

# ChnCom2

JEDNOTKA SÉRIOVÉ KOMUNIKACE  
RS232 / RS485 S OBVODEM I8250 BEZ  
VYUŽITÍ PŘERUŠOVACÍHO SYSTÉMU

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: 22.05.2003

Datum posledního uložení dokumentu: 22.05.2003

(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.Popis konstant a typů	6
4.1. Konstanty kódů příkazů pro metodu ChGetBinParam	6
4.2. Konstanty kódů příkazů pro metodu ChSetBinParam	7
4.3. Konstanty a typy přijímacích a vysílacích bufferů	7
4.4. Konstanty a typy pro počet obsluhovaných COM portů	8
5.Proměnné	8
6.Objekty	8
6.1. tChnCom2	8
6.1.1. Položky	8
6.1.2. Metody	9
6.1.2.1. Init konstruktor	9
6.1.2.2. ChInitParam konstruktor	9
6.1.2.3. Done destruktork	9
6.1.2.4. ChSetOneParam funkce	9
6.1.2.5. ChGetParam funkce	11
6.1.2.6. ChSetBinParam procedura	11
6.1.2.7. ChGetBinParam funkce	11
6.1.2.8. ChOpen procedura	12
6.1.2.9. ChClose procedura	12
6.1.2.10. ChConnect procedura	12
6.1.2.11. ChDisconnect procedura	12
6.1.2.12. ChSendTick procedura	12
6.1.2.13. ChSend procedura	12
6.1.2.14. ChSendReady funkce	13
6.1.2.15. ChSendFlush procedura	13
6.1.2.16. ChReceiveReady funkce	13
6.1.2.17. ChReceiveChar funkce	13
6.1.2.18. ChReceive procedura	13
6.1.2.19. ChReceiveFlush procedura	13
6.2. tAddChnCom2	14
6.2.1. Metody	14
6.2.1.1. ChInit funkce	14
7.Procedury	14
7.1. MakeCom2	14
7.2. SetIntrCom2	14
7.3. ResetIntrCom2	14
8.Příklad	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.

### 1.2. Účel dokumentu

---

Tento dokument slouží jako popis jednotky sériové komunikace RS232 / RS485 s obvodem I8250 bez využití přerušovacího systému.

### 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 rozhraní svých potomků a 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 ChnCom2 definuje objekt **tChnCom2**, jehož instance vytváří fyzickou vrstvu v komunikačním kanálu tvořenou sériovým rozhraním RS232 nebo RS485 s obvodem i8250. Ke své činnosti nevyužívá přerušovacího režimu počítače, ale dotazovacího (task) režimu. Knihovnu lze použít pro obsluhu až čtyř komunikačních linek. Pro každou komunikační linku se musí vytvořit samostatná instance komunikačního objektu. Všechny instance jsou najednou obsluhovány procedurou **MakeCom2**, která musí být aplikací pravidelně vyvolávána. Frekvence jejího volání by měla být vyšší nebo rovna přenosové rychlosti nejrychlejšího kanálu.

Znaky přicházející po komunikaci jsou ukládány do vstupního kruhového vyrovnávacího bufferu, z kterého jsou předávány metodami **ChReceive** a **ChReceiveChar** k dalšímu zpracování. Znaky určené k odeslání jsou zasílány z výstupního bufferu do linky. Určení parametrů komunikace je voleno parametrem nastavovací metody **ChSetParam**.

Knihovna rovněž definuje objekt **tAddChnCom2**, který je dědicem od rodičovského objektu **tAddChnVirt**. Objekt **tAddChnCom2** zajistí, aby daný komunikační objekt (objekt **tChnCom2**) byl k aplikaci připojen a popřípadě zajistí vytvoření instance tohoto objektu. Po přilinkování této jednotky do aplikace (uses ChnCom2), se jméno objektu **tChnCom2** automaticky vloží do seznamu objektů správců komunikačních objektů pro případné použití.

Protože je objekt **tChnCom2** 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   = 'COM2';
```

Konstanta **cName** definuje jméno komunikačního objektu **tChnCom2**.

#### 4.1. Konstanty kódů příkazů pro metodu ChGetBinParam

---

```
cmd_GetMSR   = $0101;
```

Konstanta definuje kód příkazu pro vrácení stavu všech modemových signálů (Modem Status Registeru).

