

ChnT03

JEDNOTKA SÉRIOVÉ KOMUNIKACE
RS232 / RS485 S OBVODEM I8251 NA
V40 V TERMINÁLU TERM03

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>

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: 05.05.2005

Datum posledního uložení dokumentu: 05.05.2005

(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.	Popis konstant a typů	6
4.1.	Konstanty statusu přijatého znaku	7
4.2.	Konstanty a typy přijímacích a vysílacích bufferů	7
4.3.	Konstanty kódů příkazů pro metodu ChGetBinParam	8
5.	Objekty	8
5.1.	tChnT03	8
5.1.1.	Položky	8
5.1.2.	Metody	9
5.1.2.1.	Init konstruktor	9
5.1.2.2.	ChInitParam konstruktor	9
5.1.2.3.	Done destruktork	10
5.1.2.4.	ChSetOneParam funkce	10
5.1.2.5.	ChGetParam funkce	11
5.1.2.6.	ChGetBinParam funkce	11
5.1.2.7.	ChOpen procedura	11
5.1.2.8.	ChClose procedura	12
5.1.2.9.	ChConnect procedura	12
5.1.2.10.	ChDisconnect procedura	12
5.1.2.11.	ChSendTick procedura	12
5.1.2.12.	ChSend procedura	12
5.1.2.13.	ChSendReady funkce	12
5.1.2.14.	ChSendFlush procedura	13
5.1.2.15.	ChReceiveReady funkce	13
5.1.2.16.	ChReceiveChar funkce	13
5.1.2.17.	ChReceive procedura	13
5.1.2.18.	ChReceiveFlush procedura	13
5.2.	tAddChnT03	13
5.2.1.	Metody	14
5.2.1.1.	ChInit funkce	14
6.	Příklad	14

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.
1.20	4.XX	Wil	05.05.2005	Zrušena nadbytečná obsluha modemových signálů. Do přerušovací rutiny přidány čítače počtu přerušení. Přidány chybové kódy res_ErrRecBuffOvf, res_ErrOverrun, res_ErrParity, res_ErrFraming, res_BreakInt, res_ErrSendTimeout.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky sériové komunikace RS232 / RS485 s obvodem i8251 na V40 v terminálu Term03.

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, uDispT03 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 ChnT03 definuje objekt **tChnT03**, jehož instance vytváří fyzickou vrstvu v komunikačním kanálu tvořenou sériovým rozhraním RS232 nebo RS485 s obvodem i8251 obsaženým v procesoru V40 v grafickém terminálu TERM03. Ke své činnosti využívá přerušovacího systému počítače. Tato knihovna je obdobou knihoven ChnV40 a ChnV40_ pro obsluhu sériové komunikace s obvodem i8251 obsaženým v procesoru V40 na procesorové desce KitV40. Rozdíl je v obsluze modemových signálů na RS232 a v obsluze RS485 (na terminálu TERM03 se linkový budič vysílače Tx aktivuje programově, kdežto na KitV40 monostabilním obvodem).

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 **tAddChnT03**, který je dědicem od rodičovského objektu **tAddChnVirt**. Objekt **tAddChnT03** zajistí, aby daný komunikační objekt (objekt **tChnT03**) byl k aplikaci připojen a popřípadě zajistí vytvoření instance tohoto objektu. Po přilinkování této jednotky do aplikace (příkazem "uses ChnT03"), se jméno objektu **tChnT03** automaticky vloží do seznamu správců komunikačních objektů pro případné použití.

Protože je objekt **tChnT03** 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   = 'T03';
```

Konstanta **cName** definuje jméno komunikačního objektu **tChnT03**.

```
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 }
  r485Wait   : Word;   { čekací pauza pro shození EN vysilace pri
                       pouziti RS485 }
  rCntIrq    : Word;   { citac poctu vseh preruseni }
```

```

    rCntRecB   : Word;   { citac poctu preruseni vyvolanych prijmem
                        znaku nebo statusem }
    rCntSendB  : Word;   { citac poctu preruseni vyvolanych
                        odvysilanim znaku }
end;

```

Tato struktura je používána pro globální proměnné v přerušovací rutině.

