CoBase

ZÁKLADNÍ KNIHOVNA PRO IMPLEMENTACI SÍŤOVÝCH PROTOKOLŮ

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.11

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 22.04.2004
Datum posledního uložení dokumentu: 22.04.2004
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Čr		První vydání
1.10	1.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000.
1.11	1.XX	Wil	22.04.2004	Grafické a jazykové úpravy.

1.2. Účel dokumentu

Tento dokument popisuje základní knihovnu pro implementaci síťových protokolů.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu není potřeba číst žádný další manuál, ale je potřeba orientovat se v používání programového vybavení SofCon.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

Používané termíny a definice jsou popsány v samostatném dokumentu „Termíny a definice“.

3. Úvod

3.1. Účel knihovny CoBase

Knihovna **CoBase** je určena pro implementaci různých síťových protokolů a ovladačů zařízení (např. ovladače síťové karty, protokoly TCP/IP, IPX/SPX apod.). Základem této knihovny je třída **TCoDevice**. Tato třída je předkem všech dalších protokolů a ovladačů zařízení. Knihovna definuje dvě rozhraní: **Aplikační rozhraní** (tj. soubor funkcí konstant a typů pro tvorbu aplikací) a **rozhraní pro vytváření protokolů** (tj. rozhraní pro programátory protokolů a ovladačů zařízení). Tento dokument se zabývá pouze aplikačním rozhraním.

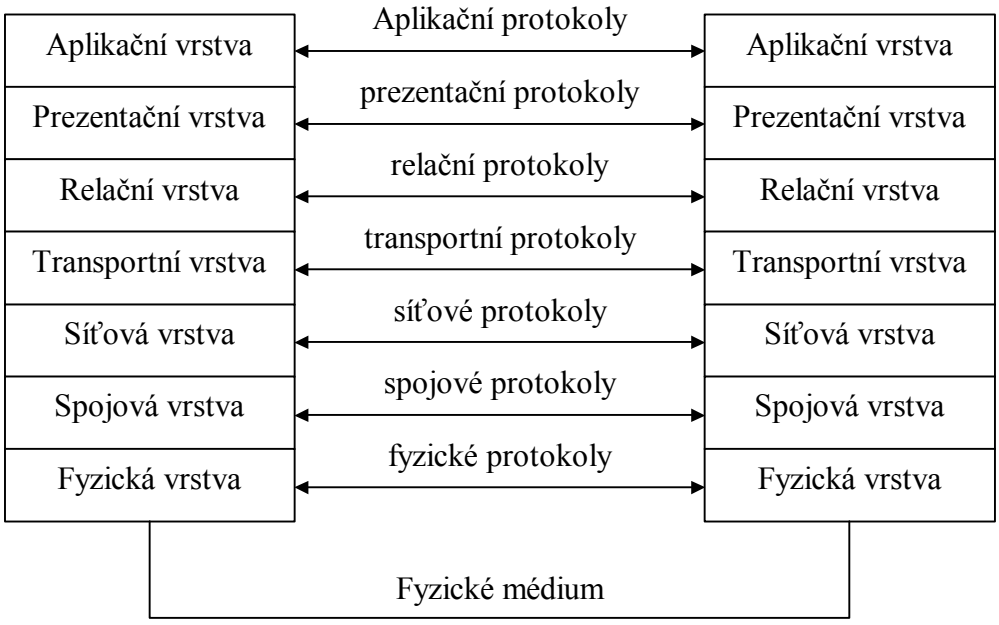
3.2. Síťové architektury

Síťová architektura představuje souhrn řídících činností umožňujících vzájemnou komunikaci mezi komunikujícími systémy. Komunikace a její řízení je složitý problém sestávající z celé řady dílčích problémů. Přistoupilo se tedy k rozdělení tohoto problému do několika problémových skupin, tzv. vrstev. Proto se o síťových architekturách hovoří také jako o architekturách vrstvových.

Každá vrstva síťové architektury je definovaná službami, které je schopna poskytovat vyšší vrstvě, a funkcemi, které vykonává v rámci protokolu. Protokol (nebo také komunikační protokol) je souhrn pravidel, formátů a procedur potřebný pro výměnu dat mezi dvěma komunikujícími prvky. Každá vrstva tedy vykonává jasně definovanou skupinu funkcí potřebnou pro komunikaci s jinými systémy a pro svoji činnost využívá služeb nižší vrstvy a své služby pak poskytuje vyšší vrstvě.

Zásadními výhodami vrstvových architektur je nezávislost na implementaci a snadná výměna protokolu jedné vrstvy za jiný. K tomu je nezbytně nutné mít mezi každou dvojicí sousedních vrstev jasně definované rozhraní nezávislé na implementaci obou vrstev. Protože jednotlivé vrstvy jsou obvykle implementovány programově, jedná se o programové rozhraní definující funkce a postupy, pomocí kterých lze poskytovat služby vyšší vrstvě a pomocí kterých jsou zpřístupněny služby vrstvy nižší.

Nejznámější síťovou architekturou, navrženou již v 70. letech organizací ISO (International Standard Organization), je referenční model OSI (Open System Interconnection). Referenční model OSI je sedmivrstvý. Úlohou OSI je poskytnou společnou základnu pro vypracování norem pro účely propojování systémů.