```
cmd_GetDCTS  = $0102;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu DCTS v 0.bitu výsledku funkce.

```
cmd_GetDDSR  = $0103;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu DDSR v 0.bitu výsledku funkce.

```
cmd_GetTERI  = $0104;
```

Konstanta definuje kód příkazu pro vrácení stavu modemového signálu TERI v 0.bitu výsledku funkce.

cmd\_GetDRLSD = \$0105;  
 Konstanta definuje kód příkazu pro vrácení stavu modemového signálu DRLSD v 0.bitu výsledku funkce.

cmd\_GetCTS = \$0106;  
 Konstanta definuje kód příkazu pro vrácení stavu modemového signálu CTS v 0.bitu výsledku funkce.

cmd\_GetDSR = \$0107;  
 Konstanta definuje kód příkazu pro vrácení stavu modemového signálu DSR v 0.bitu výsledku funkce.

cmd\_GetRI = \$0108;  
 Konstanta definuje kód příkazu pro vrácení stavu modemového signálu RI v 0.bitu výsledku funkce.

cmd\_GetRLSD = \$0109;  
 Konstanta definuje kód příkazu pro vrácení stavu modemového signálu RLSD v 0.bitu výsledku funkce.

## 4.2. Konstanty kódů příkazů pro metodu ChSetBinParam

---

cmd\_SetMCR = \$0101;  
 Konstanta definuje kód příkazu pro nastavení všech modemových signálů (Modem Control Registeru) dle nejnižšího byte parametru funkce Param.

cmd\_SetDTR = \$0102;  
 Konstanta definuje kód příkazu pro nastavení modemového signálu DTR dle 0.bitu parametru funkce Param.

cmd\_SetRTS = \$0103;  
 Konstanta definuje kód příkazu pro nastavení modemového signálu RTS dle 0.bitu parametru funkce Param.

## 4.3. Konstanty a typy přijímacích a vysílacích bufferů

---

tTChar = byte;  
 Typ pro položku vysílacího bufferu.

tRChar = record  
 Hod : byte;  
 Sts : byte;  
 end;  
 Typ pro položku přijímacího bufferu.

MaxRBuf = 65500 div SizeOf(tRChar);  
 Maximální velikost přijímacího bufferu (32750).

MaxTBuf = 65500 div SizeOf(tTChar);  
 Maximální velikost vysílacího bufferu (65500).

tAByte = array[0..65500] of Byte;  
 Typ pole byte.

pAByte = ^tAByte;  
 Typ ukazatele na pole byte.

tTBuf = array[0..MaxTBuf] of tTChar;  
 Typ vysílacího bufferu.

tRBuf = array[0..MaxRBuf] of tRChar;  
 Typ přijímacího bufferu.

pTBuf = ^tTBuf;  
 Typ ukazatele na vysílací buffer.

pRBuf = ^tRBuf;  
 Typ ukazatele na přijímací buffer.

## 4.4. Konstanty a typy pro počet obsluhovaných COM portů

---

MaxCom2 = 4;

Maximální počet obsluhovaných sériových portů. Hodnota této konstanty je závislá na počtu definovaných interních datových struktur v této knihovně pro obsluhu přerušeni od obvodu i8250.

tCtCom2 = 1..MaxCom2;

Intervalový typ počtu sériových portů.

## 5. Proměnné

---

AChnCom2 : array [tCtCom2] of pChnCom2 = (nil, nil, nil, nil);

Pole ukazatelů na komunikační objekty tChnCom2 pro obsluhu obvodu i8250.

## 6. Objekty

---

### 6.1. tChnCom2

---

#### 6.1.1. Položky

---

CH\_Cislo : Byte;

Položka **CH\_Cislo** určuje číslo sériového kanálu. Jen pro vnitřní použití.

CH\_Addr : Word;

Položka **CH\_Addr** obsahuje adresu sériového komunikačního adaptéru i8250.

CH\_Rate : tRate;

Položka **CH\_Rate** obsahuje požadovanou přenosovou rychlost v bitech za sekundu.

CH\_Parity : tParity;

Položka **CH\_Parity** obsahuje požadovanou paritu přenášeného znaku.

CH\_Stop : tStop;

Položka **CH\_Stop** obsahuje počet stop bitů přenášeného znaku.

CH\_Length : tLength;

Položka **CH\_Length** obsahuje počet datových bitů přenášeného znaku.

CH\_RSDelay1 : Longint;

Položka **CH\_RSDelay1** definuje minimální časovou prodlevu v ms před vysíláním.

CH\_RSDelay2 : Longint;

Položka **CH\_RSDelay2** definuje minimální časovou prodlevu v ms po vysílání.

CH\_RecOn : Boolean;

Položka **CH\_RecOn** určuje, zda se má během vysílání ponechat povolený příjem znaků či zda ho zakázat.

CH\_RecDel : Boolean;

Položka **CH\_RecDel** určuje, zda se má (při nastaveném zakázání příjmu během vysílání) vypustit první přijatý znak po skončeném vysílání.

CH\_STime : tTimer;

Položka **CH\_STime** je určena pro vnitřní použití, slouží pro odměřování časových intervalů vysílače.

CH\_STick : Boolean;



Položka **CH\_STick** je určena pro vnitřní použití, pro určení, zda je vykonávána činnost vysílacího automatu.

