COM_400, KNIHOVNA KOMUNIKACÍ PRO AUTOMATY ŘADY 400 V JAZYCE SIMPLE4

edice 12.2019
verze 2.0

Popis podpůrné knihovny komunikací v SIMPLE4.

http:/www.micropel.cz

Historie změn knihovny a dokumentu

04.2017

- První verze knihovny a dokumentu.

06.2018

- U SMS brány pracující dle nastavení z editoru opravena chyba zastavení zpracovávání dalších SMS po příjmu požadavku s neúplným počtem parametrů.
- V dokumentu opraveny formulace a doplněny některé informace.

04.2019

- Do SMS brány pracující dle nastavení z editoru doplněn telefonní seznam upravitelný za běhu aplikace.
- Do SMS brány pracující dle nastavení z editoru vestavěn SMS příkaz GPRS propojení automatu / komunikátoru s GSM modulem na IP adresu v režimu komunikátoru.
- Mezi synchronizační funkce doplněna příkazová funkce *Sync_ConnectPC* pro možnost propojení automatu / komunikátoru s ETHERNET nebo GPRS rozhraním na IP adresu v režimu komunikátoru.

OBSAH

REALIZACE GSM BRÁNY PRACUJÍCÍ PODLE NASTAVENÍ Z SMS EDITORU

1 Úvod

Následující text popisuje realizaci SMS brány pracující podle nastavení SMS editorem. Kód obsluhy brány využívá prostředky dále popsané obecné knihovny pro GSM bránu CA5. Do programu PLC řady 400 lze v projektu StudioWin přidat i editor SMS. Editor umožňuje nastavit seznam telefonních čísel, definovat příchozí a odchozí zprávy (SMS) resp. prozvonění a sestavit odchozí dávky. Také definuje umístění, kde má program najít rozhraní pro komunikaci s GSM modulem CA5. Editor před zahájením překladu programu pro PLC automaticky vygeneruje tabulku s nastavením chování brány. Do kódu programu je potřeba doplnit řádek s voláním obslužné funkce, do níž bude tabulka předávána.

1.1 Nastavení propojení na GSM modul CA5 v editoru

GSM bránu je třeba vždy realizovat za pomoci převodníku CA5 (MCA45) s přívlastkem G, pouze takový převodník disponuje GSM modulem. Program obsluhující GSM modul může běžet přímo uvnitř MCA45, nebo i v libovolném jiném PLC řady 400 s přístupem k CA5 v síti MICROPEL.
Pokud obsluha poběží přímo v MCA45, je potřeba v editoru nastavit číslo řídícího EXBUS uzlu na 12. Jinak je třeba nastavit, zda bude CA5 dostupná v síti PESNET, v tom případě se zadává přímo PESNET adresa CA5. Poslední možností je dostupnost v síti EXBUS, pak je třeba nastavit hodnotu EXBUS adresy CA5 zvětšené o nastavené číslo vnějšího uzlu SMS brány v rámci nastavení ovladače GSM u CA5.

2 Ovládání SMS brány z programu

V jednom PLC, v tom se SMS editorem v projektu, bude probíhat obsluha rozhraní pro ovládání GSM modulu převodníku CA5. Pouze toto PLC smí přistupovat k rozhraní GSM modulu CA5. V hlavní programové smyčce PLC je pak třeba volat rutinu *SmsGtCA5* s parametrem *@sms_mem*.

```
; Kód programu, kde hlavní smyčka volá pouze obslužnou funkci SMS brány.
;=== Začátek hlavní smyčky ===
SmsGtCA5(@sms_mem)
;=== Konec hlavní smyčky ===
RESET = 0
end
```

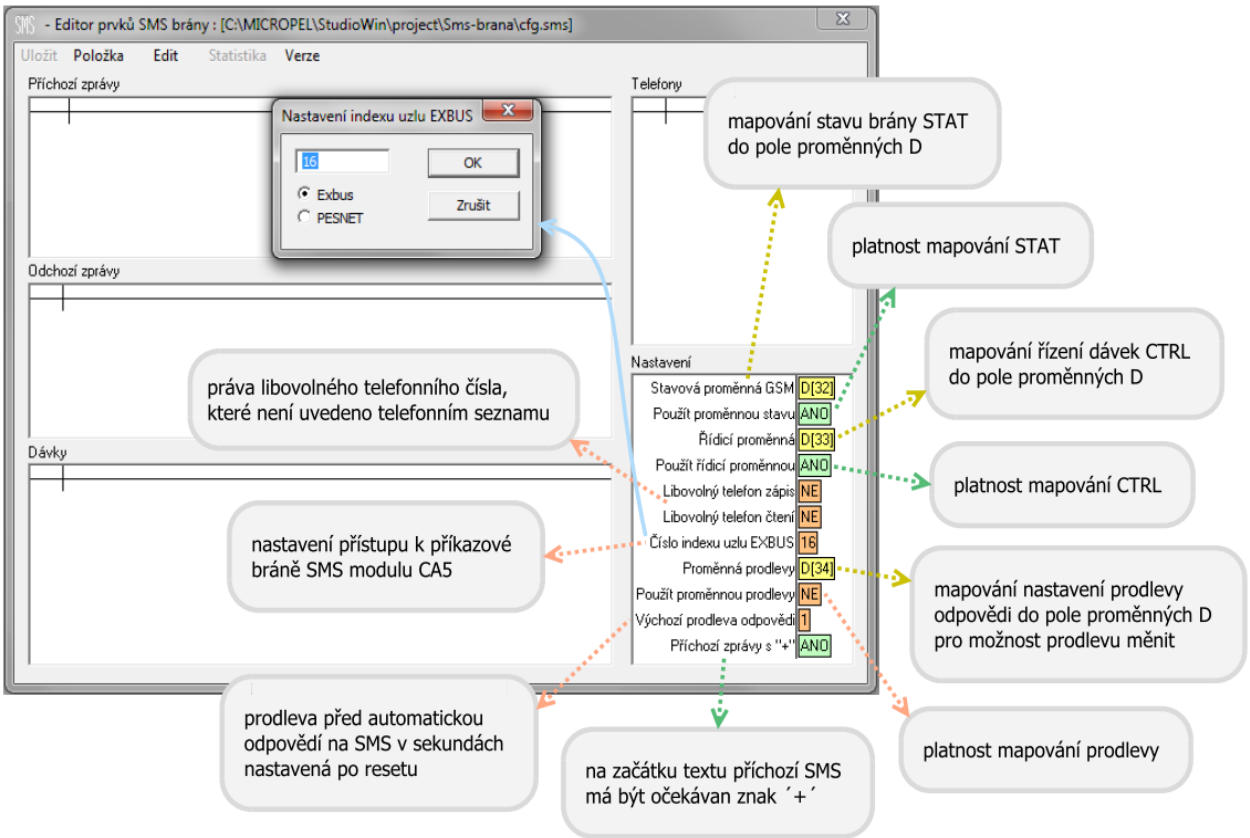
Obslužná funkce zajistí zpracování obsahu příchozích SMS od povolených čísel a odvysílání případných automatických odpovědí. Kromě toho na vyžádání zahájí zpracování přednastavené odchozí dávky.
Funkce neustále obnovuje aktuální stav brány do vyhrazeného wordu v poli *D* (o 64 položkách) dále zmiňovaného jako *STAT*. Dávky lze spouštět nastavením čísla dávky do dalšího vyhrazeného wordu v poli *D* dále zmiňovaného jako *CTRL*. Novou dávku je možno zadat jen za podmínky *(STAT=0* and *CTRL=0)*. Umístění *STAT* a *CTRL* do pole *D* je nastavitelné v editoru SMS, wordy *D[0]* až *D[31]* jsou lokální, wordy *D[32]* až *D[63]* jsou síťové, tedy sdílené mezi všemi automaty v síti PESNET.
Hodnota proměnné *STAT* má následující význam (nezmíněné bity budou vždy nulové):

