ChnV40T

JEDNOTKA SÉRIOVÉ KOMUNIKACE RS232 / RS485 S OBVODEM I8251 PROCESORU V40 S MĚŘENÍM ČASŮ PŘÍCHOZÍCH ZNAKŮ

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: 06.08.2004
Datum posledního uložení dokumentu: 06.08.2004
(Datum vydání a posledního uložení dokumentu musí být stejné)

Upozornění:
V dokumentu použité názvy výrobků, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Wi		První vydání.
1.10	1.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Doplněný typ tRecCom a proměnná aRecV40T.
1.20	1.XX	Wil	06.08.2004	Přidána automatická kompenzace zrychlování systémového časovače metod ChOpen a ChClose a úprava jejich popisu. Přidáno automatické zjišťování parametrů TFR a THI.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky sériové komunikace RS232 / RS485 s obvodem i8251 na procesoru V40 s měřením časů příchozích znaků. Ke své činnosti nevyužívá služeb MCP BIOSu, ale přistupuje přímo k hardware.

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 jednotné rozhraní všech komunikačních objektů a dále knihovna používá některé typy z knihovny „ChTypes“.

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 ChnV40T definuje objekt **tChnV40T**, jehož instance vytváří fyzickou vrstvu v komunikačním kanálu tvořenou sériovým rozhraním RS232 nebo RS485 s obvodem i8251 obsaženým v procesoru V40. Ke své činnosti využívá přerušovacího systému počítače bez využití služeb BIOSu MCP. Knihovna ChnV40T dále umožňuje zjišťovat status přijetí dalšího znaku po navolené časové pauze. K tomu využívá již dříve naprogramovaného (prostřednictvím BIOSu MCP) obvodu i8253. 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 ChnV40T správně dekódovala pauzy mezi příchozími znaky, je nutno v přerušení INT 08h (systémový časovač) volat proceduru IncRecTime_1 (viz. 6. Procedury). Pokud je systémový časovač zrychlen (například pomocí knihovny **Tick**), musí se tato procedura volat v tomto zrychleném časovači (při použití knihovny Tick v proceduře **UserTick1**).

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 **tAddChnV40T**, který je dědicem od rodičovského objektu tAddChnVirt. Objekt tAddChnV40T zajistí, aby daný komunikační objekt (objekt tChnV40T) byl k aplikaci připojen a popřípadě zajistí vytvoření instance tohoto objektu. Po přilinkování této jednotky do aplikace (příkazem "uses ChnV40T"), se jméno objektu tChnV40T automaticky vloží do seznamu správců komunikačních objektů pro případné použití.

Protože je objekt **tChnV40T** 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. 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     = 'V40_';
```

Konstanta **cName** definuje jméno komunikačního objektu **tChnV40T**.

```
tRecCom = record
    rComAdrIIR : Word;   { adresa pro Interrupt Enable Register }
    rBufRec    : pRBuff; { přijímací buffer }
    rURec      : Word;   { ukazovátko do přijímacího bufferu na první
                           volné místo }
```

