Komunikace pomocí sériové linky RS485 pod Windows 95/98/NT

(obsluha komunikačních karet Tedia PCI585, PCI685 ve 32-bitovém prostředí)

Úvod

Tento dokument je určen vývojovým pracovníkům, kteří jsou postaveni před úkol propojit zařízení komunikující s PC smyčkou RS485 s moderními 32-bitovými operačními systémy Windows 95/98/NT. Jeho úkolem je tento přechod zjednodušit co možná nejvíce. Proto se zde kromě deklarací procedur a funkcí v jazyce C objevují i deklarace a příklady pro Borland Delphi.

Jako hardwarovou platformu pro komunikaci předpokládá komunikační kartu Tedia PCI-685.

Zjištění volných prostředků v PC

Před vlastní instalací karty je nutno ujasnit si, které systémové prostředky je možné kartě přiřadit.

Windows 95/98

Pravým tlačítkem myši klikneme na ikonu Tento počítač na pracovní ploše a vybereme položku Vlastnosti. Zvolíme kartu Správce zařízení, ve stromu jednou klikneme na položku Počítač (první v pořadí) a stiskneme tlačítko Vlastnosti. Zde je třeba vybrat volné IRQ a I/O adresu podle možností nastavení karty. Zatímco IRQ můžeme zvolit takřka libovolně (pozor na IRQ9 – tam je mapováno IRQ2), bude výhodné zvolit I/O adresu odpovídající běžné adrese pro COM3 (tj. 3E8h), popř. pro COM4 (tj. 2E8h). To však není podmínkou.

Hardwarová instalace

Podle návodu nastavíme na kartě následující parametry:

- I/O port a IRQ podle možností PC. Ty jsme zjistili v minulé kapitole.
- Řízení přenosu signálem DTR, resp. RTS, popř. automatické řízení přenosu obvody ADFC (v závislosti na konkrétní aplikaci – viz dále).
- Typ smyčky RS422/485.

Poté zasuneme kartu do PC. Po ubezpečení se, že jsme zákrok provedli korektně, můžeme pustit PC.

Instalace ovladače

Komunikace s kartou PCI-685 bude probíhat pomocí standardního ovladače sériového portu. V dalších kapitolách si ukážeme, jak s tímto ovladačem pracovat a jaké rozdíly se uplatňují při práci s RS485. K tomu je nejprve potřeba sdělit systému, že jsme nainstalovali nový port.

Windows 95/98

Nenajde-li systém Windows po spuštění port automaticky, je třeba přidat jej ručně volbou Přidat nový hardware v Ovládacích panelech, ruční výběr HW, komunikační zásuvka. Změny parametrů (bázová adresa, IRQ) provedeme po instalaci ve Správci zařízení. Poklepeme na nově vytvořený port a na kartě Prostředky nastavíme patřičnou konfiguraci. Základní konfigurace 1..4 odpovídají běžným nastavením pro COM1..COM4. Chceme-li využívat jiné prostředky, je třeba zvolit Základní konfiguraci 8 a prostředky nadefinovat ručně. Na záložce nastavení zásuvky pod tlačítkem Upřesnit zkontrolujeme, že je zapnuto využívání vyrovnávací paměti FIFO. Naši volbu potvrdíme.

Komunikační rychlost nebudeme nastavovat zde, provedeme to uvnitř aplikace.

Základní procedury a funkce pro práci se sériovým portem RS485 v prostředí Win32

Nejprve je třeba podotknout, že si tento dokument neklade v žádném případě za cíl popsat do detailu níže zmiňované funkce. Zde čtenáře musím odkázat na referenční příručky, bez kterých se při hlubším pronikání do problematiky neobejde. Popis zde uvedený se omezuje pouze na základní obsluhu RS485 ve 32-bitovém operačním systému. Pro zkušenější vývojáře bych rád podotknul, že práce s RS485 se pod 32-bitovými Windows liší jen nepatrně od práce s RS232 a dokonce se nemusí lišit vůbec.

Definice procedur jsou uvedeny v jazyce C a Borland Delphi, příklady jsou určeny pro Borland Delphi (3.0 a vyšší).