- bity 0 až 3 … kód aktuálního stavu vyřizování příchozí odchozí akce
  - hodnota 0 → neprobíhá příchozí ani odchozí akce
  - hodnota 1 → probíhá zpracování příchozí SMS / automatické odesílání odpovědi
  - hodnota 2 → probíhá zpracování dávky
  - hodnota 7 → probíhá průchod programovou smyčkou s nastaveným bitem RESET
  - jiná hodnota → nedefinovaný stav
- bit 4 … indikátor stavu *offline*, tj. stavu, kdy modul není připojen do GSM sítě
- bit 7 … indikátor nedostupnosti příkazové brány GSM modulu

- bity 8 až 12 … kód poslední chyby komunikace se SMS bránou
  - hodnota 0 → bez chyby
  - hodnota 1 → nelze odeslat zprávu, modul není připojen v síti
  - hodnota 2 → ve vyrovnávací paměti není místo, nelze zapsat další text
  - hodnota 3 → zpráva je příliš dlouhá
  - hodnota 4 → zadán nepodporovaný příkaz
  - hodnota 5 → není žádná zpráva ke smazání
  - hodnota 6 → zprávu nelze přeskočit
  - hodnota 7 → zprávu nelze odeslat
  - hodnota 8 → nelze se dovolat
  - hodnota 15 → chyba nastavená knihovnou při resetu
  - hodnota 16 → zadaný příkaz nebyl vyřízen do časového limitu
  - hodnota 17 → došlo k chybě při vyřizování příkazu na lince PESnet
- bit 13 … v příchozí zprávě chyběl některý z očekávaných parametrů
- bit 14 … v příchozí zprávě byl nalezen parametr s chybným formátem nebo došlo k chybě v průběhu zpracování odchozích akcí

Bity 8 až 15 proměnné *STAT* jsou nulovány při úspěšném vyčtení aktuálního stavu SMS brány.

3 Obrázek s popisky parametrů brány v SMS editoru



4 Úprava telefonů bez nutnost přeprogramování

Knihovna SMS brány nyní též pracuje se seznamem telefonních čísel, v němž lze jednotlivá čísla za běhu aplikace (dynamicky) měnit. K uložení dynamického seznamu slouží vnitřní proměnná s kapacitou pro prvních **40** čísel seznamu vytvořeného editorem SMS. Po startu programu PLC bude provedena kontrola, zda proměnná obsahuje platný telefonní seznam. Pokud seznam není platný, dojde k inicializaci proměnné seznamem z kódové paměti, jenž byl vytvořen v editoru SMS. Pokud seznam v proměnné platný je, bude

rovnou použit, bez inicializace. K zachování změn v seznamu dojde i v případě, kdy existující program PLC přehrajeme novou verzí programu, kde je proměnná telefonního seznamu umístěna na stejném místě v paměti, a editorem SMS jsme v nahrávané verzi programu neprovedli jinou úpravu tel. seznamu, než je přidání nových čísel.
Obecně lze doporučit fixaci proměnné telefonního seznamu v programu PLC. Např. pro fixaci seznamu na adresu 200 v uživatelské RAM v PLC bude program obsahovat řádky:

```
fix SMSGT_PhoneList = (200, sizeof(SMSGT_PhoneList))
SmsGtCA5(@sms_mem); obsluha SMS brány v hlavní smyčce
```

Pro využití editovatelného telefonního seznamu v PLC bez zálohované uživatelské RAM (např. MCA45, MT424) je třeba v programu definovat novou proměnnou se seznamem umístěnou na zásobníku (stack, velikosti 4kB u zmíněných „malých" PLC). Tuto proměnnou je pak potřeba předávat obslužné funkci SMS brány. Např. pro umístění seznamu od adresy 200 na zásobníku v PLC bude program obsahovat řádky:

```
var _sms_phones_list TelSeznamVar mapon(StackB[200])
SmsGtCA5(@sms_mem, TelSeznamVar); obsluha SMS brány v hlavní smyčce
```

Jestliže máme v plánu možnost úpravy seznamu za běhu aplikace využít, je třeba program PLC obsluhujícího SMS bránu dle nastavení z editoru SMS přeložit a nahrát ze StudioWin verze 8.200 nebo vyšší! Pro účely budoucího doplnění nových kontaktů za běhu aplikace pak editor SMS umožňuje definovat též kontakt s prázdným (nevyplněným) telefonním číslem - odpovědní SMS nebo prozvonění směřující na prázdné tel. číslo v rámci dávkového příkazu budou vynechávány.

4.1 Práva telefonního čísla

Každé telefonní číslo v seznamu má přiřazena práva přístupu k proměnným automatu a k administraci telefonního seznamu. Práva jsou v seznamu uložena v jednom bajtu. Nejnižší (první bit) povoluje čtení proměnných, druhý bit zápis proměnných, třetí bit povoluje administraci tel. seznamu, vyšší bity zatím nemají přiřazen žádný účel. Např. hodnota bajtu práv rovna 7 povoluje číslu úplný přístup, hodnota 1 povoluje pouze čtení proměnných (tj. zápis proměnných ani administraci seznamu tel. číslu neumožňuje).

4.2 Editaci seznamu lze provádět dvěma způsoby:

Knihovna obsahuje rutinu začleňující editor seznamu telefonních čísel do menu vytvořeného knihovnou MENULIB3_400. Rutinu tak lze jednoduše použít při obsluze SMS brány v programu PLC s displejem.
Seznam lze též číst a upravovat pomoci SMS zpráv. Telefonní číslo, z něhož zprávy zasíláme, pak potřebuje mít nastaveno v aktuálně platném tel. seznamu SMS brány právo administrace (Admin). Z tel. čísla s právem administrace lze editovat ostatní čísla v telefonním seznamu včetně jejich práv, nelze pouze editovat položku seznamu příslušející jemu (tel. číslu odesílatele). V seznamu je proto vhodné mít definována alespoň dvě tel. čísla, administrátor pak např. může před změnou svého čísla (dočasně) nastavit práva administrace jinému číslu, z toho čísla poté upravit svůj záznam.

4.3 Vestavěné editační SMS příkazy

Příkazy přístupu k telefonnímu seznamu pomocí SMS jsou: *PLGetItems*, *PLSetPhone*, *PLSetRights*. Stejným textem bude začínat text příkazové SMS (za případným povinným znakem „+" – jde o volitelný parametr brány v editoru SMS), velikost písmen v SMS nehraje roli.
Pokud odesílatel nemá právo administrace, v SMS odpovědi dorazí text „*DENIED*". Při chybě v parametrech příkazu nebo chybě vykonání příkazu v SMS odpovědi dorazí text „*ERROR X*", kde místo *X* bude uvedeno číslo.

Příkaz PLGetItems

Příkaz čtení několika položek z aktuálně platného telefonního seznamu.
Např. požadavek vyčtení 3 položek od 2. pozice v tel. seznamu zadáme textem „+*PLGETITEMS 2 3*".

