ChnV40

JEDNOTKA SÉRIOVÉ KOMUNIKACE RS232 / RS485 S OBVODEM I8251 NA V40 PROSTREDNICTVÍM BIOSU MCP

Príručka uživatele a programátora

SofCon **spol. s r.o.**
Strešovická 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 peclive zkontrolovány a SofCon verí, že jsou spolehlivé, presto SofCon nenese odpovednost za prípadné nepresnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel prebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvoreny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvoril.
SofCon si vyhrazuje právo zmeny obsahu tohoto dokumentu bez predchozího oznámení a nenese žádnou odpovednost za dusledky, 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)

Upozornení:
V dokumentu použité názvy výrobku, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami príslušných vlastníku.

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis zmen
1.00	1.XX	Wi		První vydání
1.10	4.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000 Pridaný popis objektu tAddChnV40
1.20	4.XX	Wil	06.08.2004	Pridána automatická kompenzace zrychlování systémového casovace metod ChConnect a ChDisconnect a úprava jejich popisu.

1.2. Úcel dokumentu

Tento dokument slouží jako popis jednotky sériové komunikace RS232 / RS485 s obvodem i8251 v procesoru V40. Ke své cinnosti využívá služeb MCP BIOSu.

1.3. Rozsah platnosti

Urcen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro ctení tohoto dokumentu je potreba seznámit se s manuálem „ChnVirt“ popisujícím jednotné rozhraní všech komunikacních objektu, dále knihovna používá nekteré typy z knihovny „ChTypes“ a z knihovny „uComM“ používá procedury na volání obsluhy komunikacního kanálu V40.

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 vytvárí fyzickou vrstvu v komunikacním kanálu tvorenou sériovým rozhraním RS232 nebo RS485 s obvodem i8251 obsaženým v procesoru V40. Ke své cinnosti využívá prerušovacího systému pocítace prostrednictvím služeb obsažených v BIOSu MCP. Znaky pricházející po komunikacní lince jsou v prerušovací procedure ukládány do vstupního kruhového vyrovnávacího bufferu, z kterého jsou predávány metodami **ChReceive** a **ChReceiveChar** k dalšímu zpracování. Znaky urcené k odeslání jsou zasílány z výstupního bufferu rovnež s využitím prerušovacího systému.

Knihovna rovnež definuje objekt **tAddChnV40**, který je dedicem od rodicovského objektu tAddChnVirt. Objekt tAddChnV40 zajistí, aby daný komunikacní objekt (objekt tChnV40) byl k aplikaci pripojen a poprípade zajistí vytvorení instance tohoto objektu. Po prilinkování této jednotky do aplikace (príkazem "uses ChnV40"), se jméno objektu tChnV40 automaticky vloží do seznamu správcu komunikacních objektu pro prípadné použití.

Protože je objekt **tChnV40** dedicem rodicovského komunikacního objektu **tChnVirt**, jsou v této prírucce popsány jen odlišnosti a speciality pro tento druh sériové komunikace. Ostatní naleznete v prírucce **ChnVirt**.

Nekteré použité konstanty a typy jsou preddefinované v jednotce **ChnTypes**.

4. Popis konstant a typu

```
cVerNo = napr. $0251; { BCD formát }
cVer   = napr. '02.51,07.08.2003';
```

Císlo verze jednotky v BCD tvaru a v textové podobe vcetne datumu zmeny.

```
cName     = 'V40';
```

Konstanta **cName** definuje jméno komunikacního objektu **tChnV40**.

5. Objekty

5.1. tChnV40

5.1.1. Položky

```
Ch_Rate     : tRate;
```

Položka **Ch_Rate** obsahuje požadovanou prenosovou rychlost v bitech za sekundu.

```
Ch_Parity   : tParity;
```

Položka **Ch_Parity** obsahuje požadovanou paritu prenášeného znaku.

```
Ch_Stop     : tStop;
```

Položka **Ch_Stop** obsahuje pocet stop bitu u prenášeného znaku.

```
Ch_Length   : tLength;
```

Položka **Ch_Length** obsahuje pocet datových bitu u prenášeného znaku.

```
CH_RSDelay1 : Longint;
```

Položka **CH_RSDelay1** definuje minimální casovou prodlevu v ms pred vysíláním.

```
CH_RSDelay2 : Longint;
```

Položka **CH_RSDelay2** definuje minimální casovou 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** urcuje, zda se má behem vysílání ponechat povolený príjem znaku ci zda ho zakázat.

```
CH_STime    : tTimer;
```

Položka **CH_STime** je urcena pro vnitrní použití, pro odmerování casových intervalu vysílace.

```
CH_STick    : Boolean;
```

Položka **CH_STick** je urcena pro vnitrní použití, pro urcení, zda je vykonávána cinnost vysílacího automatu.

