microPEL

PES lib

(C + PASCAL)
KNIHOVNY
KOMUNIKAČNÍCH FUNKCÍ

03/2000

PESlib
Popis knihoven PASCAL a C
03.2000
2. verze dokumentu

Změny a doplňky
proti 1. verzi dokumentu :
pouze reedice, změny žádné

http://www.micropel.cz

1. ÚVOD

Knihovny jsou řešeny jako samostatné bloky pro začlenění do programů psaných v jazycích PASCAL a C. Jako výchozí prostředí byly zvoleny překladače fy Borland - TURBO PASCAL nebo BORLAND C, C++ . Kromě již přeložených souborů (*.TPU nebo *.OBJ) jsou součástí dodávky i úplné zdrojové texty (*.PAS a *.CPP).

Pro dokonalé pochopení funkce a významu jednotlivých funkcí v knihovnách je třeba vědět ještě základní minimum o funkci síťového protokolu PESNET, používaném automaty PES (viz dále).

Knihovny obsahují všechny základní nástroje pro komunikaci se sítí automatů PES (s protokolem PESNET verze 3.xx) přes standardní PC sériový port COM1 - COM4. Komunikační procedury pracují tak, aby bylo možné pro komunikaci využívat převodník RS323 -> RS485 typu PES-CA1 i bez externího napájení.

1.1. Adresy proměnných

U funkcí, které čtou nebo zapisují data do programových registrů (ReadSTPBit, WriteSTPBit, ReadSTPWord, WriteSTPWord) je třeba patřičnou proměnnou vždy zadat její absolutní adresou.

U proměnných umísťovaných automaticky překladačem SIMPLE (s využitím "otazníkového" zápisu při definici symbolických názvů proměnných) je třeba jejich skutečnou adresu zjistit z příslušného *.DNL souboru (soubor vzniklý překladem zdrojového textu). Tyto adresy se však např. po zadefinování dalších proměnných mohou po dalším překladu posunout a v takovém případě je nutné opětovně zkontrolovat, zda nedošlo v adresách ke změně. Z uvedeného vyplývá, že používání automatického přiřazování proměnných pomocí otazníku je pro spolupráci s knihovnami PESLIB nevhodné a jen komplikuje situaci.

Seznam absolutních adres všech proměnných používaných jazykem SIMPLE:

proměnné	typ	popis	absolutní adresa
X0 - X31	bit	digitální vstupy	BIT 0 - 31
Y0 - Y31	bit	digitální výstupy	BIT 32 - 63
M0 - M127	bit	uživatelské bity	BIT 64 - 191
B0 - B127	bit	speciální funkční bity	BIT 192 - 319
I0 - I31	word	analogové vstupy	WORD 0 - 31
O0 - O31	word	analogové výstupy	WORD 32 - 63
D0 - D63	word	uživatelské registry	WORD 64 - 127
W0 - W127	word	speciální funkční registry	WORD 128 - 255

1.2. Stručný popis funkce sítě PESNET V3.xx

PESNET je síť typu "multi-master, token-passing". Jinak řečeno : všechny stanice v síti jsou na stejné hierarchické úrovni, všechny mají oprávnění k vysílání dat a pokyn k vysílání (tzv. TOKEN) si navzájem předávají v logickém kruhu. I když stanice nemají žádná data k vysílání, síť běží "naprázdno" - stanice si neustále dokola předávají token.

Pokud dojde k narušení komunikace na síti (např. zkratem na vedení, vnucením log. úrovně nula na lince, nebo vlivem extrémně silného rušení), přeruší se provoz na síti a stanice začnou časovat tzv. "mrtvý čas", po jehož uplynutí se opět pokusí nastartovat provoz na síti. Tento čas je mimo jiné závislý i na adrese stanice (aby se zabránilo kolizím při restartu). Stanice která první dočasuje mrtvý čas do konce zahájí vysílání a začne zkoušet předání token na všechny síťové adresy dokud nedostane kladnou odpověď. Tento mechanismus funguje rovněž vždy po zapnutí stanic a zabezpečuje tak spolehlivý start sítě. Maximální mořná délka mrtvého času (závisí též na adresách použitých v síti) je asi 5 sekund. Pokud je právě zapnuta další stanice na již aktivní síti, trvá její zařazení do logického kruhu o něco déle (zpravidla do 20 sekund - závisí i na použité baudové rychlosti) - každá stanice totiž neustále prohledává volný prostor adres před sebou a toto prohledávání se nemůže dít příliš často, aby se zbytečně nesnižovala propustnost sítě.

