Komunikační knihovna

LnkSofMA.DLL

Střešovická 49
162 00 PRAHA 6
Tel/Fax: (02) 20180 454

OBSAH

Příručka knihovny LnkSofMA.DLL

1. Určení knihovny

Dynamická knihovna LnkSofMA.DLL je určena pro snadnou implementaci komunikačního napojení počítače PC na řídicí systémy firmy ***SofCon® s.r.o.*** s protokolem SofCon Level 2 přes sériové rozhraní COM nebo telefonní modem.

Knihovna LnkSofMA.DLL je navržena tak, aby ji bylo možno používat v mnoha programovacích jazycích, resp. programech, které pracují na počítači PC s operačním systémem MS Windows 95/98/NT. Předpokladem pro použití knihovny LnkSofMA.DLL je schopnost zvoleného programovacího jazyka nadeklarovat a zavolat externí proceduru či funkci z dynamické knihovny DLL.

Jako příklady programů a jazyků, které dovolují knihovnu LnkSofMA.DLL používat lze uvést:

- programy napsané v Object Pascalu Delphi od verze 2.0 výše
- programy napsané v jazyce C, resp. C++, např. MS Visual C++ V6.0
- MS Visual Basic - MS Access 97, MS Excel 97, MS Word 97
- Macro jazyk v textovém procesoru 602Text
- a další 32-bitové aplikace pod MS Windows

Z uvedeného výčtu vyplývají následující možnosti použití:

- Snadné začlenění komunikace na řídicí systémy firmy SofCon do vlastních uživatelových aplikací.
- Uživatel MS Office 97 může pomocí maker vytvářet soubory protokolů, které budou obsahovat aktuálně nakomunikovaná data z řídicích systémů. Stejně může postupovat uživatel textového procesoru 602Text firmy Software602.

Komunikační knihovna LnkSofMA.DLL může komunikaci navázat nejen po lince RS232, ale také přes telefonní modem. Z hlediska uživatelského rozhraní knihovny LnkSofMA.DLL není mezi komunikací přes kabel linky RS 232 a komunikací přes telefonní modem žádný rozdíl. Pokud je použita komunikační sběrnice RS485/422, pak je možno komunikovat současně s více "slave" stanicemi na sběrnici RS485/422, neboť komunikační protokol SofCon Level 2 je síťovým protokolem.

Knihovna LnkSofMA.DLL má vestavěny diagnostické funkce, dokáže otevřít vlastní výpisové okénko, do kterého bude jako na consoli vypisovat diagnostické zprávy.

Pokud je komunikace realizována přes modem, pak knihovna nabízí plovoucí okénko, ve kterém lze sledovat činnost modemu (postup vytáčení čísla a navázaní spojení), zadávat cílové telefonní číslo a pomocí tlačítek na okénku ručně ovládat vytáčení a zavěšení.

Jméno knihovny LnkSofMA.DLL je odvozeno od skutečnosti, že knihovna vnitřně používá tzv. "Master Automat" komunikace. Tento programový automat realizuje všechny fáze komunikace. Nejdříve inicializuje komunikační port, potom naváže spojení a po navázání spojení udržuje běžící komunikaci. Master Automat ošetří krátkodobé výpadky či poruchy komunikační linky, při přerušení spojení se snaží automaticky o jeho nové navázání. Ošetřovat tyto nežádoucí stavy komunikační linky na úrovni uživatelského programu není třeba, Master Automat pracuje za Vás.

2. Nabízené služby knihovny

Nabízené služby knihovny LnkSofMA.DLL lze rozdělit do několika kategorií:

- Služby pro identifikaci verze knihovny
- Služby pro nastavení Debug Mode
- Služby pro obsluhu komunikačního kanálu
- Pomocné funkce

2.1 Služby pro identifikaci verze knihovny

Knihovna byla implementována v jazyce Object Pascal ve vývojovém prostředí Borland Delphi 4. V tomto prostředí jsou jednoduché datové typy implementovány následovně:

Typ	Implementace	Poznámka
byte	1 byte	1 byte = 8 bitů
char	1 byte	znak v kódu ASCII
Boolean	1 byte	0=false, 1=true
word	2 byte	celé číslo bez znaménka
integer	4 byte	celé číslo se znaménkem
longint	4 byte	celé číslo se znaménkem
longword	4 byte	celé číslo bez znaménka
cardinal	4 byte	celé číslo bez znaménka
Real48	6 byte	reálné číslo používané v Pascalu
Single	4 byte	reálné číslo standard
Double	8 byte	reálné číslo standard
pointer	4 byte	typ ukazatel v prostředí WIN 32 platforma CPU intel

Typ (short) string používaný v Pascalu není v rozhraní knihovny LnkSofMA.DLL používán. Pokud je na rozhraní knihovny LnkSofMA.DLL pracováno s textovým řetězcem, pak je práce s tímto textovým řetězcem realizována jako práce s řetězcem jednobytových znaků v kódu ASCII, který je zakončen znakem s kódem 0, tj. "null terminated string".

Jazyk Object Pascal používá pro práci s "null terminated" řetězci datový typ PChar, který je definován jako ukazatel na pole znaků (array[..] of char).

2.1.1 function LnkSofMA_GetVersion

Deklarace

```
function  LnkSofMA_GetVersion(AVerBuff:PChar;AVerBuffSize:word):longint;
stdcall; external 'LnkSofMA.DLL';
```

Parametry

Funkční hodnota	:longint;	- Binárni identifikace verze knihovny LnkSofMA.DLL.
AVerBuff	:PChar;	- Ukazatel na buffer, do kterého funkce zapíše textovou representaci verze knihovny LnkSofMA.DLL jako null terminated string.
AVerBuffSize	:word;	- Velikost bufferu, omezuje maximální délku předaného stringu.

Činnost

Funkce do zadaného bufferu pro "null terminated string" předá textovou representaci verze knihovny LnkSofMA.DLL. Binární representaci verze funkce předá jako svoji funkční hodnotu.

Příklad

```
var APom :array[0..127] of char;
    QVer :longint;
begin
  {=== V O L A N I   L n k S o f M A . D L L =======}
  QVer:=LnkSofMA_GetVersion(Addr(APom),SizeOf(APom));
  {================================================}
  ShowMessage('Ver=$'+LongintStrHex(QVer)+chCRLF+
              StrPas(APom));
end;
```

2.2 Služby pro nastavení Debug Mode

2.2.1 procedure LnkSofMA_SetDebugMode

Deklarace

```
procedure LnkSofMA_SetDebugMode(ADM:longint;ASysfFl:byte);stdcall;
external 'LnkSofMA.DLL';
```

Parametry

ADM	:longint;	- Množina bitů pro nastavení debug výpisů.
ASysfFl	:byte;	- <>0 = požadavek na zapisovaní do *.TXS souboru

Činnost

Procedura nastaví Debug Mode pro knihovnu LnkSofMA.DLL.

Příklad

```
begin
  { Nejvyšší bit v parametru ADM otevře okénko Debug Console }
  { Otevření okénka Debug Console knihovny LnkSofMA.DLL }
  LnkSofMA_SetDebugMode($80000000,ord(false));
end;
```

2.2.2 function LnkSofMA_ExecDlgDebugDM

Deklarace

```
type
  { Typ bufferu predavaneho dialogu pro mastaveni DebugMode }
  TDebugDM_TrRec = record
    DebugDM_Value :longint; { mnozina bitu pro nastaveni debug vypisu }
    SysfFl_Value  :Boolean; { pozadavek zapisovani do *.TXS souboru   }
  end;

{-------------------------------------------------------------------}
{ Editace zaznamu typu TDebugDM_TrRec pomoci dialogu
  implementovaneho v knihovne LnkSofMA }
function  LnkSofMA_ExecDlgDebugDM(lpDebugDM_TrRec:pointer):integer;
stdcall; external 'LnkSofMA.DLL';
```

Parametry

Funkční hodnota	:integer;	- 0 = Storno (Cancel) v dialogu, <>0 = OK v dialogu
lpDebugDM_TrRec	:pointer;	- Pointer na editovaný záznam typu TDebugDM_TrRec.

Činnost

Procedura otevře dialog pro nastavení Debug Mode v předaném záznamu typu TDebugDM_TrRec.

Příklad

```
var DebugDM_TrRec :TDebugDM_TrRec;
begin
  with DebugDM_TrRec do
  begin { inicializace položek }
    DebugDM_Value := 0;
    SysfFl_Value  := false
  end;
  { Otevření editačního dialogu záznamu }
  if LnkSofMA_ExecDlgDebugDM(Addr(DebugDM_TrRec))<>0 then
  with DebugDM_TrRec do
  begin
    { Použití nových hodnot položek záznamu z dialogu }
    LnkSofMA_SetDebugMode(DebugDM_Value,ord(SysfFl_Value));
  end;
end;
```

2.2.3 procedure LnkSofMA_ExecSetDebugMode

Deklarace

```
procedure LnkSofMA_ExecSetDebugMode; stdcall;
external 'LnkSofMA.DLL';
```

Činnost

Procedura otevře dialog pro přímé nastavení DebugMode knihovny LnkSofMA.DLL.

2.3 Služby pro obsluhu komunikačního kanálu

Knihovna LnkSofMA.DLL nabízí vytvoření jedné instance komunikačního kanálu realizovaného Master Automatem protokolu SofCon Level 2. Komunikační kanál může být realizován přes standardní

sériové rozhraní COM RS232 počítače PC. K rozhraní COM počítače PC může být být připojen telefonní modem, nebo převodník RS232-RS485/422.

2.3.1 Typ hlavičky zprávy v protokolu SofCon Level 2

Každá datová zpráva, kterou Master Automat přebírá nebo předává, začíná záznamem typu TProcMessHeader. Tento typ záznamu definuje hlavičku zprávy v protokolu SofCon Level 2. Hlavička zprávy obsahuje položky, které definují délku zprávy, adresu cíle zprávy, adresu odesílatele zprávy a kód zprávy. Kód zprávy je identifikací datové části zprávy, která následuje bezprostředně za hlavičkou zprávy.

Deklarace

```
type
{-------------------------------------------------------------------}
{ Úvodní blok (hlavička) každé zprávy přenášené protokolem SofL2 }
{-------------------------------------------------------------------}
TProcIdent      = byte;   { typ identifikátoru procesu ve zprávě }
TProcInst       = byte;   { typ instance procesu ve zprávě       }
TProcLogA       = word;   { typ logické adresy ve zprávě         }
TPMCode         = byte;   { typ kódu zprávy                      }

TProcAddr = record        { --- celkem  (4) byty --------------- }
   XIdent    :TProcIdent; {(1) identifikátor procesu }
   XInst     :TProcInst;  {(1) číslo instance procesu }
   XLogA     :TProcLogA;  {(2) logická adresa }
end;

TProcMessHeader = record  { --- celkem (14) bytu --------------- }
   BuffSize  :word;       {(2) délka včetně následujících Data   }
   DDETrans  :byte;       {(1) ddet_xxxx (vyuziti pro DDE)       }
   Destin    :TProcAddr;  {(4) adresa příjemce                   }
   Source    :TProcAddr;  {(4) adresa odesílatele                }
   MNo       :byte;       {(1) číslo vysílané zprávy             }
   ANo       :byte;       {(1) číslo poslední přijaté zprávy     }
   MCode     :TPMCode;    {(1) identifikátor/kód příkazu zprávy  }
end;
```

Z hlediska funkce Master Automatu v knihovně LnkSofMA.DLL je třeba ve vysílané zprávě správně vyplnit položky BuffSize a Destin. Ostatní položky jsou využívány aplikací, která volá služby knihovny LnkSofMA.DLL.

2.3.2 procedure LnkSofMA_ChannelCreate

Vytvoření instance Master Automatu protokolu SofCon Level 2 a otevření komunikačního kanálu.

Deklarace

```
procedure LnkSofMA_ChannelCreate(ASofMACallBackProc:TSofMACallBackProc;
                                 ASofMANotifWndHandle:HWND;
                                 ASofMANotifUserMsg  :longword); stdcall;
external 'LnkSofMA.DLL';
```

Parametry

ASofMACallBackProc	:TSofMACallBackProc; - Adresa uživatelské CallBack procedury.
ASofMANotifWndHandle	:HWND; - Handle okna, na které má být posílána uživatelská

	Windows message, která nese informace o přijaté zprávě z komunikačního kanálu.
ASofMANotifUserMsg	:longword; - Kód uživatelské Windows message (wm_User+UserID).

