

# ChnComT

JEDNOTKA SÉRIOVÉ KOMUNIKACE  
RS232 / RS485 S OBVODEM I8250 S  
MĚŘENÍM ČASŮ PŘÍCHOZÍCH ZNAKŮ

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 nenes 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 nenes žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 23.09.2005

Datum posledního uložení dokumentu: 23.09.2005

(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 jednoduché typy	6
4.1.	Konstanty statusu přijatého znaku	6
4.2.	Konstanty kódů pro metodu ChGetBinParam	7
4.3.	Konstanty kódů příkazů pro metodu ChSetBinParam	8
4.4.	Konstanty a typy přijímacích a vysílacích bufferů	8
4.5.	Konstanty a typy pro počet obsluhovaných COM portů	9
5.	Proměnné	9
6.	Objekty	9
6.1.	tChnComT	9
6.1.1.	Položky	9
6.1.2.	Metody	10
6.1.2.1.	Init konstruktor	10
6.1.2.2.	ChInitParam konstruktor	11
6.1.2.3.	Done destruktork	11
6.1.2.4.	ChSetOneParam funkce	11
6.1.2.5.	ChGetParam funkce	14
6.1.2.6.	ChSetBinParam procedura	14
6.1.2.7.	ChGetBinParam funkce	14
6.1.2.8.	ChOpen procedura	15
6.1.2.9.	ChClose procedura	15
6.1.2.10.	ChConnect procedura	15
6.1.2.11.	ChDisconnect procedura	16
6.1.2.12.	ChSendTick procedura	16
6.1.2.13.	ChSend procedura	16
6.1.2.14.	ChSendReady funkce	16
6.1.2.15.	ChSendFlush procedura	16
6.1.2.16.	ChReceiveReady funkce	16
6.1.2.17.	ChReceiveChar funkce	16
6.1.2.18.	ChReceive procedura	17
6.1.2.19.	ChReceiveFlush procedura	17
6.2.	tAddChnComT	17
6.2.1.	Metody	17
6.2.1.1.	ChInit funkce	17
7.	Procedury	17
8.	Příklad	18



## 1. O dokumentu

---

### 1.1. Revize dokumentu

---

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Wil		První vydání.
1.10	1.XX	Tum	22.05.2003	Úprava dokumentu dle ISO9000.
1.20	1.XX	Wil	23.09.2005	Přidána obsluha FIFO. Přidány čítače počtu přerušení. Přidána detekce nefungujícího IRQ při vysílání. Upraven popis pro nastavování IRQ.

### 1.2. Účel dokumentu

---

Tento dokument slouží jako popis jednotky sériové komunikace RS232 / RS485 s obvodem i8250 s měřením časů příchozích znaků.

### 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 rodičovský prvek komunikačních objektů. Dále s manuálem ChnTypes definujícím typy a konstanty pro komunikační knihovny.

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 ChnComT definuje objekt **tChnComT**, jehož instance vytváří fyzickou vrstvu v komunikačním kanálu tvořenou sériovým rozhraním RS232 nebo RS485 s obvodem i8250. Ke své činnosti využívá přerušovacího systému počítače. Knihovna ChnComT dále umožňuje zjišťovat status přijetí dalšího znaku po navolené časové pauze. K tomu využívá již dříve naprogramovaného (prostřednictvím BIOSu) obvodu i8253 a přerušování BIOSu INT 1Ah (uživatelský časovač). V případě přijetí znaku po navolené časové pauze se nejdříve vygeneruje znak s nastaveným adresním bitem ve statusu (příznak, že se jedná o adresní znak) a poté se vygeneruje tentýž znak již bez nastaveného adresního bitu ve statusu. To znamená, že příjem jednoho znaku po navolené časové pauze se generuje ve formě dvou znaků. Tento způsob se provádí vzhledem ke kompatibilitě s knihovnou ChnComBR, kdy se přijetí adresního znaku generuje také ve formě dvou znaků, z nichž první určuje, že jde právě o adresní znak (viz. příručka ChnComBR). Tím je zajištěno jednotné rozhraní rozlišování adresních znaků od normálních, čehož využívají některé vyšší komunikační protokoly pro určení začátku zprávy. Aby knihovna ChnComT správně dekodovala pauzy mezi příchozími znaky, je nutno v přerušování INT 08h (systémový časovač) volat příslušnou proceduru IncRecTime (viz. 7. Procedurey).