```
var aRecCom    : tRecCom;
```

aRecCode je globální proměnná pro přerušovací rutinu.

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.MSCUOvrErr;
```

Maska pro dekódování chyby Overrun.

```
res_ErrParity     = ChnTypes.MSCUParErr;
```

Maska pro dekódování chyby Parity

```
res_ErrFraming    = ChnTypes.MSCUFrmErr;
```

Maska pro dekódování chyby Framing.

```
res_BreakInt     = ChnTypes.MSCUBrkDet;
```

Maska pro dekódování Break Interrupt.

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

```
res_ErrSendTimeout = $0030;
```

Timeout při vysílání (nechodí přerušení).

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.

4.3. Konstanty kódů příkazů pro metodu ChGetBinParam

cmd_GetCntIrq = \$0110;
Konstanta definuje kód příkazu pro vrácení počtu všech 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 odvysláním znaku.

5. Objekty

5.1. tChnT03

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 vysláním.

CH_RSDelay2 : Longint;
Položka **CH_RSDelay2** definuje minimální časovou prodlevu v ms po vysláním.

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_RS485` : Boolean;
 Položka **CH_RS485** určuje, že terminál bude komunikovat po rozhraní RS485 2drát nebo 4drát a vysílač Tx se bude obsluhovat programově.

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

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

`CH_CntSendB` : word;
 Položka **CH_CntSendB** je určena pro vnitřní použití pro uložení počtu přerušení od vysílače.

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_RecOn     := true;
CH_RecDel    := false;
CH_RS485     := false;
CH_STick     := false;
CH_OldTimer  := 3;
CH_FreqKit   := GetFreqID*8;
```

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;
```

Destruktork **Done** slouží ke zrušení instance komunikačního objektu. Pokud je v paměti alokován přijímací buffer, bude odstraněn a poté je zavolan 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 tChnT03 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 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 vysláním.

RS2=ggg

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

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.

485=iii

Parametr **485** určuje, zda má být programově obsluhován linkový budič vysílače na rozhraní RS485. iii může nabývat hodnot ON nebo OFF. Hodnota OFF se zpravidla používá pro rozhraní RS232.

Příklad:

Příklad ukazuje, jak je možné v jednotce T03 nastavit parametry komunikace na sudou paritu, přenosovou rychlost 9600 Bd a velikost vstupního vyrovnávacího bufferu na 1000 položek.

```
ChSetParam('NAM=T03 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. ChGetBinParam funkce

```
function ChGetBinParam(NumName: tChNumName; Code: Word): longint;
```

Metodou **ChGetBinParam** s parametrem NumName příslušným číselnému jménu tohoto komunikačního objektu, lze podle parametru Code vrátit tyto nastavení:

```
Code = cmd_GetCntIrq;
```

Vrátí počet všech přerušení.

```
Code = cmd_GetCntRecB;
```

Vrátí počet přerušení způsobených příjmem znaku nebo statusem.

```
Code = cmd_GetCntSendB;
```

Vrátí počet přerušení způsobených odvysíláním znaku.

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**.

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**.

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 odvíšlat.

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** započítí 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** (čekačí smyčka do odvíšlení 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 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.

5.1.2.17. 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** přijetí 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 přijímače kanálu **CH_RCtrl** na stabilní stav **CHS_ReceiveNoReady**.

5.2. tAddChnT03

Typ **tAddChnT03** je typem objektu, který slouží k definování prvku v seznamu správců komunikačních objektů (tzv. správce komunikačního objektu **tChnT03** v seznamu správců). Objekt **tAddChnT03** 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 tChnT03 a ukazatel na instanci tohoto objektu vrací jako svoji funkční hodnotu.

6. Příklad

Příklad ukazuje použití komunikační jednotky ChnT03. Je vytvořen komunikační kanál definovaných vlastností, po kterém je zasílána zpráva a z kterého je poté očekáván příjem zpráv.

```
uses
  uString,
  ChnVirt,
  ChnT03,
  ...

const
  ParamStr : tParamStr =
    'NAM=T03 BD=1200 BIT=8 PAR=E STOP=2 LRB=1000 485=ON';

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(ChnT03.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 }
    ...
  end
end
```

```
{ 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.
```