Činnost

Procedura LnkSofMA_ChannelCreate dovoluje komunikační kanál inicializovat několika způsoby, které se liší v mechanismu předávání přijatých zpráv. Přijaté zprávy z komunikační linky lze z Master Automatu přebírat jedním z následujících způsobů:

1) Pomocí uživatelské CallBack procedury.

2) Prostřednictvím Windows User Message zaslané na okno v uživatelské aplikaci.

3) Převzetím zprávy z výstupní fronty Master Automatem přijatých zpráv.

O tom, který způsob předávání bude používán, rozhodují předané parametry procedury LnkSofMA_ChannelCreate.

Pokud ASofMACallBackProc<>nil, pak je použit mechanismus CallBack procedury.

Pokud (ASofMACallBackProc=nil) and (ASofMANotifWndHandle<>0) and (ASofMANotifUserMsg<>0), pak je použit mechanismus posílání Windows User Message.

Pokud (ASofMACallBackProc=nil) and (ASofMANotifWndHandle=0) and (ASofMANotifUserMsg=0), pak je používána výstupní fronta přijatých zpráv.

Další konfigurační parametry pro vytvoření instance komunikačního kanálu jsou čerpány z INI-souboru LnkSofMA.INI, který musí být umístěn ve stejném adresáři jako knihovna LnkSofMA.DLL. INI-soubor obsahuje důležité informace pro vytvoření kaskády instancí komunikačních objektů, které dohromady tvoří komunikační kanál. Při vytváření kaskády objektů lze pomocí položek v INI-souboru navolit použití komunikace přes telefonní modem, definovat počet a parametry Slave stanic na síti RS485/422 a mnoho dalších parametrů.

V INI-souboru se nastavuje volba portu COM, přenosová rychlost a parita.

Důležité upozornění

Pokud v uživatelské aplikaci vytvoříme procedurou LnkSofMA_ChannelCreate komunikační kanál, je třeba zajistit, aby tako aplikace před svým ukončením komunikační kanál regulárním způsobem uzavřela voláním procedury LnkSofMA_ChannelDestroy. Tuto proceduru je vhodné volat při uzavírání aplikace vždy, ať byl nebo nebyl kanál vytvořen. Není-li kanál vytvořen, je činnost procedury prázdná. Pokud není kanál před ukončením aplikace zrušen, zůstane knihovna DLL "viset" v paměti operačního systému a počítač je zpravidla třeba restartovat. Procedura LnkSofMA_ChannelDestroy vykoná regulární uzavření komunikačního kanálu, tj. případné zavěšení modemu, uzavření portu COM a ukončení vykonávacího toku (thread), ve kterém běží Master Automat.

Poznámka: Ačkoliv knihovna LnkSofMA.DLL ve svém exit kódu sama provede uzavření kanálu, v praxi se ukazuje, že v okamžiku vykonávání tohoto kódu je již na některé činnosti příliš pozdě, a takovéto uzavření je z hlediska nežádoucích efektů neúčinné.

2.3.2.1 Příjem zprávy pomocí CallBack procedury

V tomto odstavci je popsána hlavička tzv. "CallBack" procedury, kterou definuje uživatel ve svém programu a adresu této procedury předá knihovně LnkSofMA.DLL jako parametr při otevírání komunikačního kanálu. CallBack procedura typu TSofMACallBackProc slouží pro předání přijaté zprávy z komunikačního kanálu do uživatelova programu. CallBack proceduru volá knihovna LnkSofMA.DLL v okamžiku, kdy má k dispozici přijatou zprávu. Tělo CallBack procedury implementuje uživatel ve svém vývojovém prostředí.

Implementace příjmu zpráv prostřednictvím CallBack procedury je ideálním řešením příjmu zpráv, neboť CallBack procedura je zavolána právě v okamžiku příjmu zprávy z komunikačního kanálu. Při použití CallBack procedury odpadá nutnost v uživatelově programu implementovat čekací smyčku na příjem zprávy.

Pokud uživatelovo vývojové prostředí nedovoluje nadefinovat CallBack proceduru, pak knihovna LnkSofMA.DLL nabízí také jiná klasická řešení.

Deklarace

```
const
  { CallBack type = hodnoty parametru XTYP:word pro TSofMACallBackProc }
  CBTYPE_None       =0;{ nedefinovany }
  CBTYPE_Connect    =1;{ Master Automat oznamuje navazani komunikace,
                         lpMessBuff^=Node }
  CBTYPE_DisConnect =2;{ Master Automat oznamuje ztratu   komunikace,
                         lpMessBuff^=Node }
  CBTYPE_Mess       =3;{ Master Automat predava prijatou zpravu,
                         lpMessBuff^="zprava"  }

type
TSofMACallBackProc = procedure (XTYP:word;
                                lpMessBuff:pointer;
                                wMessLen:word);stdcall;
```

Parametry

Pozor, zde parametry definuje knihovna LnkSofMA.DLL, uživatel je používá v těle procedury. Platnost hodnot parametrů je omezena na dobu volání CallBack procedury.

XTYP	:word;	- Hodnota identifikuje typ dat předávaných v bufferu. Viz. CBTYPE_XXX
lpMessBuff	:pointer;	- Ukazatel na buffer se zprávou.
wMessLen	:word;	- Délka zprávy v bufferu.

Činnost

V těle procedury musí uživatel naprogramovat přijetí tří typů zpráv:

1) Přijetí zprávy Connect - XTYP= CBTYPE_Connect
V tomto případě Master Automat oznamuje navázání komunikace se Slave stanicí, jejíž síťová adresa NODE:byte je předána v bufferu zprávy, tj. lpMessBuff^=byte(NODE) a wMessLen=SizeOf(NODE).

2) Přijetí zprávy DisConnect - XTYP= CBTYPE_DisConnect
V tomto případě Master Automat oznamuje ztrátu komunikace se Slave stanicí, jejíž síťová adresa NODE:byte je předána v bufferu zprávy, tj. lpMessBuff^=byte(NODE) a wMessLen=SizeOf(NODE).

3) Přijetí zprávy Mess - XTYP= CBTYPE_Mess
V tomto případě Master Automat předává v bufferu zprávu přijatou z komunikačního kanálu. Každá zpráva uložená do bufferu začíná záznamem hlavičky dle definice protokolu SofCon Level 2, za hlavičkou zprávy následuje její datová část. Délka celé zprávy včetně hlavičky je předána v parametru wMessLen.

V těle uživatelem nadefinované CallBack procedury musí uživatel převzít zprávu z bufferu lpMessBuff^ do své aplikace a musí CallBack proceduru co nejrychleji opustit, aby mohla komunikace dále pokračovat. Hodnota ukazatele lpMessBuff je definována pouze po dobu volání CallBack procedury, proto je třeba zprávu z bufferu lpMessBuff^ v těle procedury překopírovat do nějaké datové struktury uživatelské aplikace.

Příklad

```
{ Otevření kanálu se zadáním adresy CallBack procedury }
LnkSofMA_ChannelCreate(LnkSofMA_MessRxD,0,0);


{ CallBackProcedure pro LnkSofMA.DLL }
procedure LnkSofMA_MessRxD(XTYP:word;
                           lpMessBuff:pointer;
                           wMessLen:Word);stdcall;
begin
  case XTYP of
   CBTYPE_Connect:
     if Assigned(lpMessBuff) and (wMessLen>=SizeOf(byte){Node}) then
     begin
       { NODE= byte(lpMessBuff^) }
     end;
   CBTYPE_DisConnect:
     if Assigned(lpMessBuff) and (wMessLen>=SizeOf(byte){Node}) then
     begin
       { NODE= byte(lpMessBuff^) }
     end;
   CBTYPE_Mess:
     if Assigned(lpMessBuff) and (wMessLen>=SizeOf(word){BuffSize}) then
     begin
       { Mess = lpMessBuff^ }
     end;
  end;
end;
```

2.3.2.2 Příjem zprávy pomocí Windows User Message

Zasílaná User Message má definovány položky wParam a lParam následovně:

Parametry

Msg.wParam	= CBTYPE_XXX	- Typ volání, identifikace dat v bufferu
Msg.lParam	= pointer na buffer	- viz popis lpMessBuff pro TSofMACallBackProc

Příklad

```
{ Otevření kanálu se zadáním Handle okna a User Message }
LnkSofMA_ChannelCreate(nil,AppMainForm.Handle,wm_User+wm_SofMANotif);
```

```
{ Příjem wm_User+wm_SofMANotif }
procedure TAppMainForm.WMSofMANotif(var Msg:TMessage);
begin
  case Msg.wParam of
   CBTYPE_Connect: { SofMA oznamuje navazani komunikace }
     if pointer(Msg.lParam)<>nil then
     begin
        { NODE = byte(pointer(Msg.lParam)^) }
     end;
   CBTYPE_DisConnect: { SofMA oznamuje ztratu/zruseni komunikace }
     if pointer(Msg.lParam)<>nil then
     begin
       { NODE = byte(pointer(Msg.lParam)^) }
     end;
   CBTYPE_Mess: { SofMA posila prijatou zpravu }
     if pointer(Msg.lParam)<>nil then
     begin
       { Mess = pointer(Msg.lParam)^ }
     end;
  end;
end;
```

2.3.2.3 Příjem zprávy převzetím z výstupní fronty

Pokud při vytvoření komunikačního kanálu procedurou LnkSofMA_ChannelCreate není zadána ani CallBack procedura ani Handle okna a User Message, pak Master Automat ukládá přijaté zprávy z komunikačního kanálu do výstupní fronty. K dotazování na stav výstupní fronty slouží funkce LnkSofMA_QOutFIFOEmpty, k převzetí zprávy z fronty je určena funkce LnkSofMA_GetFromOutFIFO. Jmenované funkce jsou popsány v samostatných odstavcích.

V uživatelské aplikaci je třeba realizovat periodické dotazování na stav výstupní fronty a pokud je tato neprázdná, je možno zprávu z fronty převzít.

Příklad

```
{ Otevření kanálu s použitím výstupní fronty }
LnkSofMA_ChannelCreate(nil,0,0);
```

```
procedure TAppMainForm.QGetFromOutFIFO;
var QXTYP       :word;
    MyMessBuff  :array[0..1000] of byte;
    MessLen     :word;
begin
  if LnkSofMA_QOutFIFOEmpty=0 then
  begin
    MessLen:=LnkSofMA_GetFromOutFIFO(QXTYP,
                                     Addr(MyMessBuff),
                                     SizeOf(MyMessBuff));
    case QXTYP of
     CBTYPE_Connect:
       if (MessLen>=SizeOf(byte){Node}) then
       begin
         ...
       end;
     CBTYPE_DisConnect:
       if (MessLen>=SizeOf(byte){Node}) then
       begin
         ...
       end;
     CBTYPE_Mess:
       if (MessLen>=SizeOf(word){BuffSize}) then
       begin
         ...
       end;
    end;
  end;
end;
```

Je třeba říci, že implementace čekací smyčky na zprávu ve výstupní frontě je kritickým místem uživatelské aplikace. Pod dobu čekání na zprávu ve frontě je třeba umožnit činnost Windows. Zároveň však rychlost dotazování má zásadní vliv na celkovou rychlost komunikace.

Vhodným místem, kam zařadit dotazování, je např. metoda OnIdle objektu Application v Delphi. Dotazování lze volat také v obsluze události od časovače (timeru). Pro systémy, které neposkytují vhodné systémové nástroje pro realizaci čekací smyčky, poskytuje knihovna LnkSofMA.DLL pomocné funkce LnkSofMA_WaitForFIFO a LnkSofMA_QuietWaitForFIFO. Obě funkce vnitřně realizují čekací smyčku hlídanou časovým limitem (timeout). Po dobu "čekání" v obou funkcích je umožněna běžná činnost Windows.

2.3.3 procedure LnkSofMA_ChannelDestroy

Procedura slouží k uzavření a zrušení komunikačního kanálu.

Deklarace

```
procedure LnkSofMA_ChannelDestroy; stdcall;
  external 'LnkSofMA.DLL';
```

Parametry

Procedura LnkSofMA_ChannelDestroy má skrytý parametr, který je zadáván v INI-souboru LnkSofMA.DLL. Tímto parametrem je časový limit (timeout), po který může procedura čekat na regulární ukončení činnosti komunikačního Master Automatu. Tento čas má Master Automat na to, aby regulárním způsobem vykonal operaci DisConnect, tj. ukončení komunikace, případné zavěšení modemu a uzavření používaného portu COM. Pokud operaci DisConnect Master Automat nestihne v zadaném časovém limitu (timeout), je jeho činnost ukončena explicitně bez ohledu na jeho stav.