CreateFile

```
HANDLE CreateFile(
    LPCTSTR  lpFileName,       // ukazatel na jméno souboru
    DWORD  dwDesiredAccess,    // režim přístupu
    DWORD  dwShareMode,        // režim sdílení
    LPSECURITY_ATTRIBUTES  lpSecurityAttributes,     // ukazatel na
popisovač ochrany (security descriptor)
    DWORD  dwCreationDistribution,   // režim vytvoření
    DWORD  dwFlagsAndAttributes,     // atributy souboru
    HANDLE  hTemplateFile      // identifikátor (handle) předlohy
   );
```

```
function CreateFile(lpFileName: PChar; dwDesiredAccess,
dwShareMode: Integer; lpSecurityAttributes: PSecurityAttributes;
dwCreationDisposition, dwFlagsAndAttributes: DWORD; hTemplateFile:
THandle): THandle;
```

Otevírá přístup k sériovému portu. Použití jednotlivých parametrů:

lpFileName – Ukazatel na jméno portu (COM1, COM2, …). Řetězec je ukončen znakem nula (null-terminated string)

dwDesiredAccess – Kombinace konstant GENERIC_READ a GENERIC_WRITE.

dwShareMode – Vložíme hodnotu 0.

lpSecurityAttributes – Vložíme hodnotu NUL, resp. **nil** .

dwCreationDistribution - Vložíme hodnotu OPEN_EXISTING .

dwFlagsAndAttributes - Vložíme hodnotu FILE_ATTRIBUTE_NORMAL + FILE_FLAG_WRITE_THROUGH.

hTemplateFile - Vložíme hodnotu 0.

Vrací identifikátor (handle) na otevřený COM-port nebo hodnotu INVALID_HANDLE_VALUE . Na port se budeme nadále odvolávat pomocí této hodnoty.

Příklad: Otevření portu COM3 pro čtení i zápis– Borland Delphi 3.0 :

```
var Soubor:THandle;
begin
  Soubor:=CreateFile('COM3',GENERIC_READ or GENERIC_WRITE, 0, nil,
  OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL or FILE_FLAG_WRITE_THROUGH,
  0);
```

Přestože funkce má relativně velký počet parametrů, není třeba se všemi obšírně zabývat. Je nutno mít na zřeteli, že tato funkce slouží k otevírání mnoha typů komunikačních kanálů, ať to je soubor, sériový kabel anebo vzdálený počítač v Internetu. Pro naše účely postačí pochopení pouze prvního parametru, ve specifických případech i druhého.

CloseHandle

```
BOOL CloseHandle(
    HANDLE  hObject  // identifikátor (handle) otevřeného portu
 );

function CloseHandle(hObject: THandle): BOOL;
```

Uzavírá přístup k portu, který byl otevřen funkcí CreateFile. Použití jednotlivých parametrů:
hObject – Identifikátor (handle) otevřeného portu získaný z funkce CreateFile.
Vrací TRUE při úspěchu operace, FALSE v opačném případě.

Příklad: uzavření portu – Borland Delphi 3.0 :

```
CloseHandle(Soubor);
```

V touto instrukcí jsme uzavřeli port, který jsme otevřeli v předchozím příkladu.

WriteFile

```
BOOL WriteFile(
    HANDLE  hFile,               // identifikátor (handle) portu
    LPCVOID  lpBuffer,           // adresa dat k odeslání
    DWORD  nNumberOfBytesToWrite,    // počet bytů k odeslání
    LPDWORD  lpNumberOfBytesWritten, // adresa proměnné, do které
se uloží počet odeslaných bitů
    LPOVERLAPPED  lpOverlapped   // ukazatel na strukturu pro
overlapped I/O
   );

function WriteFile(hFile: THandle; const Buffer;
nNumberOfBytesToWrite: DWORD; var lpNumberOfBytesWritten: DWORD;
lpOverlapped: POverlapped): BOOL;
```

Provádí zápis dat na RS485. Použití jednotlivých parametrů:
hFile – Identifikátor (handle) otevřeného portu získaný z funkce CreateFile.
lpBuffer – Ukazatel na data, která mají být odeslána.
nNumberOfBytesToWrite – Počet bytů, které mají být odeslány.
lpNumberOfBytesWritten – Adresa proměnné (v Borland Delphi parametr volaný odkazem), do které bude uložen počet bytů, které byly odeslány.
lpOverlapped – vložíme hodnotu NUL, resp. **nil** .
Vrací TRUE při úspěchu operace, FALSE v opačném případě.