Přenos dat po síti probíhá po malých kvantech - tzv. "rámcích". Každý rámec obsahuje adresu cílové stanice (lze použít i speciální globální adresu - pak je rámec určen všem stanicím v síti) a kontrolní součet. Pokud stanice detekuje při příjmu rámce chybu kontrolního součtu, ignoruje celý tento rámec.

Při používání knihoven PESNET je třeba mít na paměti logický fakt, že stanice při příjmu rámce s globální adresou nedává žádnou odezvu - nelze tedy nastavit globální adresu a pak vyčítat data (v tomto případě interface hlásí chybu "RX TIMEOUT").

1.3. Používání knihoven pro PESNET3

Na lince neustále běží předávání TOKEN v logickém kruhu mezi stanicemi. Na vyšších rychlostech již standardní počítač PC se svým COM portem není schopen se začlenit do logického kruhu aniž by jej zdržoval, v horším případě dokonce úplně zrušil. Komunikace s automaty proto probíhá v dávkovém režimu. Během přenosu dat mezi PC a stanicemi je logický kruh zrušen, předávání TOKEN je zastaveno a PC pracuje jako master. Po přenesení potřebných dat je vhodné nastartovat opět předávání TOKEN (pomocí "SendToken") a nechat chvíli prostor pro přenos globálních síťových proměnných mezi automaty. Délku časových prodlev zvolíme v závislosti na množství využívaných síťových proměnných a na množství stanic v síti (maximální doba vysílání jedné stanice je asi 15 ms při rychlosti 57600 Bd, tedy při 30 stanicích v síti může trvat jedna obrátka logického kruhu maximálně asi 450 ms). Na disketě jsou i příklady (EXAMPLES), které ukazují typické použití funkcí modulů PESNET3.

1.4. Detekce chyb

Po zavolání kterékoliv knihovní funkce se vždy nastaví příznaková proměnná Success. Pokud je nastavena (PASCAL : boolean = true, nebo C++ : unsigned char = 1), akce proběhla úspěšně a pokud není (PASCAL - false, C++ - 0), znamená to, že došlo k chybě a v proměnné COM_Error je kód chyby.

Chyby jsou definovány jako symbolické konstanty takto :

Symbol. název	Kód chyby	Popis
ERR_NONE	00	žádná chyba
Chyby vzniklé a detekované na straně PC		
ERR_INITCOM	01	chyba procedury COM_Init při inicializaci COM portu
ERR_TIMEOUT	02	obecná chyba překročení časového limitu na akci
ERR_TIMEOUTTX	03	příliš dlouhé čekání na uvolnění vysílače
ERR_TIMEOUTRX	04	příliš dlouhé čekání na příjem znaku z linky
ERR_RXSUM	05	chyba kontrolního součtu přijatého rámce
ERR_DATASIZE	06	nesouhlasí délka datového pole při příjmu rámce
ERR_ADDRESS	07	adresa stanice mimo povolený rozsah
ERR_DEBUGFLAG	08	interní chybový kód
ERR_MOREBAUD	09	detekovány stanice s různou komunikační rychlostí
ERR_NETEMPTY	10	žádná aktivní stanice v síti
Chyby vzniklé a detekované v připojeném automatu		
ERR_BADERROR	128	neznámý chybový kód ze subprocesu
ERR_BADCMD	129	neznámý příkaz v rámci
ERR_TARGETSUM	130	chybný kontrolní součet v přijatém rámci
ERR_BADFRAME	131	neznámý typ rámce
ERR_SELFRX	132	chyba při zpětné kontrole vlastního vysílání
ERR_IOFULL	133	plný vstupní buffer při operaci CmIO - ztráta dat
ERR_NODATA	134	ve výstupním bufferu nejsou data (operace CmIO)

2. KNIHOVNA PASCAL - MODUL PESNET3.PAS

Rozhraní modulu obsahuje jednak proměnné, jednak procedury a funkce.