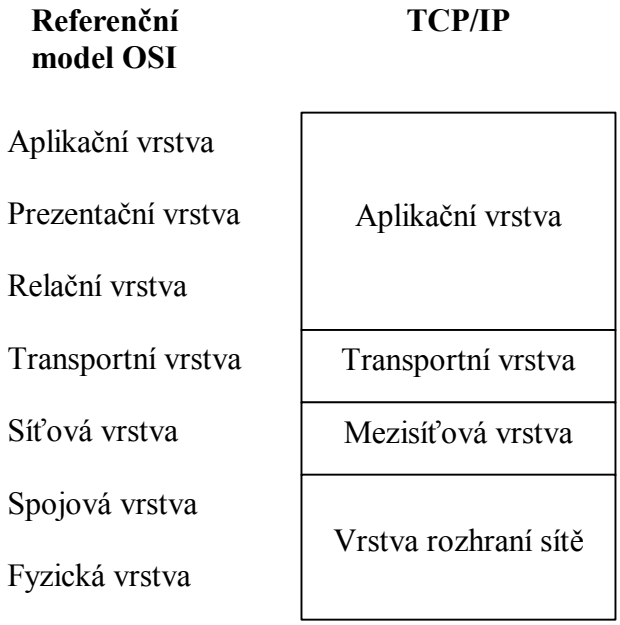
- **Fyzická vrstva** zajišťuje fyzickou komunikaci mezi komunikujícími systémy. Definuje mechanické a elektrické vlastnosti přenosových prostředků. Provádí fyzický přenos dat, příp. navazování fyzických spojení apod.
- **Spojová vrstva** zajišťuje navazování, udržování a rušení spojení, identifikaci stanic, detekci a opravu chyb v rámci lokální sítě.
- **Síťová vrstva** zajišťuje přenos dat mezi stanicemi, které spolu nemusí nutně sousedit (mohou být v jiné lokální síti). Základními funkcemi jsou síťové adresování, zahajování, rušení síťových spojení a uspořádání přenášených fragmentů v paketech.
- **Transportní vrstva** poskytuje transparentní, obvykle spolehlivý přenos dat s požadovanou kvalitou. Poskytuje relační vrstvě navázání, udržování a závěr transportních spojení. Dalšími funkcemi transportní vrstvy je detekce a oprava chyb, multiplexování transportních spojení apod.
- **Relační vrstva** organizuje a synchronizuje dialog mezi spolupracujícími stanicemi a řídí výměnu dat.
- **Prezentační vrstva** zajišťuje transformaci různorodých přenášených dat. Jedná se především o transformace kódů a abeced, kompresi dat apod.
- **Aplikační vrstva** poskytuje aplikačním procesům přístup ke komunikačnímu systému. Mezi základní služby aplikační vrstvy patří přenos zpráv, identifikace komunikujících procesů apod.

Konkrétní síťové architektury nemusí některé ze zmíněných funkcí poskytovat. Dále se již referenčním modelem OSI nebudeme zabývat.

3.3. TCP/IP architektura

Rozvrstvení TCP/IP architektury neodpovídá rozvrstvení referenčního modelu OSI, neboť vznikla ještě před jeho oficiálním přijetím. TCP/IP architektura obsahuje pouze čtyři vrstvy, z nichž každá zahrnuje jednu nebo více vrstev referenčního modelu OSI. Nejnižší vrstva TCP/IP architektury, **vrstva rozhraní sítě,** odpovídá svými funkcemi vrstvě spojové a síťové. Druhá nejnižší vrstva, **vrstva mezisíťová,** odpovídá vrstvě síťové. **Transportní vrstva** odpovídá transportní vrstvě OSI. **Aplikační vrstva** zahrnuje všechny zbývající vrstvy referenčního modelu OSI, tj. relační, prezentační a aplikační vrstvu.

Obrázek 3.1 Srovnání referenčního modelu OSI s TCP/IP



- **Vrstva rozhraní sítě** umožňuje přístup k fyzickému přenosovému médiu. V našem případě je tato vrstva reprezentovaná ovladačem dané síťové karty (např. knihovna CoEth01 pro kartu IOETH01).
- **Vrstva mezisíťová** je zodpovědná za síťovou adresaci, směrování a předávání datagramů přes komunikační podsíť. Dalším úkolem této vrstvy je provádět segmentaci a znovusestavování datagramů do a z rámců specifikovaných protokolem vrstvy rozhraní sítě. Mezisíťová vrstva se skládá z několika protokolů, z nichž nejdůležitější jsou protokoly IP, ICMP a ARP. Více o této vrstvě lze nalézt v dokumentaci k protokolu IP (CoIPv4).
- **Transportní vrstva** je určena pro koncový přenos dat mezi dvěma stanicemi. TCP/IP definuje nabízí transportní služby se spojením a bez spojení za pomoci použití jednoho ze dvou protokolů: TCP nebo UDP. (viz. knihovny CoUdp a CoTcp)
- **Aplikační vrstva** je nejvyšší vrstvou síťové architektury TCP/IP a obsahuje všechny protokoly poskytující uživatelům konkrétní aplikace. Některé aplikační protokoly jsou přesně závislé na typu transportní vrstvy, proto vyžadují buď protokol TCP nebo UDP. Některé z aplikačních protokolů

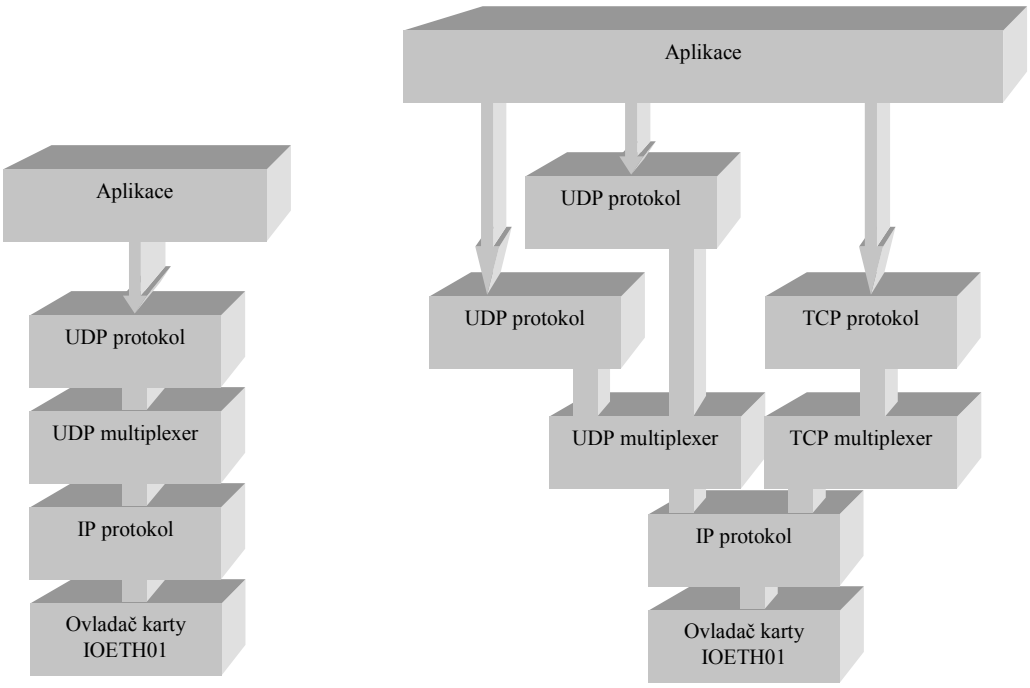
mohou používat oba transportní protokoly. Mezi nejpoužívanější transportní protokoly patří: TELNET, FTP, HTTP, NFS, POP3, SMTP, SNMP apod. Protože aplikační protokoly vyžadují obvykle úplně jiné rozhraní než protokoly nižších vrstev (obvykle se jedná o soubor specifických funkcí pro daný protokol), nejsou implementovány pomocí třídy TCoDevice, ale jsou ve většině případů přímo součástí aplikace (např. knihovna HTTP).