Parametr 1: Číslo požadované položky v seznamu, z rozsahu 1 až N, kde N představuje celkový počet položek seznamu.
Parametr 2: Počet požadovaných položek seznamu, z rozsahu 1 až 5 (maximální možná délka odchozí SMS je 119 znaků!).
Odpověď: Text v odpovědní SMS bude obsahovat požadovaný počet záznamů ve formátu „*N R TEL;*“, se středníkem na konci, kde představují: N číslo položky, R nastavení práv a TEL telefonní číslo (u neplatného tel. čísla namísto TEL zobrazí text „---“). Např. text obsahujícího druhý záznam může být „*02 3 +420123456789;*“.

Příkaz PLSetPhone

Příkaz výměny telefonního čísla v aktuálně platném telefonním seznamu. Lze uvést i novou hodnotu práv.
Např. požadavek výměny tel. čísla +420123456789 za číslo +420111222333 zadáme textem „+*plsetphone +420123456789 +420111222333*“. Požadavek výměny tel. čísla na 2. pozici v seznamu za číslo +420111222333 a nastavení práv čtení a zápisu zadáme textem „+*PLSETPHONE 2 +420111222333 3*“.
Parametr 1: Buď číslo požadované položky v seznamu, z rozsahu 1 až N, kde N představuje celkový počet položek seznamu. Nebo telefonní číslo (např. +420123456789), položka v seznamu se zadaným tel. číslem bude automaticky vyhledána.
Parametr 2: Telefonní číslo (např. +420987654321), kterým má být původní tel. číslo nahrazeno. Případně znak „-“ (mínus) pro zneplatnění tel. čísla.
Parametr 3: Je nepovinný. Nová hodnota práv telefonního čísla, hodnota z rozsahu 0 až 7 - obecně hodnota menší než 256, nastavení práv čtení, zápisu a administrace je ale uloženo ve spodních 3 bitech hodnoty.
Odpověď: Text v odpovědní SMS bude „*OK*“.

Příkaz PLSetRights

Příkaz změny práv telefonního čísla v aktuálně platném telefonním seznamu.
Např. požadavek nastavení práv pouze číst hodnoty tel. číslu +420123456789 zadáme textem „+*plsetrights +420123456789 1*“. Požadavek nastavení všech práv číslu na 2. pozici v seznamu zadáme textem „+*PLSETRIGHTS 2 7*“.
Parametr 1: Buď číslo požadované položky v seznamu, z rozsahu 1 až N, kde N představuje celkový počet položek seznamu. Nebo telefonní číslo (např. +420123456789), položka v seznamu se zadaným tel. číslem bude automaticky vyhledána.
Parametr 2: Nová hodnota práv telefonního čísla, hodnota z rozsahu 0 až 7 - obecně hodnota menší než 256, nastavení práv čtení, zápisu a administrace je ale uloženo ve spodních 3 bitech hodnoty.
Odpověď: Text v odpovědní SMS bude „*OK*“.

5 Vestavěný SMS příkaz GPRS propojení s komunikačním programem

Pomocí programu SMS brány přeloženého ve verzi prostředí StudioWin 8.200 nebo vyšší je možno vyvolat GPRS spojení automatu / komunikátoru s GSM modulem na libovolnou IPv4 adresu pomocí SMS. Pro správnou funkci potřebuje mít SIM karta v GSM modulu komunikátoru povoleny a správně nastaveny GPRS datové přenosy. Nastavení lze upravovat programem PlcConfig. V případě programu přímo pro automat s GSM modulem, nebo pokud je komunikátor s GSM modulem dostupný v síti PESNET, stačí v nastavení GPRS povolit datové přenosy. Jestliže program neběží přímo v automatu s GSM modulem, ale k odesílání SMS využívá komunikátor dostupný v síti EXBUS jako Slave, je též potřeba komunikátoru programem PlcConfig povolit zpřístupnění SMS i GPRS datových přenosů na síť EXBUS.
Možnost poslouží např. ke vzdálenému propojení se do sítě se SMS bránou, třeba programem DDEAP kvůli ladění, kdy předem neznáme IP adresu komunikátoru přidělenou mobilním operátorem, což je obvyklý případ.

5.1 Příkaz GPRSConnect

Zasláním SMS ve tvaru „+*GPRSConnect 100.150.50.204 10001 3210*“ se automat / komunikátor s GSM modulem pokusí datově propojit s adresou 100.150.50.204 na portu 10001, následně bude očekávat EPNP požadavky s heslem komunikace 3210. Velikost písmen v SMS zprávě nehraje roli, úvodní znak „+“ je třeba vynechat, pokud SMS brána očekává příkazy bez úvodního plus (jde o volitelný parametr brány v SMS editoru).
Jako odpověď dorazí SMS s textem „OK“, nebo „*ERROR X*“ v případě neúspěchu, kde místo *X* bude uvedeno číslo chyby požadavku spojení synchronizační knihovny.
Po úspěšném spojení začne automat či komunikátor fungovat jako běžný převodník EPNP protokolu. Jestliže připojený program nebude udržovat spojení, pomocí zasílání EPNP požadavků, dojde po cca 30 sekundách k automatickému odpojení.

5.2 Postup navázání GPRS propojení s DDEAP

1. V síti, v níž máme PC s programem DDEAP, zajistíme přesměrování připojení z internetu na vybraném „vnějším“ portu na naše PC a libovolném (stejném či jiném) „vnitřním” portu.
2. V DDEAP zapneme čekání na TCP spojení s automatem na vnitřním (přesměrovaném) portu a heslem komunikace.
3. Pošleme SMS pro navázání spojení. Jako IP adresu zadáme veřejnou adresu našeho PC, tu lze snadno zjistit z internetových stránek, pokud ji neznáme. Jako číslo portu uvedeme vnější port zvolený v prvním bodě. Heslo komunikace musí být zadáno stejné jako v parametrech připojení v DDEAP.

Použití příkazu lze zkoušet též na simulátoru v prostředí StudioWin. Pokud na lokálním PC spustíme DDEAP, pak stačí jako cílovou zadat IP adresu 127.0.0.1, a u simulátoru komunikátoru GSM je ještě potřeba pomocí kontextového menu povolit GSM propojení do sítě Ethernet.

KNIHOVNA OVLÁDÁNÍ GSM BRÁNY CA5

1 Úvod

Převodník CA5 (MCA45) s GSM modulem a platnou SIM kartou umožňuje přijmout a odeslat SMS a prozvonit telefonní číslo. Přijaté SMS se čtou z obsahu SIM, takže je třeba SMS po přečtení mazat, aby nedošlo k zaplnění SIM a tím pádem i nemožnosti přijmout další zprávu.
Zmíněné úkony lze provádět z SIMPLE4 programu zadáváním příkazů na komunikační bránu v CA5. Zde popisovaná knihovna nabízí sadu funkcí, které uvnitř řeší veškerou potřebnou komunikaci na bráně.

1.1 SMS brána

Brána pro přístup k SMS funkcím GSM modulu CA5 je tvořena 2 EXBUS uzly, uzlem WR_BUF a uzlem RD_BUF. Díky možnosti nastavení přístupnosti brány po EXBUS z jiného automatu není použití knihovny omezeno jen na program běžící uvnitř MCA45. GSM modul tak lze ovládnout i z programu jiného PLC řady 400, a navíc nejen přes EXBUS, ale díky využití firmwarové podpory u řady 400 i přes linku PESNET - u programovaného automatu je potřeba mít nastaven ovladač PLC-DMA.