Znaky přicházející po komunikační lince jsou v přerušovací proceduře 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 rovněž s využitím přerušovacího systému.

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

Protože je objekt **tChnCom** 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 jednoduché 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 = 'COMT';
```

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

#### 4.1. Konstanty statusu přijatého znaku

---

Následující chyby vrací metoda **ChReceiveResult** po příjmu znaku metodou **ChReceiveChar**. Jelikož v některých případech může nastat i více chyb najednou,

definují tyto konstanty tzv. bitové masky pro spodní byte funkční hodnoty metody **ChReceiveResult**.

```
res_ErrRecBuffOvf = $0080;
```

Přijatý znak způsobil přetečení přijímacího bufferu – buffer byl vymazán.

```
res_ErrOverrun = ChnTypes.MErrOvr;
```

Maska pro dekódování chyby Overrun.

```
res_ErrParity = ChnTypes.MErrPar;
```

Maska pro dekódování chyby Parity

```
res_ErrFraming = ChnTypes.MErrFrm;
```

Maska pro dekódování chyby Framing.

```
res_BreakInt = ChnTypes.MBrkInt;
```

Maska pro dekódování Break Interrupt.

Následující chyby vrací metoda **ChSendResult** po metodách **ChSend**, **ChSendReady**, **ChSendTick**.

```
res_ErrSendTimeout = $0030;
```

Timeout při vysílání (nechodí přerušení).

Následující chyby vrací metoda **ChResult** po metodě **ChOpen**.

