ChnMBox

JEDNOTKA PRO OBSLUHU SÉRIOVÉ KOMUNIKACE S VYUŽITÍM SCHRÁNEK SYSTÉMU REÁLNÉHO ČASU RETOS

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 1.10

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: 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

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 pro obsluhu sériové komunikace s využitím schránek systému reálného času RETOS.

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.

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 ChnMBox definuje objekt **tChnMBox**, jehož instance vytváří fyzickou vrstvu v komunikačním kanálu tvořenou simulací přenosu dat pomocí schránek operačního systému reálného času Retos. Tohoto způsobu přenosu se využívá převážně při ladění aplikací se simulátory komunikujících procesů. Oproti fyzickým sériovým komunikacím, pracujícím na pozadí pod přerušením, lze tuto jednotku v ladicím prostředí TurboPascalu neomezeně přerušovat. Objekt **tChnMBox** definuje dvě schránky systému reálného času Retos, z nichž jedna je vstupní a jedna je výstupní. Přes tyto dvě schránky se sériově po bytech předávají data. Schránky musí být asynchronní, s proměnnou délkou. Nejlepší jsou schránky dynamické, protože u nich nehrozí problém přetečení a tím přepsání dat. Schránky se musí nainicializovat v aplikaci a objektu tChnMBox se předává pouze ukazatel na již nainicializovanou schránku.

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

Protože je objekt **tChnMBox** 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  = 'MBOX';
```

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

```
pMailBox  = ^MailBox;
```

pMailBox je typ ukazatele na obecnou schránku operačního systému reálného času Retos.

5. Objekty

5.1. tChnMBox

5.1.1. Položky

```
Ch_SendMailBox : pMailBox;
```

Položka **Ch_SendMailBox** definuje ukazatel na schránku, ze které budou zprávy vysílány.

```
Ch_RecMailBox  : pMailBox;
```

Položka **Ch_RecMailBox** definuje ukazatel na schránku, na kterou budou příchozí zprávy ukládány.

```
Ch_RecCt       : Word;
```

Položka **Ch_RecCt** definuje počet přijatých znaků ve vstupní schránce.

```
Ch_RecId       : Word;
```

Položka **Ch_RecId** definuje index aktuální pozice ve vstupní schránce.

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_RecMailBox  := nil;
  CH_SendMailBox := nil;
  CH_RecCt       := 0;
  CH_RecId       := 0;
```

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. Ve svém těle volá zděděný destruktor **Done** z rodičovského 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 **ChsSetOneParam** komunikačního objektu tChnMBox dekóduje tyto parametry:

RMB=‘$’+Segment+’:$’+Offset

Parametrem **RMB** ("Receive Mail Buffer") se předá objektu tChnMBox adresa schránky, do které se budou přijímané zprávy ukládat. Uvedená syntaxe hodnoty parametru je nejsnadněji splněna při použití funkce **PtrStr** z knihovny uString (např: `'RMB=' + PtrStr(RecMailBox)`, kde RecMailBox je ukazatel na inicializovanou instanci přijímací schránky);

SMB=‘$’+Segment+’:$’Offset

Parametrem **SMB** ("Send Mail Buffer") se předá objektu tChnMBox adresa schránky, ze které budou zprávy vysílány. Uvedená syntaxe hodnoty parametru je nejsnadněji splněna při použití funkce **PtrStr** z knihovny uString (např: `'SMB=' + PtrStr(SendMailBox)`, kde SendMailBox je ukazatel na inicializovanou instanci vysílací schránky).

LRB=Size

Parametrem **LRB** ("Length of Receive Buffer") se definuje velikost přijímacího bufferu (přijímací schránky) v bytech.

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. 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.

5.1.2.7. ChSendReady funkce

```
function  ChSendReady: TChState;
```

Pokud je kanál ve stavu **CHS_Connect** a vysílací schránka je prázdná, vrátí metoda **ChSendReady** stav **CHS_SendReady**. V opačném případě vrátí stav **CHS_SendNoReady**.

5.1.2.8. ChReceiveReady funkce

```
function  ChReceiveReady: TChState;
```

Pokud je kanál ve stavu **CHS_Connect** a přijímací schránka není prázdná, vrátí metoda **ChReceiveReady** stav **CHS_ReceiveReady**. V opačném případě vrátí stav **CHS_ReceiveNoReady**.