Zde je stručný přehled:

Proměnné

BaudRate	: longint	baudová rychlost
ComPort	: byte	číslo COM portu, použitého pro komunikaci
COM_Error	: byte	kód chyby vzniklé při poslední akci
Success	: boolean	úspěšnost poslední akce (true=OK, false=ERROR)
StationNum	: byte	počet stanic v síti detekovaný procedurou "NetScan"
StationList	: array[0..31] of byte	seznam adres nalezených v síti pomocí "NetScan"

Procedury a funkce

```
procedure   COM_Init;
procedure   COM_Close;
 function   BaudScan:longint;
procedure   NetScan;
procedure   SelectStation(StAddr : byte);
procedure   NetBreak;
procedure   SendToken;
 function   ReadSTPWord(Addr:word):word;
 function   ReadSTPBit(Addr:word):boolean;
procedure   WriteSTPWord(Addr:word; Dato:word);
procedure   WriteSTPBit(Addr:word; Bit:boolean);
 function   ReadStack(var Buffer; Position : word; Count : word) : word;
 function   WriteStack(var Buffer; Position : word; Count : word) : word;
 function   GetErrorString : string;
```

procedure COM_Init ;

Procedura, kterou je třeba volat vždy jako úplně první - inicializuje příslušný COM port. Před voláním této procedury je nutno uložit do proměnné ComPort číslo portu s kterým budeme pracovat (1..4) a do proměnné BaudRate příslušnou baudovou rychlost (jsou povoleny hodnoty 2400, 9600, 19200, nebo 57600). Hodnota BaudRate není povinná, pro její zjištění lze použít např. proceduru BaudScan - viz dále.

procedure COM_Close ;

Tuto proceduru je třeba volat vždy před ukončením programu, slouží ke korektnímu uvedení COM portu do klidového stavu.

function BaudScan : longint ;

Funkce nejprve přeruší síťový provoz na lince (zavoláním procedury NetBreak) a poté vyzkouší komunikaci na všech podporovaných rychlostech. Pokud nalezne v síti pouze jedinou komunikační rychlost, vrátí ji jako výsledek a nastaví Success=true. Pokud nalezne stanice komunikující na různých rychlostech, nebo naopak nenalezne vůbec žádné stanice, vrací nulu, nastaví Success=false a v proměnné COM_Error je detailní kód chyby.

V případě, že funkce byla úspěšná, je zároveň do proměnné BaudRate uložena detekovaná baudová rychlost. Chceme-li se tedy připojit k síti automatů a nevíme jakou mají nastavenou baudovou rychlost, stačí použít tuto posloupnost příkazů :

```
ComPort := CISLO_COM_PORTU ;
COM_Init ;
if not Success then writeln('CHYBA INICIALIZACE COM PORTU') ;
BaudScan ;
if not Success then begin
  if COM_Error=ERR_NETEMPTY then writeln('ZADNA STANICE V SITI');
  if COM_Error=ERR_MOREBAUD then writeln('RUZNE BD-RYCHLOSTI V SITI !') ;
end ;
COM_Init ;
```

Poslední volání COM_Init je proto, aby se port nastavil na nalezenou baudovou rychlost.

procedure NetScan ;

Procedura nejprve přeruší provoz na síti zavoláním procedury NetBreak a poté provede komplexní průzkum sítě. Pokud nenalezne žádné stanice, nebo pokud nalezne nekorektnosti v baudových rychlostech na síti, nastaví Success=false a příslušný kód chyby do COM_Error. Pokud je vše v pořádku, nastaví do proměnné BaudRate nalezenou baudovou rychlost a do StationNum počet nalezených aktivních stanic na síti. Dále do pole StationList (počínaje prvkem 0) vypíše adresy nalezených stanic.

procedure SelectStation(StAddr : byte) ;

Používá se k nastavení adresy stanice s kterou bude vedena komunikace. Tato adresa je platná až do další změny. Adresy 0 až 30 jsou lokální, adresa 31 je tzv. globální adresa. Vzhledem k tomu, že služby modulu PESNET3.PAS jsou určeny pro komunikaci vždy s vybranou jednou stanicí, doporučujeme globální adresu vůbec nevyužívat.

procedure NetBreak ;

Slouží k přerušení síťového provozu na lince. Používá se vždy před prováděním akcí se sítí automatů. Pokud mezi jednotlivými akcemi není prodleva delší než 0.5 sec., není nutné volat NetBreak před každou akcí, ale jen na začátku vykonávané sekvence.

procedure SendToken ;