Doba vykonáni procedury LnkSofMA_ChannelDestroy může tudíž trvat až dobu zadaného časového limitu, stihne-li to Master Automat dříve, je doba vykonání rovna době skutečného uzavírání.

Činnost

Procedura regulárním způsobem uzavře a zruší instanci komunikačního kanálu.

Poznámka:
Je třeba si uvědomit, že aplikací předaná zpráva Master Automatu nemusí být bezprostředně odvysílána, může čekat ve frontě zpráv na odvysílání, nebo se může právě vysílat. Pokud by byl Master Automat v tomto stavu explicitně ukončen, nemusela by být taková zpráva doručena ke Slave stanici. Tato skutečnost může být na závadu, pokud taková zpráva vykonává operaci typu uzavření, odpojení nebo zrušení registrace ve Slave stanici. Regulární uzavření kanálu pomocí procedury LnkSofMA_ChannelDestroy výše zmíněné negativní jevy eliminuje, neboť k ukončení komunikace dojde v některém stabilním stavu Master Automatu.

2.3.4 function LnkSofMA_MessTxD

Funkce vykoná předání zprávy do Master Automatu k odvysílání.

Deklarace

```
function  LnkSofMA_MessTxD(lpMessBuff:pointer;wMessLen:word):integer;
  stdcall;
```

Parametry

Funkční hodnota	:integer;	- <>0...kód chyby; =0...O.K.
lpMessBuff	:pointer;	- Ukazatel na buffer se zprávou určenou k odvysílání. Zpráva v bufferu musí začínat vyplněnou hlavičkou zprávy typu TProcMessHeader, za hlavičkou následuje datová část zprávy.
wMessLen	:word;	- Délka celé zprávy v bufferu (vyjádřená v počtu byte), tj. délka hlavičky + délka datové části.

Činnost

Uživatelská aplikace použije funkci LnkSofMA_MessTxD k odvysílání zprávy protokolem SofCon Level 2.

Uživatelská aplikace musí nejdříve zprávu sestavit v bufferu, definovat správně položky hlavičky zprávy a naplnit obsah datové části zprávy. Po sestavení zprávy v bufferu, předá aplikace adresu bufferu a délku zprávy jako parametry funkce LnkSofMA_MessTxD. Funkce vrátí jako funkční hodnotu kód chyby. Je-li funkční hodnota rovna nule, převzetí zprávy Master Automatem proběhlo OK. Předáním zprávy Master Automatu končí pro uživatelskou aplikaci operace vysílání, o skutečné vyslání zprávy se postará Master Automat ve vlastní režii. Master Automat má ošetřeny chybové stavy komunikační linky a pomocí mechanismu opakování zpráv doručí zprávu Slave stanici i po krátkodobé poruše na lince.

Příklad

```
procedure TAppMainForm.TxDGlbKbd_GetParamVal(AIx:byte);
type TGetParamValMess = record
                          HED :TProcMessHeader;
                          REC :TKbd_GetParamVal;
                        end;
var ErrCode         :integer;
    GetParamValMess :TGetParamValMess;
    AdrD,AdrS       :TProcAddr;
begin
  { -- Definovani polozek hlavicky zpravy -- zacatek -------------- }
  FillChar(GetParamValMess,SizeOf(GetParamValMess),0); { vynulovani }
  with GetParamValMess.HED do
  begin
    BuffSize:=SizeOf(GetParamValMess);
    with Destin do    {(4) adresa prijemce    [86,01,4001] }
    begin
      XIdent :=$86;   { identifikator procesu PRT_KBPAR_S  }
      XInst  :=$01;   { cislo instance procesu             }
      XLogA  :=$4001; { logicka adresa procesu             }
    end;
    with Source do    {(4) adresa odesilatele [85,01,2001] }
    begin
      XIdent :=$85;   { identifikator procesu PRT_KBPAR_M  }
      XInst  :=$01;   { cislo instance procesu             }
      XLogA  :=$2001; { logicka adresa procesu             }
    end;
    MNo   :=0;                {(1) cislo vysilane zpravy         }
    ANo   :=0;                {(1) cislo posledni prijate zpravy }
    MCode :=gcmd_GetParamVal; {(1) identifikator zpravy          }
  end;
  { -- Definovani polozek hlavicky zpravy -- konec ---------------- }
  GetParamValMess.REC:=GlbKbd_GetParamVal[AIx];{ Datova cast zpravy }
  {========== V O L A N I   L n k S o f M A . D L L ================}
  ErrCode:=LnkSofMA_MessTxD(Addr(GetParamValMess),SizeOf(GetParamValMess));
  {================================================================}
  if ErrCode=0
  then { OK }
  else { ERROR };
end;
```

2.3.5 function LnkSofMA_QOutFIFOEmpty

Funkce slouží jako dotaz na stav výstupní fronty Master Automatem přijatých zpráv z komunikační linky.

Deklarace

```
function  LnkSofMA_QOutFIFOEmpty:byte; stdcall;
```

Parametry

Funkční hodnota	:byte;	- 0=ord(false) ... ve frontě jsou zprávy 1=ord(true) ... fronta je prázdná (Empty)

Činnost

Funkce jako funkční hodnotu vrací Boolean hodnotu dotazu na prázdnost výstupní fronty Master Automatu.

Poznámka:

Výstupní fronta Master Automatu je používána pouze tehdy, není-li použita pro příjem zpráv CallBack procedura nebo posílání User Message na Handle okna uživatelské aplikace.

2.3.6 function LnkSofMA_GetFromOutFIFO

Vyzvednutí zprávy z výstupní fronty Master Automatem přijatých zpráv z komunikační linky.

Deklarace

```
function LnkSofMA_GetFromOutFIFO(var XTYP:word;
                                 lpOutBuff:pointer;
                                 wOutBuffSize:word):word;
stdcall;
```

Parametry

Funkční hodnota	:word;	- Délka převzaté zprávy do bufferu lpOutBuff^, tj. délka platných dat zprávy předaných do bufferu.
var XTYP	:word;	- Proměnná, do které bude zapsána hodnota CBTYPE_XXX, identifikující převzatou zprávu.
lpOutBuff	:pointer;	- Ukazatel na buffer pro převzetí zprávy v uživatelské aplikaci.
wOutBuffSize	:word;	- Délka bufferu pro převzetí zprávy v uživatelské aplikaci, hodnota omezuje maximální možnou délku přijaté zprávy.

Činnost

Funkce je určena k vyzvednutí zprávy z výstupní fronty Master Automatem přijatých zpráv. Před zavoláním funkce LnkSofMA_GetFromOutFIFO zpravidla voláme dotazovací funkci LnkSofMA_QOutFIFOEmpty. Pokud tato dotazovací funkce vrátí hodnotu 0, tj. ord(false0, vyzvedneme zprávu z fronty do vlastního bufferu voláním funkce LnkSofMA_GetFromOutFIFO.

Poznámka:
Výstupní fronta Master Automatu je používána pouze tehdy, není-li použita pro příjem zpráv CallBack procedura nebo posílání User Message na Handle okna uživatelské aplikace.

Bližší informace o způsobu implementace výběru zpráv z výstupní fronty lze nalézt v odstavci 2.3.2.3 Příjem zprávy převzetím z výstupní fronty.

2.3.7 function LnkSofMA_WaitForFIFO

Pomocná funkce realizující čekání na zprávu ve výstupní frontě Master Automatem přijatých zpráv z komunikační linky.

Deklarace

```
function  LnkSofMA_WaitForFIFO(dwTimeOut:longint):integer;
  stdcall;
```

Parametry

Funkční hodnota	:integer;	- 0 = mrNone ... bez významu 1 = mrOK ... FIFO obsahuje zprávu 2 = mrCancel ... uplynutí dwTimeOut 3 = mrAbort ... přerušeno explicitně operátorem
dwTimeOut	:longint;	- horní časový limit čekání v [ms]

Činnost

Funkce je určena k realizaci čekání na zprávu ve výstupní frontě Master Automatu. Po dobu vykonávání funkce je otevřen modální dialog s tlačítkem ABORT. Stisknutím tlačítka ABORT lze dialog a tudíž čekání na zprávu ukončit. Funkce vrátí řízení zpět do volající aplikace až po uzavření dialogu. Po dobu otevřeného dialogu je umožněna činnost Windows.

Pokud se ve frontě neobjeví zpráva do uplynutí časového limitu, je čekání ukončeno a řízení je předáno zpět do volající aplikace.

Funkce LnkSofMA_WaitForFIFO je určena spíše pro ladění, v odladěné aplikaci je vhodnější použít funkci LnkSofMA_QuietWaitForFIFO, která realizuje čekání bez otevřeného dialogu.

Poznámka:
Pokud aplikace skutečně komunikuje, dochází k rychlému otevírání a zavírání dialogu, proto je funkce určena spíše pro fázi ladění.

2.3.8 function LnkSofMA_QuietWaitForFIFO

Pomocná funkce realizující čekání na zprávu ve výstupní frontě Master Automatem přijatých zpráv z komunikační linky.

Deklarace

```
function  LnkSofMA_QuietWaitForFIFO(dwTimeOut:longint):integer;
  stdcall;
```

Parametry

Funkční hodnota	:integer;	- 0 = mrNone ... bez významu 1 = mrOK ... FIFO obsahuje zprávu 2 = mrCancel ... uplynutí dwTimeOut
dwTimeOut	:longint;	- horní časový limit čekání v [ms]

Činnost

Funkce je určena k realizaci čekání na zprávu ve výstupní frontě Master Automatu. Po dobu vykonávání funkce je vykonáván speciální Message Loop s funkcí PeekMessage, který umožňuje běžnou činnost Windows. Funkce vrátí řízení zpět do volající aplikace po objevení se zprávy ve frontě, nebo po uplynutí časového limitu (timeout).

Funkce je určena pro aplikace, které nedovolují implementovat čekací smyčku vlastními systémovými prostředky.

2.3.9 procedure LnkSofMA_ConnectRq

Vydání explicitního požadavku na operaci Connect pro Master Automat.

Deklarace

```
procedure LnkSofMA_ConnectRq; stdcall;
```

Činnost

Zavoláním procedury LnkSofMA_ConnectRq vydáme na Master Automat požadavek na vykonání operace Connect. Tato operace naváže spojení, v případě modemu provede vytáčení telefonní linky a navázání spojení. Pokud je spojení v době volání funkce navázáno, je činnost prázdná.

2.3.10 procedure LnkSofMA_DisConnectRq

Vydání explicitního požadavku na operaci DisConnect pro Master Automat.

Deklarace

```
procedure LnkSofMA_DisConnectRq; stdcall;
```

Činnost

Zavoláním procedury LnkSofMA_DisConnectRq vydáme na Master Automat požadavek na vykonání operace DisConnect. Tato operace ukončí spojení, v případě modemu provede ukončení spojení a zavěšení telefonní linky. Pokud není spojení v době volání funkce navázáno, je činnost prázdná.

2.3.11 function LnkSofMA_ChannelType

Deklarace

```
const chtMA_None  =0;
      chtMA_COM   =1; { Kanal inicializovan jako TComChannel }
      chtMA_MODEM =2; { Kanal inicializovan jako TComModemChannel }

function  LnkSofMA_ChannelType:byte; stdcall;
```

Parametry

Funkční hodnota	:byte;	- chtMA_XXX ... typ komunikačního kanálu

Činnost

Funkce vrací typ vytvořeného komunikačního kanálu. O tom, jaký typ kanálu bude vytvořen, rozhoduje konfigurace zapsaná v INI-souboru LnkSofMA.INI. Aplikace pomocí funkce LnkSofMA_ChannelType může zjistit, zda komunikuje přes sériový kabel nebo přes telefonní modem.

2.3.12 function LnkSofMA_GetMABaseState

Deklarace

```
const
 { Identifikace stabilnich stavu Master Automatu }
  BAS_Closed      =0;{ Vychozi stav, komunikacni kanal uzavren }
  BAS_Opened      =1;{ Komunikacni kanal otevren,
                       tj. inicializovan COM port }
  BAS_Connected   =2;{ Spojeni navazano, bezi komunikace,
                       Master Automat udrzuje spojeni }
  BAS_DisConnected=3;{ Spojeni zruseno/ modem zavesen,
                       Master Automat ceka na prikaz }

function  LnkSofMA_GetMABaseState:word; stdcall;
```

Parametry

Funkční hodnota	:word;	- BAS_XXX ... identifikace stabilního stavu Master Automatu

Činnost

Pokud aplikace potřebuje explicitně zjistit v jakém stavu se nachází Master Automat komunikace, může provést dotaz voláním funkce LnkSofMA_GetMABaseState.

