ChnCAN1

PROGRAMOVÁ JEDNOTKA PRO OBSLUHU KOMUNIKAČNÍHO KANÁLU S ROZHRANÍM

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: 16.05.2003
Datum posledního uložení dokumentu: 16.05.2003
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Bu		První vydání
1.10	2.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Doplněná deklarace metody ChSetOneParam. Odstraněná funkce AddChnCan1.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky pro obsluhu komunikačního kanálu s rozhraním CAN.

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 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 ***ChnCAN1*** obsahuje objekt **TChnCan1**, který je určen pro obsluhu fyzického komunikačního kanálu s rozhraním **CAN** (**C***ontroller* **A***rea* **N***etwork*) realizovaného řadičem Philips SJA1000 na desce **IOCAN**. Instanci objektu **TChnCan1** lze zařadit do kaskády instancí objektů vyšších protokolů, neboť rozhraní služeb objektu **TChnCan1** je tvořeno virtuálními metodami komunikačního objektu **TChnVirt**, který je objektovým rodičem všech typů komunikačních objektů.

Jednotka dále definuje objekt **TAddChnCan1**, který zajistí, aby daná komunikační knihovna byla k aplikaci připojena a byla v aplikaci k dispozici pro případné využití.

Knihovna ***ChnCAN1*** je implementována v jazyce Pascal a svým rozhraním nabízí následující služby:

- HW inicializaci a konfiguraci řadiče SJA1000
- příjem CAN rámců
- příjem CAN rámců pod přerušením do vyrovnávací paměti
- vysílání CAN rámců
- vysílání CAN rámců pod přerušením z vyrovnávací paměti
- detekci chybových stavů

Knihovna ***ChnCAN1*** podporuje oba typy rámců dle specifikace CAN 2.0A (11-bit ID) a CAN 2.0B (29-bit ID).

Knihovna dovoluje jednoduché nastavení doporučených (CiA DS 102 V2.0) komunikačních rychlostí a časování bitů: 10 kbit/s, 20 kbit/s, 50 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s, 800 kbit/s a 1 Mbit/s. Možnost přímého nastavení konfiguračních registrů řadiče SJA1000 je zachována.

Tento text si neklade za cíl popisovat architekturu obvodu SJA1000, proto je třeba jako doplňkovou literaturu použít Data Sheet firmy Philips Semiconductors SJA1000 Stand-alone CAN controller, který je dostupný na internetu **http://www.semiconductors.philips.com** ve formátu ***.pdf.

Poznámka k používané terminologii:

Termínem komunikační kanál je označována kaskáda instancí komunikačních objektů. Instance objektu v kaskádě je potom označována jako objekt kanálu. V našem případě je komunikační kanál na úrovni CAN rámců tvořen právě jednou instancí komunikačního objektu typu TChnCan1.

4. Konstanty

4.1. Obecné konstanty

```
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.

```
CAN1_cName      = 'CAN1';
```

Konstanta definuje jméno komunikačního objektu **TChnCan1.** Toto jméno slouží pro identifikaci objektu v seznamu tzv. správců komunikačních objektů **ChnCollection**.

4.2. Kódy chybových hlášení

Konstanty definují výsledky operací s komunikačním kanálem. Tyto kódy jsou navraceny spolu s identifikací komunikačního objektu funkcemi **ChResult**, **ChReceiveResult** a **ChSendResult**. Kód výsledku je typu word, dolní byte je naplněn kódem chyby, horní byte je naplněn číselnou identifikací komunikačního objektu odvozenou funkcí **CH_NumName** od položky **CH_Name**.

```
res_ErrCan1_BadIOAddr       = $00A0;
```

Chyba nastavené I/O adresy desky IOCAN, test připojení nedostal očekávanou odezvu od desky IOCAN.

```
res_ErrCan1_SJA1000Fail     = $00A1;
```

Chyba nastavení kanálu na desce IOCAN, zápis do testovacího registru v řadiči SJA1000 se nepodařil.

```
res_ErrCan1_ResetActive     = $00A2;
```

Řadič SJA1000 je stále ve stavu Reset, ač by neměl být.

```
res_ErrCan1_NoChnAvailable  = $00A3;
```

Chyba knihovny ChnCAN1, není k dispozici volný komunikační kanál, proto nelze přidělit číslo kanálu.

```
res_ErrCan1_RxMsgLost       = $00A4;
```

Chyba přetečení přijímače, z přijímače CAN rámců v SJA1000 nebyly rámce včas odebrány, a proto byly přepsány nově příchozími rámci.

```
res_ErrCan1_RxBuffOvr       = $00A5;
```

Chyba přetečení vyrovnávací paměti přijímače, vyrovnávací paměť byla zcela zaplněna přijatými rámci, proto nelze do vyrovnávací paměti další rámce zapsat.

```
res_ErrCan1_TxBuffOvr       = $00A6;
```

Chyba přetečení vyrovnávací paměti vysílače, tato chyba nemůže ve stávající implementaci nastat.