5.1.2. Metody

5.1.2.1. Init konstruktor

```
constructor Init;
```

Konstruktor **Init** slouží k vytvorení a inicializaci instance komunikacního objektu. Ve svém tele nejdríve zavolá zdedený konstruktor **Init** (inherited Init) z rodicovského objektu tChnVirt a poté inicializuje položky objektu. Telo konstruktoru vypadá následovne:

```
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_STick    := false;
```

5.1.2.2. ChInitParam konstruktor

```
constructor ChInitParam(const S: tParamStr);
```

Konstruktor **ChInitParam** slouží ke zkrácenému vytvorení instance komunikacního objektu s definovaným nastavením parametru kanálu. Ve svém tele nejprve volá konstruktor **Init** a poté metodu **ChSetParam**.

5.1.2.3. Done destruktor

```
destructor  Done;
```

Destruktor **Done** slouží ke zrušení instance komunikacního objektu. Pokud je v pameti alokován prijímací buffer, bude odstranen a poté je zavolán zdedený 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 prostrednictvím metody **ChSetParam**. Metoda **ChSetOneParam** komunikacního objektu tChnV40 dekóduje tyto parametry:

BD=aaa

Parametr **BD** ("BaudRate") urcuje prenosovou rychlost požadované sériové komunikace. aaa muže nabývat hodnot 25, 50, 75, 100, 110, 150, 300, 600, 1200, 2400, 4800, 9600 nebo 19200 Bd. Pri rychlostech 9600 a 19200 Bd je automaticky nastaven jiný pomer v preddelici vstupních hodin, což ovlivní i všechny k nemu pripojená zarízení. Pri rychlosti 9600 Bd bude systémový casovac zrychlen dvakrát, pri rychlosti 19200 Bd ctyrikrát.

BIT=bbb

Parametr **BIT** ("Number of Data Bits") urcuje pocet datových bitu v prenášeném znaku. bbb muže nabývat hodnot 5 až 8.

PAR=ccc

Parametr **PAR** ("Parity") urcuje paritu prenášeného znaku. ccc muže 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") urcuje pocet stop-bitu v prenášeném znaku. ddd muže nabývat hodnot 1 nebo 2.

LRB=eee

Parametr **LRB** ("Length of Receive Buffer") urcuje velikost vstupního kruhového vyrovnávacího bufferu. Buffer je alokován na heapu a každá jeho položka zaujímá v pameti prostor o velikosti 2 byte (prijatý znak a status). Platí, cím vetší je vstupní buffer, tím více znaku 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 prijaté znaky zpracováváme.

RS1=fff

Parametr **RS1** urcuje zpoždení v ms pred vysíláním.

RS2=ggg

Parametr **RS2** urcuje zpoždení v ms po vysílání.

REC=hhh

Parametr **REC** ("Receive While Sending") urcuje, zda má být prí vysílání povolen príjem znaku. hhh muže nabývat hodnot ON nebo OFF.

Príklad:

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

```
ChSetParam('NAM=V40 BD=9600 PAR=E LRB=1000');
```

Pozn: Všimnete 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 parametru komunikacního objektu. Nejprve vrátí nastavení parametru rodicovského komunikacního objektu tChnVirt a poté k nim pripojí seznam svých parametru. Seznam parametru je uveden výše u popisu metody **ChSetOneParam**.

5.1.2.6. ChConnect procedura

```
procedure ChConnect;
```

Pred voláním této metody musí být kanál ve stavu **CHS_Open** – což se zajistí úspešným voláním metody **ChOpen**. Metoda **ChConnect** provede inicializaci pro príjem a vysílání znaku a pokud nastavení probehlo v porádku, zpusobí prechod do stavu **CHS_Connect**. To znamená, že je možno po daném kanále komunikovat. V tomto stavu je možno prijímat data z komunikacní linky, naopak je možno požadovaná data odvysílat.

Pri komunikacní rychlosti 9600Bd a vyšší muže dojít ke zrychlení systémového casovace. Proto je zavolána automatická korekce, která se pokusí toto zrychlení kompenzovat systémovými prostredky knihovny **Tick**. Více podrobností o této korekci se dozvíte v dokumentu knihovny Tick.

5.1.2.7. ChDisConnect procedura

```
procedure ChDisConnect;
```

Metoda **ChDisConnect** ukoncí vysílací i prijímací cinnost komunikacního kanálu (je prerušen príjem a vysílání zpráv) a uvede kanál do stavu **CHS_DisConnect**. Po volání této metody lze opetovne volat metodu **ChConnect**.

Pri komunikacní rychlosti 9600Bd a vyšší muže dojít naopak ke zpomalení systémového casovace. Proto je opet zavolána automatická korekce, která se pokusí toto zpomalení kompenzovat systémovými prstredky knihovny **Tick**. Více podrobností o této korekci se dozvíte v dokumentu knihovny Tick.