```
res_ErrFIFOunavailable = $0040;
```

Vyžaduje se FIFO, které ale není k dispozici.

## 4.2. Konstanty kódů 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.

`cmd_GetCntIrq = $0110;`  
Konstanta definuje kód příkazu pro vrácení počtu všech vyvolaných přerušení.

`cmd_GetCntRecB = $0111;`  
Konstanta definuje kód příkazu pro vrácení počtu přerušení způsobených příjmem znaku nebo statusem.

`cmd_GetCntSendB = $0112;`  
Konstanta definuje kód příkazu pro vrácení počtu přerušení způsobených odvysíláním znaku.

`cmd_GetFIFOenabled = $0120;`  
Konstanta definuje kód příkazu pro vrácení příznaku, je-li skutečně povoleno FIFO (volat po metodě ChOpen).

`cmd_GetRecTOut = $0201;`  
Kód pro vrácení navolené časové pauzy pro rozlišení dlouhých mezer mezi přijatými znaky.

`cmd_GetRecTOutChar = $0202;`  
Kód pro vrácení statusu přijetí znaku po navolené časové pauze po volání metody ChReceiveChar.

### 4.3. 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.

### 4.4. 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.

`tTBuff = array[0..MaxTBuf] of tTChar;`



Typ vysílacího bufferu.  
 tRBuff = array[0..MaxRBuf] of tRChar;  
 Typ přijímacího bufferu.  
 pTBuff = ^tTBuff;  
 Typ ukazatele na vysílací buffer.  
 pRBuff = ^tRBuff;  
 Typ ukazatele na přijímací buffer.

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

---

MaxCom = 4;  
 Maximální počet obsluhovaných sériových portů. Hodnota této konstanty je závislá na počtu definovaných interních datových struktur v této knihovně pro obsluhu přerušení od obvodu i8250.  
 tCtCom = 1..MaxCom;  
 Intervalový typ počtu sériových portů.

## 5. Proměnné

---

AChnComT : array [tCtCom] of pChnComT = (nil, nil, nil, nil);  
 Pole ukazatelů na komunikační objekty tChnComT pro obsluhu obvodu i8250.

## 6. Objekty

---

### 6.1. tChnComT

---

#### 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\_Irq : tIrq;  
 Položka **CH\_Irq** obsahuje číslo přerušení IRQ, na kterém adaptér i8250 žádá o přerušení.

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 vysláním.

`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 nechat 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_FifoEnb` : Boolean;  
Položka **CH\_FifoEnb** určuje, zda se má používat přijímací a vysílací FIFO.

`CH_FifoErr` : Boolean;  
Položka **CH\_FifoErr** určuje, zda se má hlásit chyba, když daný komunikační obvod i8250 neumožňuje FIFO a aplikace ho vyžaduje.

`CH_STime` : tTimer;  
Položka **CH\_STime** je určena jen pro vnitřní použití, pro odměřování časových intervalů vysílače.

`CH_STick` : Boolean;  
Položka **CH\_STick** je určena jen pro vnitřní použití, pro určení, zda je vykonávána činnost vysílacího automatu.

`CH_FlDTR` : Boolean;  
Položka **CH\_FlDTR** definuje výstupní signál DTR (hodnota True odpovídá nastaví signálu na hodnotu 0).

`CH_FlRTS` : Boolean;  
Položka **CH\_FlRTS** definuje výstupní signál RTS (hodnota True odpovídá nastaví signálu na hodnotu 0).

`CH_TCU_Frekv`: Real;  
Položka **CH\_TCU\_Frekv** určuje hodnotu vstupní frekvenci do systémového časovače v kHz. Při změně frekvence systémového časovače uživatel musí dodatečně změnit patřičně i hodnotu této proměnné.

`CH_TCU_Hi` : Word;  
Položka **CH\_TCU\_Hi** určuje horní mez pro dekrementaci systémového časovače. Při změně systémového časovače (např. pomocí jednotky Tick) uživatel musí dodatečně změnit patřičně i hodnotu této proměnné.

`CH_RecTOut` : Word;  
Položka **CH\_RecTOut** určuje maximální časovou pauzu v 0,1 ms pro rozlišení dlouhých pauz mezi přijatými znaky.

`CH_RecTOutChar`: Boolean;  
Položka **CH\_RecTOutChar** určuje příznak přijetí znaku po nastavené časové pauze. Tato položka se nastavuje voláním metody `ChReceiveChar`.

## 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_Irq       := Irq4;
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_FifoEnb   := false;
CH_FifoErr   := false;
CH_STick     := false;
CH_FlDTR     := false;
CH_FlRTS     := true;
CH_TCU_Frekv:= ITimerClk/1000;
if ITCU>$FFFF then
    CH_TCU_Hi:= $FFFF
else CH_TCU_Hi:= ITCU;
CH_RecTOut   := $FFFF;
CH_RecTOutChar:=false;

```

### 6.1.2.2. 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**.

### 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 a poté je zavolán zděděný destruktork **Done** z objektu tChnVirt (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**, tzn že metoda **ChSetOneParam** by se v aplikaci neměla volat nikdy (volat vždy jen metodu **ChSetParam**, která zajistí správné volání metody **ChSetOneParam**). Metoda **ChSetOneParam** komunikačního objektu tChnCom dekoduje 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** "Adres" určuje adresu sériového kanálu COM. Použije se jen v případě, kdy komunikační adaptér i8250 není na standardní vstupně/výstupní adrese. bbb může nabývat hodnot 0 až \$ffff.

**IRQ=ccc**

Parametr **IRQ** určuje číslo přerušení IRQ, na kterém adaptér i8250 žádá o zpracování přerušení. Hodnota ccc je závislá na použitém typu procesoru. Výčet použitelných IRQ definuje intervalový typ tIRQ v knihovně Hwsyst.

**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 alokovan 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 vysíláním.

