

Chn188

JEDNOTKA SÉRIOVÉ KOMUNIKACE RS232 SYSTÉMOVÝ PORT KIT188

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: 04.03.2005

Datum posledního uložení dokumentu: 04.03.2005

(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.	Konstanty a jednoduché typy	6
4.1.	Konstanty statusu přijatého znaku	6
4.2.	Konstanty a typy přijímacích a vysílacích bufferů	7
5.	Proměnné	7
6.	Objekty	7
6.1.	tChn188	7
6.1.1.	Položky	7
6.1.2.	Metody	8
6.1.2.1.	Init konstruktor	8
6.1.2.2.	ChInitParam konstruktor	8
6.1.2.3.	Done destruktor	9
6.1.2.4.	ChSetOneParam funkce	9
6.1.2.5.	ChGetParam funkce	10
6.1.2.6.	ChOpen procedura	10
6.1.2.7.	ChClose procedura	10
6.1.2.8.	ChConnect procedura	10
6.1.2.9.	ChDisConnect procedura	11
6.1.2.10.	ChSendTick procedura	11
6.1.2.11.	ChSend procedura	11
6.1.2.12.	ChSendReady funkce	11
6.1.2.13.	ChSendFlush procedura	11
6.1.2.14.	ChReceiveReady funkce	11
6.1.2.15.	ChReceiveChar funkce	11
6.1.2.16.	ChReceive procedura	12
6.1.2.17.	ChReceiveFlush procedura	12
6.2.	tAddChn188	12
6.2.1.	Metody	12
6.2.1.1.	ChInit funkce	12
7.	Příklad	12

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.0X	Wil	04.03.2005	První vydání.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky sériové komunikace s interním systémovým obvodem RS232 řídicího systému Kit188ER.

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 rodičovský prvek komunikačních objektů 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 Chn188 definuje objekt **tChn188**, jehož instance vytváří fyzickou vrstvu v komunikačním kanálu tvořenou sériovým rozhraním RS232 na konektoru X2 procesorové desky Kit188ER. Ke své činnosti využívá přerušovacího systému počítače. 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 **tAddChn188**, který je dědicem od rodičovského objektu **tAddChnVirt**. Objekt **tAddChn188** zajistí, aby daný komunikační objekt (objekt **tChn188**) byl k aplikaci připojen a popřípadě zajistí vytvoření instance tohoto objektu. Po přilinkování této jednotky do aplikace (příkazem "uses Chn188"), se jméno objektu **tChn188** automaticky vloží do seznamu správců komunikačních objektů pro případné použití.

Protože je objekt **tChn188** 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 = '188';
```

Konstanta **cName** definuje jméno komunikačního objektu **tChn188**.

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   = $0001;
```

Maska pro dekódování chyby Overrun.

```
res_ErrParity    = $0002;
```

Maska pro dekódování chyby Parity

```
res_ErrFraming   = $0004;
```

Maska pro dekódování chyby Framing.

```
res_BreakInt     = $0004;
```

Maska pro dekódování Break Interrupt.

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.

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. Proměnné

AChnCom : pChn188 = nil;
Pole ukazatelů na komunikační objekty tChn188.

6. Objekty

6.1. tChn188

6.1.1. 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 přenášeného znaku.

CH_Length : tLength;
Položka **CH_Length** obsahuje počet datových bitů 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_RecOn` : Boolean;
 Položka **CH_RecOn** určuje, zda se má během vysílání nechat povolený příjem znaků či zda ho zakázat.

`CH_RecDel` : Boolean;
 Položka **CH_RecDel** určuje, zda se má (při nastaveném zakázání příjmu během vysílání) vypustit první přijatý znak po skončení vysílání.