5.1.2.8. ChSendTick procedura

```
procedure ChSendTick;
```

Metoda **ChSendTick** zpusobí provedení kroku vysílacích automatu. Je nutné ji periodicky volat behem vysílání. **ChSendTick** je rovnež automaticky volána vmetodách **ChSendReady** a **ChSend**.

5.1.2.9. ChSend procedura

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

Pokud je kanál ve stavu **CHS_Connect**, zpusobí metoda **ChSend** zapocetí vysílání zprávy délky `Len` uložené na adrese urcené ukazatelem `Buff`. Pokud je parametr Len = 0, nebude se vysílat žádná zpráva. Po volání této metody by melo následovat volání metody **ChSendReady** s testem na **CHS_SendReady** (cekací smycka do odvysílání zprávy).

5.1.2.10. ChSendReady funkce

```
function  ChSendReady: TChState;
```

Metoda **ChSendReady** zpusobí provedení kroku vysílacího automatu na základe volání metody **ChSendTick**. Jako svoji funkcní hodnotu vrátí aktuální stav automatu vysílace komunikacního kanálu, který je uložen v položce **CH_SCtrl**. Pokud kanál není ve stavu **CHS_Connect**, vrací metoda stav **CHS_SendNoReady**.

5.1.2.11. ChReceiveReady funkce

```
function  ChReceiveReady: TChState;
```

Pokud je kanál ve stavu **CHS_Connect** a v prijímacím bufferu jsou prijata nejaká data, vrátí metoda **ChReceiveReady** stav **CHS_ReceiveReady**. V opacném prípade vrátí stav **CHS_ReceiveNoReady**.

5.1.2.12. ChReceiveChar funkce

```
function  ChReceiveChar: Byte;
```

Pokud je kanál ve stavu **CHS_Connect** a v prijímacím bufferu jsou prijata nejaká data, navrací metoda **ChReceiveChar** jeden prijatý znak z prijímacího bufferu a nastaví výsledek operace prijímace na status tohoto prijatého znaku. Metodu **ChReceiveChar** je možno volat pouze ve stavu automatu prijímace **CHS_ReceiveReady**, proto pred voláním této metody musí predcházet volání metody **ChReceiveReady** s testem na stav **CHS_ReceiveReady**, jinak v prípade neprijetí žádného znaku skoncí volání metody **ChReceiveChar** chybou.

5.1.2.13. ChReceive procedura

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

Pokud je kanál ve stavu **CHS_Connect** a je nadefinován buffer pro uložení prijaté zprávy (metodou **ChReceiveBuffer**), provede metoda **ChReceive** prijmutí zprávy a její uložení do prijímacího bufferu. V promenné `Len` navrací délku prijaté zprávy. Ve svém tele volá metodu **ChReceiveChar** (pro prijetí jednoho znaku zprávy) tak dlouho, dokud je prijímac ve stavu **CHS_ReceiveReady**.

5.1.2.14. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

Metoda **ChReceiveFlush** zpusobí vyprázdnení prijímacích bufferu a nastaví stav prijímace kanálu **CH_RCtlr** na stabilní stav **CHS_ReceiveNoReady**.

5.2. tAddChnV40

Typ **tAddChnV40** je typem objektu, který slouží k definování prvku v seznamu správcu komunikacních objektu (tzv. správce komunikacního objektu tChnV40 v seznamu správcu).

5.2.1. Metody

5.2.1.1. ChInit funkce

```
function ChInit: pChnVirt;
```

Metoda **ChInit** slouží k vytvorení instance komunikacního objektu tChnV40 a ukazatel na instanci tohoto objektu vrací jako svoji funkcní hodnotu.

6. Príklad

Príklad ukazuje použití komunikacní jednotky ChnV40. Je vytvoren komunikacní kanál definovaných vlastností, po kterém je zasílána zpráva a z kterého je poté ocekáván príjem 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 parametru 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 prjatá 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;
    ...
    { naplnení zprávy daty }
    ...
    { vyslání zprávy }
    if ChSendReady=CHS_SendReady then
    begin
      ChSend(SMess, LSMess);
      { cekání na odvysílání zprávy }
      repeat
        if ChSendResult<>res_Ok then WriteLn('Chyba');
      until ChSendReady=CHS_SendReady;
      if ChSendResult<>res_Ok then WriteLn('Chyba');
      ...
    end;
    ...
    { cekání na príjem zprávy }
    while not ChReceiveReady=CHS_ReceiveReady do
    begin
      if ChReceiveResult<>res_Ok then WriteLn('Chyba');
    end;
    { príjem zprávy }
    ChReceive(LRMess);
    if ChReceiveResult<>res_Ok then WriteLn('Chyba');
    ...
    { ukoncení }
    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.
```