```
CH_FlDTR      : Boolean;
```

Položka **CH\_FIDTR** definuje výstupní signál DTR (hodnota True odpovídá nastaví signálu na hodnotu 1).

```
CH_FlRTS     : Boolean;
```

Položka **CH\_FIRTS** definuje výstupní signál RTS (hodnota True odpovídá nastaví signálu na hodnotu 1).

## 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 nejdříve 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_Cislo     := 1;
CH_Addr      := ACom1;
CH_Rate      := 9600;
CH_Parity    := ParOdd;
CH_Stop      := Stop1;
CH_Length    := Bits8;
CH_RSDelay1  := 0;
CH_RSDelay2  := 0;
CH_RecOn     := true;
CH_RecDel    := false;
CH_STick     := false;
CH_FlDTR     := true;
CH_FlRTS     := false;
```

### 6.1.2.2. ChInitParam konstruktor

```
constructor ChInitParam(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**.

### 6.1.2.3. Done destruktork

```
destructor Done;
```

Destruktor **Done** slouží ke zrušení instance komunikačního objektu. Pokud je v paměti alokovan přijímací buffer, bude odstraněn z paměti a poté je zavolán zděděný destruktork **Done** z rodičovského objektu (inherited Done).

### 6.1.2.4. ChSetOneParam funkce

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

Metoda **ChSetOneParam** slouží k dekodování a nastavení jednoho konkrétního parametru, který je zadán v parametru S. Tato metoda se volá v aplikaci

prostřednictvím metody **ChSetParam**. Metoda **ChSetOneParam** komunikačního objektu tChnCom2 dekóduje tyto parametry:

**COM=aaa**

Parametr **COM** určuje číslo sériového kanálu COM. aaa může nabývat hodnot 1 až 8. Adresy odpovídající jednotlivým kanálům COM jsou uvedeny v jednotce ChnTypes.

**ADD=bbb**

Parametr **ADD** "Address" určuje adresu sériového kanálu COM. Použije se jen v případě, kdy komunikační adapter i8250 není na standardní vstupně/výstupní adrese. bbb může nabývat hodnot 0 až \$ffff.

**BD=ddd**

Parametr **BD** "BaudRate" určuje přenosovou rychlost požadované sériové komunikace. ddd může nabývat hodnot 25 až 115200 Bd.

**BIT=eee**

Parametr **BIT** "Number of Data Bits" určuje počet datových bitů v přenášeném znaku. eee může nabývat hodnot 5 až 8.

**PAR=fff**

Parametr **PAR** "Parity" určuje paritu přenášeného znaku. fff může nabývat hodnot 0 "Odd" pro lichou paritu, E "Even" pro sudou paritu a N "None" pro znak bez parity.

**STO=ggg**

Parametr **STO** "Number of Stop Bits" určuje počet stop-bitů v přenášeném znaku. ggg může nabývat hodnot 1 nebo 2.

**LRB=hhh**

Parametr **LRB** "Length of Receive Buffer" určuje velikost vstupního kruhového vyrovnávacího bufferu. Buffer je alokován na heapu a každá jeho položka zaujímá v paměti prostor o velikosti 2 byte (přijatý znak a status). Platí, čím větší je vstupní buffer, tím více znaků dokáže udržet, aniž by byly znaky metodami **ChReceive** a **ChReceiveChar** odebírány. Velikost bufferu je shora omezena na 32750. Je proto nutno volit kompromis mezi velikostí bufferu a periodou, kterou přijaté znaky odebíráme.

**RS1=iii**

Parametr **RS1** určuje zpoždění v ms před vysláním.

**RS2=jjj**

Parametr **RS2** určuje zpoždění v ms po vyslání.

**REC=kkk**

Parametr **REC** "Receive While Sending" určuje, zda má být při vyslání povolen příjem znaků. kkk může nabývat hodnot ON, OFF nebo OFF2. Hodnota ON znamená, že příjem znaků je povolen i během vyslání. Hodnota OFF znamená, že během vyslání je příjem znaků potlačen (pro RS232 a RS485 - 4drát). Pro RS422 - 2drát se poslední vyslaný znak zprávy automaticky zpětně přijímá. Proto hodnota OFF2 má stejný význam jako hodnota OFF, ale navíc se vypouští první zpětně přijatý znak po odvyslání zprávy.

**DTR=lll**

Parametr **DTR** "Setting of DTR Signal" určuje, zda má být nastaven výstupní signál DTR na hodnotu 1 (lll=ON) či na hodnotu 0 (lll=OFF).

**RTS=mmm**

Parametr **RTS** "Setting of RTS Signal" určuje, zda má být nastaven výstupní signál RTS na hodnotu 1 (mmm=ON) či na hodnotu 0 (mmm=OFF).

Příklad:

Příklad ukazuje, jak je možné v jednotce COM2 nastavit parametry komunikace na sudou paritu, přenosovou rychlost 4800 Bd a velikost vstupního vyrovnávacího bufferu na 1000 položek.