```
    rVRec      : Word;   { ukazovátko do přijímacího bufferu na ještě
                           nevyzvednutÿ znak }
    rMaxURec   : Word;   { maximální velikost přijímacího bufferu }
    rBufTrans  : pTBuff; { vysílací buffer }
    rUTrans    : Word;   { ukazovátko do vysílacího bufferu na první
                           ještě neodvysílaný znak }
    rMaxUTrans : Word;   { velikost vysílané zprávy }
    rRecOn     : Word;   { příznak zda se má po odeslání zprávy
                           povolit příjem znaku }
    rRecDel    : Word;   { příznak zda se má po odeslání zprávy první
                           přijatý znak vypustit }
    rRecTFlIrq0: Word;   { příznak že se má měřit pomoci IRQ0 nebo
                           pomocí obvodu i8253 }
    rRecTi8253 : Word;   { maximální hodnota v impulsech
                           (dekrementačních cyklu) pro měření pomocí
                           obvodu i8253 }
    rRecTirq0  : Word;   { maximální hodnota v počtu Irq0 pro měření
                           pomoci Biosu Int 1Ah }
    rRecHiT    : Word;   { horní mez pro dekrementaci časovače }
    rRecOldT   : Word;   { stará hodnota přečtená z i8253 }
    rRecTimer  : Word;   { čítač přerušení Irq0 }
  end;
```

var aRecV40T : tRecCom;

aRecV40T je globální proměnná pro globální proměnné přerušovací rutiny.

4.1. Konstanty kódů pro metodu ChGetBinParam

`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.

4.2. 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.

5. Objekty

5.1. tChnV40T

5.1.1. Položky

`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 u přenášeného znaku.

`CH_Length : tLength;`
Položka **CH_Length** obsahuje počet datových bitů u 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_SMess : Pointer;`
Položka **CH_SMess** definuje ukazatel na vysílanou zprávu.

`CH_LSMess : Word;`
Položka **CH_LSMess** definuje délku vysílané zprávy.

`CH_RecOn : Boolean;`
Položka **CH_RecOn** určuje, zda se má během vysílání ponechat povolený příjem znaků či zda ho zakázat.

`CH_RedDel : 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, 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_TCU_Frekv: Real;`

Položka **CH_TCU_Frekv** určuje hodnotu vstupní frekvence do systémového časovače v kHz. Při změně komunikační rychlosti na 9600 Bd a výše se vstupní frekvence do systémového časovače automaticky v násobcích čísla 2 zvyšuje. Proto uživatel musí dodatečně změnit patřičně i hodnotu této proměnné podle použité komunikační rychlosti příslušným parametrem metody ChSetParam.

`CH_TCU_Hi   : Word;`

Položka **CH_TCU_Hi** určuje horní mez pro dekrementaci systémového časovače.

`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 po volání metody ChReceiveChar.

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 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_Rate     := 9600;
CH_Parity   := ParOdd;
CH_Stop     := Stop1;
CH_Length   := Bits8;
CH_RSDelay1 := 0;
CH_RSDelay2 := 0;
CH_SMess    := nil;
CH_LSMess   := 0;
CH_RecOn    := true;
CH_RecDel   := false;
CH_STick    := false;
CH_TCU_Frekv:= (ITimerClk*2)/1000;
    { „*2" je kvůli zrychlení při implicitní rychlosti 9600Bd }
CH_TCU_Hi   := ITCU;
CH_RecTOut  := $FFFF;
CH_RecTOutChar:=false;
CH_OldTimer := 3;
```

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. Done destruktor

```
destructor  Done;
```

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

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 tChnV40T dekóduje tyto parametry:

BD=aaa

Parametr **BD** ("BaudRate") určuje přenosovou rychlost požadované sériové komunikace. aaa může nabývat hodnot 25, 50, 75, 100, 110, 150, 300, 600, 1200, 2400, 4800, 9600 nebo 19200 Bd. Při rychlostech 9600 a 19200 Bd je automaticky nastaven jiný poměr v předděliči vstupních hodin, což ovlivní i všechny k němu připojená zařízení. Při rychlosti 9600 Bd bude systémový časovač zrychlen dvakrát, při rychlosti 19200 Bd čtyřikrát.

BIT=bbb

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

PAR=ccc

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

STO=ddd

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

LRB=eee

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 zpracováváme.

RS1=fff

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

RS2=ggg

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

REC=hhh

Parametr **REC** ("Receive While Sending") určuje, zda má být pří vysílání povolen příjem znaků. hhh 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.

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. Při změně komunikační rychlosti na 9600 Bd a výše se vstupní frekvence do systémového časovače automaticky v násobcích čísla 2 zvyšuje. Ve starší verzi této knihovny musel uživatel tímto parametrem dodatečně změnit patřičně i hodnotu proměnné CH_TCU_Frekv podle použité komunikační rychlosti. Současná verze této knihovny provádí tyto změny automaticky.

THI=kkk

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

Při zrychlení systémového časovače pomocí jednotky Tick se horní mez pro dekrementaci systémového časovače zmenšuje. Ve starší verzi této knihovny uživatel musel tímto parametrem patřičně změnit i hodnotu proměnné CH_TCU_Hi. Současná verze této knihovny provádí tyto změny automaticky.