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

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

6.1.2. Metody

6.1.2.1. 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_STick     := false;
```

6.1.2.2. 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**.

6.1.2.3. Done destruktork

destructor Done;

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

6.1.2.4. 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** (zděděná metoda ChSetParam od svého předka). Metoda **ChSetOneParam** komunikačního objektu tChn188 dekoduje tyto parametry:

BD=ddd

Parametr **BD** "BaudRate" určuje přenosovou rychlost požadované sériové komunikace. ddd může nabývat hodnot 25 až 115200 Bd.

BIT=eee

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

PAR=fff

Parametr **PAR** "Parity" určuje paritu přenášeného znaku. fff může nabývat hodnot 0 "Odd" pro lichou paritu, E "Even" pro sudou paritu a N "None" pro znak bez parity.

STO=ggg

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

LRB=hhh

Parametr **LRB** "Length of Receive Buffer" určuje velikost přijímacího 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 odebíráme. Tento parametr je povinný. Pokud by zadán nebyl, vrátil by ChResult po ChConnect chybu res_ErrNoRecBuff. Pokud se přijímací buffer nadefinuje malý, tj. COM přijme v přerušení větší množství znaků, než aplikace zpracuje metodami ChReceive nebo ChReceiveChar, dojde k jeho přetečení. V tomto případě se obsah celého bufferu vymaže a poslední přijatý znak, který způsobil jeho přetečení, je do bufferu uložen se statusem res_ErrRecBuffOvf.

RS1=iii

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

RS2=jjj

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

REC=kkk

Parametr **REC** "Receive While Sending" určuje, zda má být při vysílání povolen příjem znaků. kkk 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 objektu nastavit parametry komunikace na sudou paritu, přenosovou rychlost 4800 Bd a velikost vstupního vyrovnávacího bufferu na 1000 položek.

```
ChSetParam('NAM=188 PAR=E BD=4800 LRB=1000');
```

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

6.1.2.5. 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**.

6.1.2.6. ChOpen procedura

```
procedure ChOpen;
```

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

6.1.2.7. 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**.

6.1.2.8. 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 odvysílat.

6.1.2.9. 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**.

6.1.2.10. 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**.

6.1.2.11. 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 **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).

6.1.2.12. 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**.

6.1.2.13. ChSendFlush procedura

```
procedure ChSendFlush;
```

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

6.1.2.14. 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**.

6.1.2.15. 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.

6.1.2.16. 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 proměnné **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**.

6.1.2.17. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

Metoda **ChReceiveFlush** způsobí vyprázdnění přijímacích bufferů a nastaví stav přijímače kanálu **CH_RCtrl** na stabilní stav **CHS_ReceiveNoReady**.

6.2. tAddChn188

Typ **tAddChn188** je typem objektu, který slouží k definování prvku v seznamu správců komunikačních objektů (tzv. správce komunikačního objektu **tChn188** v seznamu správců). Objekt **tAddChn188** je dědicem od rodičovského objektu **tAddChnVirt**.

6.2.1. Metody

6.2.1.1. ChInit funkce

```
function ChInit: pChnVirt;
```

Metoda **ChInit** slouží k vytvoření instance komunikačního objektu **tChn188** a ukazatel na instanci tohoto objektu vrací jako svoji funkční hodnotu.

7. Příklad

Příklad ukazuje použití komunikační jednotky **Chn188**. Je vytvořen komunikační kanál definovaných vlastností, po kterém je zasílána zpráva a z kterého je poté očekáván příjem zpráv.

```
uses
  uString,
  ChnTypes,
  ChnVirt,
  Chn188,
  ...

const
  ParamStr : tParamStr =
    'NAM=188 BD=9600 PAR=N BIT=8 STOP=2 LRB=1000';
```

```

type
  tMess      = array [0..65500] of Byte;

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

begin
  ...
  New(SMess);
  New(RMess);
  ...
  { inicializace Chn }
  Chn:=ChnCollection^.ChNewInit(Chn188.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);
      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;
end;

```

```
{ zrušení instance objektu }  
Dispose (Chn, Done) ;  
...  
end.
```