Vyšle TOKEN na zvolenou síťovou adresu (nastavenou pomocí procedury SelectStation). Používá se vždy na závěr sekvence akcí pro urychlení startu sítě. Během komunikace mezi PC a stanicemi totiž nefunguje globální síťový přenos dat používaný ve stanicích (protože před započetím komunikace musí být volán NetBreak). Po ukončení komunikace mezi PC a automaty totiž ještě běží časování "mrtvého času" a potom teprve automaty startují síťový provoz. Tento krátký výpadek může být v některých aplikacích fatální. Proto je vhodné po ukončení komunikační sekvence zavolat proceduru SendToken (před tím nastavit pomocí SelectStation některou existující adresu) - procedura zajistí vyslání TOKEN na danou stanici a tím okamžitý restart síťové komunikace v logickém kruhu aby se mohly předávat sdílené proměnné mezi automaty.

function ReadSTPWord (Addr:word) : word ;

Přečte datovou proměnnou z daného automatu (adresa automatu se nastavuje pomocí SelectStation) a vrátí ji jako typ word. Adresa proměnné je dána parametrem Addr, viz stať Adresy proměnných - tedy adresy proměnných typu WORD.

function ReadSTPBit (Addr:word) : boolean ;

Přečte bitovou proměnnou z daného automatu (adresa automatu se nastavuje pomocí SelectStation) a vrátí ji jako typ boolean. Adresa bitu je dána parametrem Addr, viz stať Adresy proměnných - tedy adresy proměnných typu BIT.

procedure WriteSTPWord (Addr:word; Dato:word) ;

Zapíše datovou proměnnou do daného automatu (adresa automatu se nastavuje pomocí SelectStation). Adresa proměnné je dána parametrem Addr, viz stať Adresy proměnných - tedy adresy proměnných typu WORD. Zapisovaná hodnota je dána parametrem Dato.

procedure WriteSTPBit (Addr:word; Bit:boolean) ;

Zapíše bitovou proměnnou do daného automatu (adresa automatu se nastavuje pomocí SelectStation). Adresa bitu je dána parametrem Addr, viz stať Adresy proměnných - tedy adresy proměnných typu BIT. Zapisovaná hodnota je dána parametrem Bit.

function ReadStack (var Buffer; Position : word; Count : word) : word;

Slouží k vyčítání hodnot z uživatelského zásobníku (pro tento účel totiž nelze použít spec. registry POINTER a STACK, neboť ty mohou být právě využívány programem běžícím v automatu a mohlo by dojít ke kolizím). Funkce ReadStack vyčte ze zásobníku blok dat (délka bloku je dána parametrem Count) a uloží je do pole Buffer. Jako Buffer se předpokládá pole prvků typu word (pole dimenzujeme tak, aby pojmulo počet prvků Count). Parametr Position určuje adresu v zásobníku, od které se začne vyčítat blok dat. Návratová hodnota funkce typu word udává, kolik položek se do bufferu skutečně vyčetlo.

function WriteStack (var Buffer; Position : word; Count : word) : word ;

Procedura WriteStack zapíše do zásobníku blok dat (délka bloku je dána parametrem Count). Data bere z pole Buffer. Jako Buffer se předpokládá pole prvků typu word (pole dimenzujeme tak, aby pojmulo počet prvků Count). Position udává adresu v zásobníku, odkud se začne ukládat blok dat. Návratová hodnota funkce typu word udává, kolik položek se do zásobníku skutečně zapsalo.

POZN.:

Funkce provádějící operace se zásobníkem si automaticky kontrolují důležité systémové proměnné z automatu, mimo jiné i informace o případném přemístění zásobníku. U automatů s rozšířenou pamětí 32kB je totiž možné dynamicky měnit pozici a velikost zásobníku v paměti (provádí se automaticky při downloadu programu do automatu). Funkce se tedy automaticky přizpůsobují konfiguraci paměti toho automatu, se kterým se zrovna komunikuje. Funkce kontrolují i velikost paměti vyhrazené pro zásobník v tom kterém automatu a svojí návratovou hodnotou mimo jiné informují i o tom, zda již náhodou nebylo dosaženo horní meze zásobníku.

function GetErrorString : string ;

Procedura vrátí slovní popis poslední chyby.

3. KNIHOVNA C - MODUL PESNET3.CPP

Rozhraní modulu je zadefinováno v hlavičkovém souboru PESNET3.H a obsahuje jednak proměnné, jednak funkce.