Příklad: zápis řetězce – Borland Delphi 3.0 :

```
function SendString(S:String):Boolean;
var Počet,Delka:Integer;
begin
  Delka:=Length(S);
  Result:=WriteFile(Soubor,S[1],Delka,Pocet,nil)
     and (Pocet=Delka);
end;
```

Takto může vypadat zápis na RS485, je-li tok dat řízen obvody ADFC integrovanými na kartě. Jak bude rutina vypadat, bude-li přenos řízen jedním ze signálů RTS/DTR, se dozvíme později.

Poznámka: funkce WriteFile skončí poté, co vyšle kartě poslední znak. Znak se však teprve odesílá a z tohoto důvodu je sériový port ještě nějakou dobu zaneprázdněn a nelze na něj volat další funkce.

ReadFile

```
BOOL ReadFile(
    HANDLE  hFile,                // identifikátor (handle) portu
    LPVOID  lpBuffer,             // adresa paměti pro příjem dat
    DWORD  nNumberOfBytesToRead,  // počet bytů k přečtení
    LPDWORD  lpNumberOfBytesRead, // adresa proměnné, do níž se
                                     uloží počet přečtených bytů
    LPOVERLAPPED  lpOverlapped    // adresa struktury pro
                                     overlapped I/O
   );
```

```
function ReadFile(hFile: THandle; var Buffer; NumberOfBytesToRead:
DWORD; var lpNumberOfBytesRead: DWORD; lpOverlapped: POverlapped):
BOOL;
```

Provádí čtení dat z RS485. Použití jednotlivých parametrů:

hFile – Identifikátor (handle) otevřeného portu získaný z funkce CreateFile.

lpBuffer – Ukazatel na vyrovnávací paměť, kam budou data zapsána.

nNumberOfBytesToRead – Počet bytů, které mají být načteny (popř. velikost vyrovnávací paměti).

lpNumberOfBytesRead – Adresa proměnné (v Borland Delphi parametr volaný odkazem – **var**), do které bude uložen počet bytů, které byly načteny.

lpOverlapped – vložíme hodnotu NUL, resp. **nil** .

Vrací TRUE při úspěchu operace, FALSE v opačném případě.

Příklad: načtení krátké řetězcové zprávy z přístroje – Borland Delphi 3.0 :

```
function GetString:String;
var Pocet:Integer;
    Buf:array[0..20] of Char;
begin
  FillChar(Buf,SizeOf(Buf),#0);
  ReadFile(Soubor,Buf,SizeOf(Buf),Pocet,nil);
  Result:=StrPas(@Buf);
end;
```

EscapeCommFunction

```
BOOL EscapeCommFunction(
    HANDLE  hFile,    // identifikátor (handle) portu
    DWORD  dwFunc     // identifikace funkce
   );

function EscapeCommFunction(hFile: THandle; dwFunc: DWORD): BOOL;
```

Zprostředkovává volání speciálních funkcí sériového rozhraní. Umožňuje přímé řízení signálů DTR a RTS, což lze použít pro řízení přenosu, není-li na kartě nastaven automatický režim. Použití jednotlivých parametrů:

hFile – Identifikátor (handle) otevřeného portu získaný z funkce CreateFile.

dwFunc – Tento parametr určuje, která funkce se má provést. Obsahuje jednu z hodnot CLRDTR, CLRRTS, SETDTR, SETRTS, které způsobí nastavení, resp. vynulování příznaku DTR (data-terminal-ready), resp. RTS (request-to send).

Vrací TRUE při úspěchu operace, FALSE v opačném případě.

