Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

2 SÉRIOVÝ PORT COM

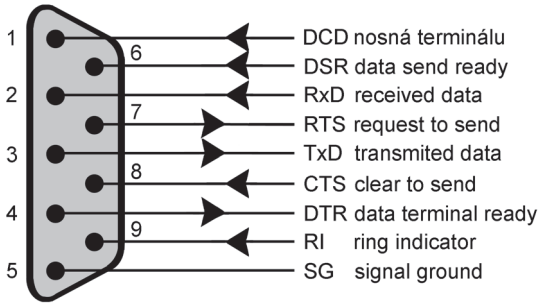
Dříve než se pustíme do vlastního objasňování, jak napsat program pro obsluhu sériového portu, bude nutné si nejdříve vyčerpávajícím způsobem popsat jeho hardwarové vlastnosti.

Vlastní obvod pro sériovou komunikaci tvoří zpravidla obvod UART 8250, a to u starších typů počítačů, nebo obvod 82550 u novějších typů. Jádro tohoto obvodu pro sériovou komunikaci může být obsaženo i v jiných čipech, které jsou použity pro řešení hardwaru paralelních i sériových portů najednou. Hovoříme-li však o sériovém rozhraní COM, musí tyto definované vlastnosti obvod realizující sériové rozhraní obsahovat.

2.1 Popis hardwaru sériového portu

Signály na rozhraní sériového portu můžeme rozdělit z hlediska své funkce do dvou skupin. První skupinu tvoří signály pro vlastní sériový přenos dat, a to signál pro vysílání dat (TxD) a signál pro příjem dat (RxD).

Druhou skupinou jsou tzv. signály modemu sériového rozhraní. Tyto signály přenášejí informaci o stavu vysílače a přijímače sériových dat. Přenáší informace při navazování spojení a zároveň se používají pro řízení vlastního přenosu dat. Jsou to signály (RTS, CTS, DSR, DTR). Zvláštním signálem je signál DCD, který přenáší informaci o navázání



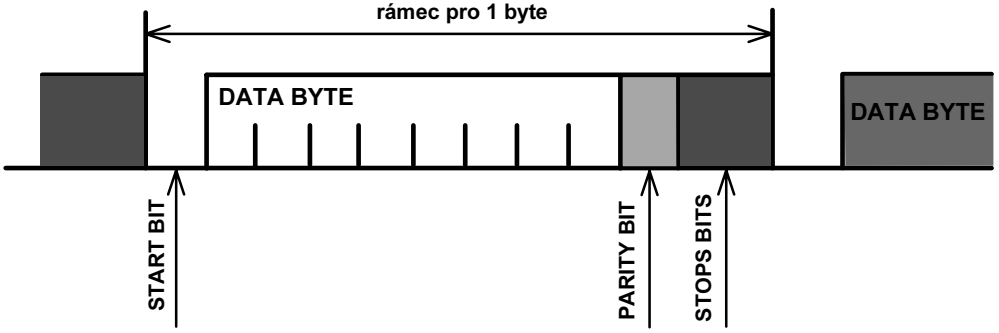
Obr. 2.1

spojení. Zapojení 9pinového konektoru CANNON sériového portu je vyobrazeno na *obr. 2.1*.

Vlastní princip sériové komunikace si popíšeme v následující kapitole.

2.2 Princip sériové komunikace

Při vysílání znaku jsou data nahrána z registru Transmiter Holding Register do posuvného registru vysílače a bit po bitu posílána na výstup jako signál TxD. Struktura přenášeného byte je znázorněna na *obr. 2.2*.



Obr. 2.2 *Struktura přenášeného byte*

Jak je z *obr. 2.1* zřejmé, kromě datové informace, která může mít různou délku, jsou s každým znakem vysílány i informace synchronizační, protože jde o tzv. asynchronní přenos dat. Každý rámec oznámí svůj začátek přijímači nízkou úrovní L signálu TxD. Toto se nazývá START BIT. Po uplynutí start bitu následují bity datové. Počet těchto bitů se může pohybovat od 5 do 8. Po odvysílání datových bitů může (ale nemusí) následovat tzv. paritní bit, který slouží pro kontrolu chyb sériového přenosu. Posledními vysílanými bity jednoho rámce jsou STOP BITY, které rámec ukončují. Počet těchto stop bitů může být jeden, jeden a půl nebo dva. Stop bit má proti start bitu vysokou úroveň H. Jakmile se tato vysoká úroveň změní v nízkou, považuje to přijímač za start bit dalšího rámce.

Přijímač při sériové komunikaci očekává přijímané bity jako signál RxD a ve svém posuvném registru z nich skládá přijatý znak. Některé obvody UART vzorkují vstup RxD čtyřnásobnou rychlostí a provádějí vyhodnocení odebraných vzorků na principu pravděpodobnosti dva ze tří. Tím jsou eliminovány krátké poruchy nebo rušení v přenosu dat. Přijatý signál RxD je ukládán do posuvného registru a postupně je z něho skládán přijatý znak. Ten po přijetí jeho posledního bitu uloží rámec do registru Receiver Buffer Register, ze kterého je poté odebrán k dalšímu zpracování.