**RS2=jjj**

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

**REC=kkk**

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

**DTR=lll**

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

**RTS=mmm**

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

#### **FIF=ON / OFF / IFAVAIL**

Parametr **FIF** "FIFO" určuje, zda se má používat vysílací a přijímací FIFO. Hodnota **ON** znamená zapnutí FIFO, přičemž pokud daný obvod i8250 FIFO neumožňuje, nastaví se chyba **res\_ErrFIFOunavailable**. Hodnota **OFF** znamená vypnutí FIFO (implicitně). Hodnota **IFAVAIL** znamená zapnutí FIFO, přičemž pokud daný obvod i8250 FIFO neumožňuje, nehlásí se žádná chyba a k obvodu se přistupuje jako k obvodu bez FIFO. Z praktického hlediska se doporučuje spíše používat hodnotu **IFAVAIL** než hodnotu **ON**.

#### **RTM=iii**

Parametr **RTM** (Receive TimeOut) určuje maximální časovou pauzu v 0,1 ms pro rozlišení dlouhých pauz mezi přijatými znaky. Pokud byl přijat znak po delší časové prodlevě než  $iii/10$  ms od minule přijatého znaku, vyhodnotí se jako tzv. adresní znak. V opačném případě jde o normální znak zprávy. Vyhodnocování adresních/normálních znaků viz. metoda **ChGetBinParam**.

#### **TFR=jjj**

Parametr **TFR** (Timer Frequency) určuje hodnotu vstupní frekvence do systémového časovače v kHz. Tento parametr je od počátku inicializace instance objektu (voláním konstrukturu Init) nastaven na správnou hodnotu a v současné době není třeba tuto hodnotu měnit. Tento parametr je zde pouze pro úplnost a případné další použití.

#### **THI=kkk**

Parametr **THI** (Timer Hi Value) určuje horní mez pro dekrementaci systémového časovače. Tento parametr je od počátku inicializace instance objektu (voláním konstrukturu Init) nastaven na správnou hodnotu a pokud aplikace nezrychluje systémový časovač, není třeba tento parametr vůbec použít. Jinak je tomu při zrychlování systémového časovače.

**Důležité! : Při zrychlení systémového časovače pomocí jednotky Tick se horní mez pro dekrementaci systémového časovače zmenšuje. Proto uživatel musí tímto parametrem patřičně změnit i hodnotu proměnné CH\_TCU\_Hi.**

Způsob nastavení hodnoty parametru THI:

Pokud uživatel volá metodu **SetTickDivider(TD)** z knihovny **Tick** pro zrychlení systémového časovače (TD udává násobek zrychlení), bude výpočet hodnoty parametru THI následující:

Při používání procesoru KitV40: (54926 div TD) and \$FFFF

Při používání procesoru PC či Kit386: (65536 div TD) and \$FFFF

Čísla 54926 nebo 65536 je hodnota konstanty **ITCU** z knihovny **HWSyst** pro danou platformu (procesor).

Pokud uživatel volá metodu **SetTimeUser1(TM)** z knihovny **Tick** pro zrychlení systémového časovače (TM udává periodu zrychleného časovače v ms), bude výpočet hodnoty parametru THI následující:

Při používání procesoru KitV40:  $\text{Trunc}(1000,000 * TM + 0,5)$  and \$FFFF

Při používání procesoru Kit386 25MHz  $\text{Trunc}(1190,476 * TM + 0,5)$  and \$FFFF

Při používání procesoru Kit386 33MHz  $\text{Trunc}(1178,572 * TM + 0,5)$  and \$FFFF

Při používání procesoru PC: `Trunc(1191,563*TM+0,5)` and `$FFFF`

Čísla 1000,000 až 1191,563 je hodnota konstanty **ItimerClk/1000** z knihovny **HWSyst** pro daný procesor.