3.4. Zřetězení komunikačních zařízení

Zařízením (Device) budeme v dalším textu rozumět programový kód, zajišťující funkce části nebo celé vrstvy protokolové architektury. Zařízením je například ovladač síťové karty, ovladač protokolu IP apod. Všechna zařízení byla implementována pomocí obecné třídy **TCoDevice** deklarované v této knihovně, která definuje společné vlastnosti a jednotné rozhraní všech dalších implementací protokolů a ovladačů. **Třídou zařízení** budeme rozumět třídu objektů daného zařízení a **Instancí zařízení** instanci objektu daného zařízení. Skupina propojených zařízení tvoří **protokolový zásobník (stack)**, např. TCP/IP zásobník tvoří zařízení ovladače síťové karty, zařízení protokolu IP a zařízení protokolu TCP a UDP.

Jednotlivé vrstvy uvedené v kapitole 3.3 jsou reprezentovány komunikačními zařízeními (potomky třídy TCoDevice). Třída TCoDevice poskytuje jednotné rozhraní pro zřetězení zařízení.

Na následujícím obrázku jsou ukázky dvou příkladů zřetězení komunikačních zařízení. Třída TCoDevice umožňuje variabilní zřetězení podle potřeby aplikace. Je vidět, že zřetězení nemusí být lineární, ale může mít stromovou strukturu.

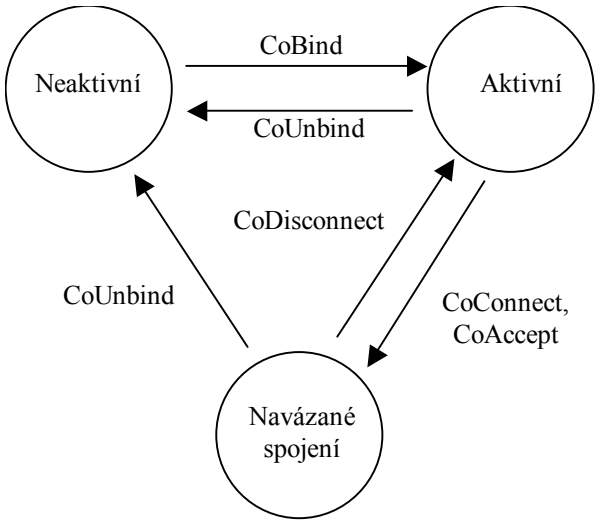


3.5. Stavy zařízení

Každé zařízení implementované pomocí třídy TCoDevice se může nacházet ve třech elementárních stavech:

- Neaktivní (Inactive). Nebyla provedena základní inicializace zařízení.
- Aktivní (Active). Základní inicializace zařízení byla provedena.
- S navázaným spojením (Connected). Spojení s cílovou stanicí bylo navázáno.

Přechody mezi jednotlivými stavy zajišťují metody CoBind, CoUnbind, CoConnect, CoDisconnect, příp. CoAccept.



Některá zařízení nemusí implementovat všechny tři stavy, ale pouze první dva (např. zařízení ovladače ethernetové karty, zařízení protokolu IP a UDP).

4. Konstanty a jednoduché typy

4.1. Konstanty

4.1.1. Návratové kódy CST_xxx

Identifikátory s prefixem **CST_** jsou návratové kódy metod zařízení. Konstanty CST_ definují implementace ovladačů konkrétních zařízení. Obvykle za prefixem CST_ následuje textový identifikátor zařízení. Seznam konstant CST_ konkrétního ovladače zařízení je uveden v dokumentaci k tomuto zařízení.

Je definována celá řada obecných návratových kódů, které nejsou poplatné žádnému konkrétnímu zařízení:

CST_SUCCESS
: Požadavek byl úspěšně proveden.

CST_PENDING
: Požadavek probíhá a bude dokončen později. CST_PENDING vracejí tzv. neblokující operace. viz. kapitola 7.2.

CST_ERR_NOTSUPP (Not supported)
: Funkce nebo operace není u daného zařízení podporována. Např. zařízení síťového adaptéru nepodporuje volání metod CoAccept a CoListen a proto tyto metody vracejí tento návratový kód.

CST_ERR_FATAL (Fatal error)
: Interní chyba knihovny CoBase. Tato chyba by neměla za normálních okolností nastávat. Vyskytnout se může v případě poškození vnitřních datových struktur zařízení.

CST_ERR_INVALID (Invalid parameter)
: Požadavek nemohl být proveden, protože při volání metody byly předány neplatné parametry.

CST_ERR_MEMOVF (Memory overflow)
: Požadavek nemohl být dokončen v důsledku nedostatku operační paměti vyhrazené pro tento typ požadavku.