Počet datových bitů, přítomnost paritního bitu, typ parity, počet stop bitů a rychlost přenosu dat, tedy dobu existence 1 bitu na přenosové lince lze nakonfigurovat a jak se to udělá, si vysvětlíme při vlastním objasňování tvorby programu obsluhy sériového portu.

2.3 Signály modemu sériového portu

Jak již bylo výše řečeno, signály modemu se používají při navazování spojení i při vlastním datovém přenosu. U vlastní asynchronní sériové komunikace není nezbytně nutné tyto signály využívat, ale některé aplikace či hardwarové periferie používání těchto signálů vyžadují. Mezi tato zařízení patří i telefonní datový modem. Signály modemu sériového portu tvoří vždy dvojice pro daný typ aktivity. Signály $\overline{\text{DTR}}$ a $\overline{\text{DSR}}$ se používají k navázání spojení, zatímco signály $\overline{\text{RTS}}$ a $\overline{\text{CTS}}$ se používají pro řízení vlastního přenosu dat:

$\overline{\text{DTR}}$ data terminal ready – PC	→	modem – PC je svolné k připojení, deaktivace tohoto signálu přeruší spojení.
$\overline{\text{DSR}}$ data set ready – modem	→	PC – modem je připraven spojit se s PC.
$\overline{\text{RTS}}$ request to send – PC	→	modem – PC je připraveno vysílat.
$\overline{\text{CTS}}$ clear to send – modem	→	PC – modem je připraven přijmout další data.
$\overline{\text{DCD}}$ data carrier detect – modem	→	PC – modem zachytil nosnou vysílanou PC.
$\overline{\text{RI}}$ ring indicator – modem	→	PC – někdo volá a chce se spojit.

Při navazování spojení jsou využívány signály $\overline{\text{DSR}}$ a $\overline{\text{DTR}}$. Svou připravenost spojit se s PC dává modem najevo signálem $\overline{\text{DSR}}$. Svou připravenost připojení k modemu dává PC modemu najevo signálem $\overline{\text{DTR}}$. Datový telefonní modem při deaktivaci signálu $\overline{\text{DTR}}$ zpravidla přeruší spojení, ale tuto funkci je možné u některých typů modemu zablokovat.

Svou připravenost k vysílání dává PC modemu najevo signálem $\overline{\text{RTS}}$. To že je modem připraven přijmout data z PC dává modem najevo signálem $\overline{\text{CTS}}$. Není-li tento signál aktivní, nesmí PC odesílat do modemu žádná data, jelikož by došlo k jejich ztrátě.

Signálem $\overline{\text{DCD}}$ dává modem PC najevo, že bylo navázáno spojení, a že je možné započít datovou komunikaci. Nastal-li případ, že navázané spojení mezi modemy bylo přerušeno, signál $\overline{\text{DCD}}$ přestane být aktivní a spojení je nutné navázat znovu.

Posledním signálem RI modem dává PC najevo, že někdo požaduje spojení. Jde o tzv. indikaci vyzvánění.

Na závěr je ještě nutné podotknout, že u signálu modemu platí, že jsou aktivní při nízké úrovni L, a proto jsou v textu označeny jako negované.

Tímto jsme uzavřeli kapitolu popisu sériového portu COM, a to z hlediska významu a funkce jednotlivých signálů, a jako další bude nutné popsat vybrané procedury a datové struktury používané při programování obsluhy sériového portu.

6 POPIS PROSTŘEDKŮ TAPI TELEFONNÍHO MODEMU

Seznam vybraných funkcí TAPI pro obsluhu telefonního modemu:

DEVICECONFIG

Kategorie:

Struktura

Definice:

```
typedef struct device_config_tag {
   VARSTRING vs;
   DWORD dwSize;
   DWORD dwVersion;
   WORD fwOptions;
   WORD wWaitBong;
   COMMCONFIG commconfig;
} DEVICECONFIG, FAR *LPDEVICECONFIG;
```

Popis:

Struktura komplexně popisující nastavení komunikačního rozhraní místního modemu. Je použita při volání funkcí *lineGetDevConfig* a *lineSetDevConfig* při zjišťování a nastavování parametrů sériového portu místního modemu. V parametru těchto funkcí je přetypována na proměnný řetězec typu VARSTRING.

Parametry:

vs	–	Proměnná typu VARSTRING obsahující popis konstrukce a velikost dat ve struktuře.
dwSize	–	Absolutní velikost struktury sestavené proměnným řetězcem typu VARSTRING.
dwVersion	–	Proměnná identifikující verzi konstrukce struktury.
fwOptions	–	Řídicí parametr dat ve struktuře.
wWaitBong	–	Není použit, musí být 0.
commconfig	–	Proměnná typu COMMCONFIG obsahující strukturu typu DCB s nastavením sériového portu místního modemu.

MODEM_INFO

Kategorie:

Struktura

Definice:

```
typedef struct modem_info_tag {
   VARSTRING vs;
   HANDLE hComm;
   char szDeviceName[255];
} MODEM_INFO, FAR *LPMODEM_INFO;
```

Popis:

Struktura obsahující informace předávané prostředím TAPI o modemu, který zprostředkovává spojení. Struktura je naplněna daty voláním funkce *lineGetID*.