5.1.2.9. ChReceiveChar funkce

```
function  ChReceiveChar: Byte;
```

Pokud je kanál ve stavu **CHS_Connect** a v přijímacím bufferu (schránce) jsou nějaká přijatá data, navrací metoda **ChReceiveChar** jeden přijatý znak zprávy. 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.

5.1.2.10. 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 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**.

5.1.2.11. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

Metoda **ChReceiveFlush** způsobí vyprázdnění přijímacího bufferu na základě cyklického volání metody **ChReceiveChar** až do dosažení stavu přijímače **CHS_ReceiveNoReady**.

5.2. tAddChnMBox

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

6. Příklad

Příklad ukazuje použití komunikační jednotky ChnMBox. Jsou vytvořeny dva komunikační kanály Chn1 a Chn2 pracující nad dvěma schránkama MailBox1 a MailBox2. Zpráva je zasílána z Chn2 do Chn1.

```
uses
  uString,
  ChnVirt,
  ChnMBox,
 ...

const
  LMess    = 40;

  ParamStr1 : tParamStr = 'NAM=MBOX';
  ParamStr2 : tParamStr = 'NAM=MBOX';

type
  tMess    = array [0..LMess+10] of Byte;

var
  Chn1    ,
  Chn2    : pChnVirt;
  SMess1  ,
```

```
  SMess2  : ^tMess;
  RMess1  ,
  RMess2  : ^tMess;
  LSMess1 ,
  LSMess2 : Word;
  LRMess1 ,
  LRMess2 : Word;
  MailBox1,
  MailBox2: pMailBox;

begin
  ...
  New(SMess1);
  New(RMess1);
  New(SMess2);
  New(RMess2);
  ...
  InitVarMailBox(MailBox1, 'Send1', normal);
  InitVarMailBox(MailBox2, 'Send2', normal);
  Paramstr1:=ParamStr1+
             ' RMB='+PtrStr(MailBox1)+
             ' SMB='+PtrStr(MailBox1);
  Paramstr2:=ParamStr2+
             ' RMB='+PtrStr(MailBox2)+
             ' SMB='+PtrStr(MailBox2);

  { inicializace Chn1 }
  Chn1:=ChnCollection^.ChNewInit(ChnMBox.cName);
  with Chn1^ do
  begin
    { nastavení parametrů komunikace }
    ChSetParam(ParamStr1);
    if ChResult<>res_Ok then WriteLn('Chyba');
    ChOpen;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_Open;
    ChReceiveBuffer(RMess1,SizeOf(tMess));
    if ChReceiveResult<>res_Ok then WriteLn('Chyba');
    ChConnect;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_Connect;
  end;

  { inicializace Chn2 }
  Chn2:=ChnCollection^.ChNewInit(ChnMBox.cName);
  with Chn2^ do
  begin
    { nastavení parametrů komunikace }
    ChSetParam(ParamStr2);
    if ChResult<>res_Ok then WriteLn('Chyba');
    ChOpen;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_Open;
    ChReceiveBuffer(RMess2,SizeOf(tMess));
    if ChReceiveResult<>res_Ok then WriteLn('Chyba');
    ChConnect;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_Connect;
  end;
  ...
  { vyslání zprávy z Chn2 do Chn1}
  with Chn2^ do
  begin
```

```
    ...
    { naplnění zprávy daty }
    ...
    if ChSendReady=CHS_SendReady then
    begin
      ChSend(SMess2, LSMess2);
      { čekání na odvysílání zprávy }
      repeat
        if ChSendResult<>res_Ok then WriteLn('Chyba');
      until ChSendReady=CHS_SendReady;
      if ChSendResult<>res_Ok then WriteLn('Chyba');
    end;
  end;
  ...
  { čekání Chn1 na příjem zprávy }
  with Chn1^ do
  begin
    while not ChReceiveReady=CHS_ReceiveReady do
    begin
      if ChReceiveResult<>res_Ok then WriteLn('Chyba');
    end;
    { příjem zprávy }
    ChReceive(LRMess1);
    if ChReceiveResult<>res_Ok then WriteLn('Chyba');
  end;
  ...
  { ukončení Chn1 }
  with Chn1^ do
  begin
    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 Chn1 }
  Dispose(Chn1,Done);

  with Chn2^ do
  begin
    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 Chn2 }
  Dispose(Chn2,Done);
  ...
end.
```