2.3.13 function LnkSofMA_GetSLConnectFlg

Funkce slouží jako dotaz na skutečnost, zda komunikuje zadaná Slave stanice na síti.

Deklarace

```
function  LnkSofMA_GetSLConnectFlg(ANode:byte):byte; stdcall;
```

Parametry

Funkční hodnota	:byte;	- 0=Ord(false) ... Slave stanice nekomunikuje 1=Ord(true) ... Slave stanice komunikuje
ANode	:byte;	- Hodnota NODE, tj. síťová adresa, Slave stanice na síti.

Činnost

Pomocí funkce LnkSofMA_GetSLConnectFlg může uživatelská aplikace explicitně zjistit, zda zvolená Slave stanice na síti RS485/422 komunikuje nebo nekomunikuje.

Pokud Master Automat komunikuje alespoň s jednou stanicí na síti RS485/422 je ve stavu BAS_Connected. Některé Slave stanice na síti však mohou být vypnuty, tudíž nekomunikují. Pomocí funkce LnkSofMA_GetSLConnectFlg může aplikace explicitně zjistit, které Slave stanice na síti komunikují a které nekomunikují.

Poznámka: Pozor na rozdílné chápání BAS_Connected a SLConnected v síti.

2.3.14 procedure LnkSofMA_GetSLMessCounters

Procedura LnkSofMA_GetSLMessCounters předá aktuální hodnoty čítačů přenesených zpráv na zvolenou Slave stanici v síti.

Deklarace

```
procedure LnkSofMA_GetSLMessCounters(ANode:byte;
                                     lpOKCounter,lpRepCounter:pointer);
stdcall;
```

Parametry

ANode	:byte;	- Hodnota NODE, tj. síťová adresa, Slave stanice na síti.
lpOKCounter	:pointer;	- Ukazatel na proměnnou typu longint, do které bude zapsána hodnota čítače přenesených zpráv.
lpRepCounter	:pointer;	- Ukazatel na proměnnou typu longint, do které bude zapsána hodnota čítače opakovaní přenosu zpráv.

Činnost

Pokud aplikace potřebuje indikovat běžící komunikaci a případně zjišťovat chybovost kanálu, může k tomu použít volání procedury LnkSofMA_GetSLMessCounters. Tato procedura naplní první proměnnou aktuální hodnotou čítače přenesených zpráv a druhou proměnnou aktuální hodnotou čítače opakování přenosu zpráv. Zvětšující se hodnota čítače opakování indikuje poruchovost komunikační linky.

2.3.15 procedure LnkSofMA_OpenModemOVLForm

Otevření okénka pro ovládání telefonního modemu.

Deklarace

```
procedure LnkSofMA_OpenModemOVLForm; stdcall;
```

Parametry

Procedura používá parametr INT z INI-souboru LnkSofMA.INI. Parametr INT=1 nastavuje použití mezinárodní verze dialogu, tj. v dialogu použité textové řetězce jsou čerpány z String Resource Table. String Resource Table implicitně obsahuje anglické texty

Činnost

Zavoláním procedury LnkSofMA_OpenModemOVLForm dojde k otevření plovoucího okénka, ve kterém jsou umístěny stavové a ovládací prvky telefonního modemu.

V dialogu lze zadat cílové telefonní číslo a případně alternativní telefonní číslo, na které bude voláno po neúspěšném volání na prvotní telefonní číslo. Dále lze zadat počet opakování, tj. počet pokusů o navázání spojení. Zadané parametry lze uložit tlačítkem "Ulož" do příslušné sekce parametrů modemu v INI-souboru LnkSofMA.INI.

Tlačítka "Volba" a "Závěr" slouží k explicitnímu vytáčení čísla a zavěšení (závěru) linky.

Tlačítko "Zavři" uzavře okénko, tato operace nijak neovlivní činnost modemu, pouze zavře "pohled" na činnost modemu.

Pomocí funkce LnkSofMA_ChannelType (viz.2.3.11 function LnkSofMA_ChannelType) může uživatelská aplikace zjistit, zda je komunikační kanál vytvořen přes modem, a podle toho povolit či zakázat otevření dialogu pomocí procedury LnkSofMA_OpenModemOVLForm.

2.4 Pomocné funkce a procedury

Knihovna LnkSofMA.DLL implementuje několik pomocných funkcí a procedur, které napomohou s konverzí datových typů používaných v řídicích systémech firmy SofCon na datové typy používané v různých vývojových prostředích na počítači PC a naopak.

2.4.1 function LnkSofMA_PtrToPChar

Přetypování typu pointer pro jazyky s důslednou kontrolou doménových typů.

Deklarace

```
function LnkSofMA_PtrToPChar(APtr:pointer):PChar;  stdcall;
```

Parametry

Funkční hodnota	:PChar;	- Funkce vrátí hodnotu svého argumentu APtr.
APtr	:pointer;	- Hodnota určená k přetypování.

Činnost

V jazycích, které používají důslednou typovou kontrolu doménových typů ukazatele, v některých případech při použití knihovny LnkSofMA potřebujeme provést přetypování. Ve vývojovém prostředí takového jazyka však můžeme deklarovat funkci LnkSofMA_PtrToPChar pomocí více deklarací této funkce. Rozdílné deklarace s různými typy ukazatelů mohou odkazovat na jedinou funkci LnkSofMA_PtrToPChar v knihovně LnkSofMA.DLL. Tímto způsobem bude "uspokojena" typová kontrola překladače programovacího jazyka.

Příklad

```
{ ------------- Macro 602Text --------------------------)
  type
    tString255=string[255];
    pString255=^tString255;

    TGetParamValMess = record
      HED :TProcMessHeader;
      REC :TKbd_GetParamVal;
    end;
    PGetParamValMess=^TGetParamValMess;

    TPutParamValMess=record
      HED :TProcMessHeader;
      REC :TKbd_PutParamVal;
    end;
    PPutParamValMess=^TPutParamValMess;

  (* Finta pro přetypovaní na universální PChar *)
  function  PtrStr255ToPChar(APtr:pString255):PChar;
  external "LnkSofMA.DLL" name "LnkSofMA_PtrToPChar";

  (* Finta pro přetypovaní na universální PChar *)
  function  GetParamValMessToPChar(APtr:PGetParamValMess):PChar;
  external "LnkSofMA.DLL" name "LnkSofMA_PtrToPChar";

  (* Finta pro pretypovani na universalni PChar *)
  function  PutParamValMessToPChar(APtr:PPutParamValMess):PChar;
  external "LnkSofMA.DLL" name "LnkSofMA_PtrToPChar";
```

2.4.2 function LnkSofMA_BytePtrToByte

Funkce vrátí hodnotu položky typu byte, na kterou odkazuje zadaný pointer.

Deklarace

```
function LnkSofMA_BytePtrToByte(APtr:pointer):byte; stdcall;
```

Parametry

Funkční hodnota	:byte;	- Funkce vrátí hodnotu byte(APtr^).
APtr	:pointer;	- Ukazatel na hodnotu typu byte.

2.4.3 function LnkSofMA_WordPtrToDWord

Funkce vrátí hodnotu položky typu word, na kterou odkazuje zadaný pointer.

Deklarace

```
function LnkSofMA_WordPtrToDWord(APtr:pointer):longword;  stdcall;
```

Parametry

Funkční hodnota	:longword;	- Funkce vrátí hodnotu word(APtr^).
APtr	:pointer;	- Ukazatel na hodnotu typu word.

2.4.4 procedure LnkSofMA_DWordToWordPtr

Procedura zapíše hodnotu svého prvního parametru do položky typu word, která je umístěna na adrese druhého parametru.

Deklarace

```
procedure LnkSofMA_DWordToWordPtr(ADWord:longword;APtr:pointer);
 {^.word} stdcall;
```

Parametry

ADWord	:longword;	- Hodnota určená k zápisu jako word na word(APtr^).
APtr	:pointer;	- Adresa pro uložení výsledku.

2.4.5 function LnkSofMA_IntPtrToLInt

Funkce vrátí hodnotu položky typu short, na kterou odkazuje zadaný pointer.

Deklarace

```
function LnkSofMA_IntPtrToLInt(APtr:pointer):longint; stdcall;
```

Parametry

Funkční hodnota	:longint;	- Funkce vrátí hodnotu short(APtr^).
APtr	:pointer;	- Ukazatel na hodnotu typu short.

2.4.6 procedure LnkSofMA_IntToIntPtr

Procedura zapíše hodnotu svého prvního parametru do položky typu short, která je umístěna na adrese druhého parametru.

Deklarace

```
procedure LnkSofMA_IntToIntPtr(AInt:Short;APtr:pointer);{^.short}stdcall;
```

Parametry

AInt	:Short;	- Hodnota určená k zápisu jako short na short(APtr^).
APtr	:pointer;	- Adresa pro uložení výsledku.

2.4.7 function LnkSofMA_LIntPtrToLInt

Funkce vrátí hodnotu položky typu longint, na kterou odkazuje zadaný pointer.

Deklarace

```
function LnkSofMA_LIntPtrToLInt(APtr:pointer):longint; stdcall;
```

Parametry

Funkční hodnota	:longint;	- Funkce vrátí hodnotu longint(APtr^).
APtr	:pointer;	- Ukazatel na hodnotu typu longint.

2.4.8 procedure LnkSofMA_LIntToLIntPtr

Procedura zapíše hodnotu svého prvního parametru do položky typu longint, která je umístěna na adrese druhého parametru.

Deklarace

```
procedure LnkSofMA_LIntToLIntPtr(ALInt:longint;APtr:pointer); {^.longint}
stdcall;
```

Parametry

ALInt	:longint;	- Hodnota určená k zápisu jako longint na longint(APtr^).
APtr	:pointer;	- Adresa pro uložení výsledku.

2.4.9 function LnkSofMA_FDatePtrToDate

Funkce provede konverzi hodnoty datumu a času ve formátu MS-DOS, tj. DWord, na formát času typu TDateTime, tj. Double.

Deklarace

```
function LnkSofMA_FDatePtrToDate(APtr:pointer):TDateTime;{Double}
stdcall;
```

Parametry

Funkční hodnota	:TDateTime;	- Funkce vrátí konvertovanou hodnotu datumu a času. Hodnota typu TDateTime je zakódována do hodnoty reálného čísla typu Double.
APtr	:pointer;	- Ukazatel na hodnotu typu longint, která je zakódována jako datum a čas používaný v systému MS DOS.

2.4.10 procedure LnkSofMA_DateToFDatePtr

Procedura provede konverzi hodnoty datumu a času z formátu TDateTime, tj. Double, na formát MS-DOS, tj. DWord.

Deklarace

```
procedure LnkSofMA_DateToFDatePtr(ADateTime:TDateTime;
                                  APtr:pointer);{^.FDate}
stdcall;
```

Parametry

ADateTime	:TDateTime;	- Hodnota typu TDateTime určená ke konverzi na longword(APtr^) v zakódování jako datum a čas používaný v MS DOS.
APtr	:pointer;	- Adresa pro uložení výsledku konverze.

2.4.11 LnkSofMA_PStrToSzStr

Procedura provede konverzi Pascalského string na null terminated string.

Deklarace

```
procedure LnkSofMA_PStrToSzStr(InPStr:pointer;
                               OutSzBuff:PChar;
                               OutSzBuffSize:word);
stdcall;
```

Parametry

InPStr	:pointer;	- Ukazatel na Pascalský string určený k převodu na null terminated string.
OutSzBuff	:PChar;	- Adresa cílového bufferu pro null terminated string.
OutSzBuffSize	:word;	- Délka cílového bufferu pro null terminated string.

2.4.12 procedure LnkSofMA_SzStrToPStr

Procedura provede konverzi null terminated string na Pascalský string.

Deklarace

```
procedure LnkSofMA_SzStrToPStr(InSzStr:PChar;
                               OutPStrBuff:pointer;
                               OutPStrBuffSize:word);
stdcall;
```

Parametry

InSzStr	:PChar;	- Ukazatel na null terminated string určený k převodu na Pascalský string.
OutPStrBuff	:pointer;	- Adresa cílového bufferu pro Pascalský string.
OutPStrBuffSize	:word;	- Délka cílového bufferu pro Pascalský string.

2.4.13 procedure LnkSofMA_ConvertReal6To4

Procedura je určena k převodu pole Pascalských real hodnot (size 6) na pole real hodnot typu Single (size 4).