Příklad:

Příklad ukazuje, jak je možné v jednotce COMT nastavit parametry komunikace na sudou paritu, přenosovou rychlost 4800 Bd, velikost vstupního vyrovnávacího bufferu na 1000 položek a maximální časovou pauzu pro dekodování dlouhých pauz mezi přijatými znaky na 3 ms. Systémový časovač není zrychlen.

```
ChSetParam('NAM=COMT PAR=E BD=4800 LRB=1000 RTM=30');
```

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;
```

Metoda **ChGetBinParam** s parametrem `NumName` příslušným 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.

Code = cmd\_GetCntIrq  
Vrátí počet všech vyvolaných přerušení.

Code = cmd\_GetCntRecB  
Vrátí počet přerušení způsobených příjmem znaku nebo statusem.

Code = cmd\_GetCntSendB  
Vrátí počet přerušení způsobených odvíláním znaku.

Code = cmd\_GetFIFOenabled  
Vrátí v 0.bitu příznak, je-li skutečně povoleno FIFO (volat po metodě ChOpen).

Code = cmd\_GetRecTOut  
Vrátí hodnotu navolené časové pauzy pro rozlišení dlouhých mezer mezi přijatými znaky v 0,1 ms.

Code = cmd\_GetRecTOutChar  
Vrátí příznak přijetí znaku po navolené časové pauze při posledním přijetí znaku metodou **ChReceiveChar**. Je-li vrácená hodnota = 0, potom byl přijat znak před dosažením navoleného času (normální znak). Je-li vrácená hodnota = 1, potom byl přijat znak po navolené časové pauze (adresní znak).

#### 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 deinitializace 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** započetí 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 odvysílání 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, navrácí metoda **ChReceiveChar** jeden přijatý znak z přijímacího bufferu. Vždy se dekóduje status přijatého znaku na nastavení bitu, který znamená přijetí



znaku po navolené časové pauze. Podle tohoto testu se nastaví položka `CH_RecTOutChar` a bit překročení časové pauzy se ze statusu odstraní. Výsledek operace přijímače se nastaví na výsledný status 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` navrácí 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. tAddChnComT

---

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

## 7. Procedury

---

```
procedure IncRecTime(I:tCtCom);
```

Procedura `IncRecTime` provede inkrementaci čítače pauz mezi příchozími znaky pro příslušný kanál `I`. **Tuto proceduru je nutno volat v přerušení INT 08h (systémový časovač) nebo (pokud se používá knihovna Tick) v uživatelské proceduře UserTick1 (ve zrychleném časovači)!**

## 8. Příklad

---

Příklad ukazuje použití komunikační jednotky ChnComT. Je vytvořen komunikační kanál definovaných vlastností, po kterém je zasílána zpráva a z kterého je poté očekáván příjem zpráv.

```

uses
  uString,
  ChnVirt,
  ChnComT,
  ...

const
  ParamStr : tParamStr =
    'NAM=COMT COM=1 IRQ=4 BD=1200 PAR=N BIT=8 STOP=2 LRB=1000'+
    'RTM=30';

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

var
  Chn        : pChnVirt;
  SMess      : ^tMess;
  RMess      : ^tMess;
  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);
  ...
  { inicializace Chn }
  Chn:=ChnCollection^.ChNewInit(ChnCom.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,SizeOf(tMess));
    if ChReceiveResult<>res_Ok then WriteLn('Chyba');
    ChConnect;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_Connect;
    ...
  end;
end;

```

```
{ naplnění zprávy daty }
...
{ vyslání zprávy }
if ChSendReady=CHS_SendReady then
begin
  ChSend(SMess, LSMess);
  repeat
    if ChSendResult<>res_Ok then WriteLn('Chyba');
  until ChSendReady=CHS_SendReady;
  if ChSendResult<>res_Ok then WriteLn('Chyba');
  ...
end;
...
{ čekání na příjem zprávy }
repeat
until ChReceiveReady=CHS_ReceiveReady;
{ příjem zprávy }
while ChReceiveReady=CHS_ReceiveReady do
begin
  if ChReceiveResult<>res_Ok then WriteLn('Chyba')
  else
  begin
    RecChar:=ChReceiveChar;
    if ChGetBinParam(CH_NumName,cmd_GetRecToutChar)<>0 then
      Write(' [Addr Byte]')
    else
      Write(RecChar);
  end;
end;
...
{ 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);
...
end.
```