Zde je stručný přehled:

Proměnné

unsigned long	BaudRate	baudová rychlost
unsigned char	ComPort	číslo COM portu, použitého pro komunikaci
unsigned char	COM_Error	kód chyby vzniklé při poslední akci
unsigned char	Success	úspěšnost poslední akce (1=OK, 0=ERROR)
unsigned char	StationNum	počet stanic v síti detekovaný funkcí "NetScan"
unsigned char	StationList[32]	seznam adres nalezených v síti funkcí "NetScan"

Funkce

```
          void   COM_Init();
          void   COM_Close();
unsigned long    BaudScan();
          void   NetScan();
          void   SelectStation(unsigned char StAddr);
          void   NetBreak();
          void   SendToken();
  unsigned int   ReadSTPWord(unsigned int Addr);
unsigned char    ReadSTPBit(unsigned int Addr);
          void   WriteSTPWord(unsigned int Addr, unsigned int Dato);
          void   WriteSTPBit(unsigned int Addr, unsigned char Bit);
  unsigned int   ReadStack(void *Buffer, unsigned int Position, unsigned int Count);
  unsigned int   WriteStack(void *Buffer, unsigned int Position, unsigned int Count);
          char   *GetErrorString(char *ErrString);
```

void COM_Init() ;

Procedura, kterou je třeba volat vždy jako úplně první - inicializuje příslušný COM port. Před voláním této procedury je nutno uložit do proměnné ComPort číslo portu s kterým budeme pracovat (1..4) a do proměnné BaudRate příslušnou baudovou rychlost (jsou povoleny hodnoty 2400, 9600, 19200, nebo 57600). Hodnota BaudRate není povinná, pro její zjištění lze použít např. proceduru BaudScan - viz dále.

void COM_Close() ;

Tuto proceduru je třeba volat vždy před ukončením programu, slouží ke korektnímu uvedení COM portu do klidového stavu.

unsigned long BaudScan() ;

Funkce nejprve přeruší síťový provoz na lince (zavoláním procedury NetBreak) a poté vyzkouší komunikaci na všech podporovaných rychlostech. Pokud nalezne v síti pouze jedinou komunikační rychlost, vrátí ji jako výsledek a nastaví Success=1. Pokud nalezne stanice komunikující na různých rychlostech, nebo naopak nenalezne vůbec žádné stanice, vrací nulu, nastaví Success=0 a v proměnné COM_Error je detailní kód chyby.

V případě, že funkce byla úspěšná, je zároveň do proměnné BaudRate uložena detekovaná baudová rychlost. Chceme-li se tedy připojit k síti automatů a nevíme jakou mají nastavenou baudovou rychlost, stačí použít tuto posloupnost příkazů :

```
ComPort = CISLO_COM_PORTU ;
COM_Init() ;
if (! Success) printf("CHYBA INICIALIZACE COM PORTU") ;
BaudScan() ;
if (! Success) {
  if (COM_Error == ERR_NETEMPTY) printf("ZADNA STANICE V SITI") ;
  if (COM_Error == ERR_MOREBAUD) printf("RUZNE BD-RYCHLOSTI V SITI !") ;
}
COM_Init() ;
```

Poslední volání COM_Init je proto, aby se port nastavil na nalezenou baudovou rychlost.

void NetScan() ;

Procedura nejprve přeruší provoz na síti zavoláním procedury NetBreak a poté provede komplexní průzkum sítě. Pokud nenalezne žádné stanice, nebo pokud nalezne nekorektnosti v baudových rychlostech na síti, nastaví Success=0 a příslušný kód chyby do COM_Error. Pokud je vše v pořádku, nastaví do proměnné BaudRate nalezenou baudovou rychlost a do StationNum počet nalezených aktivních stanic na síti. Dále do pole StationList (počínaje prvkem 0) vypíše adresy nalezených stanic.

void SelectStation(unsigned char StAddr) ;

Používá se k nastavení adresy stanice s kterou bude vedena komunikace. Tato adresa je platná až do další změny. Adresy 0 až 30 jsou lokální, adresa 31 je tzv. globální adresa. Vzhledem k tomu, že služby modulu PESNET3.PAS jsou určeny pro komunikaci vždy s vybranou jednou stanicí, doporučujeme globální adresu vůbec nevyužívat.

void NetBreak() ;

Slouží k přerušení síťového provozu na lince. Používá se vždy před prováděním akcí se sítí automatů. Pokud mezi jednotlivými akcemi není prodleva delší než 0.5 sec., není nutné volat NetBreak před každou akcí, ale jen na začátku vykonávané sekvence.