Deklarace

```
procedure LnkSofMA_ConvertReal6To4(R:PRealp;S:PSingle;Cnt:word); stdcall;
```

Parametry

R	:PRealp;	- Ukazatel na zdrojové pole Pascalských real hodnot, tj. pole typu array[0..Cnt-1] of Real48.
S	:PSingle;	- Ukazatel na cílové pole real hodnot typu Single, tj. pole typu array[0..Cnt-1] of Single.
Cnt	:word;	- Počet položek ve zdrojovém poli R a zároveň také v cílovém poli S.

2.4.14 procedure LnkSofMA_ConvertReal6To8

Procedura je určena k převodu pole Pascalských real hodnot (size 6) na pole real hodnot typu Double (size 8).

Deklarace

```
procedure LnkSofMA_ConvertReal6To8(R:PRealp;D:PDouble;Cnt:word); stdcall;
```

Parametry

R	:PRealp;	- Ukazatel na zdrojové pole Pascalských real hodnot, tj. pole typu array[0..Cnt-1] of Real48.
D	:PDouble;	- Ukazatel na cílové pole real hodnot typu Double, tj. pole typu array[0..Cnt-1] of Double.
Cnt	:word;	- Počet položek ve zdrojovém poli R a zároveň také v cílovém poli D.

2.4.15 procedure LnkSofMA_ConvertReal4To6

Procedura je určena k převodu pole real hodnot typu Single (size 4) na pole Pascalských real hodnot (size 6).

Deklarace

```
procedure LnkSofMA_ConvertReal4To6(S:PSingle;R:PRealp;Cnt:word); stdcall;
```

Parametry

S	:PSingle;	- Ukazatel na zdrojové pole real hodnot typu Single, tj. pole typu array[0..Cnt-1] of Single.
R	:PRealp;	- Ukazatel na cílové pole Pascalských real hodnot, tj. pole typu array[0..Cnt-1] of Real48.
Cnt	:word;	- Počet položek ve zdrojovém poli S a zároveň také v cílovém poli R.

2.4.16 procedure LnkSofMA_ConvertReal8To6

Procedura je určena k převodu pole real hodnot typu Double (size 8) na pole Pascalských real hodnot (size 6).

Deklarace

```
procedure LnkSofMA_ConvertReal8To6(D:PDouble;R:PRealp;Cnt:word);stdcall;
```

Parametry

D	:PDouble;	- Ukazatel na zdrojové pole real hodnot typu Double, tj. pole typu array[0..Cnt-1] of Double.
R	:PRealp;	- Ukazatel na cílové pole Pascalských real hodnot, tj. pole typu array[0..Cnt-1] of Real48.
Cnt	:word;	- Počet položek ve zdrojovém poli D a zároveň také v cílovém poli R.

2.4.17 procedure LnkSofMA_BuffMove

Procedura provede překopírování obsahu zdrojového bufferu do cílového bufferu.

Deklarace

```
procedure LnkSofMA_BuffMove(SrcBuffPtr,DstBuffPtr:pointer;
                            DstBuffSize:cardinal); stdcall;
```

Parametry

SrcBuffPtr	:pointer;	- Adresa zdrojového bufferu.
DstBuffPtr	:pointer;	- Adresa cílového bufferu.
DstBuffSize	:cardinal;	- Délka cílového bufferu nebo počet byte, které budou kopírovány ze SrcBuffPtr^ na DstBuffPtr^.

2.4.18 function LnkSofMA_ByteStrHex

Funkce převede hodnotu typu byte na hexadecimální stringovou representaci do cílového null terminated stringu. Jako funkční hodnotu vrátí délku vytvořeného stringu.

Deklarace

```
function LnkSofMA_ByteStrHex(B:Byte;OutStr:PChar;MaxLen:word):word;
stdcall;
```

Parametry

B	:Byte;	- Hodnota typu byte určená k převodu do stringu.
OutStr	:PChar;	- Adresa cílového bufferu pro null terminated string.
MaxLen	:word;	- Délka cílového bufferu pro null terminated string.

2.4.19 function LnkSofMA_WordStrHex

Funkce převede hodnotu typu word na hexadecimální stringovou representaci do cílového null terminated stringu. Jako funkční hodnotu vrátí délku vytvořeného stringu.

Deklarace

```
function LnkSofMA_WordStrHex(W:Word;OutStr:PChar;MaxLen:word):word;
stdcall;
```

Parametry

W	:Word;	- Hodnota typu word určená k převodu do stringu.
OutStr	:PChar;	- Adresa cílového bufferu pro null terminated string.
MaxLen	:word;	- Délka cílového bufferu pro null terminated string.

2.4.20 function LnkSofMA_LongIntStrHex

Funkce převede hodnotu typu longint na hexadecimální stringovou representaci do cílového null terminated stringu. Jako funkční hodnotu vrátí délku vytvořeného stringu.

Deklarace

```
function LnkSofMA_LongIntStrHex(L:LongInt;OutStr:PChar;MaxLen:word):word;
stdcall;
```

Parametry

L	:longint;	- Hodnota typu longint určená k převodu do stringu.
OutStr	:PChar;	- Adresa cílového bufferu pro null terminated string.

MaxLen	:word;	- Délka cílového bufferu pro null terminated string.

2.4.21 function LnkSofMA_PointerStrHex

Funkce převede hodnotu typu pointer na hexadecimální stringovou representaci do cílového null terminated stringu. Jako funkční hodnotu vrátí délku vytvořeného stringu.

Deklarace

```
function LnkSofMA_PointerStrHex(P:pointer;OutStr:PChar;MaxLen:word):word;
stdcall;
```

Parametry

P	:pointer;	- Hodnota typu pointer určená k převodu do stringu.
OutStr	:PChar;	- Adresa cílového bufferu pro null terminated string.
MaxLen	:word;	- Délka cílového bufferu pro null terminated string.

2.4.22 function LnkSofMA_BufferStrHex

Funkce převede pole hodnot typu byte na hexadecimální stringovou representaci do cílového null terminated stringu. Jako funkční hodnotu vrátí délku vytvořeného stringu.

Deklarace

```
function LnkSofMA_BufferStrHex(P:pointer;L:word;
                               OutStr:PChar;MaxLen:word):word;stdcall;
```

Parametry

P	:pointer;	- Ukazatel na buffer určený k výpisu do stringu.
L	:word;	- Délka bufferu;
OutStr	:PChar;	- Adresa cílového bufferu pro null terminated string.
MaxLen	:word;	- Délka cílového bufferu pro null terminated string.

2.4.23 function LnkSofMA_RealStrDec

Funkce převede hodnotu typu Pascalský real (size 6) na dekadickou stringovou representaci do cílového null terminated stringu. Jako funkční hodnotu vrátí délku vytvořeného stringu.

Deklarace

```
function LnkSofMA_RealStrDec(R:PRealp;Len:Byte;Flt:Byte;
                             OutStr:PChar;MaxLen:word):word;stdcall;
```

Parametry

R	:PRealp;	- Ukazatel na Pascalský real určený k převodu.
Len	:Byte;	- Délka požadované stringové representace.
Flt	:Byte;	- Požadovaný počet desetinných míst. Pokud Len=Flt, pak výsledný string bude délky Len s plovoucí desetinnou tečkou.
OutStr	:PChar;	- Adresa cílového bufferu pro null terminated string.
MaxLen	:word;	- Délka cílového bufferu pro null terminated string.

2.4.24 function LnkSofMA_RealStrExp

Funkce převede hodnotu typu Pascalský real (size 6) na dekadickou exponenciální stringovou representaci do cílového null terminated stringu. Jako funkční hodnotu vrátí délku vytvořeného stringu.

Deklarace

```
function LnkSofMA_RealStrExp(R:PRealp;Len:Byte;
                             OutStr:PChar;MaxLen:word):word;stdcall;
```

Parametry

R	:PRealp;	- Ukazatel na Pascalský real určený k převodu.
Len	:Byte;	- Délka požadované stringové representace.
OutStr	:PChar;	- Adresa cílového bufferu pro null terminated string.
MaxLen	:word;	- Délka cílového bufferu pro null terminated string.

2.4.25 Časovač LnkSofMA_UserTimer

Pomocí procedur knihovny LnkSofMA.DLL lze realizovat jednu instanci časovače, který bude volat uživatelskou CallBack proceduru.

2.4.25.1 procedure LnkSofMA_UserTimerCreate

Vytvoření uživatelského časovače, který bude v zadané periodě volat uživatelskou CallBack proceduru.

Deklarace

```
type
  TUserTimerCallBackProc = procedure; stdcall;

procedure LnkSofMA_UserTimerCreate(APeriod:longword;
                    AUserCallBackProc:TUserTimerCallBackProc;
                    AEnabled:Boolean);stdcall;
```

Parametry

APeriod	:longword;	- Perioda časovače v [ms].
AUserCallBackProc	:TUserTimerCallBackProc;	- CallBack procedura, která bude volána periodicky při "ticku" časovače.
AEnabled	:Boolean;	- Povolení/zakázání činnosti uživatelského časovače.

2.4.25.2 procedure LnkSofMA_UserTimerSetEnabled

Povolení/zakázání činnosti uživatelského časovače.

Deklarace

```
procedure LnkSofMA_UserTimerSetEnabled(AEnabled:Boolean);stdcall;
```

2.4.25.3 procedure LnkSofMA_UserTimerDestroy

Zrušení uživatelského časovače.

Deklarace

```
procedure LnkSofMA_UserTimerDestroy; stdcall;
```

3. Soubor LnkSofMA.INI

V souboru LnkSofMA.INI jsou uloženy systémové konfigurační parametry pro knihovnu LnkSofMA.DLL. Soubor LnkSofMA.INI musí ležet ve stejném adresáři, jako soubor knihovny LnkSofMA.DLL. Bez informací získaných ze souboru LnkSofMA.INI nemůže knihovna LnkSofMA.DLL pracovat. Do souboru LnkSofMA.INI jsou explicitně při běhu knihovny LnkSofMA.DLL nebo při ukončení jejího běhu ukládány vybrané konfigurační parametry.

3.1 Sekce [SECTIONS]

Základní informací v INI-souboru jsou položky uvedené v sekci [SECTIONS].

```
[SECTIONS]
RD_SECTION=SofMACFG
WR_SECTION=SofMACFG
```

3.1.1 Položka s klíčem RD_SECTION

Položka s klíčem RD_SECTION obsahuje jméno sekce, odkud budou čteny konfigurační parametry pro knihovnu LnkSofMA.DLL. Parametry z uvedené sekce zpravidla knihovna LnkSofMA.DLL použije při své inicializaci, tj. po spuštění programu, který knihovnu používá, nebo při vytváření instance objektu Master Automatu.

3.1.2 Položka s klíčem WD_SECTION

Položka s klíčem WR_SECTION obsahuje jméno sekce, kam budou případně zapisovány při běhu knihovny změněné konfigurační parametry. Pokud nepotřebujeme chránit sekci uvedenou v položce s klíčem RD_SECTION proti zápisu, můžeme v položce s klíčem WR_SECTION uvést stejné jméno sekce jako v položce RD_SECTION. V takovém případě může knihovna LnkSofMA.DLL modifikovat některé své parametry a takto modifikované parametry budou použity i při příštím spuštění programu s použitím knihovny LnkSofMA.DLL.

Je-li jméno sekce uvedené v položce WR_SECTION jiné, než jméno sekce uvedené v položce RD_SECTION, pak se modifikované parametry neuplatní při novém startu knihovny LnkSofMA.DLL.

3.2 Konfigurační sekce pro knihovnu LnkSofMA.DLL

Jméno sekce je libovolné, pokud však mají být parametry ze sekce použity, musí být jméno sekce uvedeno v položce RD_SECTION (viz.3.1.1 Položka s klíčem RD_SECTION).

V konfigurační sekci jsou uvedeny tzv. inicializační textové řetězce pro instance objektů. Z inicializačního textového řetězce získává objekt zpravidla ve svém constructoru hodnoty některých svých položek. Klíčem řádky, na které je uveden inicializační řetězec, je zpravidla jméno přiřazené

instanci objektu. Samotný inicializační řetězec se skládá z přiřazovacích příkazů pro jednotlivé položky objektu. Položky objektu jsou identifikovány krátkým, zpravidla tříznakovým identifikátorem.

V následujících odstavcích budou popsány inicializační řetězce pro instance objektů, které se používají v knihovně LnkSofMA.DLL.

3.2.1 Aplikační objekt LnkSofMA

Aplikační objekt LnkSofMA slouží pro uložení všeobecných konfiguračních parametrů aplikace.