2 Popis knihovny

Knihovna obsahuje funkce zabezpečující zadávání požadavků ovladači GSM (SMS) modulu převodníku CA5. Název veškerých funkcí knihovny začíná řetězcem „*Sms_*“. Kvůli vestavěné možnosti ovládání modulu přes PESNET knihovna používá i prostředky knihovny *SYNC*, ta zprostředkuje přenos dat pomocí PLC-DMA.

2.1 Proměnné určené k použití

SMS_GStat

Proměnná typu *s_gsm_status* s informacemi o stavu GSM brány naplněná na základě úspěšného vyřízení příkazové funkce *Sms_GetStatus*. Pro více informací viz popis zmíněné funkce.

2.2 Obecné funkce

Sms_Init

Deklarace :	**subroutine Sms_Init(word node)** **subroutine Sms_Init_Pn(byte pnadr, word node)**	
Parametr :	**node**	číslo vnitřního/EXBUS uzlu komunikační brány GSM-SMS
	plc	PESNET adresa automatu, který má přímý přístup k uzlům brány, typicky rovnou adresa CA5-G
Výstup :	- - -	

Funkce inicializuje potřebné proměnné. V programu zavolat jednou před prvním použitím jakékoli jiné „*Sms_*“ funkce.
První verze funkce se použije v programu pro MCA45-G (programovatelná CA5), nebo v programu pro EXBUS-master, kde bude CA5-G EXBUS-slave.

- Parametrem funkce je číslo komunikačního uzlu WR_BUF GSM (SMS) brány.
  - V programu pro MCA45 (programovatelnou CA5) se použije číslo vnitřního uzlu vyhrazeného pro GSM (SMS) ovladač, tedy **12**.
  - Při obsluze CA5 (EXBUS-slave) z programu pro EXBUS-master se použije jako číslo uzlu EXBUS adresa CA5 zvětšená o hodnotu nastavenou v parametrech GSM (SMS) ovladače u CA5.

```
Sms_Init(12) ; Použít v programu do MCA45-G (programovatelné CA5).
Sms_Init(22) ; Použít v programu do PLC EXBUS-MA, kde má CA5-G EXBUS adresu
        ; 20 a nastavenu viditelnost ovladače SMS na vnějším uzlu 2.
```

Druhá verze funkce se použije v programu pro automat, u něhož bude CA5-G přístupná prostřednictvím linky PESNET.

- Číslo EXBUS uzlu zde má význam čísla uzlu, jak jej vidí vzdálený automat dostupný přes PESNET. Přístup k CA5 bude probíhat s využitím vnitřního ovladače PLC-DMA - ovladač je třeba mít u programovaného automatu povolen.

```
Sms_Init_Pn(4,12) ; Použít v programu, kde má CA5-G PESNET adresu 4.
Sms_Init_Pn(4,22) ; Použije se jen výjimečně, např. v programu pro MT424
        ; (s PESNET), kde má MPC400 (s PESNET + EXBUS-MA) adresu 4 na lince
        ; PESNET a na EXBUS připojen převodník CA5-G s EXBUS adresou 20
        ; a s nastavením viditelnosti ovladače SMS na vnějším uzlu 2.
```

Sms_CmState

Deklarace :	**function byte Sms_CmState()**	
Parametr :	- - -	
Výstup :	**state**	stav vyřizování posledního zadaného příkazu

Funkce vrátí stav naposled zadaného příkazu na SMS bránu. Získaná hodnota má význam:

- 0 … poslední zadaný příkaz zatím probíhá.
- 1 … poslední zadaný příkaz byl úspěšně dokončen.
- > 1 … poslední zadaný příkaz byl ukončen chybou.

2.3 Příkazové funkce

Funkce zadávají příkazy, případně posloupnosti příkazů na rozhraní SMS brány. Pro vyřízení akce je třeba funkci opakovaně volat, typicky v každém průchodu programovou smyčkou, dokud nevrátí hodnotu různou od 0. Dokud funkce vrací 0, nesmí se v programu volat žádná jiná příkazová funkce pro SMS rozhraní. Při úspěšném dokončení akce pak funkce vrátí hodnotu 1. V případě chybové odpovědi na příkaz nebo po timeoutu komunikace vrátí hodnotu > 1. Výstupní hodnotu lze vždy dodatečně získat zavoláním funkce *Sms_CmState()*.

Sms_GetStatus

Deklarace :	**function byte Sms_GetStatus()**	
Parametr :	- - -	
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro vyčtení a uložení aktuálního stavu GSM brány. Uložení proběhne do proměnné *SMS_GStat* se strukturou *s_gsm_status*. Proměnná obsahuje následující informace:

- Stav připojení do GSM sítě.
- Kód poslední chyby.
- Indikátor přítomnosti SMS zprávy k vyčtení. Pokud je přítomna, jsou k ní dostupné další informace:
  - telefonní číslo odesílatele
  - čas přijetí zprávy SMS centrem
  - formát textu zprávy
  - identifikační číslo zprávy (relevantní u SMS rozdělených kvůli délce na více částí)

- celkový počet částí SMS
- číslo konkrétní dostupné části SMS

Definice struktury v jazyce simple pak vypadá následovně:

```
type struct
   bit  gsm_on,         ; příznak připojení do GSM sítě
   byte gsm_rssi,       ; síla signálu 0 až 31 (99 znamená neznámý stav)
   byte gsm_error,      ; chyba posledního zadaného příkazu vyjma GetStatus
   byte sms_status,     ; sada příznaků s významem uvedeném níže
   longword sms_time,   ; čas odeslání aktuálně dostupné SMS
   byte sms_id,         ; ID číslo dostupné SMS (u SMS o více částech)
   byte sms_part_num,   ; číslo dostupné části SMS (u SMS o více částech)
   byte sms_part_total, ; celk. počet částí SMS (SMS o více částech)
   byte[17] sms_phone   ; tel. číslo, z něhož SMS dorazila (textový řetězec)
end s_gsm_status
```

Formát položky *sms_status*:

číslo bitu :	7	6	5	4	3	2	1	0
význam :	sms připravena	x	x	x	x	x	16b kódování	sms přečtena

Když není nastaven příznak 16bit kódovaní znaků (UCS-2), je použito 7bit kódování.

Sms_ReadMsg

Sms_ReadDeleteMsg

Deklarace :	**function byte Sms_ReadMsg(var _circular_buffer cbuf, byte bufclr)** **function byte Sms_ReadDeleteMsg(var _circular_buffer cbuf, byte bufclr)**	
Parametr :	**cbuf**	zásobník pro uložení vyčteného textu
	bufclr	příznak, zda před zápisem iniciovat cbuf
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro vyčtení textu SMS zprávy aktuálně dostupné v GSM modulu. Funkce *Sms_ReadDeleteMsg* nakonec zprávu i smaže ze SIM karty - uvnitř volá *Sms_ReadMsg* a poté *Sms_DeleteMsg*.
Vyčítaný text je postupně přidáván do *cbuf*. Pokud je zadán parametr *bufclr* různý od 0, dojde nejdřív k resetu ukazatelů čtení a zápisu *cbuf* (vyprázdnění) a to zavoláním vestavěné funkce *TrResetBuffer*. To, že je nějaká zpráva k dispozici oznamuje GSM modul nastaveným stavovým bitem (viz *Sms_GetStatus*).