Příklad:

Příklad ukazuje, jak je možné v jednotce V40T nastavit parametry komunikace na sudou paritu, přenosovou rychlost 19200 Bd s úpravou vstupní frekvence do systémového časovače, velikost vstupního vyrovnávacího bufferu na 1000 položek a maximální časovou pauzu pro dekódování dlouhých pauz mezi přijatými znaky na 3ms. Systémový časovač není zrychlen, ale je 4* zvětšena jeho vstupní frekvence (proto parametr TFR má hodnotu 4000).

```
ChSetParam('NAM=V40T BD=19200 TFR=4000 PAR=E LRB=1000 RTM=30');
```

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

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. 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_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).

5.1.2.7. ChOpen procedura

```
procedure ChOpen;
```

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

Při komunikační rychlosti 9600Bd a vyšší může dojít ke zrychlení systémového časovače. Proto je zavolána automatická korekce, která se pokusí toto zrychlení kompenzovat systémovými prostředky knihovny **Tick**. Více podrobností o této korekci se dozvíte v dokumentu knihovny Tick.

5.1.2.8. 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**.

Při komunikační rychlosti 9600Bd a vyšší může dojít naopak ke zpomalení systémového časovače. Proto je opět zavolána automatická korekce, která se pokusí toto zpomalení kompenzovat systémovými prstředky knihovny **Tick**. Více podrobností o této korekci se dozvíte v dokumentu knihovny Tick.

5.1.2.9. 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 odvysílat.

5.1.2.10. 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**.

5.1.2.11. 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**.

5.1.2.12. 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).

5.1.2.13. 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**.

5.1.2.14. ChSendFlush procedura

```
procedure ChSendFlush;
```

Pokud je kanál ve stavu **CHS_Connect** způsobí metoda **ChSendFlush** ukončení vysílání a přechod automatu vysílače do stavu **CHS_SendReady**.

5.1.2.15. 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**.

5.1.2.16. 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. 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.

5.1.2.17. 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** přijmutí zprávy a její uložení do přijímacího bufferu. V proměné `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**.

5.1.2.18. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

Metoda **ChReceiveFlush** způsobí vyprázdnění přijímacích bufferů a nastaví stav příjímače kanálu **CH_RCtlr** na stabilní stav **CHS_ReceiveNoReady**.

5.2. tAddChnV40T

Typ **tAddChnV40T** je typem objektu, který slouží k definování prvku v seznamu správců komunikačních objektů (tzv. správce komunikačního objektu tChnV40T v seznamu správců). Objekt tAddChnV40T 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 tChnV40T a ukazatel na instanci tohoto objektu vrací jako svoji funkční hodnotu.

6. Procedury

```
procedure IncRecTime_1;
```

Procedura IncRecTime_1 provede inkrementaci čítače pauz mezi příchozími znaky. **Tuto proceduru je nutno volat v přerušení INT 08h (systémový časovač) nebo (pokud se používá knihovna Tick) v uživatelské proceduře UserTick1 (ve zrychleném časovači)!**

7. Příklad

Příklad ukazuje použití komunikační jednotky ChnV40T. Je vytvořen komunikační kanál definovaných vlastností, po kterém je zasílána zpráva a z kterého je poté očekáván příjem zpráv.

```
uses
  uString,
  ChnVirt,
```

```
  ChnV40T,
  Tick
  ...

const
  ParamStr : tParamStr =
             'NAM=V40T BD=2400 TFR=1000 RTM=30 BIT=8 PAR=E STOP=2 ' +
             'LRB=1000';

type
  tMess    = array [0..32750] of Byte;

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

begin
  ...
  { inicializace systémového časovače }
  UserTick1:=IncRecTime_1;
  InitTickI;
  ...
  New(SMess);
  New(RMess);
  ...
  { inicializace Chn }
  Chn:=ChnCollection^.ChNewInit(ChnV40.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;
    ...
    { naplnění zprávy daty }
    ...
    { 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 }
    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.
```