CST_ERR_INVREQUEST (Invalid request)
: Interní chyba knihovny CoBase. Tato chyba by neměla za normálních okolnost nastávat. Vyskytnout se může v případě chyby implementace zařízení.

CST_ERR_INVDEVICE (Invalid device)
: Tato chyba může nastat v případě, že přistupujeme k řetězci zařízení, který není správně vytvořen. Běžně by tato chyba neměla nastávat.

CST_ERR_INVRANGE (Invalid range)
: Neplatný rozsah parametru. Operace byla volána s parametrem mimo platný rozsah.

CST_ERR_SYNTAX (Syntax error)
Chyba syntaxe textového řetězce s parametry zařízení. Tento návratový kód může navracet metoda **CoSetOptionString**.

CST_ERR_INVCLASS (Invalid class)
Neplatný identifikátor třídy zařízení. Tato chyba může nastávat v operací, kde je nutné specifikovat identifikátor třídy zařízení, např. **CoIoctl**, **CoGetOption** apod.

CST_ERR_INVADDR (Invalid address)
Tuto návratovou hodnotu mohou vrátit operace (např. **CoSendBufferTo**, **CoConnect** apod.), pokud byla v parametrech uvedena neplatná adresa stanice.

CST_ERR_ISCONN (Is connected)
Spojení je již navázáno. Tento návratový kód vrací metody, které vyžadují, aby v okamžiku jejich volání nebylo navázáno spojení, např. **CoConnect**.

CST_ERR_NOTCONN (Not connected)
Spojení nebylo navázáno. Tento návratový kód mohou vrátit operace, které vyžadují, aby zařízení bylo ve stavu, kdy je navázané spojení s jinou stanicí.

CST_ERR_ISBOUND (Is bound)
Lokální adresa byla již přiřazena. Tento návratový kód vrací metody, které vyžadují, aby v okamžiku jejich volání nebyla přiřazena lokální adresa.

CST_ERR_NOTBOUND (Not bound)
Lokální adresa nebyla přiřazena. Tento návratový kód mohou vrátit operace, které vyžadují, aby zařízení bylo v aktivním stavu, tj. stavu, kdy je zařízení nainicializováno a je mu přiřazena lokální adresa.

CST_ERR_BUFFSIZE (Buffer size)
Předávaný buffer nemá správnou velikost.

CST_ERR_ABORTED (Aborted)
Operace byla předčasně zrušena, nejčastěji voláním metody **CoCancelIO**.

CST_ERR_TIMEOUT (Timeout)
Operace nebyla dokončena v důsledku vypršení časového limitu. Časové limity jednotlivých operací lze nastavit pomocí metody **CoSetTimeouts**.

CST_ERR_LISTFULL (List full)
Seznam zařízení připojených k nižší vrstvě je plný a není možné připojit další zařízení. Tuto chybu vrací metoda **CoAttachDevice**, nebo funkce **CoCreateDevice**.

CST_ERR_NOTFOUND (Not found)
Zařízení není v seznamu. Tato chyba může nastat pouze v případě chyby implementace konkrétního zařízení a běžně by nastávat neměla.

CST_ERR_DUPLICATE (Duplicate)

CST_ERR_ISATTACHED (Is attached)
Zařízení je již připojeno k nižší vrstvě. Tento návratový kód vrací metoda CoAttachDevice, pokud se pokusíme připojit již připojené zařízení.

CST_ERR_DUPNAME (Duplicate name)
Duplicitní název instance zařízení. Tento návratový kód může vrátit např. funkce **CoCreateDevice**, pokud vytvářené zařízení pojmenujeme stejně jako některé již existující zařízení.

CST_ERR_INVOPT (Invalid option)
Neplatný identifikátor parametru specifikovaný v metodě **CoSetOption** nebo **CoGetOption**.

CST_ERR_PROTOREG (Protocol registered)
Protokol byl již zaregistrován. Tento návratový kód vrací operace připojení protokolu k nižší vrstvě, pokud stejný protokol je již k dané vrstvě připojen. *Např. k zařízení ETH01 (ovladač síťového adaptéru) lze připojit pouze jednu instanci protokolu třídy IPV4 (IP protokol verze 4).*

CST_ERR_QUEUEFULL *(Queue full)*
*Fronta je plná, paket byl zahozen. Tento návratový kód vracejí některé interní funkce. Uživatelské metody **CoXXX** jej nikdy nevrací.*

CST_ERR_DROPPED *(Packet dropped)*
*Paket byl zahozen. Tento návratový kód vracejí některé interní funkce. Uživatelské metody **CoXXX** jej nikdy nevrací.*

CST_ERR_HOSTUNREACH (Host unreachable)
Cílová stanice je nedosažitelná. Vrací metody pro navazování spojení (**CoConnect**), odesílaní dat (**CoSendBuffer**) apod.

CST_ERR_NETUNREACH (Net unreachable)
Cílová síť je nedosažitelná. Tento návratový kód vrací metody pro navazování spojení (např. **CoConnect**), odesílání dat (např. **CoSendBuffer** apod).

Návratové kódy lze přeložit pomocí funkce **CoStatusToStr** na textový řetězec. Existují dvě varianty této funkce.

- Globální funkce CoStatusToStr (viz. kapitola 6.1.5)
- Metoda CoStatusToStr (viz. kapitola 7.1.2.22)

Metoda **CoStatusToStr** definovaná ve třídě **TCoDevice** správně převede na řetězec i ty návratové kódy, které jsou specifické konkrétnímu zařízení. Globálně definovaná funkce **CoStatusToStr** v jednotce **CoBase** převede na řetězec správně pouze standardní chybové kódy uvedené výše. Ostatní chybové kódy budou uvedený pouze v číselné podobě.

4.1.2. Identifikátory tříd zařízení DEV_CLASS_xxx

Každá třída zařízení má přiřazený vlastní jednoznačný identifikátor, tj. konstantu s prefixem **DEV_CLASS_.** Tento identifikátor určuje zařízení v řetězci při volání funkcí **CoIoctl**, **CoGetOption**, **CoSetOption** apod.