Sms_SendMsg

Deklarace :	**function byte Sms_SendMsg(var dstring phone, var _circular_buffer cbuf)** **function byte Sms_SendMsg(const string phone, var _circular_buffer cbuf)**	
Parametr :	**phone**	textově zadané telefonní číslo
	cbuf	zásobník pro uložení vyčteného textu
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro odeslání SMS zprávy.
Cílové telefonní číslo je dáno parametrem *phone*, např. "+420111222333". Jako text zprávy se použije řetězec uložený v *cbuf*.
Pro možnost úspěšného provedení je každopádně potřeba, aby byl modul se SIM kartou přihlášen v síti GSM (kontrolovat hodnotu *SMS_GStat.gsm_on*).

Aby bylo zaručeno správné zobrazení znaků SMS na straně přijímače, je třeba text omezit jen na velká a malá písmena bez diakritiky, číslice, mezery a znaky:

```
! " # % & ' ( ) * + , - . / : ; < = > ?
```

Sms_DropCall

Deklarace :	**function byte Sms_DropCall(var dstring phone, byte time)** **function byte Sms_DropCall(const string phone, byte time)**	
Parametr :	**phone** **time**	textově zadané telefonní číslo maximální doba vyzvánění v sekundách
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro prozvonění telefonního čísla.
Cílové telefonní číslo je dáno parametrem *phone*, např. "+420111222333". Parametr *time* udává maximální dobu vyzvánění v sekundách.
Pro možnost úspěšného provedení je každopádně potřeba, aby byl modul se SIM kartou přihlášen v síti GSM (kontrolovat hodnotu *SMS_GStat.gsm_on*).

Sms_DeleteMsg

Deklarace :	**function byte Sms_DeleteMsg()**	
Parametr :	- - -	
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro smazání aktuálně dostupné SMS zprávy v GSM modulu.
Zpráva bude odstraněna ze SIM karty, následně GSM modul sám začne procházet SIM kartu a hledat další přijatou zprávu (uloženou na SIM).

Sms_SkipMsg

Deklarace :	**function byte Sms_SkipMsg()**	
Parametr :	- - -	
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro přeskočení aktuálně dostupné SMS zprávy v GSM modulu.
Přeskočená zpráva bude ponechána na SIM kartě a při správné obsluze ji GSM modul za čas opět nabídne jako aktuálně dostupnou SMS zprávu.
Použije se v případě, kdy se čeká na příjem všech částí zprávy rozdělené do více částí. Jinak je obecně potřeba zprávy po přečtení mazat, aby nedošlo k zaplnění SIM karty a tím pádem i znemožnění příjmu dalších zpráv.

3 Příklad použití

Ukázky funkčních programů lze najít v *SMS-Demo* projektu pro StudioWin. V tomto projektu je pro všechny automaty použit společný kód obsluhy GSM brány CA5-G. Ve společném kódu ale chybí definice některých konstant, proměnných a funkcí (chybějící jsou zmíněny v komentářích ve zdrojovém souboru), ty jsou pak dodefinovány a upraveny podle potřeb v konkrétních ukázkových programech jednotlivých automatů. Projekt vznikl s myšlenkou, že i programátor ve své aplikaci použije společně sdílený kód mezi automaty v demo-projektu a pouze si uzpůsobí tu část kódu, kterou mají jednotlivé programy odlišnou.

KNIHOVNA URČENÁ K PŘENOSŮM DAT PŘES ETHERNET (CA5), GPRS (CA5) A PESNET

1 Úvod

Pro výměnu dat mezi systémy na základě protokolu EPNP může převodník CA5 disponovat rozhraním ETHERNET či GSM modulem s možností propojení přes GPRS. Kromě toho je v automatech řady 400 k dispozici ovladač komunikace PLC-DMA, který umožní zápisy a čtení do/z jiných automatů řady 400 po lince PESNET. Přenosy iniciované z programu napsaném v SIMPLE4 na zmíněných třech rozhraních se realizují stejným způsobem na příkazové bráně. Proto vznikla společná knihovna, která uvnitř svých příkazových funkcí řeší veškerou potřebnou komunikaci na příkazových bránách rozhraní.

1.1 Komunikační brána

Brána každého rozhraní je tvořena 2 EXBUS uzly, uzlem WR_BUF a uzlem RD_BUF. Díky možnosti nastavení přístupnosti brány po EXBUS z jiného automatu není použití knihovny omezeno jen na program běžící uvnitř automatu s daným rozhraním.

1.2 Přístupné paměťové prostory

Knihovnu lze využít pro přístup z programu automatu jak do vlastní (RAM) paměti programovaného automatu, tak do (RAM) paměti ostatních automatů řady 400 v síti MICROPEL. V programu lze předdefinovat základní přístupné paměti automatů konstantami, např.:

```
const _EPNP_RAM_USER = 0x30000000 ; proměnné programu (včetně fixovaných)
    , _EPNP_RAM_STACK = 0x23000000 ; zásobník
    , _EPNP_RAM_IOND = 0x21000000 ; aktuální stav uzlů EXBUS
    , _EPNP_RAM_IONDDSC = 0x21010000 ; deskriptory uzlů EXBUS
    , _EPNP_RAM_IONDLIST = 0x21030000 ; příznaky přítomnosti EXBUS uzlů
    , _EPNP_RAM_SYSTEM = 0x24000000 ; systémové proměnné (NetLW, D aj.)
```

Velikosti jednotlivých pamětí jsou obecně závislé na typu automatu. Podrobná mapa paměti automatů řady 400 je uvedena ve speciálním dokumentu.

2 Popis knihovny

Knihovna obsahuje funkce zabezpečující komunikaci s ovladači rozhraní ETHERNET, GPRS a PLC-DMA. Název veškerých funkcí začíná řetězcem „*Sync_*“.

2.1 Proměnné určené k použití

SYNC_SelComID

Proměnná typu *byte*, pomocí níž se v programu rozlišují požadavky na jedno synchronizační rozhraní. Její nastavování je potřeba v programu řešit v případě, kdy není zajištěno, že se program pokusí přistupovat k rozhraní až jen poté, co byl ukončen předchozí požadavek. Proměnnou nastavovat před každým voláním funkce *Sync_Select…* Pro více informací viz popis „Řízení přístupu k rozhraní“.

SYNC_DptrOffset

Proměnná typu *longword*, pomocí níž lze funkci *Sync_Read…* stanovit posunutí začátku místa pro ukládání přijatých dat, resp. funkci *Sync_Write…* stanovit posunutí začátku místa, odkud se odesílaná data berou. Pro více informací viz popis funkcí *Sync_Read…* a *Sync_Write…*

2.2 Obecné funkce

Sync_Select

Deklarace :	**function byte Sync_SelectETH(word node)** **function byte Sync_SelectGPRS(word node)** **function byte Sync_SelectPNET(word node)**	
Parametr :	**node**	číslo vnitřního/EXBUS uzlu komunikační brány rozhraní
Výstup :	**selected**	hodnota udávající, zda došlo ke zpřístupnění rozhraní

