

# ChnV40\_

JEDNOTKA SÉRIOVÉ KOMUNIKACE  
RS232 / RS485 S OBVODEM I8251 NA  
V40

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: 16.05.2003

Datum posledního uložení dokumentu: 16.05.2003

(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 a typy přijímacích a vysílacích bufferů	7
5.Objekty	7
5.1. tChnV40_	7
5.1.1. Položky	7
5.1.2. Metody	8
5.1.2.1. Init konstruktor	8
5.1.2.2. ChInitParam konstruktor	8
5.1.2.3. Done destruktor	8
5.1.2.4. ChSetOneParam funkce	8
5.1.2.5. ChGetParam funkce	10
5.1.2.6. ChOpen procedura	10
5.1.2.7. ChClose procedura	10
5.1.2.8. ChConnect procedura	10
5.1.2.9. ChDisConnect procedura	10
5.1.2.10. ChSendTick procedura	10
5.1.2.11. ChSend procedura	10
5.1.2.12. ChSendReady funkce	11
5.1.2.13. ChSendFlush procedura	11
5.1.2.14. ChReceiveReady funkce	11
5.1.2.15. ChReceiveChar funkce	11
5.1.2.16. ChReceive procedura	11
5.1.2.17. ChReceiveFlush procedura	11
5.2. tAddChnV40_	12
5.2.1. Metody	12
5.2.1.1. ChInit funkce	12
6.Příklad	12



## 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	16.05.2003	Úprava dokumentu dle ISO9000 Doplněný typ tRecCom a proměnná aRecCom

### 1.2. Účel dokumentu

---

Tento dokument slouží jako popis jednotky sériové komunikace RS232 / RS485 s obvodem i8251 na V40.

### 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 ChnV40\_ definuje objekt **tChnV40\_**, 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. Na rozdíl od knihovny ChnV40 nevyužívá služeb obsažených v BIOSu MCP, což má za výhody vytvoření dědičných knihoven nebo pomocí jednoduchých úprav (aniž by byly nutné zásahy do BIOSu MCP) vytvoření dalších knihoven s obvodem i8251 na V40.

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 **tAddChnV40\_**, který je dědicem od rodičovského objektu **tAddChnVirt**. Objekt **tAddChnV40\_** zajistí, aby daný komunikační objekt (objekt **tChnV40\_**) byl k aplikaci připojen a popřípadě zajistí vytvoření instance tohoto objektu. Po přilinkování této jednotky do aplikace (příkazem "uses ChnV40\_"), se jméno objektu **tChnV40\_** automaticky vloží do seznamu správců komunikačních objektů pro případné použití.

Protože je objekt **tChnV40\_** 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 **tChnV40\_**.

```
tRecCom = record
  rComAdrIIR : Word;    { adresa pro Interrupt Enable Register }
  rBufRec    : pRBuf;  { přijímací buffer }
  rURec      : Word;    { ukazovátka do přijímacího bufferu na první
                        volné místo }
  rVRec      : Word;    { ukazovátka do přijímacího bufferu na ještě
                        nevyzvednutý znak }
  rMaxURec   : Word;    { maximální velikost přijímacího bufferu }
  rBufTrans  : pTBuf;  { vysílací buffer }
  rUTrans    : Word;    { ukazovátka 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 }
end;
```

**aRecCode** je globální proměnná pro obslužné procedury pro COM

```
var aRecCom      : tRecCom;
```

## 4.1. 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.
```

## 5. Objekty

---

### 5.1. tChnV40\_

---

#### 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\_RedDel** 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.

## 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_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 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).

### 5.1.2.4. 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 tChnV40\_ dekoduje 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 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 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 při 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.

**Příklad:**

Příklad ukazuje, jak je možné v jednotce V40\_ nastavit parametry komunikace na sudou paritu, přenosovou rychlost 9600 Bd a velikost vstupního vyrovnávacího bufferu na 1000 položek.

```
ChSetParam('NAM=V40_ BD=9600 PAR=E LRB=1000');
```

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** 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**.

#### 5.1.2.6. ChOpen procedura

```
procedure ChOpen;
```

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

#### 5.1.2.7. 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**.

#### 5.1.2.8. 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.

#### 5.1.2.9. 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**.

#### 5.1.2.10. 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**.

#### 5.1.2.11. ChSend procedura

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

Pokud je kanál ve stavu **CHS\_Connect**, způsobí metoda **ChSend** započetí vysílání zprávy délky Len uložené na adrese určené ukazatelem Buffer. 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** (čekačí smyčka do odovysílání zprávy).

#### 5.1.2.12. 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**.

#### 5.1.2.13. ChSendFlush procedura

```
procedure ChSendFlush;
```

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

#### 5.1.2.14. 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**.

#### 5.1.2.15. 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 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.

#### 5.1.2.16. 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 proměné `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**.

#### 5.1.2.17. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

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

---

## 5.2. tAddChnV40\_

---

Typ **tAddChnV40\_** je typem objektu, který slouží k definování prvku v seznamu správců komunikačních objektů (tzv. správce komunikačního objektu **tChnV40\_** v seznamu správců). Objekt **tAddChnV40\_** 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 **tChnV40\_** a ukazatel na instanci tohoto objektu vrací jako svoji funkční hodnotu.

---

## 6. Příklad

---

Příklad ukazuje použití komunikační jednotky **ChnV40\_**. Je vytvořen komunikační kanál definovaných vlastností, po kterém je zasílána zpráva a z kterého je poté očekáván příjem zpráv.

```
uses
  uString,
  ChnVirt,
  ChnV40_,
  ...

const
  ParamStr : tParamStr =
    'NAM=V40_ BD=1200 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;

begin
  ...
  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 }
while not ChReceiveReady=CHS_ReceiveReady do
begin
  if ChReceiveResult<>res_Ok then WriteLn('Chyba');
end;
{ příjem zprávy }
ChReceive(LRMess);
if ChReceiveResult<>res_Ok then WriteLn('Chyba');
...
{ 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.
```