DEV_CLASS_NIC	Network Interface Card
DEV_CLASS_IPV4	Internet Protocol
DEV_CLASS_IPXMUX	IPX Multiplexer
DEV_CLASS_IPX	IPX Socket

DEV_CLASS_UDPMUX	UDP Multiplexer
DEV_CLASS_UDP	UDP Socket
DEV_CLASS_TCPMUX	TCP Multiplexer
DEV_CLASS_TCP	TCP Socket

Identifikátory zařízení jsou automaticky registrovány v inicializačních částech knihoven. K registraci zařízení stačí pouze uvést název odpovídající jednotky za klíčovým slovem **uses**.

4.1.3. Identifikátory řídicích operací IOCTL_xxx

Identifikátory s prefixem **IOCTL_** specifikují operaci prováděnou funkcí **CoIoctl** v řetězci zařízení. Konstanty IOCTL_ definují implementace konkrétních zařízení. Obvykle za prefixem IOCTL_ následuje textový identifikátor zařízení. Seznam konstant IOCTL_ konkrétního zařízení je uveden v dokumentaci k tomuto zařízení.

Příklad identifikátorů řídicích operací:

IOCTL_NIC_GETMACADDRESS
IOCTL_ETH01_RESETCOUNTERS

4.1.4. Identifikátory parametrů zařízení COPT_xxx

Identifikátory s prefixem **COPT_** specifikují parametr zařízení ve funkci **CoGetOption** příp. **CoSetOption**. Konstanty COPT_ definují implementace konkrétních zařízení. Obvykle za prefixem COPT_ následuje textový identifikátor zařízení. Seznam konstant COPT_ konkrétního zařízení je uveden v dokumentaci k tomuto zařízení.

Příklady identifikátorů parametrů zařízení:

COPT_IPV4_DEFGW
COPT_ETH01_PADFRAMES

4.1.5. Upřesňující příznaky volání operací COF_xxx

Konstanty s prefixem **COF_** se používají pro změnu chování standardních funkcí zařízení. Konstanty COF_ definují implementace konkrétních zařízení. Obvykle za prefixem COF_ následuje textový identifikátor zařízení. Seznam konstant COF_ konkrétního zařízení je uveden v dokumentaci k tomuto zařízení.

Kromě COF_ konstant specifických danému zařízení existují standardní konstanty, definované v jednotce **CoBase**.

COF_PROPAGATE (viz. CoBind, CoUnbind, CoConnect, CoDisconnect)
Provedení operace ve všech vrstvách protokolového řetězce.

COF_QUIET (viz. CoBind, CoUnbind, CoConnect, CoDisconnect)
Volaná metoda nevrací chybu, pokud požadovaná operace byla již provedena.

COF_IFNOTUSED (viz. CoUnbind, CoDisconnect)
Volaná operace bude provedena jen tehdy, pokud zařízení není využívané.

COF_WAIT (viz. CoGetIOResult, CoCancelIO)
Aktivní čekání na dokončení operace.

COF_NOTICK (viz. CoGetIOResult, CoCancelIO)
Interní příznak. V rámci volané funkce se nevolá metoda CoTick.

4.2. Typy

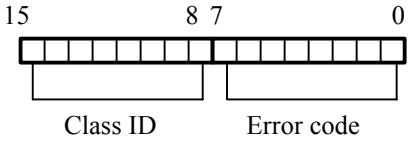
4.2.1. Třída TCoDevice

```
PCoDevice = ^TCoDevice;
TCoDevice = object;
```

4.2.2. Struktura TCoStatus

```
TCoStatus = Word;
```

TCoStatus je typ pro návratový kód naprosté většiny funkcí a metod knihovny **CoBase**. Návratový kód má následující strukturu:



Nižších 8 bitů návratového kódu obsahuje jednu z konstant **CST_** jak byly uvedeny v kapitole 4.1.1. Vyšších 8 bitů obsahuje identifikátor třídy zařízení, tj. konstanty **DEV_CLASS_** a identifikují zařízení, které chybu vyvolalo. Pokud konkrétní zařízení není specifikováno, je vyšších 8 bitů nulových. V případě že návratová hodnota funkce je **CST_SUCCESS** nebo **CST_PENDING**, pak je **vždy** všech osm horních bitů specifikujících třídu zařízení nulových.

4.2.3. Struktura TCoAddress

```
PCoAddress = ^TCoAddress;
TCoAddress = record
  Size     : Byte;
  Reserved : array[0..MAX_COMM_ADDR_SIZE - 1] of Byte;
end;
```

Typ **TCoAddress** je zobecněná adresa stanice. Obecná adresa rezervuje až 15 (MAX_COMM_ADDR_SIZE) bajtů pro specifikaci adresy stanice. Konkrétní zařízení definují své specifické struktury o stejné délce jako je tato struktura (kvůli jednoduchému přetypování). První položka struktury adresy **Size** musí vždy určovat správnou velikost adresy. Obsah položky **Reserved** definují konkrétní zařízení.

Pro práci s obecnou adresou jsou určeny funkce **CoIsAddrValid** a **CoCopyAddress**.

4.2.4. Struktura TCoTime

```
TCoTime = Longint;
```

Čas v milisekundách.

4.2.5. Struktura TCoDeviceTimeouts

```
TCoDeviceTimeouts = record
  SendTimeout    : TCoTime;
  RecvTimeout    : TCoTime;
  IoctlTimeout   : TCoTime;
  ConnectTimeout : TCoTime;
  DisconnTimeout : TCoTime;
  BindTimeout    : TCoTime;
  UnbindTimeout  : TCoTime;
  AcceptTimeout  : TCoTime;
  ListenTimeout  : TCoTime;
  OptTimeout     : TCoTime;
end;
```

Struktura **TCoDeviceTimeouts** společně s funkcemi **CoSetTimeouts** a **CoGetTimeouts** slouží k manipulaci s nastavení časových limitů pro provádění všech standardních operací se zařízeními.