```
ChSetParam('NAM=COM2 PAR=E BD=4800 LRB=1000');
```

Pozn: Všimněte si, že není volána metoda ChSetOneParam, ale metoda ChSetParam.

### 6.1.2.5. ChGetParam funkce

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

Metoda **ChGetParam** navrácí 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. ChSetBinParam procedura

```
procedure ChSetBinParam(NumName: tChNumName; Code: Word;  
                        Param: longint);
```

Metodou **ChSetBinParam** s parametrem NumName příslušným číselnému jménu tohoto komunikačního objektu, lze podle parametru Code nastavit tyto nastavení:

```
Code = cmd_SetMCR
```

Podle nejnižšího byte parametru Param nastaví obsah Modem Control Registeru - obraz všech výstupních modemových signálů.

```
Code = cmd_SetDTR
```

Podle 0.bitu parametru Param nastaví nebo vynuluje modemový signál DTR.

```
Code = cmd_SetRTS
```

Podle 0.bitu parametru Param nastaví nebo vynuluje modemový signál RTS.

### 6.1.2.7. ChGetBinParam funkce

```
function ChGetBinParam(NumName: tChNumName; Code: Word): longint;
```

Metodou **ChGetBinParam** s parametrem NumName příslušným číselnému jménu tohoto komunikačního objektu, lze podle parametru Code vrátit tyto nastavení:

```
Code = cmd_GetMSR
```

Vrátí obsah Modem Status Registeru - obraz všech vstupních modemových signálů.

```
Code = cmd_GetDCTS
```

Vrátí v 0.bitu příznak modemového signálu DCTS.

```
Code = cmd_GetDDSR
```

Vrátí v 0.bitu příznak modemového signálu DDSR.

```
Code = cmd_GetTERI
```

Vrátí v 0.bitu příznak modemového signálu TERI.

```
Code = cmd_GetDRLSD
```

Vrátí v 0.bitu příznak modemového signálu DRLSD.

```
Code = cmd_GetCTS
```

Vrátí v 0.bitu příznak modemového signálu CTS.