Položky LnkSofMA

UPD - UpDate flag
0 ... aktualizace položek objektu zakázána
1 ... aktualizace položek objektu povolena
Je-li aktualizace povolena, pak při uzavírání aplikace jsou do INI-souboru zapsány hodnoty právě platné v aplikaci.

DM - DebugMode
bitově orientovaná položka representující množinu nastavených flagů pro ladící výpisy do okénka.

TXS - Zapnutí výpisu do debug souboru *.TXS
0 ... výpis do *.TXS vypnut
1 ... výpis do *.TXS zapnut
Je-li výpis zapnut, pak se ladící výpisy píší paralelně také do souboru s příponou ".TXS". Jméno souboru je odvozeno od jména programu a času spuštění programu.
Výpis do souboru pracuje pouze v ladící verzi knihovny LnkSofMA.DLL. Ladící verzi identifikujeme dle řetězce "TXS" v řetězci programové verze knihovny LnkSofMA.

TMA - Master Automat Thread priority in [1..7]
Priorita vlákna, ve kterém běží Master Automat. Zpravidla volíme prioritu tpNormal=4, nebo tpHigher=5.

SMA - Thread Sleep time [ms]
Parametr pro proceduru Win32 Sleep(), udávající dobu, po kterou je vlákno Master Automatu "uspáno", tj. odevzdá řízení operačnímu systému a ten přidělí čas procesoru jinému vláknu.

DMA - Destroy timeout [ms]
Maximální časový limit, ve kterém může Master Automat provést své regulární uzavření při ukončování aplikace. Nestihne-li to v zadaném čase, je Master Automat explicitně ukončen bez ohledu na jeho vnitřní stav.

INT - International
0 ... knihovna LnkSofMA.DLL používá český jazyk
1 ... knihovna LnkSofMA.DLL používá textové řetězce z resource string table, Originální resource string table obsahuje anglické textové řetězce.

3.2.2 Adresa instance Master Automatu PL_SofMA

Objekt se jménem PL_SofMA slouží pro uložení systémové adresy Master Automatu a dalších parametrů použitých při vytváření instance objektu Master Automatu.

Položky PL_SofMA

UPD - UpDate flag
0 ... aktualizace položek objektu zakázána
1 ... aktualizace položek objektu povolena
Je-li aktualizace povolena, pak při uzavírání aplikace jsou do INI-souboru zapsány hodnoty právě platné v aplikaci.

ANY - systémová adresa s libovolným XIdent a XInst, rozhodující je pouze XLogA, tj. [?,?,XlogA]

ADR - systémová adresa s definovaným XIdent, XInst a XLogA, tj. [XIdent,XInst,XLogA]

PRS - process static - povinné nastavení pro LnkSofMA.DLL, nastavení typu procesu, zpravidla rovno XIdent, typ přiřazeného Master Automatu

NAM - jméno instance Master Automatu

3.2.3 Objekt instance Master Automatu &SOFMA

Objekt obsahuje parametry použité při vytváření instance Master Automatu.

Položky &SOFMA

DWN - Číslo okénka pro debug výpisy.

NOD - NODE síťová adresa, kterou je Master Automat adresován v komunikačním protokolu.

L0T - Jméno typu komunikačního objektu L0 protokolu, zde implicitně 'SOFL0'.

L0N - Jméno instance komunikačního objektu L0 protokolu.

LRB - Velikost alokovaného přijímacího bufferu, implicitní hodnota je $2FFF.

SCN - Jméno SlaveChanColl, tj. jméno instance seznamu SLAVE stanic na síti.

TOP - TimeOut pro MA_ChnL0.CHOpen v [ms].

TIC - TimeOut pro MA_ChnL0.CHConnect v [ms].

TRC - Prodleva mezi pokusy o novy Connect v [ms]. 0=vypnuto.

TID - TimeOut pro MA_ChnL0.CHDisConnect v [ms].

TCL - TimeOut pro MA_ChnL0.CHClose v [ms].

3.2.4 Seznam Slave stanic na síti %SLCHANCOLL

Definování seznamu SLAVE stanic na síti pro Master Automat.

Položka %SLCHANCOLL

UPD - UpDate flag
0 ... aktualizace položek objektu zakázána
1 ... aktualizace položek objektu povolena
Je-li aktualizace povolena, pak při uzavírání aplikace jsou do INI-souboru zapsány hodnoty právě platné v aplikaci.

PFX - Prefix jména jednotlivých položek seznamu (collection)

MAX - Počet položek seznamu-1, tj. maximální hodnota indexu položky, jména klíčů položek jsou tvořena pomocí řetězce zadaného jako prefix PFX a hexadecimálně vyjádřeného indexu položky, tj. 00 až MAX.

Položky seznamu %SLCHANCOLL

NOD - NODE síťová adresa, kterou je Slave adresován na síti komunikačním protokolem.

LOG - Logická adresa, kterou je Slave adresován (Dest.XLogA), případně dolní mez intervalu logických adres.

HIL - Horní mez intervalu log. adres, kterými je Slave adresován (Dest.XLogA).

LO2 - Další logická adresa, kterou je Slave adresován.

HI2 - Horní hranice intervalu další logické adresy.

TTF - TstTimeoutFirst[ms], timeout příjmu testovací zprávy.

RTF - RecTimeoutFirst[ms], timeout příjmu datové zprávy.

REP - Max. počet opakování žádosti M ->S při neobdržení odpovědi či při chybě komunikace.

EW - ErrWait - prodleva po chybě, čas k zotavení se z chybového stavu.

RC - ReadCnt - počet dotazovacích zpráv protokolu za sebou v jednom cyklu před přechodem na případné datové zprávy z fronty zpráv na odvysílání.

WC - WriteCnt- počet případných datových zpráv (z fronty zpráv na odvysílání) za sebou v jednom cyklu před přechodem na dotazovací zprávy protokolu.

TI - Délka intervalu periodického testování připojení nepřipojené Slave stanice na síti v [ms].

TN - Minimální počet testem prošlých testovacích zpráv pro ukončení testovacího režimu.

3.2.5 Objekt komunikačního protokolu úrovně L0 !SOFL0

Paramery pro instanci komunikačního objektu na úrovni protokolu SofCon Level 0.

Položky !SOFL0

P_CHT - Jméno typu podřízeného komunikačního objektu.
Jméno typu kanálu =COM
Jméno typu kanálu =MODEM

P_CHN - Jméno instance podřízené komunikační jednotky.

P_LSB - Velikost alokovaného vysílacího bufferu, implicitně =$1FFF.

3.2.6 Objekt obsluhy sériového portu COM

Parametry pro instanci komunikačního objektu, který přímo ovládá pomocí služeb Win32 sériový port počítače PC.

Položky !COM

LRB - Velikost alokovaného přijímacího bufferu.
Implicitní hodnota =0.

COM - Číslo COM portu na počítači PC

BD - Bd rychlost, přenosová rychlost
Implicitní hodnota =9600.

BIT - Počet bitů znaku.
Implicitní hodnota =8.

PAR - Parita přenášeného znaku (N-none,E-even,O-odd).
Implicitní hodnota =N.

IRT - Min. prodleva mezi příjmem a vysíláním v [ms].
Implicitní hodnota =0.

IQU - Délka Input Queue pro MS Windows. (Pozn. Délka musí být dělitelná 4.)
Implicitní hodnota = 4096.

OQU - Délka Output Queue pro MS Windows. (Pozn. Délka musí být dělitelná 4.)
Implicitní hodnota = 2048.

CTR - 0 ... k nastavení DTR=1 dojde již při CHOpen.
1 ... k nastavení DTR=1 dojde až při CHConnect.
Implicitní hodnota =0.

RTS - Nastavení signálu RTS:
=$FF ... signál ovládán programem
=0 ... nastavení signálu RTS=0
=1 ... nastavení signálu RTS=1
Možnost explicitního nastavení signálu RTS na hodnotu, kterou program neovlivní.

DTR - Nastavení signálu DTR:
=$FF ... signál ovládán programem
=0 ... nastavení signálu DTR=0
=1 ... nastavení signálu DTR=1
Možnost explicitního nastavení signálu DTR na hodnotu, kterou program neovlivní.

3.2.7 Objekt obsluhy telefonního modemu

Parametry pro instanci komunikačního objektu, který přímo ovládá pomocí služeb Win32 sériový port počítače PC, na kterém je připojen externí telefonní modem ovládaný tzv. AT-příkazy.

Položky !MODEM

Pro modem platí také stejné položky jako pro objekt COM.

MOD - Jméno seznamu řádek s parametry pro modem

Příklad

```
; parametry pro COM RS232C modemové propojení
!MODEM=COM=2 BD=38400 MOD=@SUPRAMASTER
```

Pro typ kanálu MODEM je třeba zadat položku MOD. Tato položka udává jméno seznamu s parametry pro modem.

Příklad

```
@SUPRAMASTER=PFX=SM_ MAX=$05
SM_00= BD=38400 PAR=N MAS=1 QD1=30000 HWI=0 HWH=1
SM_01= DHL=1000 DHH=500 DI1=1000 DI2=1000 DRC=0
SM_02= QD1=45000 QD2=5000 QD3=5000
SM_03= I1S='ATZ'
SM_04= I2S='ATLM3S0=0S95.0=1S95.2=1S95.3=1'
SM_05= PNS='ATDP' PNN=561
```

Seznam parametrů pro modem

PFX - Prefix jména jednotlivých položek seznamu (collection)

MAX - Počet položek seznamu-1, tj. maximální hodnota indexu položky, jména klíčů položek jsou tvořena pomocí řetězce zadaného jako prefix PFX a hexadecimálně vyjádřeného indexu položky, tj. 00 až MAX.

Položky parametrů pro modem

ARB - Velikost přijímacího bufferu odezev AT příkazů

DES - Prodleva před a po ESC-řetězci při SW závěru (položení sluchátka)

DIL - Doba trvání signálu DTR=0 při HW inicializaci modemu

DIH - Doba trvání signálu DTR=1 při HW inicializaci modemu

DHL - Doba trvání signálu DTR=0 při HW závěru modemu

DHH - Doba trvání signálu DTR=1 při HW závěru modemu

I1S - 1. inicializační řetězec s AT-příkazy pro modem

I2S - 2. inicializační řetězec s AT-příkazy pro modem

I3S - 3. inicializační řetězec s AT-příkazy pro modem

DI1 - Prodleva po vyslání 1. inicializačního AT-řetězce do modemu

DI2 - Prodleva po vyslání 2. inicializačního AT-řetězce do modemu

DI3 - Prodleva po vyslání 3. inicializačního AT-řetězce do modemu

DBU - Prodleva mezi opakovaným vytáčením při obsazení (BUSY)

QCS - Časovka (timeout) pro vyslání AT-příkazu [ms]

QCA - Časovka (timeout) pro příjem odpovědi na AT příkazy

QD1 - Časovka (timeout) pro 1. odpověď na PNS+PNN [ms]

QD2 - Časovka (timeout) pro 2. odpověď na PNS+PNN [ms], 0-nečeká se

QD3 - Časovka (timeout) pro 3. odpověď na PNS+PNN [ms], 0-nečeká se

PNS - Řetězec vytáčecího AT-příkazu, např. "ATDP"

PNN - Řetězec vytáčeného telefonního čísla, např. "0,20180561"

PNA - Řetězec alernativního vytáčeného telefonního čísla

OKS - Řetězec pro komparaci odpovědi modemu "OK", odpověď se komparuje pouze do délky zadaného řetězce

COS - Řetězec pro komparaci odpovědi modemu "CONNECT", odpověď se komparuje pouze do délky zadaného řetězce, tj. "CONNECT"="CONNECT 19200"

BUS - Řetězec pro komparaci odpovědi modemu "BUSY", odpověď se komparuje pouze do délky zadaného řetězce

NCA - Řetězec pro komparaci odpovědi modemu "NO CARRIER", odpověď se komparuje pouze do délky zadaného řetězce

ESC - Řetězec pro SW přechod do Cmd režimu modemu bez zavěšení, tzv. únikový kód "+++"

HUP - Řetězec pro závěr (položení sluchátka) modemu, např. "ATH"

REP - Max. počet opakovaní inicializace a vytáčení při navazování spojení

HWI - Flag použití HW inicializace (DTR: 1-->0-->1) (1= HW inicializace použita)

HWH - Flag použití HW závěru (DTR: 1-->0-->1) (RTS: 1-->0-->1) (1=HW závěr použit)

MAS - 1=Master / 0=Slave , Master volá, Slave čeká na zavolání

DRC - Flag zákazu automatického obnovení spojení po výpadku DCD (Data Carrier Detect)
(0=obnovení povoleno, 1=obnovení zakázáno)
Je-li obnovení spojení povoleno, pak modemový komunikační kanál se jako Master snaží obnovit spojení. Po dobu obnovování spojení není modemový komunikační kanál pro nadřízený kanál READY, tj. nadřízený kanál čeká buď na CHReceiveReady=CHS_ReceiveReady nebo CHSendReady=CHS_SendReady.
Je-li obnovení spojení zakázáno, dojde po výpadku DCD k zavěšení modemu. Obnovení spojení je v takovém případě na nadřízeném komunikačním kanálu.
S ohledem na volbu obnovování je třeba nastavit příslušné časovky (timeouts) v nadřízeném komunikačním kanálu. Je-li automatické obnovení spojení povoleno, pak časovky v nadřízeném kanálu musí být delší než předpokládaná doba obnovení spojení. V takovém případě v nadřízeném kanálu dojde po obnovení spojení k uplynutí časovky, neboť zpravidla nepřišla odpověď od Slave, proto Master zopakuje předchozí dotaz, na tento dotaz Slave nyní po obnoveném spojení odpoví, a tak komunikace mezi Master a Slave může pokračovat.