Jednotlivé položky reprezentují časové limity v milisekundách. Hodnota nula znamená, že daná operace nemá žádný omezující časový limit. Následující tabulka uvádí, ke kterým metodám (operacím) se vztahují jednotlivé položky.

Položka	**Funkce**
SendTimeout	CoSendBuffer, CoSendBufferTo
RecvTimeout	CoRecvBuffer, CoRecvBufferTo
IoctlTimeout	CoIoctl
ConnectTimeout	CoConnect
DisconnTimeout	CoDisconnect
BindTimeout	CoBind
UnbindTimeout	CoUnbind
AcceptTimeout	CoAccept
ListenTimeout	CoListen
OptTimeout	CoSetOption, CoGetOption, CoSetOptionString, CoGetOptionString

Implicitní nastavení časových limitů je specifické pro konkrétní zařízení a je uvedené v dokumentaci příslušného zařízení.

4.2.6. Typ TCoAcceptFilterProc

```
TCoAcceptFilterProc = function ( Address: PCoAddress ): Boolean;
```

Tento procedurální typ reprezentuje funkci volanou při navazování spojení s protější stanicí v rámci operace **CoAccept**. Jedná se o „callback“, kterým lze filtrovat příchozí spojení. Viz. metoda **CoAccept** v kapitole 7.1.2.10.

4.2.7. Typ TCoSleepProcedure

```
TCoSleepProcedure = procedure ( Miliseconds: Word );
```

viz. proměnná **CoSleep** v kapitole 5.1.1

4.2.8. Struktura TOverlapped

```
POverlapped = ^TOverlapped;
TOverlapped = record
  Reserved: array[0..5] of Byte;
end;
```

Struktura TOverlapped slouží k uchovaní stavu prováděné tzv. neblokující operace (ukazatel na strukturu overlapped jednoznačně identifikuje neblokující operaci pro další použití v metodách CoGetIOResult příp. CoCancelIO). viz. kapitola 7.2. Aplikace nesmí v žádném případě měnit obsah struktury TOverlapped po dobu provádění operace.

5. Proměnné

5.1.1. Proměnná CoSleep

```
CoSleep : TCoSleepProcedure;
```

CoSleep je proměnná procedurálního typu. Obsahuje ukazatel na proceduru, která provede uspání současného procesu na zadaný počet milisekund. Tato funkce se volá z blokujících metod. Standardně proměnná **CoSleep** ukazuje na prázdnou proceduru. Pokud by však byl v aplikaci použít o.s. ReTOs, je potřeba tuto proměnnou přesměrovat na vlastní proceduru, která provede zastavení současně běžícího procesu.

Příklad:

```
procedure RetosCoSleep( Miliseconds: Word );
begin
  { podle zrychlení časovače: }
  Wait( (Milisecods + 49) div 50 );
end;
```

Ihned po inicializaci o.s. Retos provést přiřazení:

```
CoBase.CoSleep := RetosCoSleep;
```

6. Funkce a procedury

6.1.1. Funkce CoCreateDevice

Funkce **CoCreateDevice** slouží k vytvoření instance specifikovaného zařízení nebo protokolu.

```
function  CoCreateDevice( const AClassName, AInstanceName,
            AParams: string; AAttachTo: PCoDevice;
            var ADevice: PCoDevice ): TCoStatus;
```

Parametry:

AClassName	Název registrované třídy zařízení.
AInstanceName	Název pro vytvořenou instanci zařízení. Pokud je uveden prázdný řetězec, instance zařízení nebude pojmenovaná.
AParams	Inicializační parametry zařízení. Obsah tohoto textového řetězce je poplatný danému zařízení (viz. metoda **CoSetOptionString**)
AAttachTo	Zařízení nižší vrstvy, ke kterému se má vytvářené zařízení připojit. Pokud je uvedena hodnota **nil**, je potřeba později provést propojení pomocí metody **CoAttachDevice**.
ADevice	Proměnná, do které se uloží ukazatel na vytvořenou instanci zařízení.

Návratové hodnoty:

V případě úspěchu vrací funkce návratový kód CST_SUCCESS. V opačném případě může vrátit jednu z následujících hodnot:

CST_ERR_INVCLASS	Požadovaná třída zařízení **AClassName** není zaregistrovaná.
CST_ERR_MEMOVF	Není dostatek operační paměti pro vytvoření datových struktur instance zařízení.
CST_ERR_DUPNAME	Uvedený název instance **AInstanceName** byl již zaregistrován.
CST_ERR_INVALID	Tato operace je pro dané zařízení neplatná (např. zařízení síťového adaptéru nelze připojit k nižší vrstvě)
CST_ERR_LISTFULL	Seznam zařízení nižší vrstvy je plný, zařízení nelze připojit.
CST_ERR_SYNTAX	Chyba syntaxe konfiguračního řetězce **AParams**.
CST_ERR_INVRANGE	Hodnota některého z parametrů konfiguračního řetězce není v platném rozsahu.

Poznámky:

Pokud se instance zařízení pojmenuje (položka **AInstance**), pak bude zaregistrována v seznamu pojmenovaných instancí zařízení. Z tohoto seznamu ji lze kdykoli vyjmout pomocí funkce **CoDeleteDevice**.

Ukazatel na instanci předaný parametrem **ADevice** je potřeba v okamžiku ukončení používání zařízení uvolnit funkcí **CoReleaseDevicePtr**.

*Zařízení by se vždy mělo vytvářet pomocí funkce **CoCreateDevice**. S ohledem na budoucí změny knihovny není vhodné inicializovat zařízení přímo pomocí konstruktoru třídy zařízení.*

6.1.2. Funkce CoOpenDevice

Funkce **CoOpenDevice** slouží k otevření dříve vytvořeného a pojmenovaného zařízení (získání ukazatele na zařízení).

```
function  CoOpenDevice( const AInstanceName: string ): PCoDevice;
```

Parametry:

AInstanceName Název instance dříve vytvořeného zařízení.