```
res_ErrCan1_TxComplete      = $00A7;
```
Posledně zapsaný rámec do vysílače řadiče SJA1000 nebyl odvysílán OK.

```
res_ErrCan1_CanBusErr       = $00A8;
```
Řadič SJA1000 detekuje dočasnou chybu na sběrnici CAN, k zotavení z této chyby může dojít po obnovení komunikace na sběrnici CAN.

```
res_ErrCan1_CanBusOff       = $00A9;
```
Řadič SJA1000 detekuje fatální chybu na sběrnici CAN, k zotavení z této chyby může dojít pouze po přeprogramování řadiče SJA1000, tj. v našem případě po uzavření a novém otevření komunikačního kanálu.

5. Objekty implementované v jednotce

V jednotce knihovny ***ChnCAN1*** jsou implementovány objekty **TChnCan1** a **TAddChnCan1**.

Komunikační objekt **TChnCan1** je objektovým dědicem abstraktního komunikačního objektu **TChnVirt**.

```
PChnCan1 = ^TChnCan1;
TChnCan1 = object(TChnVirt)
 ...
end;
```

Komunikační objekt **TChnCan1** nově definuje některé virtuální metody objektu **TChnVirt** tak, aby tyto metody obsluhovaly fyzický komunikační kanál tvořený řadičem SJA1000 na desce IOCAN.

Objekt správce komunikačních objektů **TAddChnCan1** je objektovým dědicem typu **TAddChnVirt**. Objekt je určen pro založení do seznamu správců komunikačních objektů **ChnCollection**.

```
PAddChnCan1 = ^TAddChnCan1;
TAddChnCan1 = object(TAddChnVirt)
 ...
end;
```

Prostřednictvím instance objektu typu **TAddChnCan1** je do **ChnCollection** založen objekt správce komunikačního objektu typu **TChnCan1**. Objekt správce, jako instance objektu **TAddChnCan1** v seznamu **ChnCollection,** je schopen vytvořit novou instanci komunikačního objektu typu **TChnCan1**, nebo udržovat odkaz na existující instanci objektu typu **TAddChnCan1**. Objekt správce je v seznamu **ChnCollection** identifikován jménem komunikačního objektu.

5.1. TChnCan1

V popisu budou uvedeny pouze nově zavedené položky v objektu **TChnCan1** oproti objektu **TChnVirt** a předefinované metody.

Deklarace:

```
PChnCan1 = ^TChnCan1;
TChnCan1 = object(TChnVirt)
  CH_CANNum :byte;    { cislo CAN kanalu, prideleno automaticky }
  CH_Addr   :word;    { base adresa desky SCN100 - pro Lattice  }
  CH_SJA    :tSJA;    { cislo radice SJA1000 na desce SCN100 }
  CH_Irq    :tIrq;    { cislo IRQ obvodu 8259A, na ktery bude
                        SJA1000 prerusovat [0,2..7] }
  CH_BRP    :byte;    { Baud Rate Prescaler }
  CH_SJW    :byte;    { Synchronization Jump Width }
  CH_TSEG1  :byte;    { Time Segment 1 }
  CH_TSEG2  :byte;    { Time Segment 2 }
  CH_SAM    :byte;    { Sampling }
  CH_AFM    :byte;    { Acceptance Filter Mode - Single/Dual }
  CH_ACR    :longint;{ Acceptance Code Register }
  CH_AMR    :longint;{ Acceptance Mask Register 1=don't care }
  CH_RxMsgCnt:word;   { velikost alokovaneho RxD bufferu
                        v poctu CANmessage }
  CH_TxMsgCnt:word;   { velikost alokovaneho TxD bufferu
                        v poctu CANmessage }
  CH_CommonIRQ:Boolean;{ spolecna prerusovaci rutina pro
                         CH_CANNum=1 a CH_CANNum=2 }
 ...
end;
```

5.1.1. Položky

`CH_CANNum :byte;` - Číslo CAN kanálu.
Číslo je přidělováno automaticky při otevírání kanálu a slouží jako index do pole s daty otevřených CAN kanálů. Velikost pole je implementačně omezena. Pole obsahuje záznamy s daty, která jsou sdílena a používána společně s obslužnými rutinami HW-přerušení.

`CH_Addr :word;` - Počáteční I/O adresa desky **IOCAN**
Adresu lze zadat prostřednictvím metody **CHSetParam**, hodnota musí odpovídat adrese nastavené propojkami na desce **IOCAN**.

`CH_SJA :tSJA;` - Číslo řadiče SJA1000 na desce **IOCAN**.
Deska **IOCAN** může být osazena jedním nebo dvěma CAN kanály tvořenými řadiči SJA1000.
`CH_SJA=0` ... kanál A (základní, první)
`CH_SJA=1` ... kanál B (volitelný, druhý)
Volbu kanálu lze zadat prostřednictvím metody **CHSetParam**.

`CH_Irq`	`:tIrq;`	- Číslo IRQ obvodu 8259A, kam bude připojen vodič signálu přerušení od řadiče SJA1000. Hodnotu lze nastavit na 0, 2, 3, 4, 5, 6, 7. Pokud je nastavena hodnota 0, nebude přerušení využíváno. V takovém případě je třeba při komunikaci použít tzv. metodu dotazování.
`CH_BRP`	`:byte;`	- Hodnota pro Baud Rate Prescaler v řadiči SJA1000. Hodnota zadaná prostřednictvím metody **ChSetParam** bude při inicializaci řadiče SJA1000 zapsána do registru BusTiming0, který spolu s dalšími registry definuje přenosovou rychlost pro komunikaci po CAN sběrnici.
`CH_SJW`	`:byte;`	- Hodnota pro Synchronization Jump Width v řadiči SJA1000. Hodnota zadaná prostřednictvím metody **ChSetParam** bude při inicializaci řadiče SJA1000 zapsána do registru BusTiming0, který spolu s dalšími registry definuje časování bitů na sběrnici CAN.
`CH_TSEG1`	`:byte;`	- Hodnota pro Time Segment 1 v řadiči SJA1000. Hodnota zadaná prostřednictvím metody **ChSetParam** bude při inicializaci řadiče SJA1000 zapsána do registru BusTiming1, který spolu s dalšími registry definuje časování bitů na sběrnici CAN.
`CH_TSEG2`	`:byte;`	- Hodnota pro Time Segment 2 v řadiči SJA1000. Hodnota zadaná prostřednictvím metody **ChSetParam** bude při inicializaci řadiče SJA1000 zapsána do registru BusTiming1, který spolu s dalšími registry definuje časování bitů na sběrnici CAN.
`CH_SAM`	`:byte;`	- Hodnota pro Sampling v řadiči SJA1000. Hodnota zadaná prostřednictvím metody **ChSetParam** bude při inicializaci řadiče SJA1000 zapsána do registru BusTiming1, který spolu s dalšími registry definuje časování bitů na sběrnici CAN.
`CH_AFM`	`:byte;`	- Hodnota pro Acceptance Filter Mode - Single/Dual . Hodnota zadaná prostřednictvím metody **ChSetParam** bude při inicializaci řadiče SJA1000 zapsána do registru MODE.

`CH_ACR` `:longint;` - Hodnota pro Acceptance Code Register v řadiči SJA1000.
Hodnota zadaná prostřednictvím metody **ChSetParam** bude při inicializaci řadiče SJA1000 zapsána do registru AcceptCode.

`CH_AMR` `:longint;` - Hodnota pro Acceptance Mask Register v řadiči SJA1000.
Hodnota bitu=1 ...význam "don't care"
Hodnota zadaná prostřednictvím metody **ChSetParam** bude při inicializaci řadiče SJA1000 zapsána do registru AcceptMask.

`CH_RxMsgCnt` `:word;` - Velikost alokovaného přijímacího bufferu (RxD bufferu) vyjádřená v počtu CAN rámců. Hodnotu zadáme prostřednictvím metody **ChSetParam**.

`CH_TxMsgCnt` `:word;` - Velikost alokovaného vysílacího bufferu (TxD bufferu) vyjádřená v počtu CAN rámců. Hodnotu zadáme prostřednictvím metody **ChSetParam**. Pokud je položka `CH_TxMsgCnt` nastavena na hodnotu 1, pak nebude vysílání realizováno pod přerušením. Tímto způsobem lze vypnout použití přerušení při vysílání CAN rámců.

`CH_CommonIRQ` `:Boolean;` - Flag použití společného přerušení a společné přerušovací rutiny pro oba kanály desky IOCAN (nutno nastavit propojku spojující signál SJA1000-INT).

5.1.2. Metody objektu TChnCan1

V tomto odstavci budou popsány implementačně významné metody objektu **TChnCan1**.

5.1.2.1. constructor Init

V těle constructoru Init jsou nastaveny položky objektu **TChnCan1** na svoje implicitní hodnoty. Některé položky jsou nastaveny na nedefinované hodnoty, a proto nastavení těchto položek je třeba provést následně pomocí metody **ChSetParam**.

Mezi položky, které jsou constructorem Init nastaveny na definované hodnoty patří následující položky:

`CH_SJA` - hodnota 0, nastaven kanál A

`CH_SJW, CH_BRP, CH_TSEG1, CH_TSEG2` - položky jsou nastaveny tak, aby definovaly přenosovou rychlost 20 kbit(= default bit rate)

`CH_SAM` - Sampling 1=triple..low/medium

	speed busses
`CH_AMR`	- Acceptance Mask Register - (don't care ~ 1)
`CH_RxMsgCnt`	- nastaveno na hodnotu 8
`CH_TxMsgCnt`	- nastaveno na hodnotu 4
`CH_CommonIRQ`	- nastaveno na False

5.1.2.2. constructor ChInitParam

V těle constructoru ChInitParam je zavolán constructor Init a následně metoda ChSetParam.

Deklarace:

```
constructor TChnCan1.ChInitParam(S: tParamStr);
```

`S` `:tParamStr;` - Textový řetězec s nastavením parametrů pro metodu **ChSetParam**. Význam položek v řetězci je vysvětlen v rámci popisu metody **ChSetOneParam** pro jednotlivé objektové typy z objektové hierarchie.

5.1.2.3. function ChSetOneParam

Deklarace:

```
function  TChnCan1.ChSetOneParam(S: tWordString; var CmdL:
            tCmdL):tChResult;
```

Parametry:

`S` `:tWordString;` - Inicializační řetězec.

`CmdL` `:tCmd;` - Dekódovaný řetězec parametrů.

Funkční hodnota:

Metoda **ChSetOneParam** je volána z těla metody **ChSetParam** a slouží pro nastavení hodnot položek objektu **TChnCan1** prostřednictvím tzv. inicializačního řetězce. Inicializační řetězec obsahuje přiřazovací příkazy pro nastavení jednotlivých parametrů ve tvaru:

<identifikátor>=<hodnota>

Identifikátor je tvořen zpravidla tříznakovým řetězcem, oddělovačem mezi příkazy je znak mezera. Zápis hodnoty je poplatný typu parametru.

`NAM` - Uživatelské jméno komunikačního objektu.
V inicializačním řetězci slouží pro identifikaci parametrů pro daný

objekt, musí být shodné se jménem v položce CH_Name.
Příklad: `'NAM=CAN1'`

`ADD` - Počáteční I/O adresa desky IOCAN.
Zadaná hodnota musí odpovídat adrese nastavené propojkami na desce **IOCAN**.
Příklad: `'ADD=$300'`

`SJA` - Číslo řadiče SJA1000 na desce **IOCAN**.
Kanál A ... `SJA=0,`
Kanál B ... `SJA=1,`
Příklad: `'SJA=1'`

`CIS` - Common Interrupt Service.
Flag použití společného přerušení pro oba kanály na desce IOCAN. Tento flag je třeba nastavit souhlasně u obou instancí komunikačních objektů typu TChnCan1, které obsluhují fyzické CAN kanály na jedné desce IOCAN. Pokud je nastavena hodnota `CIS=1`, pak musí být nastaveny souhlasně také čísla přerušení `IRQ`!
Příklad:
`' ADD=$300 SJA=0 CIS=1 IRQ=5'` ... pro instanci kanálu A
`' ADD=$300 SJA=1 CIS=1 IRQ=5'` ... pro instanci kanálu B

`IRQ` - Číslo IRQ pro přerušení od řadiče kanálu.
Číslo IRQ lze nastavit na hodnoty 0, 2, 3, 4, 5 a 7. Pokud je IRQ nastaveno na hodnotu 0, pak není přerušovací systém řadiče kanálu vůbec aktivován.
Příklad: `'IRQ=5'`

`BRP` - Hodnota pro Baud Rate Prescaler v řadiči SJA1000.
Zadaná hodnota musí být v intervalu [$00..$3F].
Příklad: `'BRP=39'`

`SJW` - Hodnota pro Synchronization Jump Width v řadiči SJA1000.
`SJW` může být nastaveno na hodnoty 0, 1 , 2 a 3.
Příklad: `'SJW=2'`

`TS1` - Hodnota pro Time Segment 1 v řadiči SJA1000.
TS1 může být nastaven na hodnotu v intervalu [0..15].
Příklad: `'TS1=11'`

`TS2` - Hodnota pro Time Segment 2 v řadiči SJA1000.
TS2 může být nastaven na hodnotu v intervalu [0..7].
Příklad: `'TS1=2'`

`BTR` - Hodnota Bit rate v [kbit/s].
Prostřednictvím zadání hodnoty `BTR` lze navolit hodnoty pro `SJW, BRP TS1` a `TS2` tak, aby odpovídaly zadané přenosové rychlosti `BTR` a doporučenému vzorkování bitů. Prostřednictvím `BTR` lze nastavit přenosové rychlosti dle doporučení CiA DS 102 V2.0, tj. přenosové rychlosti 10 kbit/s, 20 kbit/s, 50 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s,

800 kbit/s a 1000 kbit/s. Hodnotu `BTR` zadáme v jednotkách [kbit/s].
Příklad: `'BTR=800'`

`SAM` - Hodnota pro Sampling v řadiči SJA1000.
Povolené hodnoty jsou 0 a 1.
Hodnota 0 volí vzorkování single určené pro "high speed busses".
Hodnota 1 volí vzorkování triple určené pro "low/medium speed busses"
Příklad: `'SAM=1'`

`AFM` - Hodnota Acceptance Filter Mode pro SJA1000.
Povolené hodnoty jsou 0 a 1.
Hodnota 0 navolí Dual Filter Mode
Hodnota 1 navolí Single Filter Mode.
Zadání `AFM` ovlivňuje následné zadávání dílčích položek pro Acceptance Code Register a Acceptance Mask Register. Význam bitů v těchto registrech se mění dle navoleného modu filtrace.
Příklad: `'AFM=0'`

`ACR` - Hodnota pro Acceptance Code Register.
Prostřednictvím `ACR` lze zadat hodnotu, která bude přímo zapsána do Acceptance Code Register v řadiči SJA1000. Hodnotu pro Acceptance Code Register lze zadávat také pohodlněji prostřednictvím nastavení dílčích položek.

`AMR` - Hodnota pro Acceptance Mask Register.
Prostřednictvím `AMR` lze zadat hodnotu, která bude přímo zapsána do Acceptance Mask Register v řadiči SJA1000. Hodnotu pro Acceptance Mask Register lze zadávat také pohodlněji prostřednictvím nastavení dílčích položek.

`CS1` - Standard AccCode ID - filter 1 Single or Dual filter configuration.
Prostřednictvím `CS1` lze zadat hodnotu 11-bitového CAN identifikátoru pro Acceptance Filter. Zadaná hodnota se po transformaci stane dílčí částí registru Acceptance Code Register, která odpovídá CAN identifikátoru pro filter 1.

`MS1` - Standard AccMask ID - filter 1 Single or Dual filter configuration.
Prostřednictvím `MS1` lze zadat hodnotu 11-bitové masky
CAN identifikátoru pro Acceptance Filter. Zadaná hodnota se po transformaci stane dílčí částí registru Acceptance Mask Register, která odpovídá masce CAN identifikátoru pro filter 1.

`CS2` - Standard AccCode ID - filter 2 only for Dual filter configuration.