Parametry lze prostřednictvím zadání v INI-souboru programu nastavit na požadované hodnoty. Není třeba nastavovat všechny parametry, stačí nastavit pouze takové, u kterých je třeba zadat hodnotu různou od přednastavené (default) hodnoty.

Algoritmus obsluhy modemu v modemovém komunikačním kanálu je navržen tak, aby mohl být použit pro různé typy modemů, které se liší v detailech nastavení pomocí jejich AT-příkazů a v nastavení tzv. S-registrů. Proto veškeré parametry algoritmu jsou zadávány z INI-souboru programu. Parametry algoritmu jsou řetězce AT-příkazů, časy prodlev a časovek (timeouts), řetězce odpovědí modemu pro komparaci a další parametry ovlivňující činnost algoritmu.

Knihovna LnkSofMA má implementováno okénko tzv. ladících výpisů. Do okénka ladících výpisů lze nechat vypisovat různé kategorie systémových zpráv. Kategorie vypisovaných zpráv se zadávají v dialogu "Debug Mode", zde lze označit dvě kategorie vztahující se k obsluze modemu:

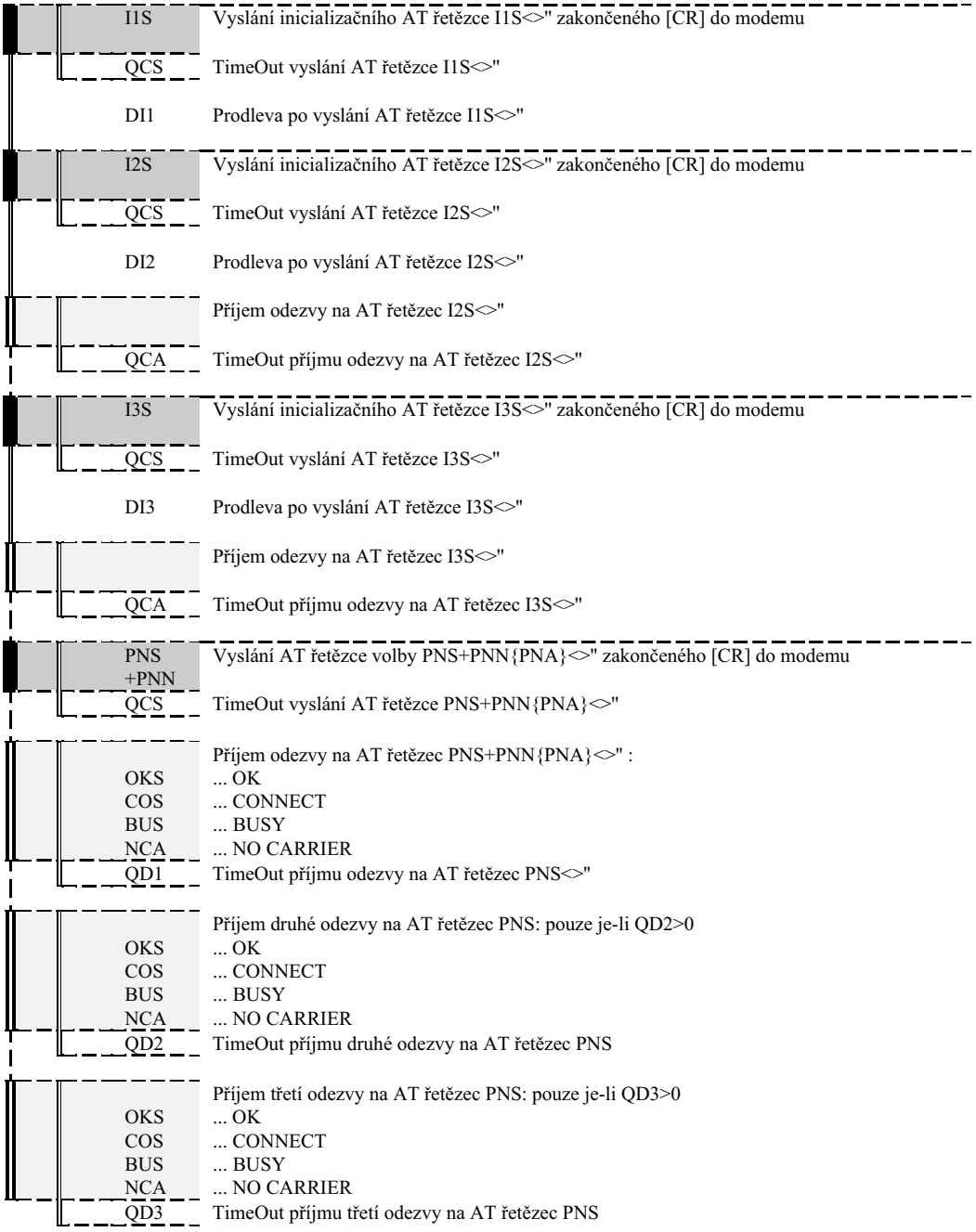
DM_Modem - vypisovány budou kroky automatu pro navazování a rušení spojení modemem

DM_ModemSMC - vypisovány budou podrobně zprávy posílané do modemu a zprávy přijaté od modemu

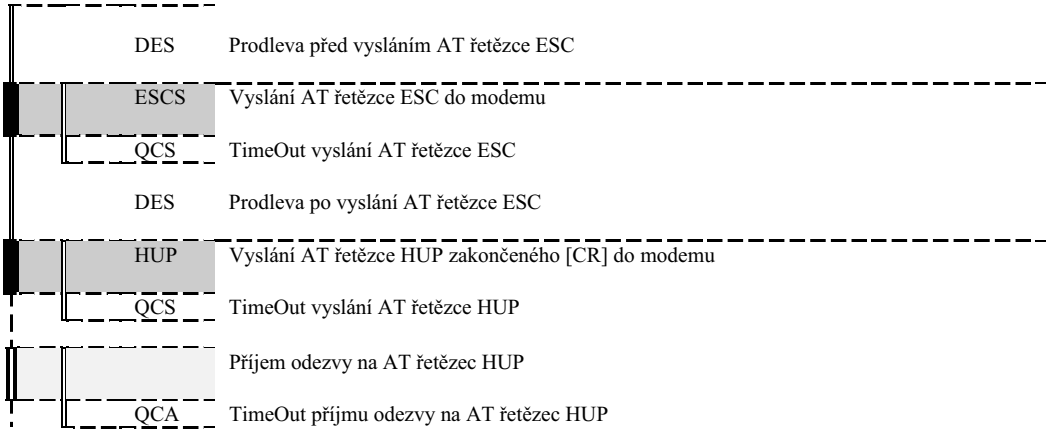
Write to *.TXS - veškeré výpisy do ladícího okénka budou zapisovány také do textového souboru s příponou ".TXS", kde si je můžeme později prohlédnout

Prostřednictvím těchto výpisů získáme dostatečné informace o komunikaci s modemem. Tyto informace využijeme při hledání optimálního nastavení nově připojovaného typu modemu bez nutnosti použít jiný program (např. Terminal z Windows).

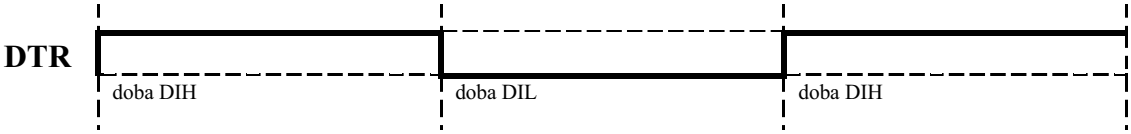
SW inicializace modemu



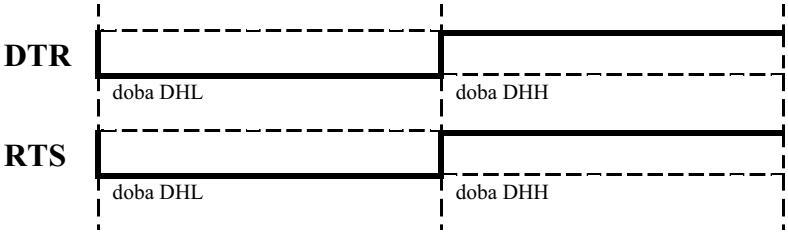
SW zavěšení modemu



HW inicializace modemu



HW zavěšení modemu

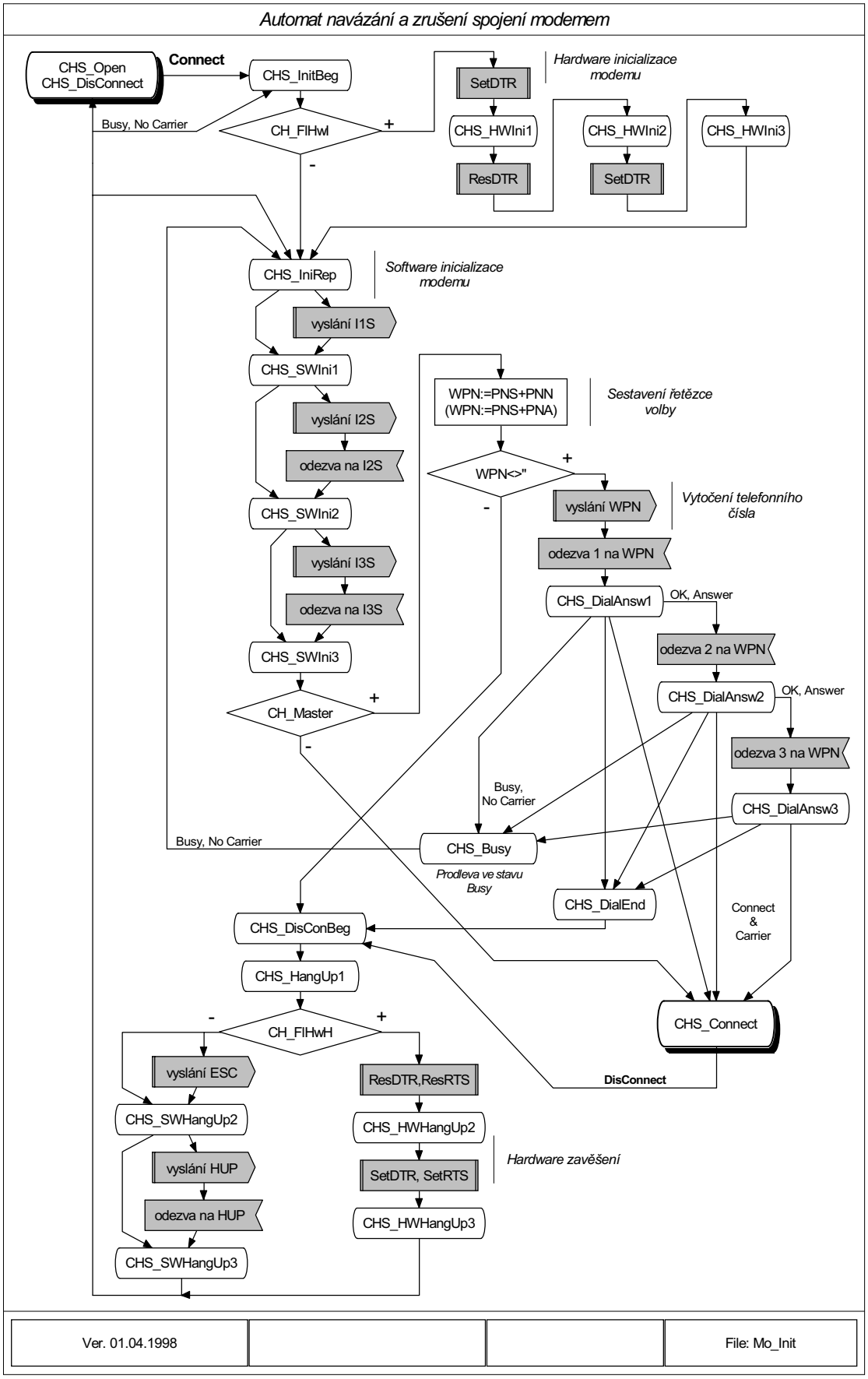


Poznámka k použití signálu DTR při nastavení CTR=1:

Je-li nastaven parametr CTR=1, je signál DTR nastavován na hodnotu 1 až ve stavu Connect. Zároveň po zavěšení modemu, není signál DTR navracen na hodnotu 1 (viz. HW zavěšení modemu - doba DHH). Po nastavení signálu DTR do hodnoty 1 je vždy před další obsluhou modemu vyčkáno dobu DIH (viz. HW inicializace modemu), aniž je nutno nastavit flag HW inicializace.