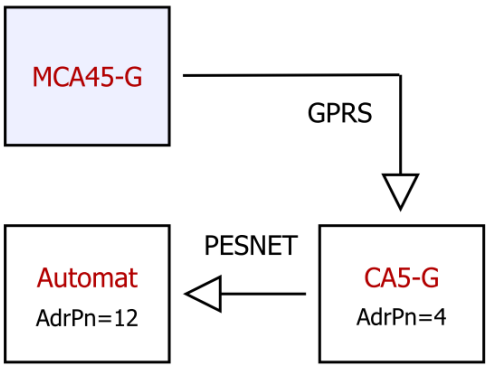
Funkce provádí nastavení cílového rozhraní pro následující volání ostatních funkcí „*Sync_*“.
Funkcí vybereme buď lokální rozhraní automatu, nebo vzdálené rozhraní dostupné po síti EXBUS (programovaný automat pak musí být EXBUS-master). Bezprostředně poté, co funkce vrátí hodnotu různou od 0, lze na rozhraní směřovat synchronizační příkazy. Návratová hodnota 0 znamená, že rozhraní nebylo zvoleno, protože je právě používáno v jiné části programu - viz „Řízení přístupu k rozhraní“.
Parametr *node* má význam čísla uzlu příslušného rozhraní. V programu pro automat, kde chceme použít lokální rozhraní (tj. rozhraní téhož automatu) se jako číslo uzlu použije konstanta:

- 8 (*_s4_nd_ETH*) … pro ETHERNET - lze použít jen v programu pro MCA45-E.
- 14 (*_s4_nd_GPRS*) … pro GPRS - lze použít jen v programu pro MCA45-G.
- 10 (*_s4_nd_PNET*) … pro PLC-DMA, tj. u PESNET synchronizace.

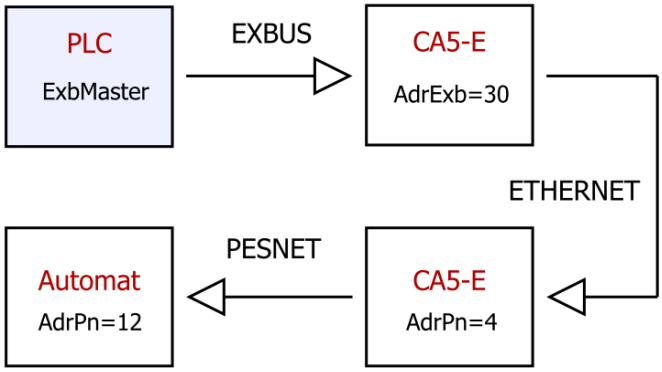
Při obsluze rozhraní vzdáleně, z programu automatu prostřednictvím EXBUS, se jako číslo uzlu použije EXBUS adresa vzdáleného automatu/převodníku zvětšená o hodnotu nastavenou v parametrech ovladače ETHERNET/GPRS/PLC-DMA (u vzdáleného automatu/převodníku).

```
Sync_SelectGPRS(8) ; Použít v programu do MCA45-G (programovatelnou CA5-G).
Sync_SelectETH(32) ; Použít v programu do PLC EXBUS-MA, kde má CA5-E EXBUS
        ; adresu 32 a nastavenu viditelnost ovladače ETH na vnějším uzlu 2.
```

Příklad zapojení pro použití parametru (8)



Příklad zapojení pro použití parametru (32)



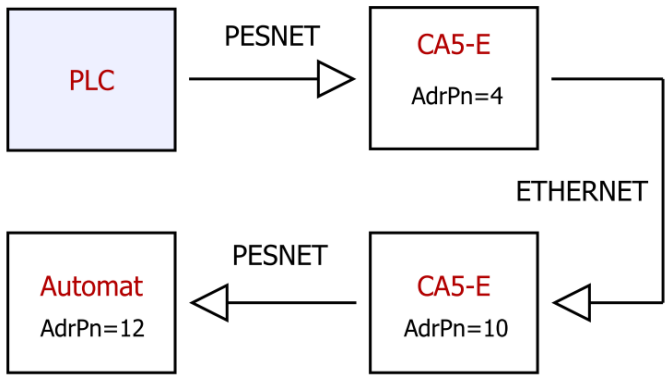
Sync_Select_Pn

Deklarace :	**function byte Sync_SelectETH_Pn(byte pnadr, word node)** **function byte Sync_SelectGPRS_Pn(byte pnadr, word node)**	
Parametr :	**pnadr**	PESNET adresa automatu, který má přímý přístup k uzlům rozhraní, typicky rovnou adresa CA5
	node	číslo vnitřního/EXBUS uzlu komunikační brány rozhraní
Výstup :	**selected**	hodnota udávající, zda došlo ke zpřístupnění rozhraní

Funkce provádí nastavení cílového rozhraní pro následující volání ostatních funkcí „*Sync_*“. Funkcí vybereme vzdálené rozhraní dostupné po síti PESNET. Bezprostředně poté, co funkce vrátí hodnotu různou od 0, lze na rozhraní směřovat synchronizační příkazy. Návratová hodnota 0 znamená, že rozhraní nebylo zvoleno, protože je právě používáno v jiné části programu - viz „Řízení přístupu k rozhraní“. Parametr *node* má, z hlediska automatu s adresou *pnadr* stejný význam jako je popsán u první verze funkce s jedním parametrem.
Se zadáním příkazu na zvolené rozhraní dojde zároveň automaticky k zahájení provedení sady příkazů na vnitřním rozhraní PLC-DMA programovaného automatu.

```
Sync_SelectETH_Pn(4,8)  ; Použít v programu, kde má CA5-E na PESNET adresu 4.
Sync_SelectETH_Pn(4,32) ; Použije se spíš jen výjimečně, např. v programu
        ; pro MT424 (s PESNET), kde má MPC400 (s PESNET + EXBUS-MA) adresu 4
        ; na lince PESNET a na EXBUS připojen převodník CA5-E s adresou 30
        ; a s nastavením viditelnosti ovladače ETH na vnějším uzlu 2.
```

Příklad zapojení pro použití parametrů (4,8)



Sync_CmState

Deklarace :	**function byte Sync_CmState()**	
Parametr :	- - -	
Výstup :	**state**	stav vyřizování posledního zadaného příkazu

Funkce vrací stav komunikace na aktuálně zvoleném rozhraní. Návratová hodnota má význam:

- 0 → poslední zadaný příkaz zatím probíhá;
- 1 → poslední zadaný příkaz byl úspěšně dokončen;
- větší než 1 → poslední zadaný příkaz byl ukončen chybou;

Řízení přístupu k rozhraní

V programu je třeba zajistit, aby se na některém z rozhraní nezahájilo vyřizování nového příkazu, jestliže na tomtéž rozhraní probíhá vyřizování jiného příkazu. Pokud budou v programu řešeny synchronizace tak, že, dokud neskončí jedna synchronizační procedura na libovolném rozhraní (ETHERNET, GPRS nebo PLC-DMA), nemůže nastat požadavek na jinou synchronizační proceduru pro totéž rozhraní, stačí volání příkazových funkcí knihovny jednoduše podmínit nenulovostí návratové hodnoty funkce *Sync_Select…* V opačném případě je navíc ještě potřeba před každým voláním funkce *Sync_Select…* jednoho rozhraní přednastavit proměnnou *SYNC_SelComID* na unikátní - pro jedno konkrétní volání funkce v kódu ale pokaždé stejnou - hodnotu z rozsahu 0 až 199 - aby šlo uvnitř knihovny jednotlivá volání rozlišovat. Po návratu z funkce *Sync_Select…* je proměnná vždy přednastavena do nuly.

```
; Příklad kódu, kde může být v jednom čase nastaveno více požadavků provedení
; synchronizačních příkazů na rozhraní (funkce 'Syn_prikazy' budou obsahovat
```

```
; posloupnost vykonávání příkazových funkcí popisovaných dále v dokumentu):
if (synEth[0]) then begin
   SYNC_SelComID = 10
   if (Sync_SelectETH(60) and Syn_prikazy_0()) then
      synEth[0] = 0 ; ukončit požadavek
   end
if (synEth[1]) then
   begin
   SYNC_SelComID = 11
   if (Sync_SelectETH(60) and Syn_prikazy_1()) then
      synEth[1] = 0 ; ukončit požadavek
   end
if (synGprs[0]) then
   begin
   SYNC_SelComID = 10 ; stejná hodnota u GPRS jakou u ETH nevadí
   if (Sync_SelectGPRS(64) and Syn_prikazy_2()) then
      synGprs[0] = 0 ; ukončit požadavek
   end
if (synGprs[1]) then
   begin
   SYNC_SelComID = 67
   if (Sync_SelectGPRS(64) and Syn_prikazy_3()) then
      synGprs[1] = 0 ; ukončit požadavek
   end
```