Návratové hodnoty:

Funkce vrací ukazatel na požadovanou instanci zařízení. Pokud instance s jménem **AInstanceName** neexistuje, pak funkce vrací hodnotu **nil**.

Poznámky:

Funkce **CoOpenDevice** zvyšuje počítadlo referencí instance o jedničku. V okamžiku kdy už nepotřebujeme ukazatel na tuto instanci, je potřeba zavolat funkci **CoReleaseDevicePtr**.

6.1.3. Procedura CoDeleteDevice

Procedura **CoDeleteDevice** odstraní instanci zařízení ze seznamu pojmenovaných instancí zařízení.

```
procedure CoDeleteDevice( ADevice: PCoDevice );
```

Parametry:

ADevice Ukazatel na instanci zařízení.

Poznámky:

Pokud zařízení instance **ADevice** nebylo pojmenované, procedura **CoDeleteDevice** neprovede žádnou akci.

6.1.4. Procedura CoReleaseDevicePtr

Procedura **CoReleaseDevicePtr** uvolní ukazatel na instanci zařízení, popřípadě uvolní instanci z operační paměti.

```
procedure CoReleaseDevicePtr( ADevice: PCoDevice );
```

Parametry:

ADevice	Ukazatel na instanci zařízení.

Poznámky:

Procedura **CoReleaseDevicePtr** sníží počítadlo referencí instance zařízení o jedničku. Pokud počítadlo referencí dosáhne nuly, provede tato procedura uvolnění instance z operační paměti.

6.1.5. Funkce CoStatusToStr

Funkce **CoStatusToStr** převádí numerickou hodnotu návratového kódu na srozumitelný textový řetězec.

```
function  CoStatusToStr( AStatus: TCoStatus ): string;
```

Parametry:

AStatus	Návratový kód dříve volané funkce.

Návratové hodnoty:

Funkce vrací textový řetězec odpovídající návratovému kódu AStatus. Tento řetězec je ve tvaru: *(název třídy): ERR popis chyby*. Pokud třída nebo návratový kód nebyl registrován, je uveden v číselné podobě.

Poznámky:

Globální funkce **CoStatusToStr** převádí do textové podoby pouze standardní návratové kódy definované v kapitole 4.1.1 a je tedy vhodná spíše pro zobrazení návratových kódů globálních funkcí z této kapitoly. Implementace konkrétních zařízení mohou definovat své vlastní návratové kódy, které nejsou touto funkcí pokryty. Všechny standardní návratové kódy i návratové kódy specifické pro dané zařízení lze převést pomocí stejnojmenné metody **CoStatusToStr** třídy **TCoDevice**.

6.1.6. Funkce CoClassIdToStr

Funkce **CoClassIdToStr** převádí numerický identifikátor třídy zařízení na textový řetězec.

```
function  CoClassIdToStr( AClassId: Word ): string;
```

Parametry:

AClassId	Identifikátor třídy zařízení (konstanta DEV_CLASS_)

Návratové hodnoty:

Funkce vrací textový identifikátor odpovídající třídě zařízení **AClassId**. Pokud třída **AClassId** nebyla registrována, funkce vrací řetězec ve tvaru *CLASSnnn*, kde nnn je identifikátor třídy zařízení v číselné podobě.

6.1.7. Procedura CoCopyAddress

Procedura **CoCopyAdresss** slouží k kopírování adresy stanice **TCoAddress**.

```
procedure CoCopyAddress( ADest: PCoAddress; ASource: PCoAddress );
```

Parametry:

ADest	Ukazatel na místo, kam bude adresa překopírovaná.
ASource	Ukazatel na adresu, která bude kopírovaná

Poznámky:

Funkce kopíruje pouze tolik bajtů adresy, kolik je uvedeno v položce **TCoAddress.Size**. Blok paměti, na který ukazuje parametr **Adest,** musí být dostatečně velký.

7. Třídy

7.1. TCoDevice

7.1.1. Položky

Programátor by nikdy neměl přímo přistupovat k položkám třídy **TCoDevice** až na dvě zde uvedené výjimky.

```
CO_ClassID      :  Byte;
```

Položka **CO_ClassID** uchovává identifikátor třídy zařízení. Tento identifikátor určuje zařízení v řetězci při volání metod **CoIoctl**, **CoGetOption**, **CoSetOption**, **CoGetOptionString** a **CoSetOptionString**. Položka **CO_ClassID** je určena pouze ke čtení, jakákoli modifikace je nepřípustná.

```
CO_InstanceName : PString;
```

Položka **CO_InstanceNam**e uchovává ukazatel na název třídy zařízení nebo **nil**, pokud zařízení není pojmenované. Položka **CO_InstanceName** je určena pouze ke čtení, jakákoli modifikace je nepřípustná.

7.1.2. Metody

7.1.2.1. Metoda CoSendBuffer

Metoda **CoSendBuffer** slouží k odeslání dat pomocí zařízení s navázaným spojením.

```
function  CoSendBuffer( ABuffer: Pointer; ABuffSize: Word;
                AFlags: Word; AOverlapped: POverlapped): TCoStatus;
```

Parametry:

ABuffer	Ukazatel na blok data k odeslání.
ABuffSize	Velikost dat k odeslání, na které ukazuje parametr ABuffer.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má odeslání proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby odesílání vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTBOUND	Zařízení není ve stavu, kdy má přiřazenou vlastní lokální adresu. Tj. nebyla zavolána metoda **CoBind**.
CST_ERR_NOTCONN	Zařízení není ve stavu navázaného spojení. Tj. nebyla zavolána metoda **CoConnect**.
CST_ERR_BUFFSIZE	Velikost odesílaných dat
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. Připadá v úvahu pouze u neblokujících operací.
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen. Odeslání pravděpodobně neproběhlo.
CST_ERR_HOSTUNREACH	Cílová stanice je nedostupná.
CST_ERR_NETUNREACH	Síť cílové stanice je nedostupná.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Metoda **CoSendBuffer** provádí odeslání dat zařízením, u kterého bylo navázáno spojení s jinou stanicí metodou **CoConnect**. Pokud spojení není navázáno, metoda **CoSendBuffer** vrací chybový kód. Adresa cílové stanice se specifikuje při volání metody **CoConnect**.