Prostřednictvím `CS2` lze zadat hodnotu 11-bitového CAN identifikátoru pro Acceptance Filter v Dual Filter Mode. Zadaná hodnota se po transformaci stane dílčí částí registru Acceptance Code Register, která odpovídá CAN identifikátoru pro filter 2.

`MS2` - Standard AccMask ID - filter 2 only for Dual filter configuration.
Prostřednictvím `MS2` lze zadat hodnotu 11-bitové masky
CAN identifikátoru pro Acceptance Filter. Zadaná hodnota se po

transformaci stane dílčí částí registru Acceptance Mask Register, která odpovídá masce CAN identifikátoru pro filter 2.

`CDW` - Standard AccCode Data - Single or Dual filter configuration.
Prostřednictvím `CDW` lze zadat vzor dat pro filter standardního (s 11-bitovým identifikátorem) CAN rámce. Je-li nastaven Dual Filter Mode, uplatní se pouze hodnota dolního byte ze zadané hodnoty typu word.

`MDW` - Standard AccMask Data - Single or Dual filter configuration.
Prostřednictvím `MDW` lze zadat masku dat pro filter standardního (s 11-bitovým identifikátorem) CAN rámce. Je-li nastaven Dual Filter Mode, uplatní se pouze hodnota dolního byte ze zadané hodnoty typu word.

`CX1` - Extended AccCode ID - filter 1 Single or Dual filter configuration.
Prostřednictvím `CX1` lze zadat hodnotu 29-bitového CAN identifikátoru pro Acceptance Filter. Zadaná hodnota se po transformaci stane dílčí částí registru Acceptance Code Register, která odpovídá CAN identifikátoru pro filter 1.

`MX1` - Extended AccMask ID - filter 1 Single or Dual filter configuration.
Prostřednictvím `MX1` lze zadat hodnotu 29-bitové masky CAN identifikátoru pro Acceptance Filter. Zadaná hodnota se po transformaci stane dílčí částí registru Acceptance Mask Register, která odpovídá masce CAN identifikátoru pro filter 1.

`CX2` - Extended AccCode ID - filter 2 only for Dual filter configuration.
Prostřednictvím `CX2` lze zadat hodnotu 29-bitového CAN identifikátoru pro Acceptance Filter. Zadaná hodnota se po transformaci stane dílčí částí registru Acceptance Code Register, která odpovídá CAN identifikátoru pro filter 2.

`MX2` - Extended AccMask ID - filter 2 only for Dual filter configuration.
Prostřednictvím `MX2` lze zadat hodnotu 29-bitové masky CAN identifikátoru pro Acceptance Filter. Zadaná hodnota se po transformaci stane dílčí částí registru Acceptance Mask Register, která odpovídá masce CAN identifikátoru pro filter 2.

`CRS` - Standard RTR AccRegBit.
Prostřednictvím `CRS` lze zadat hodnotu vzoru RTR bitu pro Acceptance Filter standardního CAN rámce. Zadaná hodnota se po transformaci stane hodnotou bitu RTR v registru Acceptance Code Register, tento bit RTR patří do filtru 1.

`MRS` - Standard RTR AccMaskBit.
Prostřednictvím `MRS` lze zadat hodnotu masky RTR bitu pro Acceptance Filter standardního CAN rámce. Zadaná hodnota se po transformaci stane hodnotou bitu RTR v registru Acceptance Mask Register, tento bit RTR patří do filtru 1.

`CR2` - Standard RTR AccRegBit - filter 2 Dual filter configuration.
Prostřednictvím `CR2` lze zadat hodnotu vzoru RTR bitu pro Acceptance Filter standardního CAN rámce. Zadaná hodnota se po transformaci stane hodnotou bitu RTR v registru Acceptance Code Register, tento bit RTR patří do filtru 2.

`MR2` - Standard RTR AccMaskBit - filter 2 Dual filter configuration.
Prostřednictvím `MR2` lze zadat hodnotu masky RTR bitu pro Acceptance Filter standardního CAN rámce. Zadaná hodnota se po transformaci stane hodnotou bitu RTR v registru Acceptance Mask Register, tento bit RTR patří do filtru 2.

`CRX` - Extended RTR AccRegBit.
Prostřednictvím `CRX` lze zadat hodnotu vzoru RTR bitu pro Acceptance Filter extended CAN rámce (29-bitový ID). Zadaná hodnota se po transformaci stane hodnotou bitu RTR v registru Acceptance Code Register.

`MRX` - Extended RTR AccMaskBit.
Prostřednictvím `MRX` lze zadat hodnotu masky RTR bitu pro Acceptance Filter extended CAN rámce (29-bitový ID). Zadaná hodnota se po transformaci stane hodnotou bitu RTR v registru Acceptance Mask Register.

`LRB` - Velikost alokovaného RxD bufferu v počtu CAN rámců.
Zadaná hodnota bude uložena do položky `CH_RxMsgCnt`.
Dle zadané hodnoty LRB je alokován buffer pro přijímané CAN rámce.

`LTB` - Velikost alokovaného TxD bufferu v počtu CAN rámců.
Zadaná hodnota bude uložena do položky `CH_TxMsgCnt`.
Dle zadané hodnoty `LTB` je alokován buffer pro vysílané CAN rámce.
Je-li zadána hodnota `LTB=1`, pak není při vysílání používáno přerušení od vysílače CAN rámců. Je-li nastavena hodnota `LTB>1`, pak jsou CAN rámce vysílány pod přerušením z takto alokovaného vysílacího bufferu.

5.1.2.4. function ChGetParam

Metoda **ChGetParam** je určena ke zpětnému získání tzv. inicializačního řetězce z komunikačního objektu.

Deklarace:

```
function  TChnCan1.ChGetParam(S: tParamStr):tParamStr;
```

Parametry:

`S :tParamStr;` - Ve stávající implementaci bez významu.

Funkční hodnota:

Metoda sestaví řetězec své funkční hodnoty z dílčích řetězců přiřazovacích příkazů pro jednotlivé položky komunikačního objektu. Hodnoty na pravé straně příkazu odpovídají aktuálně platným hodnotám položek objektu. Do výsledného

řetězce jsou začleněny přiřazovací příkazy pouze pro přímé nastavení položek objektu.

5.2. TAddChnCan1

Deklarace:

```
PAddChnCan1 = ^TAddChnCan1;
TAddChnCan1 = object(TAddChnVirt)
 ...
end;
```

Objekt správce komunikačního objektu je určen k založení do seznamu dostupných objektů kanálu **ChnCollection**. Objekt správce umí vygenerovat novou instanci komunikačního objektu typu **TChnCan1** nebo udržovat odkaz na existující instanci objektu kanálu.

5.2.1. Položky

Objekt využívá pouze zděděné položky.

5.2.2. Metody

5.2.2.1. function ChInit

Deklarace:

```
function TAddChnCan1.ChInit:PChnVirt;
```

Metoda ve svém těle alokuje novou instanci komunikačního objektu typu **TChnCan1**. Ukazatel na vytvořenou instanci bude předán jako funkční hodnota. Metoda **TAddChnCan1.ChInit** je zpravidla volána prostřednictvím metody objektu seznamu dostupných objektů kanálu **ChnCollection**:

```
function TChnCollection.ChNewInit(Name: tChName): PChnVirt;
```

Prostřednictvím volání `ChnCollection^.ChNewInit(CAN1_cName)` vytvoříme v aplikačním programu novou instanci komunikačního objektu typu **TChnCan1**. Metoda **TChnCollection.ChNewInit** vyhledá dle zadaného jména v **ChnCollection** objekt správce komunikačního objektu a zavolá jeho metodu **TAddChnCan1.ChInit.**

6. Použití komunikačního objektu TChnCan1

6.1. Datový typ CAN rámce

Komunikační objekt typu **TChnCan1** používá pro výměnu CAN rámců na svém rozhraní záznam typu **tCanFrame**. Vyrovnávací paměti jsou implementovány jako pole záznamů typu **tCanFrame**.