Příklad: vyslání řetězce s využitím ručního ovládání toku dat signálem DTR – Borland Delphi 3.0 :

```
function SendString2(S:String):Boolean;
var Pocet,Delka:Integer;
begin
  { "nažhavíme" smyčku }
  EscapeCommFunction(Soubor,SetDTR);
  Sleep(10);
  { vyslání dat }
  Delka:=Length(S);
  Result:=WriteFile(Soubor,S[1],Delka,Pocet,nil)
       and (Delka=Pocet);
  { uvolníme smyčku }
  Sleep(10);
  EscapeCommFunction(Soubor,ClrDTR);
end;
```

U tohoto příkladu si prosím všimněte způsobu práce se signálem DTR, kterým je ovládán tok dat. Procedura Sleep pozastavuje proces na dobu specifikovanou jako parametr v milisekundách (obdoba funkce Delay z Borland Pascalu). Ostatní procesy běží nerušeně dál. Možností se naskýtá více, toto je však pro jednoduchou aplikaci zcela dostačující.

Vzhledem k tomu, že po ukončení funkce WriteFile port ještě nějakou dobu pracuje, může se zejména na nižších rychlostech stát, že 10 ms bude málo (problémy se začnou objevovat při komunikaci 1200 Bd a pomaleji) a čekání bude nutné prodloužit. Naopak neúměrně dlouhá doba může ohrozit komunikaci s rychlejší periférií.

GetCommState / SetCommState

```
BOOL GetCommState(
    HANDLE  hCommDev,// identifikátor (handle) portu
    LPDCB  lpdcb     // adresa kontrolního bloku (device control
block)
   );

BOOL SetCommState(
    HANDLE  hCommDev,// identifikátor (handle) portu
    LPDCB  lpdcb     // adresa kontrolního bloku (device control
block)
   );
```

```
function GetCommState(hFile: THandle; var lpDCB: TDCB): BOOL;

function SetCommState(hFile: THandle; const lpDCB: TDCB): BOOL;
```

Umožňuje zjištění, resp. nastavení parametrů sériového rozhraní. Použití jednotlivých parametrů:
hFile – Identifikátor (handle) otevřeného portu získaný z funkce CreateFile.
lpdcb – adresa kontrolního bloku zařízení (Borland Delphi - TDCB)
Vrací TRUE při úspěchu operace, FALSE v opačném případě.

Příklad: Nastavení parametrů sériového portu – Borland Delphi 3.0 :

```
procedure SetupComm;
var DCB:TDCB;
begin
  GetCommState(Soubor,DCB);
  DCB.BaudRate:=9600;
  DCB.ByteSize:=8;
  DCB.Parity:=NOPARITY;
  DCB.StopBits:=ONESTOPBIT;
  DCB.Flags:=1;
  SetCommState(Soubor,DCB);
end;
```

Nejprve je tedy zjištěno aktuální nastavení portu, které je nato změněno podle potřeby. Nastavené parametry zřejmě není nutné podrobněji vysvětlovat. Za zmínku stojí snad pouze nastavení parametru Flags – zde se volí binární (tedy nikoliv textový) režim přenosu, je vypnuto řízení signálů DTR/RTS (ty mohou být využity pro řízení toku dat) a je vypnut protokol Xon/Xoff.

Struktura device control block (DCB) obsahuje ještě mnoho dalších příznaků, ty jsou však mimo rámec tohoto článku a jejich znalost není bezpodmínečně nutná. Potřebné konstanty jsou k dispozici v helpech, popř. MSDN.

Na tomto místě bych rovněž rád zmínil jednu zkušenost: použití této funkce může v některých případech být časově náročné - pod Windows 95 naměřeno v některých případech až 200 ms (změna přenosové rychlosti). Hodnoty během komunikace jsou sice řádově menší, mohou však přesto přinést problémy. To může při práci s rychlejšími periferiemi vést i ke ztrátě části přenášených dat.

GetCommTimeouts / GetCommTimeouts

```
BOOL GetCommTimeouts(
    HANDLE  hCommDev,    // identifikátor (handle) portu
    LPCOMMTIMEOUTS  lpctmo // adresa struktury časování
   );

BOOL SetCommTimeouts(
    HANDLE  hCommDev,    // identifikátor (handle) portu
    LPCOMMTIMEOUTS  lpctmo // adresa struktury časování
   );
```

```
function GetCommTimeouts(hFile: THandle; var lpCommTimeouts:
TCommTimeouts): BOOL;

function SetCommTimeouts(hFile: THandle; const lpCommTimeouts:
TCommTimeouts): BOOL;
```

Umožňuje zjištění, resp. nastavení parametrů časování sériového rozhraní. Použití jednotlivých parametrů:

hFile – Identifikátor (handle) otevřeného portu získaný z funkce CreateFile.

lpctmo – ukazatel na strukturu definující časování (Borland Delphi - TCommTimeouts)

Vrací TRUE při úspěchu operace, FALSE v opačném případě.