void SendToken() ;

Vyšle TOKEN na zvolenou síťovou adresu (nastavenou pomocí procedury SelectStation). Používá se vždy na závěr sekvence akcí pro urychlení startu sítě. Během komunikace mezi PC a stanicemi totiž nefunguje globální síťový přenos dat používaný ve stanicích (protože před započetím komunikace musí být volán NetBreak). Po ukončení komunikace mezi PC a automaty totiž ještě běží časování "mrtvého času" a potom teprve automaty startují síťový provoz. Tento několikasekundový výpadek může být v některých aplikacích fatální. Proto je vhodné po ukončení komunikační sekvence zavolat proceduru SendToken (a před tím nastavit pomocí SelectStation některou existující adresu) - procedura zajistí vyslání TOKEN na danou stanici a tím okamžitý restart síťové komunikace v logickém kruhu aby se mohlo realizovat předávání sdílených proměnných mezi automaty.

unsigned int ReadSTPWord (unsigned int Addr) ;

Přečte datovou proměnnou z daného automatu (adresa automatu se nastavuje pomocí SelectStation) a vrátí ji jako typ word. Adresa proměnné je dána parametrem Addr, viz stať Adresy proměnných - tedy adresy proměnných typu WORD.

unsigned char ReadSTPBit (unsigned int Addr) ;

Přečte bitovou proměnnou z daného automatu (adresa automatu se nastavuje pomocí SelectStation) a vrátí ji jako typ boolean. Adresa bitu je dána parametrem Addr, viz stať Adresy proměnných - tedy adresy proměnných typu BIT.

void WriteSTPWord (unsigned int Addr, unsigned int Dato) ;

Zapíše datovou proměnnou do daného automatu (adresa automatu se nastavuje pomocí SelectStation). Adresa proměnné je dána parametrem Addr, viz stať Adresy proměnných - tedy adresy proměnných typu WORD. Zapisovaná hodnota je dána parametrem Dato.

void WriteSTPBit (unsigned int Addr, unsigned char Bit) ;

Zapíše bitovou proměnnou do daného automatu (adresa automatu se nastavuje pomocí SelectStation). Adresa bitu je dána parametrem Addr, viz stať Adresy proměnných - tedy adresy proměnných typu BIT. Zapisovaná hodnota je dána parametrem Bit.

unsigned int ReadStack (void *Buffer, unsigned int Position, unsigned int Count) ;

Slouží k vyčítání hodnot z uživatelského zásobníku (pro tento účel totiž nelze použít spec. registry POINTER a STACK, neboť ty mohou být právě využívány programem běžícím v automatu a mohlo by dojít ke kolizím). Funkce ReadStack vyčte ze zásobníku blok dat (délka bloku je dána parametrem Count) a uloží je do pole Buffer. Jako Buffer se předpokládá pole prvků typu word (pole dimenzujeme tak, aby pojmulo počet prvků Count). Parametr Position určuje adresu v zásobníku, od které se začne vyčítat blok dat. Návratová hodnota funkce typu word udává, kolik položek se do bufferu skutečně vyčetlo.

unsigned int WriteStack (void *Buffer, unsigned int Position, unsigned int Count) ;

Funkce WriteStack zapíše do zásobníku blok dat (délka bloku je dána parametrem Count). Data bere z pole Buffer. Jako Buffer se předpokládá pole prvků typu word (pole dimenzujeme tak, aby pojmulo počet prvků Count). Position udává adresu v zásobníku, odkud se začne ukládat blok dat. Návratová hodnota funkce typu word udává, kolik položek se do zásobníku skutečně zapsalo.

POZN.:

Funkce provádějící operace se zásobníkem si automaticky kontrolují důležité systémové proměnné z automatu, mimo jiné i informace o případném přemístění zásobníku. U automatů s rozšířenou pamětí 32kB je totiž možné dynamicky měnit pozici a velikost zásobníku v paměti (provádí se automaticky při downloadu programu do automatu). Funkce se tedy automaticky přizpůsobují konfiguraci paměti toho automatu, se kterým se zrovna komunikuje. Funkce kontrolují i velikost paměti vyhrazené pro zásobník v tom kterém automatu a svojí návratovou hodnotou mimo jiné informují i o tom, zda již náhodou nebylo dosaženo horní meze zásobníku.

char * GetErrorString(char * ErrString) ;

Procedura vrátí slovní popis poslední chyby.