	C1	C2	‘?’	ETX	BCC

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až MaxCode shodné s Code od Mastera)
RW      := Rd;
Par     := tpString;
S       := '?';
```

Formát zprávy podle nového kódování s využitím SubCode:

STX	!	CH1..CH4	SbC1	SbC2	‘?’	ETX	BCC

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až 65535 shodné s Code od Mastera)
SubCode := ... (číslo 0 až 255 shodné se SubCode od Mastera)
RW      := Rd;
Par     := tpString;
S       := '?';
```

Význam položek:

EOT - úvodní znak zprávy - pevná hodnota 04h

DNode - adresa stanice pro kterou je zpráva určena. Obsahuje dvě číslice 0 až 9 v Ascii representaci.

STX - řídící znak - pevná hodnota 02h.

C1,C2 - dva Ascii znaky určující kód (Code) parametru. Kód parametru Code odpovídá číslu v rozsahu <0..MaxCode>. Následuje výpočet C1 a C2 z

kódu parametru Code:
C1 = ((Code mod 790) div 10) + Ord('0')
C2 = ((Code mod 790) mod 10) + (Code div 790) * 10 + Ord('0')
Ord('0') = $48_{dek} = 30_{hex}$ je ordinální hodnota znaku '0'
mod (modulo) je matematická funkce, která je definovaná jako zbytek po celočíselném dělení (např: 800 mod 790 = 10)
div (division) je také matematická funkce, která je definovaná jako celočíselné dělení (např.: 800 div 790 = 1).
Zpětný převod lze z těchto rovnic vypočíst, ale pro jednoduchost je uveden: Code = ((C2 - 30_{hex}) div 10)*790 + (C1 - 30_{hex})*10 + ((C2 - 30_{hex}) mod 10)

V_1,...V_n - data parametru, která můžou být ve formátu:
$VD_1..VD_{cRLen}$ - reálné číslo. Kde VD_n je Ascii representace decimální číslice, nebo znak '-', nebo znak '.'. Rozsah předávaného čísla je v intervalu <-214748.3648 .. +214748.3647>.
'H',$VH_1..VH_{cHLen}$ - číslo v hexadecimálním formátu. Kde VH_n jsou čtyři nebo 8 znaků v Ascii representaci '0'..'9','A'..'F'. VH_1 je číslo nejvyššího šestnáctkového řádu VH_{cHLen} nenižšího řádu.
'S',$VS_1..VS_{cSLen}$ - string. Kde VS_n je Ascii tištitelný znak <' '..'~'>.
'O',$VO_1..VO_{cOLen}$ - číslo v oktalovém formátu. Kde VO_n je jeden z Ascii znaků '0'..'7'.

ETX - řídící znak - pevná hodnota 03h.
BCC - byte příčné parity = xor <C1 .. ETX> nebo xor <'!' .. ETX>.
! - Ascii znak '!' = $33_{dek} = 21_{hex}$ udávající nové kódování
CH1..CH4 - kód (Code) parametru v hexadecimálním kódování v Ascii representaci.
SbC1,SbC2 - pomocný kód (SubCode) parametru v hexadecimálním kódování v Ascii representaci.
ENQ - řídící znak - pevná hodnota 05h.

6.2. Formát zprávy pro čtení parametru

MASTER posílá zprávu pro čtení parametru do zařízení.

Formát zprávy podle starého kódování (CH_UseSCD = False):

EOT	DNode	C1	C2	ENQ

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až MaxCode)
RW      := Rd;
```

Formát zprávy podle nového kódování s využitím SubCode (CH_UseSCD = True):

EOT	DNode	!	CH1..CH4	SbC1	SbC2	ENQ

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až 65535)
SubCode := ... (číslo 0 až 255)
RW      := Rd;
```

SLAVE odpovídá některou z následujících zpráv:

1. Požadavek jsem v pořádku přijal a dekódoval a posílám požadovaná data:

Formát zprávy podle starého kódování:

STX	C1	C2	V_1 ... V_n	ETX	BCC

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až MaxCode shodné s Code od Mastera)
RW      := Wr;
Par     := ... (prvek z výčtového typy tParam)
R,W,S,O := ... (podle Par naplnit příslušnou položku)
```

Formát zprávy podle nového kódování s využitím SubCode:

STX	!	CH1..CH4	SbC1	SbC2	V_1 ... V_n	ETX	BCC

Ve struktuře vysílacího bufferu se nastavují tyto položky:

```
Code    := ... (číslo 0 až 65535 shodné s Code od Mastera)
SubCode := ... (číslo 0 až 255 shodné se SubCode od Mastera)
RW      := Wr;
Par     := ... (prvek z výčtového typy tParam)
R,W,S,O := ... (podle Par naplnit příslušnou položku)
```

2. Příkaz jsem dekódoval, ale takový kód parametru neznám:

Viz stejná zpráva jako při požadavku zápisu parametru.