```
Code = cmd_GetDSR
```

Vrátí v 0.bitu příznak modemového signálu DSR.

Code = cmd\_GetRI

Vrátí v 0.bitu příznak modemového signálu RI.

Code = cmd\_GetRLSD

Vrátí v 0.bitu příznak modemového signálu RLSD.

### 6.1.2.8. ChOpen procedura

procedure ChOpen;

Metoda **ChOpen** nastaví technické vybavení komunikačního kanálu, a pokud nastavení proběhlo v pořádku, způsobí přechod kanálu do stavu **CHS\_Open**.

### 6.1.2.9. ChClose procedura

procedure ChClose;

Metoda **ChClose** uzavře komunikační kanál provedením deinicializace technického vybavení a způsobí přechod do stavu **CHS\_Close**. Lze opětovně volat metodu **ChOpen**.

### 6.1.2.10. ChConnect procedura

procedure ChConnect;

Před voláním této metody musí být kanál ve stavu **CHS\_Open**. Metoda **ChConnect** provede inicializaci pro příjem a vysílání znaků a pokud nastavení proběhlo v pořádku, způsobí přechod do stavu **CHS\_Connect**. To znamená, že je možno po daném kanále komunikovat. V tomto stavu je možno přijímat data z komunikační linky, naopak je možno požadovaná data odvíšlat.

### 6.1.2.11. ChDisconnect procedura

procedure ChDisconnect;

Metoda **ChDisconnect** ukončí navázané komunikační spojení a uvede kanál do stavu **CHS\_DisConnect**. Je přerušen příjem a vysílání zpráv. Po volání této metody lze opětovně volat metodu **ChConnect**.

### 6.1.2.12. ChSendTick procedura

procedure ChSendTick;

Metoda **ChSendTick** způsobí provedení kroků vysílacích automatů. Je nutné ji periodicky volat během vysílání. **ChSendTick** je rovněž automaticky volána v metodách **ChSendReady** a **ChSend**.

### 6.1.2.13. ChSend procedura

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

Pokud je kanál ve stavu **CHS\_Connect**, způsobí metoda **ChSend** započítí vysílání zprávy délky Len uložené na adrese určené ukazatelem Buff. Pokud je parametr Len = 0, nebude se vysílat žádná zpráva. Po volání této metody by mělo následovat volání metody **ChSendReady** s testem na **CHS\_SendReady** (čekací smyčka do odvíšlení zprávy).

#### 6.1.2.14. ChSendReady funkce

```
function ChSendReady: TChState;
```

Metoda **ChSendReady** způsobí provedení kroku vysílacího automatu na základě volání metody **ChSendTick**. Jako svoji funkční hodnotu vrátí aktuální stav automatu vysílače komunikačního kanálu, který je uložen v položce **CH\_SCtrl**. Pokud kanál není ve stavu **CHS\_Connect**, vrací metoda stav **CHS\_SendNoReady**.

#### 6.1.2.15. ChSendFlush procedura

```
procedure ChSendFlush;
```

Pokud je kanál ve stavu **CHS\_Connect** způsobí metoda **ChSendFlush** ukončení vysílání a přechod automatu vysílače do stavu **CHS\_SendReady**.

#### 6.1.2.16. ChReceiveReady funkce

```
function ChReceiveReady: TChState;
```

Pokud je kanál ve stavu **CHS\_Connect** a v přijímacím bufferu jsou přijata nějaká data, vrátí metoda **ChReceiveReady** stav **CHS\_ReceiveReady**. V opačném případě vrátí stav **CHS\_ReceiveNoReady**.

#### 6.1.2.17. ChReceiveChar funkce

```
function ChReceiveChar: Byte;
```

Pokud je kanál ve stavu **CHS\_Connect** a v přijímacím bufferu jsou přijata nějaká data, navrací metoda **ChReceiveChar** jeden přijatý znak z přijímacího bufferu a nastaví výsledek operace přijímače na status tohoto přijatého znaku. Metodu **ChReceiveChar** je možno volat pouze ve stavu automatu přijímače **CHS\_ReceiveReady**, proto před voláním této metody musí předcházet volání metody **ChReceiveReady** s testem na stav **CHS\_ReceiveReady**, jinak v případě nepřijetí žádného znaku skončí volání metody **ChReceiveChar** chybou.

#### 6.1.2.18. ChReceive procedura

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

Pokud je kanál ve stavu **CHS\_Connect** a je nadefinován buffer pro uložení přijaté zprávy (metodou **ChReceiveBuffer**), provede metoda **ChReceive** přijetí zprávy a její uložení do přijímacího bufferu. V proměnné **Len** navrací délku přijaté zprávy. Ve svém těle volá metodu **ChReceiveChar** (pro přijetí jednoho znaku zprávy) tak dlouho, dokud je přijímač ve stavu **CHS\_ReceiveReady**.

#### 6.1.2.19. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

Metoda **ChReceiveFlush** způsobí vyprázdnění přijímacích bufferů a nastaví stav přijímače kanálu **CH\_RCtrl** na stabilní stav **CHS\_ReceiveNoReady**.

---

## 6.2. tAddChnCom2

---

Typ **tAddChnCom2** je typem objektu, který slouží k definování prvku v seznamu správců komunikačních objektů (tzv. správce komunikačního objektu **tChnCom2** v seznamu správců). Objekt **tAddChnCom2** 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 **tChnCom2** a ukazatel na instanci tohoto objektu vrací jako svoji funkční hodnotu.

---

## 7. Procedury

---

Uvedené procedury nejsou součástí žádného objektu.

### 7.1. MakeCom2