2.3 Příkazové funkce

Funkce zadávají příkazy, případně posloupnosti příkazů na aktuálně zvolené rozhraní.
Pro vyřízení akce je třeba funkci opakovaně volat, typicky v každém průchodu programovou smyčkou, dokud nevrátí hodnotu různou od 0. Dokud funkce vrací 0, nesmí se v programu volat žádná jiná příkazová funkce pro totéž rozhraní. Při úspěšném dokončení akce pak funkce vrátí hodnotu 1. V případě chybové odpovědi na příkaz nebo po timeoutu komunikace vrátí hodnotu > 1. Výstupní hodnotu poslední volané příkazové funkce na aktuálně zvoleném rozhraní lze vždy dodatečně získat zavoláním funkce *Sync_CmState()*.

Sync_GetStatus

Deklarace :	**function byte Sync_GetStatus(var byte status)**	
Parametr :	**status**	bajt pro uložení stavového slova rozhraní
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro získání stavového slova rozhraní. Získaný stavový bajtu má následující význam:

8. bit ... „ready“ - ovladač (modul) připraven
7. bit ... je navázáno spojení
6. až .1 bit ... výsledek posledního zadaného příkazu vyjma *GetStatus*

U rozhraní PLC-DMA (PNET) budou 8. i 7. bit vždy nastaveny do 1.

Sync_Connect

Deklarace :	**function byte Sync_Connect(var s_sync_connection con)** **function byte Sync_Connect(const s_ sync_connection con)**	
Parametr :	**con**	IP adresa a komunikační port vzdálené CA5 (serveru)
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro navázání vzdáleného spojení s jiným komunikátorem na aktuálně zvoleném rozhraní. Určena pro rozhraní ETHERNET a GPRS, u rozhraní PESNET není třeba vůbec spojení navazovat.
Nejdříve čte stav ovladače rozhraní a čeká na příznak 'ready', pak teprve vytvoří nové spojení. Když je potřeba, ukončí se před novým spojením ještě stávající aktivní spojení. Funkce vynechá ukončení a následné navázání spojení, pokud zjistí aktivní spojení a zadaná adresa i port se shodují.
Parametry spojení se předají ve struktuře:

```
type struct
   byte[4] ip, ; IP adresa - ip[0] nese nejvyšší bajt adresy, ip[3] nejnižší,
   word port,  ; např. u adresy 192.168.1.2 bude ip[0]=192, ip[1]=168 atd.
   longint password ; heslo (0 až 999999) komunikace nastavené u vzdálené CA5
end s_sync_connection
```

Pro nekódovanou komunikaci nastavit *password* na hodnotu 0. Pokud má místní CA5 nahrán FW 1.605 nebo starší, použít hodnotu *password* -1, kódování komunikace není podporováno.

Sync_ConnectPC

Deklarace :	**function byte Sync_ConnectPC(var s_sync_connection con)** **function byte Sync_ConnectPC(const s_ sync_connection con)**	
Parametr :	**con**	IP adresa a komunikační port vzdálené CA5 (serveru)
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro navázání vzdáleného spojení s komunikačním programem na PC na aktuálně zvoleném rozhraní. Určena pro rozhraní ETHERNET a GPRS, u rozhraní PESNET nelze použít. Funkce tedy neslouží k zahájení synchronizace dat mezi sítěmi, ale k propojení sítě např. pro účely ladění či programování. Po úspěšném spojení začne automat či komunikátor fungovat jako běžný převodník EPNP protokolu. Jestliže připojený program nebude udržovat spojení, pomocí zasílání EPNP požadavků, dojde po cca 30 sekundách k automatickému odpojení.
Parametr funkce, průběh spojování i návratová hodnota odpovídají popisu u příkazové funkce *Sync_Connect*.

Sync_Disconnect

Deklarace :	**function byte Sync_Disconnect()**	
Parametr :	- - -	
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce zadá příkaz k odpojení, tj. zrušení vzdáleného spojení na aktuálně zvoleném rozhraní. Použije se v případě, kdy již dříve došlo k navázání spojení pomocí volání *Sync_Connect*, příp. *Sync_ConnectPC*, a spojení se pravděpodobně nějakou chvíli nebude používat. Pokud aktivní spojení neexistuje, vrátí funkce chybu.
Funkce je určena pro rozhraní ETHERNET nebo GPRS, u rozhraní PESNET vrací rovnou 1.

Sync_ReadBit

Sync_WriteBit

Deklarace :	**function byte Sync_ReadBit(byte plca, longword addr, byte num,** **var byte val)** **function byte Sync_WriteBit(byte plc, longword addr, byte num,** **bit val)**

Parametr :	**plc**	PESNET adresa cílového automatu (0-31)
	addr	bajtová EPNP adresa do paměti cílového automatu
	num	číslo cílového bitu (0-7) v bajtu na adrese *addr*
	val	přečtená / zapisovaná hodnota bitu – 0 nebo 1
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro přečtení (Read) resp. zápis (Write) bitu z/do automatu po aktuálně zvoleném rozhraní. Parametrem *addr* se zadává EPNP adresa v paměti – viz „Přístupné paměťové prostory".
Parametr *plc* vybere konkrétní automat v místní síti (u rozhraní PLC-DMA) nebo vzdálené síti (u rozhraní ETH/GPRS). Hodnota 31 zacílí do automatu, jehož ovladač PLC-DMA využíváme (většinou půjde přímo o programovaný automat), nebo do paměti vzdáleného převodníku, ke kterému jsme připojeni.