```
const
 frstat_MsgRdy    = $01;
 frstat_MsgLst    = $02;
 frstat_BuffOvr   = $04;

type
pCanFrame = ^tCanFrame;
tCanFrame = record  { SizeOf(tCanFrame)=16 byte }
  IDENT  :longint;  { CAN identifier 11/29 bit }
  RTR    :byte;     { RTR Remote transmission request }
  DLC    :byte;     { delka dat v polozce DATA }
  XTD    :byte;     { Extended - priznak 29-bit CAN identifier }
  STATUS :byte;     { frstat_xxx, [-,-,-,-,-,BuffOvr,MsgLst,MsgRdy] }
  DATA   :array[0..7] of byte;
end;
```

Význam položek:

`IDENT :longint;` - Hodnota CAN identifikátoru pro rámec.
Pro 11-bitový ID v rozsahu [0..2047].
Pro 29-bitový ID v rozsahu [0..536870911]

`RTR :byte;` - Remote Transmission Request.
Boolean hodnota pro bit RTR.
`RTR` v rozsahu [0..1]

`DLC :byte;` - Délka dat v položce DATA.
Délka datové části CAN rámce, určuje délku platných dat v položce DATA.
`DLC` v rozsahu [0..8].

`XTD :byte;` - Extended flag.
Boolean hodnota užití Extended CAN rámce, tj. rámce s 29-bitovým identifikátorem.
`XTD` v rozsahu [0..1].

`STATUS :byte;` - Položka pro přenos stavových informací.
`b0=MsgRdy` platný rámec
`b1=MsgLst` ztráta rámce
`b2=BuffOvr` přetečení vyrovnávací paměti

`DATA :array[0..7] of byte;` - Datový obsah CAN rámce.
Délka platných dat v poli `DATA` je dána položkou

DLC.

6.2. Vytvoření instance komun. objektu TChnCan1

Novou instanci komunikačního objektu vytvoříme prostřednictvím objektu svého správce v seznamu **ChnCollection**. Předtím, než začneme instanci vytvářet, musí být zavolána procedura **AddChnCan1**. To je zabezpečeno voláním procedury **AddChnCan1** v inicializační sekci programové jednotky ***ChnCAN1***. V aplikačním programu však musíme uvést `"uses ChnCAN1;"`.

Následující torzo programu ukazuje vytvoření a zrušení instance komunikačního kanálu obsahujícího objekt **TChnCan1**:

```
var
ChCan1 :PChnCan1; { instance komunikacniho kanalu CAN  }
 ...
begin
 ...
 { Vytvoreni instance komunikacniho kanalu }
 ChCan1:=PChnCan1(ChnCollection^.ChNewInit(CAN1_cName));
 ...
 { Komunikacni cinnost aplikacniho programu }
 ...
 { Zruseni instance komunikacniho kanalu }
 if Assigned(ChCan1) then
 begin
   ChCan1^.CHClose;
   Dispose(ChCan1,Done);ChCan1:=nil;
 end;
 ...
end;
```

6.3. Nastavení parametrů komun. objektu TChnCan1

K nastavení parametrů komunikačního objektu je určena metoda **ChSetParam**. Předtím, než tuto metodu zavoláme, si musíme připravit tzv. inicializační řetězec. Ten bude obsahovat přiřazovací příkazy pro nastavení hodnot parametrů. Dále je třeba komunikačnímu objektu předat adresu a délku proměnné, do které bude CAN rámec přijímán metodou **ChReceive**. K nastavení adresy této proměnné je určena metoda **ChReceiveBuffer**. Proměnná pro příjem CAN rámce je typu **tCanFrame**.

```
const
  SetParStrCAN1 = 'ADD=$300 SJA=0 IRQ=5 LRB=60 LTB=30 BTR=125';
  ...
var
  RxFRAME1 :tCanFrame; { promenna pro prijem CAN ramce }
  ...
begin
 ...
 { Nastaveni parametru kanalu pomoci inicializacniho retezce }
 ChCan1^.CHSetParam(SetParStrCAN1);
 if ChCan1^.CHResult<>res_Ok then AppError;
 ...
 { predani adresy promenne pro prijem CAN ramce metodou CHReceive }
 ChCan1^.ChReceiveBuffer(Addr(RxFRAME1),SizeOf(RxFRAME1));
 if ChCan1^.CHResult<>res_Ok then AppError;
 ...
end;
```

Po nastavení parametrů je komunikační kanál připraven k otevření.

6.4. Otevření komun. kanálu s objektem TChnCan1

Poté co jsou nastaveny parametry objektu **TChnCan1**, je možno kanál otevřít. Při otevírání kanálu dojde k inicializaci a konfiguraci řadiče SJA1000 dle nastavených parametrů.

```
begin
 ...
 { Otevreni komunikacniho kanalu }
 ChCan1^.CHOpen;
 repeat
   if ChCan1^.CHResult<>res_Ok then AppError;
   Wait(1);
 until ChCan1^.ChReady=CHS_Open; { ceka se na dosazeni CHS_Open }
 if ChCan1^.CHResult<>res_Ok then AppError;
 ...
end;
```

Po úspěšném otevření kanálu je třeba zavolat metodu pro připojení **CHConnect**.

6.5. Připojení komun. kanálu s objektem TChnCan1

Na úrovni CAN rámce je operace připojení prázdná, pouze je změněn vnitřní stav automatu komunikačního kanálu.

```
begin
 ...
 { Pripojeni komunikacniho kanalu }
 ChCan1^.CHConnect;
 repeat
   if ChCan1^.CHResult<>res_Ok then AppError;
   Wait(1);
 until ChCan1^.ChReady=CHS_Connect; { ceka se na CHS_Connect }
 if ChCan1^.CHResult<>res_Ok then AppError;
 ...
end;
```

6.6. Příjem a vysílání CAN rámců kanálem s objektem TChnCan1

Následující torzo aplikačního programu ukazuje cyklus, ve kterém jsou vysílány a přijímány CAN rámce pomocí virtuálních metod objektu komunikačního kanálu.