```
procedure MakeCom2;
```

Procedura **MakeCom2** provádí obsluhu všech kanálů, v jejichž komunikačním objektu se úspěšně provedla metoda **ChConnect**. Procedura přijímá a vysílá každým připojeným kanálem. Frekvence vyvolávání by měla být vyšší nebo rovna přenosové rychlosti nejrychlejšího kanálu. Například pro přenosovou rychlost 9600 Baudů je třeba volat proceduru **MakeCom2** s periodou rovno nebo menší než 1 ms. Pokud chceme použít k vyvolávání procedury přerušení, použijeme proceduru **SetIntrCom2**.

### 7.2. SetIntrCom2

```
procedure SetIntrCom2(I: Byte);
```

Procedura **SetIntrCom2** zřetězeně připojí k přerušovacímu vektoru **I** přerušovací proceduru, která vyvolává metodu **MakeCom2**. Tím je připojeno vyvolávání **MakeCom2** k interruptu **I**.

### 7.3. ResetIntrCom2

```
procedure ResetIntrCom2;
```

Procedura **ResetIntrCom2** je inverzní k proceduře **SetIntrCom2**. Procedura odstraňuje přerušovací proceduru z řetězce přerušení.

---

## 8. Příklad

---

Příklad ukazuje použití komunikační jednotky **ChnCom2**.

```
uses  
  uString,  
  ChnVirt,
```

```

    ChnCom2,
    Tick;
    ...

const
    ParamStr : tParamStr =
        'NAM=COM2 COM=1 BD=9600 BIT=8 STOP=2 LRB=1000';
    cTickDiv = 55;

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

var
    Chn        : pChnVirt;
    SMess      : ^tMess;
    RMess      : ^Byte;
    LSMess     : Word;

begin
    ...
    New(SMess);
    New(RMess);
    ...
    { zrychlení systémového časovače na 1 ms pro pravidelné vyvolávání
      MakeCom2 }
    InitTick;
    SetTickDivider(cTickDiv);
    asm
        pushf;
        cli;
    end;
    UserTick1:=MakeCom2;
    asm
        popf;
    end;
    ...
    { inicializace Chn }
    Chn:=ChnCollection^.ChNewInit(ChnCom2.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;
        { definování místa, kam se má přijatá zpráva uložit }
        ChReceiveBuffer(RMess,1);
        if ChReceiveResult<>res_Ok then WriteLn('Chyba');
        ChConnect;
        repeat
            if ChResult<>res_Ok then WriteLn('Chyba');
        until ChReady=CHS_Connect;
        ...
        { naplnění zprávy daty }
        ...
        { vyslání zprávy }
        if ChSendReady=CHS_SendReady then
        begin
            ChSend(SMess, LSMess);
            { čekání na odvyšlání zprávy }
            repeat
                if ChSendResult<>res_Ok then WriteLn('Chyba');
            until ChSendReady=CHS_SendReady;
            if ChSendResult<>res_Ok then WriteLn('Chyba');
            ...
        end;
    end;

```

```
end;
...
{ čekání na příjem zprávy }
repeat
until ChReceiveReady=CHS_ReceiveReady;
{ příjem zprávy }
RMess^:=ChReceiveChar;
if ChReceiveResult<>res_Ok then WriteLn('Chyba');
...
{ ukončení }
ChDisconnect;
repeat
if ChResult<>res_Ok then WriteLn('Chyba');
until ChReady=CHS_DisConnect;
ChClose;
repeat
if ChResult<>res_Ok then WriteLn('Chyba');
until ChReady=CHS_Close;
end;
{ zrušení instance objektu }
Dispose(Chn,Done);
{ obnovení původního časovače }
asm
pushf;
cli;
end;
UserTick1:=Nic;
asm
popf
end;
...
end.
```