Příklad: Vypnutí funkce timeout pro otevřený port – Borland Delphi 3.0 :

```
procedure TurnOffTimeouts;
var Tim:TCommTimeouts;
begin
  Tim.ReadIntervalTimeout:=MAXDWORD;
  Tim.ReadTotalTimeoutMultiplier:=0;
  Tim.ReadTotalTimeoutConstant:=0;
  Tim.WriteTotalTimeoutMultiplier:=0;
  Tim.WriteTotalTimeoutConstant:=0;
  SetCommTimeouts(Soubor,Tim);
end;
```

Pro nejjednodušší způsob komunikace je možné hlídání časování vypnout. Při čtení z portu funkcí ReadFile přečte program právě tolik bytů, kolik jich vyslalo zařízení. Neprovádí se kontrola vyslání zprávy funkcí WriteFile.

V opačném případě čeká v závislosti na nastavených časech na příjem zadaného množství dat. Při nastavení příliš dlouhé čekací doby, resp. ponechání na základních hodnotách, dojde v případě poruchy přenosu k zastavení aplikace, aniž by to mohl programátor jakkoliv ovlivnit. Proto je výhodnější příjem dat vložit do procedury, která bude periodicky volána časovačem Windows (funkce SetTimer, resp. komponenta Timer v Borland Delphi), bude obsahovat volání funkce ReadFile (timeouty vypnuty funkcí SetCommTimeouts) a jež bude sama zajišťovat obsluhu kontroly příjmu zprávy. Rovněž vysílání zprávy je vhodné ověřit podle počtu odeslaných bytů.

Podrobnější referenci najde čtenář v helpu k vývojovým nástrojům, v MSDN a jinde.

Tvorba aplikace využívající RS485

Fragmentů kódu uvedených jako příklady k jednotlivým funkcím již lze využít při tvorbě konkrétní aplikace. Postup je následující:

1. Otevření portu funkcí CreateFile. Test úspěšnosti.
2. Nastavení přenosových parametrů funkcí SetCommState
3. Zrušení čekání na příjem, resp. vyslání znaku – funkce SetCommTimeouts
4. Tělo programu
5. Uzavření portu funkcí CloseHandle

Přestože uvedeným příkladům je možné vytknout, že postrádají možnost avíza o příchozí zprávě po sériovém portu a je tudíž nutno využít poolingu, pro základní seznámení s možnostmi přechodu na 32bitovou platformu to, domnívám se, plně postačuje.

Karta PCI-685

PCI-685 je rozšiřující karta pro počítače PC, která obsahuje dva porty sériových komunikačních linek standardu RS422/485. Využívá obvodů registrově kompatibilních se standardy 8250, resp. 16C550, jejichž podpora je začleněna do všech běžných operačních systémů. Umožňuje využití méně obsazených kanálů přerušení (IRQ) a větší výběr adres I/O. S přídavných hodinovým generátorem je možná komunikace na nestandardních rychlostech. Režim přenosu je ovladatelný programově signály DTR/RTS, nebo je řízen automaticky obvody ADFC.

Svou koncepcí je karta určena zejména pro:

- distribuované systémy řízení procesů
- automatizační prostředky
- komunikace s měřícími systémy
- přenos dat mezi počítači

Kde lze kartu PCI-685 objednat

Multifunkční komunikační kartu PCI-685 lze objednat na adrese:

ADICOM Praha, spol. s r.o.
Františka Kadlece 9
180 00 Praha 8
tel.: 02 / 83842047, 0603 / 475161,
fax: 02 / 83841048
Internet: http://www.aditech.cz,
email: info@aditech.cz

Aktuální ceny jsou k dispozici na našich stránkách WWW. Zde můžete získat i podrobnější informace o dalších měřících zařízeních a jejich podpoře pro 32-bitové operační systémy.

Mnoho úspěchů při vývoji 32-bitových aplikací přeje Adicom Praha, s.r.o.

V Praze 6. 2. 1999

Michal Bureš