```
repeat
  pomw1:=ChCan1^.CHSendReady;
  repeat { cyklus vysilani CAN ramcu z TxFRAME1 }
    pomw2:=ChCan1^.CHSendResult;
    if pomw2<>res_Ok then
    begin
      TxError1:=pomw2;{kod posledni chyby}
      Inc(TxErrorCnt1);
    end;
    if (pomw1=CHS_SendReady) and (pomw2=res_Ok) then
    begin
      {== Priprava CAN ramce pro vyslani do TxFRAME1 ==}
      ChCan1^.CHSend(Addr(TxFRAME1),SizeOf(TxFRAME1));
      pomw2:=ChCan1^.CHSendResult;
      if pomw2=res_Ok
      then Inc(TxCounter1){ citac OK vyslanych ramcu }
      else begin
             TxError1:=pomw2;{kod posledni chyby}
             Inc(TxErrorCnt1);
           end;
    end;
    pomw1:=ChCan1^.CHSendReady;
    pomw2:=ChCan1^.CHReceiveReady;
  until (pomw1<>CHS_SendReady) or (pomw2=CHS_ReceiveReady)
  pomw1:=ChCan1^.CHReceiveReady;
  pomw2:=ChCan1^.CHReceiveResult;
  if pomw2<>res_Ok then
  begin
    RxError1:=pomw2;{kod posledni chyby}
    Inc(RxErrorCnt1);
  end;
  repeat { cyklus prijmu CAN ramcu do RxFRAME1 }
    if pomw1=CHS_ReceiveReady
    then begin
           ChCan1^.CHReceive(pomw2);{ prijem do RxFRAME1 }
           pomw2:=ChCan1^.CHReceiveResult;
           if pomw2<>res_Ok
           then begin
                  Inc(RxErrorCnt1);
                  RxError1:=pomw2;{kod posledni chyby }
                  ChCan1^.CHReceiveFlush;
                end
           else begin
                  Inc(RxCounter1);{ citac OK prijatych ramcu }
                  {== Zpracovani prijateho ramce z RxFRAME1 ==}
                end;
         end;
    pomw1:=ChCan1^.CHReceiveReady;
    pomw2:=ChCan1^.CHReceiveResult;
    if pomw2<>res_Ok then
    begin
      RxError1:=pomw2;{kod posledni chyby}
      Inc(RxErrorCnt1);
    end;
  until pomw1<>CHS_ReceiveReady;
  Wait(1);{ ReTOS - uspani procesu na zadanou dobu }
until FlEnd;
```

Výše uvedené torzo aplikačního programu neřeší přípravu obsahu rámce určeného k vyslání do proměnné TxFRAME1, stejně tak neřeší zpracování přijatého rámce z proměnné RxFRAME1. Ve skutečném aplikačním programu musíme položky proměnné TxFRAME1 definovat platnými hodnotami.

Pro správnou činnost komunikace je nezbytné, aby aplikační program dostatečně často volal metody **CHReceiveReady** a **CHSendReady**. Na tuto skutečnost je kladen důraz hlavně tehdy, není-li použita komunikace pod přerušením.

6.7. Odpojení komun. kanálu s objektem TChnCan1

Komunikační kanál přeruší vysílání, vynuluje vysílací vyrovnávací paměť a změní vnitřní stav automatu na stav "odpojeno".

```
begin
 ...
 { Odpojeni komunikacniho kanalu }
 ChCan1^.CHDisConnect;
 repeat
   if ChCan1^.CHResult<>res_Ok then AppError;
   Wait(1);
 until ChCan1^.ChReady=CHS_DisConnect;
 if ChCan1^.CHResult<>res_Ok then AppError;
 ...
end;
```

6.8. Zavření komun. kanálu s objektem TChnCan1

Poté co je komunikační kanál odpojen, můžeme kanál uzavřít. Při zavírání kanálu dojde k uvedení řadiče SJA1000 do pasivního stavu, tj. stavu, ve kterém se nezúčastňuje komunikace po CAN sběrnici a ani nevyžaduje obsluhu přerušení od CPU.

```
begin
 ...
 { Zavreni komunikacniho kanalu }
 ChCan1^.CHClose;
 repeat
   if ChCan1^.CHResult<>res_Ok then AppError;
 until ChCan1^.ChReady=CHS_Close;
 if ChCan1^.CHResult<>res_Ok then AppError;
 ...
end;
```

7. Zotavení z fatální chyby na sběrnici CAN

Pokud řadič SJA1000 detekuje stav CAN sběrnice "bus-off", je nezbytné řadič znovu inicializovat. Pomocí služeb objektu kanálu proto vykonáme regulární odpojení a uzavření kanálu a jeho následné nové otevření a připojení. Parametry kanálu zůstanou nezměněny.