Parametry:

vs	–	Proměnná typu VARSTRING obsahující popis konstrukce a velikost dat ve struktuře.
hComm	–	Handle pro přístup k sériovému portu modemu, který zprostředkovává spojení.
szDeviceName	–	Řetězec obsahující název modemu, který zprostředkovává spojení.

lineCallbackFunc

Kategorie:

Funkce

Syntaxe:

```
VOID FAR PASCAL lineCallbackFunc(
   DWORD hDevice,
   DWORD dwMsg,
   DWORD dwCallbackInstance,
   DWORD dwParam1,
   DWORD dwParam2,
   DWORD dwParam3
);
```

Popis:

Funkce *lineCallbackFunc* je předloha pro definici funkce obsluhy reakcí na události generované prostředím TAPI při obsluze telefonního modemu.

Parametry:

hDevice	–	Handle pro identifikaci linky nebo objektu volání, požadující obsluhu reakce na událost. Parametr musí být typu DWORD, protože typ HANDLE by mohl generovat chybu.
dwMsg	–	Parametr specifikující typ obsluhované události.
dwCallbackInstance	–	Tento parametr není v TAPI použit.
dwParam1	–	Parametr zprávy **dwMSG**.
dwParam2	–	Parametr zprávy **dwMSG**.
dwParam3	–	Parametr zprávy **dwMSG**.

Návratová hodnota:

Funkce nemá návratovou hodnotu.

LINECALLINFO

Kategorie:

Struktura

Definice:

```
typedef struct linecallinfo_tag {
   DWORD dwTotalSize;
   DWORD dwNeededSize;
   DWORD dwUsedSize;
   HLINE hLine;
   DWORD dwLineDeviceID;
   DWORD dwAddressID;
   DWORD dwBearerMode;
   DWORD dwRate;
   DWORD dwMediaMode;
   DWORD dwAppSpecific;
   DWORD dwCallID;
   DWORD dwRelatedCallID;
   DWORD dwCallParamFlags;
   DWORD dwCallStates;
   DWORD dwMonitorDigitModes;
   DWORD dwMonitorMediaModes;
   LINEDIALPARAMS DialParams;
   DWORD dwOrigin;
   DWORD dwReason;
   DWORD dwCompletionID;
   DWORD dwNumOwners;
   DWORD dwNumMonitors;
   DWORD dwCountryCode;
   DWORD dwTrunk;
   DWORD dwCallerIDFlags;
   DWORD dwCallerIDSize;
   DWORD dwCallerIDOffset;
   DWORD dwCallerIDNameSize;
   DWORD dwCallerIDNameOffset;
   DWORD dwCalledIDFlags;
   DWORD dwCalledIDSize;
   DWORD dwCalledIDOffset;
   DWORD dwCalledIDNameSize;
   DWORD dwCalledIDNameOffset;
   DWORD dwConnectedIDFlags;
   DWORD dwConnectedIDSize;
   DWORD dwConnectedIDOffset;
   DWORD dwConnectedIDNameSize;
   DWORD dwConnectedIDNameOffset;
```

lineSetDevConfig

Kategorie:

Funkce

Syntaxe:

```
LONG lineSetDevConfig(
   DWORD dwDeviceID,
   LPVARSTRING lpDeviceConfig,
   DWORD dwSize,
   LPCSTR lpszDeviceClass
);
```

Popis:

Funkce uloží informace o aktuálním nastavení přenosových vlastností vybraného linkového zařízení z paměťového bloku typu VARSTRING do prostředí TAPI.

Parametry:

dwDeviceID	–	Identifikace popisovaného linkového zařízení.
lpDeviceConfig	–	Ukazatel na paměťový blok typu VARSTRING ve kterém je uložena struktura typu DEVICECONFIG popisující aktuální nastavení linkového zařízení.
dwSize	–	Velikost alokovaného prostoru proměnné VARSTRING obsahující strukturu DEVICECONFIG.
lpszDeviceClass	–	Řetězec popisující třídu linkového zařízení. V našem případě to musí být „comm/datamodem“.

Návratová hodnota:

Proběhla-li funkce bez chyby, pak je její návratová hodnota rovna 0. Při výskytu chyby je návratová hodnota různá od nuly a význam chyby zjistíme ze seznamu konstant *LINEERR_ Constants*, který je součástí manuálu Windows SDK.

lineShutdown

Kategorie:

Funkce

Syntaxe:

```
LONG lineShutdown(
   HLINEAPP hLineApp
);
```

Popis:

Funkce zlikviduje zinicializované prostředí TAPI.

Parametry:

hLineApp – Handle aplikací použitého prostředí TAPI.

Návratová hodnota:

Proběhla-li funkce bez chyby, pak je její návratová hodnota rovna 0. Při výskytu chyby je návratová hodnota různá od nuly a význam chyby zjistíme ze seznamu konstant *LINEERR_ Constants*, který je součástí manuálu Windows SDK.