Sync_ReadBytes/Words/Lwords

Sync_WriteBytes/Words/Lwords

Deklarace :	**function byte Sync_ReadBytes(byte plc, longword addr, longword num, dataptr pd)** **function byte Sync_ReadWords(byte plc, longword addr, longword num, dataptr pd)** **function byte Sync_ReadLwords(byte plc, longword addr, longword num, dataptr pd)** **function byte Sync_WriteBytes(byte plc, longword addr, longword num, dataptr pd)** **function byte Sync_WriteWords(byte plc, longword addr, longword num, dataptr pd)** **function byte Sync_WriteLwords(byte plc, longword addr, longword num, dataptr pd)**	
Parametr :	**plc**	PESNET adresa cílového automatu (0-31)
	addr	bajtová EPNP adresa do paměti cílového automatu
	num	počet čtených/zapisovaných položek
	pd	ukazatel na proměnnou pro uložení dat resp. se zapisovanými daty
Výstup :	**state**	stav vyřízení příkazu: 0 probíhá; 1 dokončen; >1 ukončen chybou

Příkazová funkce pro přečtení (Read) resp. zápis (Write) položek typu *byte*, *word*, nebo *longword* z/do automatu po aktuálně zvoleném rozhraní. Parametr *addr* udává EPNP adresu v paměti, viz „Přístupné paměťové prostory".
Parametr *plc* vybere konkrétní automat v místní síti (u rozhraní PLC-DMA) nebo vzdálené síti (u rozhraní ETH/GPRS). Hodnota 31 zacílí na automat, jehož ovladač PLC-DMA využíváme (většinou půjde přímo o programovaný automat), nebo do paměti vzdáleného převodníku, ke kterému jsme připojeni.
Čtení nebo ukládání dat z/do proměnné proběhne standardně kopírováním požadovaného počtu datových bajtů z/na počáteční adresu proměnné. Pokud je proměnnou např. pole, může být žádoucí posunout počáteční adresu kopírování v příkazu na jiný než první prvek pole. Pro ten účel se použije proměnná *SYNC_DptrOffset* nastavující požadovaný posun. Nastavený posun platí do dalšího zavolání libovolné funkce volby rozhraní *Sync_Select…* s nenulovou návratovou hodnotou, pak je proměnná *SYNC_DptrOffset* nastavena do 0.

```
; Čtení 10 položek StackW z automatu na adrese 7 ve vzdálené síti PESNET,
; přednastavíme čtení dat do pole až od prvku [32]
SYNC_DptrOffset = 32*2 ; 32*velikost(word), o stejnou hodnotu posunujeme
                       ; i počáteční adresu v parametru funkce
;
```

```
if (Sync_ReadWords(7, _EPNP_RAM_STACK + 32*2, 10, @StackW) = 1) then
   hotovo = 1
```

Přenášená data se kopírují do cílové (resp. berou ze zdrojové) proměnné předávané pomocí odkazu *pd* typu *dataptr*. Pokud není velikost předané proměnné zmenšená o hodnotu *SYNC_DptrOffset* dostatečná vzhledem k parametru *num*, nebudou přesahující vyčtená data nikam uložena, přesahující odesílaná data budou nedefinované hodnoty.

3 Příklady

3.1 Ukázkový kód pro přenos dat mezi automaty

Příklad umožňuje nezávisle spouštět dva procesy. První proces realizuje čtení dat z automatu na lokálním PESNETu, druhý pak synchronizaci položek pole *StackW* do automatu ve vzdáleném PESNETu dostupném přes Ethernet. Každý proces lze zahájit nastavením bitu *sync_req[]*. Po úspěšném dokončení procesu bude mít bajt *sync_cmres[]* hodnotu 1, po ukončení díky chybě hodnotu > 1.

```
;--- Definice ---;
const _EPNP_RAM_USER = 0x30000000 ; proměnné programu (včetně fixovaných)
    , _EPNP_RAM_STACK = 0x23000000 ; zásobník
code s_sync_connection con_ca5eth = ((66,121,80,43), 1703, 0) ;IP port heslo
;--- Kód v hlavní smyčce ---;
var byte[40] Ram1000 ; paměť vyčtených USER dat
var bit[2] sync_req
var byte[2] sync_cmres
var byte[2] sync_cmst
if (RESET) then begin
   sync_req[0] = 0
   sync_cmres[0] = 1
   sync_req[1] = 0
   sync_cmres[1] = 1
   end
; 1. synchronizační proces
if (sync_req[0] and sync_cmres[0] <> 0) then
   begin ; požadavek zahájení + proces neprobíhá
   sync_req[0] = 0 ; shodit příznak požadavku
   sync_cmres[0] = 0 ; nastavit stav probíhajícího procesu
   sync_cmst[0] = 0 ; začít sadu příkazů od začátku
   end
if (sync_cmres[0] = 0) then ; proces probíhá
   begin
; kopírovat položky z RAM automatu(12) zafixované na adrese 1000
   if (Sync_SelectPNET(10)) then ; bude se číst z lokálního PESNETu, rozhraní
      begin                      ; PLC-DMA (PNET) je bez navazování spojení
      sync_cmres[0] = Sync_ReadBytes(12, _EPNP_RAM_USER+1000,
        sizeof(Ram1000), @Ram1000)
      end
   end
; 2. synchronizační proces
if (sync_req[1] and sync_cmres[1] <> 0) then
   begin ; požadavek zahájení + proces neprobíhá
   sync_req[1] = 0 ; shodit příznak požadavku
   sync_cmres[1] = 0 ; nastavit stav probíhajícího procesu
   sync_cmst[1] = 0 ; začít sadu příkazů od začátku
   end
```

```
if (sync_cmres[1] = 0) then ; proces probíhá
   begin
; kopírovat část zásobníku do automatu(7) ve vzdálené síti
   if (Sync_SelectETH(34)) then ; program v MPC400(EXBUS-MA), kde má CA5-E
      begin       ; EXBUS adresu 30 a Ethernet nastavený na vnější uzel 4
      if (sync_cmst[1] = 0 and Sync_Connect(con_ca5eth) = 1) then
            sync_cmst[1] = 1 ; spojení navázáno
      SYNC_DptrOffset = 40*2
      if (sync_cmst[1] = 1 and Sync_WriteWords(7, _EPNP_RAM_STACK+40*2,
        10, @StackW) = 1) then
            sync_cmst[1] = 2 ; zapsáno StackW[40] až StackW[49]
      if (sync_cmst[1] = 2 and Sync_Disconnect() = 1) then
            sync_cmst[1] = 3
   ; uložíme návratovou hodnotu posledního příkazu
      sync_cmres[1] = Sync_CmState()
      end
   end
```

3.2 Ukázková funkce tisku stavu posledního příkazu

```
table string[33] res_txt = (
    "Probiha..."
   ; - příkaz dopadl bez chyby
   ,"OK" ;0+1
   ; - příkaz skončil chybou
   ,"Jiz pripojeno" ;1+1,"Plc adresa" ;2+1
   ,"Nepripojeno" ;3+1
   ,"Parametr" ;4+1
   ,"Nelze pripojit" ;5+1
   ,"Odmitnuto" ;6+1
   ,"Bez odezvy" ;7+1
   ,"Adresa" ;8+1
   ,"Nepripraven" ;9+1
   ,"Nelze odpojit" ;10+1
   ,"-ndef-" ;11+1
   ,"-ndef-" ;12+1
   ,"-ndef-" ;13+1
   ,"-ndef-" ;14+1
   ,"Iobuf-uzel" ;15+1
   ,"-ndef-" ;16+1
   ,"-ndef-" ;17+1
   ,"-ndef-" ;18+1
   ,"-ndef-" ;19+1
   ,"-ndef-" ;20+1
   ,"-ndef-" ;21+1
   ,"-ndef-" ;22+1
   ,"-ndef-" ;23+1
   ,"Ovladac NOK" ;24+1
   ,"Prikaz vyprsel" ;25+1
   ,"OverPesnet" ;26+1
   ,"-ndef-" ;27+1
   ,"-ndef-" ;28+1
   ,"-ndef-" ;29+1
   ,"-ndef-" ;30+1
   ,"Volba rozhrani" ;31+1
```

```
   )
subroutine Sync_DisplejStav()
      var byte res
      res = Sync_CmState()
      if (res > 1) then ; skončil chybou
            Display("Ch:")
      Display(res_txt[res])
return
```