Je-li nastaven parametr CTR=1, pak se ze stavu CHS_Busy přechází dále přes zavěšení modemu.

Výše uvedená obsluha signálu DTR je určena pro GSM modem Siemens M1, kde je prostřednictvím signálu DTR=1 ovládáno připojení napětí na vstup IGNITION modemu (viz. dokumentace k modemu). Po připojení napětí je proto třeba s obsluhou modemu vyčkat, než dojde k jeho inicializaci. Po odpojení napětí ze vstupu IGNITION dojde ke zrušení hovoru. Doba odpojení musí být minimálně 550ms. Pokud není automat ve stavu Connect, je DTR=0 a GSM modem je vypnut.


Automat navázání a zrušení spojení modemem
CHS_Open
CHS_DisConnect
Connect
CHS_InitBeg
Busy, No Carrier
CH_FlHwI
+
-
SetDTR
Hardware inicializace modemu
CHS_HWIni1
ResDTR
CHS_HWIni2
SetDTR
CHS_HWIni3
CHS_IniRep
Software inicializace modemu
vyslání I1S
CHS_SWIni1
vyslání I2S
odezva na I2S
CHS_SWIni2
vyslání I3S
odezva na I3S
CHS_SWIni3
CH_Master
WPN:=PNS+PNN
(WPN:=PNS+PNA)
Sestavení řetězce volby
WPN<>"
vyslání WPN
Vytočení telefonního čísla
odezva 1 na WPN
CHS_DialAnsw1
OK, Answer
odezva 2 na WPN
CHS_DialAnsw2
OK, Answer
odezva 3 na WPN
CHS_DialAnsw3
Busy,
No Carrier
Busy, No Carrier
CHS_Busy
Prodleva ve stavu Busy
CHS_DialEnd
Connect
&
Carrier
CHS_DisConBeg
CHS_HangUp1
CH_FlHwH
CHS_Connect
DisConnect
vyslání ESC
CHS_SWHangUp2
vyslání HUP
odezva na HUP
CHS_SWHangUp3
ResDTR,ResRTS
CHS_HWHangUp2
SetDTR, SetRTS
CHS_HWHangUp3
Hardware zavěšení
Ver. 01.04.1998
File: Mo_Init

4. Komunikace na Kit-Builder

V systému Kit-Builder jsou implementovány dva procesy, se kterými lze komunikovat. Prvním komunikujícím procesem je proces OnLine sledování hodnot systémových registrů, druhým komunikujícím procesem je proces obsluhy datových bank, tj. bitmap, fontů, programů a archivů. Každý proces má přidělenu vlastní systémovou adresu.

Systémová adresa procesu OnLine sledování registrů = [**86**,01,4001]

Systémová adresa procesu obsluhy datových bank = [**80**,02,4001]

kde jednotlivé položky mají význam [<typ_procesu>,<instance>,<logická_adresa>]

Číslo instance a logická adresa v systémové adrese se mohou dle konkrétní instalace měnit, hodnota typu procesu (tj. $86 pro OnLine a $80 pro obsluhu bank) je konstantou.

4.1 OnLine sledování hodnot registrů

V systému Kit-Builder jsou implementovány dvě sady systémových registrů - sada celočíselných registrů a sada registrů typu real. Komunikační rozhraní procesu OnLine sledování nabízí služby blokového čtení a blokového zápisu hodnot registrů.

Datová část zprávy protokolu SofCon Level 2 za hlavičkou zprávy typu TProcMessHeader začíná položkami, které definují přenášený blok dat. Přenášeným blokem dat je v našem případě blok registrů systému Kit-Builder.

Položky identifikující blok registrů jsou deklarovány následovně:

Deklarace

```
const { MCODE používané v hlavičce zprávy na Kit-Builder,
        tj. (TProcMessHeader.MCode) }
  gcmd_Result     = $03; { Tgcmd_Result
                           Výsledek operace (0=O.K., >0=chyba) }
  gcmd_GetParamVal= $40; { T???_GetParamVal SLV <-  MAS
                           žádost o parametr/parametry }
  gcmd_PutParamVal= $41; { T???_ParamValue  SLV <-> MAS
                           zápis parametrů/parametru   }


const
  g_nevimco       = 1000;{ proměnná horní mez pro dynamické
                           pole parametrů }
  g_ResultSzMaxIndx=64;  { max. délka textu chybového hlášení
                           ve zprávě    }

type
  { Typ datové části potvrzovací zprávy, kterou SLAVE
    potvrzuje (ResultCode=0)bezchybné akceptování, nebo
    hlásí chybu(ResultCode<>0). V položce ResultSz může
    SLAVE předat text chybového hlášení }

  Tgcmd_Result = record { zpráva může být proměnné délky
                          dle ResultSz }
    ResultCode :word;   { 0=OK, ostatní kód chyby }
    ResultSz   :array[0..g_ResultSzMaxIndx] of char;
               { null terminated str }
  end;
```

```
type
  TDscrMsk_Type    =byte;
const  { Identifikace typů registrů }
  DscrMsk_byte     =1;{ parametr = byte    nebo pole bytu     }
  DscrMsk_word     =2;{ parametr = word    nebo pole wordu    }
  DscrMsk_integer  =3;{ parametr = integer nebo pole integeru }
  DscrMsk_longint  =4;{ parametr = longint nebo pole longintu }
  DscrMsk_dword    =5;{ parametr = dword   nebo pole dwordu   }
  DscrMsk_string   =6;{ parametr = pascalský string }
  DscrMsk_real     =7;{ parametr = pascalský 6-ti byte real  }
  DscrMsk_DosDaTi  =8;{ parametr = longint chápaný jako MS-DOS PackTime }
  DscrMsk_Bit      =9;{ parametr = byte chápaný po bitech 0..7 }


type
  {--------------------------------------------------------------------}
  {--------- Adresace bloku registrů v Kit-Builderu -------------------}
  {--------------------------------------------------------------------}
  TKbd_BlockHeader = record { identifikace bloku registrů }
    TREC   :TDscrMsk_Type;{ typ Kit-Builder registru }
    RADDR  :word;         { poč. adresa bloku registrů v Kit-Builderu }
    RCNT   :word;         { počet registrů v bloku ~ v poli registrů   }
  end;

  { Typ datové části zprávy požadavku na poslání bloku registrů,
    tj. MCode=gcmd_GetParamVal }
  TKbd_GetParamVal = record { požadavek na posláni bloku
                              registrů z Kit-Builderu }
    RqBlockHd  :TKbd_BlockHeader; { požadovaný blok registrů }
  end;

  { Typ datové části zprávy zapisující blok registrů,
    tj. MCode=gcmd_PutParamVal }
  TKbd_PutParamVal = record { posílaný blok registrů do/z Kit-Builderu }
    BlockHd  :TKbd_BlockHeader;{ identifikace posílaného bloku registrů }
    RegBlock :array[0..g_nevimco] of byte; { posílaný blok registrů }
  end;
  {--------------------------------------------------------------------}
```

Systém Kit-Builder odpovídá na požadavek gcmd_GetParamVal zprávou gcmd_PutParamVal s blokem dat, nebo zprávou gcmd_Result s kódem chyby.

Systém Kit-Builder odpovídá na zápis gcmd_PutParamVal zprávou gcmd_Result s kódem 0 při bezchybném zápisu, nebo s kódem různým od nuly při zápisu s chybou.

Pro úplnost uvedeme deklarace typů celých zpráv posílaných na Kit-Builder a přijímaných zpět tak, jak jsou tyto typy zpráv používány pro definování zpráv předávaných knihovně LnkSofMA.DLL.

Deklarace

```
type
  TGetParamValMess = record
    HED :TProcMessHeader; { MCode=gcmd_GetParamVal }
    REC :TKbd_GetParamVal;
  end;


type
  TPutParamValMess = record
    HED :TProcMessHeader; { MCode=gcmd_PutParamVal }
    REC :TKbd_PutParamVal;{ délka dle skutečné délky bloku }
  end;
```

Skutečná délka zprávy (TProcMessHeader.BuffSize) s blokem registrů je odvozena od skutečné délky tohoto bloku.

```
type
  TResultMess     = record
    HED :TProcMessHeader; { MCode=gcmd_Result }
    RES :Tgcmd_Result;
  end;
```

4.2 Obsluha datových bank

Vzhledem k rozsahu popisu komunikačního rozhraní obsluhy datových bank je tomuto popisu věnován samostaný dokument.

5. Přílohy

5.1 Programové rozhraní pro Borland Delphi V4.0

V příloze je zdrojový tvar jednotky, která implementuje programové rozhraní na knihovnu LnkSofMA.DLL. Vzhledem k tomu, že knihovna LnkSofMA.DLL byla implementována právě v Delphi V4.0, je k dispozici také testovací aplikace, která ukazuje možnosti použití knihovny při komunikaci se systémem Kit-Builder.

5.2 Programové rozhraní pro MS Visual C++ V6.0

V příloze jsou zdrojové texty, které implementují programové rozhraní na knihovnu LnkSofMA.DLL pro jazyk C++.

V jazyce C++ lze používat knihovnu LnkSofMA.DLL bez omezení.

5.3 Programové rozhraní pro MS Visual Basic

V příloze jsou zdrojové texty, které implementují programové rozhraní na knihovnu LnkSofMA.DLL v prostředí MS Visual Basic a MS Access. Zdrojové texty lze použít také v ostatních aplikacích, které používají MS Visual Basic for Application.

Příloha obsahuje také základní příklady použití a testovací programy.

V některých aplikacích nelze používat příjem zprávy prostřednictvím CallBack procedury.

5.4 Programové rozhraní pro 602Text

Textový editor 602Text z programového balíku 602Pro PC SUITE dovoluje ze svého macro jazyka, kterým je Pascal, volat procedury a funkce z externí knihovny DLL. V příloze je uveden textový tvar macra, které vytvoří z nakomunikovaných hodnot soubor s výpisem obsahu registrů systému Kit-Builder.

Pomocí vlastních macro příkazů můžete vytvořit různé soubory protokolů, které budou obsahovat aktuálně nakomunikované hodnoty z Vašeho řídicího systému.

V macro jazyku 602Text nelze používat příjem zprávy prostřednictvím CallBack procedury.

5.5 Programové rozhraní pro Váš program

Knihovnu LnkSofMA.DLL lze použít v programech, které dovolují volat služby 32-bitové knihovny DLL. Rozhraní knihovny LnkSofMA.DLL je voleno tak, aby používalo pouze základní jednoduché datové typy a typ null terminated string. Knihovna dále obsahuje pomocné procedury a funkce, které Vám pomohou překonat některá implementační omezení Vašeho programu.

6. Technická podpora

Technická podpora programu

Adresa sídla společnosti:

SofCon s.r.o.
Fr. Zvonaře 636
272 04 KLADNO 4

Adresa kanceláří společnosti:

SofCon s.r.o.
Střešovická 49
162 00 PRAHA 6

Telefony:

(02) 20610 348 ing. Karel Podaný (ředitel)
(02) 20180 452 ing. Radomír Bukovský (SW na PC)
(02) 20180 562 ing. Vladimír Kastner(harware)
(02) 20180 453 ing. Petr Tesař (ekonomika)
(02) 20180 454 ing.Jan Koďous (harware)

Tel/Fax: (02) 20180 454

Internet:
E-Mail: sofcon@sofcon.cz
WWW: http://www.sofcon.cz