Do dokončení operace nesmí aplikace měnit obsah předaných dat k odvysílání !

Pokud spojení nebylo navázáno, popř. pokud dané zařízení nepodporuje navazování spojení lze k odvysílání dat použít metodu **CoSendBufferTo**, u které je možné specifikovat adresu cílové stanice.

Pokud odvysílání není dokončeno do stanoveného časového limitu, vrací metoda **CoSendBuffer** návratový kód CST_ERR_TIMEOUT. Časový limit pro provedení operace lze nastavit pomocí metody **CoSetTimeouts**.

7.1.2.2. Metoda CoSendBufferTo

Metoda **CoSendBufferTo** slouží k odeslání dat pomocí zařízení bez navázaného spojení.

```
function  CoSendBufferTo( ABuffer: Pointer; ABuffSize: Word;
            AFlags: Word; AAddress: PCoAddress;
            AOverlapped: POverlapped): TCoStatus;
```

Parametry:

ABuffer	Ukazatel na blok dat k odeslání.
ABuffSize	Velikost dat k odeslání, na které ukazuje parametr ABuffer.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
AAddress	Ukazatel na adresu cílové stanice. *V případě, že zde uvedeme hodnotu **nil**, chování metody **CoSendBufferTo** bude identické s **CoSendBuffer**.*
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má odeslání proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

Do dokončení operace nesmí aplikace měnit obsah předaných dat k odvysílání !

V případě chyby odesílání vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTBOUND	Zařízení není ve stavu, kdy má přiřazenou vlastní lokální adresu. Tj. nebyla zavolána metoda **CoBind**.
CST_ERR_BUFFSIZE	Velikost odesílaných dat
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen. Odeslání pravděpodobně neproběhlo.
CST_ERR_HOSTUNREACH	Cílová stanice je nedostupná.
CST_ERR_NETUNREACH	Síť cílové stanice je nedostupná.
CST_ERR_INVADDR	Adresa cílové stanice je neplatná

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Metoda **CoSendBufferTo** slouží k odeslání dat pomocí zařízení bez navázaného spojení. Pro odesílání dat při navázaném spojení použijte metodu **CoSendBuffer**.

Pokud odvysílání není dokončeno do stanoveného časového limitu, vrací metoda **CoSendBufferTo** návratový kód CST_ERR_TIMEOUT. Časový limit pro provedení operace lze nastavit pomocí metody **CoSetTimeouts**.

7.1.2.3. Metoda CoRecvBuffer

Metoda **CoRecvBuffer** slouží k příjmu dat pomocí zařízení s navázaným spojením.

```
function  CoRecvBuffer( ABuffer: Pointer; ABuffSize: Word;
            ARecvLen: PWord; AFlags: Word;
            AOverlapped: POverlapped): TCoStatus;
```

Parametry:

ABuffer	Ukazatel do paměti, kam budou překopírována přijatá data.
ABuffSize	Velikost připraveného bloku, na který ukazuje ukazatel ABuffer. Tj. maximální velikost přijatých dat.
ARecvLen	Ukazatel na proměnnou, do které bude uložen počet bajtů překopírovaných dat.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má příjem proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby příjmu vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTBOUND	Zařízení není ve stavu, kdy má přiřazenou vlastní lokální adresu. Tj. nebyla zavolána metoda **CoBind**.
CST_ERR_BUFFSIZE	Velikost odesílaných dat
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Metoda **CoRecvBuffer** je určena pro příjem dat zařízeními s navázaným spojením. Adresa zdrojové stanice přijatých dat se určuje při navazování spojení metodou **CoConnect**. Pokud zařízení nenavázalo spojení, nebo navázání spojení neumožňuje, lze pro příjem použít metodu **CoRecvBufferFrom**.

Pokud příjem neproběhne do stanoveného časového limitu, vrací metoda **CoRecvBuffer** návratový kód CST_ERR_TIMEOUT. Časový limit pro provedení operace lze nastavit pomocí metody **CoSetTimeouts**.

7.1.2.4. Metoda CoRecvBufferFrom

Metoda **CoRecvBufferFrom** slouží k příjmu dat pomocí zařízení s navázaným spojením.

```
function  CoRecvBufferFrom( ABuffer: Pointer; ABuffSize: Word;
            ARecvLen: PWord; AFlags: Word; AAddress: PCoAddress;
            AOverlapped: POverlapped): TCoStatus;
```

Parametry:

ABuffer	Ukazatel do paměti, kam budou překopírována přijatá data.
ABuffSize	Velikost připraveného bloku, na který ukazuje ukazatel ABuffer. Tj. maximální velikost přijatých dat.
ARecvLen	Ukazatel na proměnnou, do které bude uložen počet bajtů překopírovaných dat.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
AAddress	Ukazatel na blok paměti, do kterého bude uložena adresa zdrojové stanice přijatých dat. Aplikace musí na tomto místě vyhradit dostatečný prostor pro překopírování adresy. *Pokud nastavíme parametr na hodnotu **nil,** chování metody bude identické s chováním metody **CoRecvBuffer.***
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má příjem proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby příjmu vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.

CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTBOUND	Zařízení není ve stavu, kdy má přiřazenou vlastní lokální adresu. Tj. nebyla zavolána metoda **CoBind**.
CST_ERR_BUFFSIZE	Velikost odesílaných dat
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Metoda **CoRecvBufferFrom** je určena k přijmu dat pomocí zařízení bez navázaného spojení. Pro příjem dat pomocí zařízení s navázaným spojením použijte metodu **CoRecvBuffer**.

Pokud odvysílání není dokončeno do stanoveného časového limitu, vrací metoda **CoSendBufferTo** návratový kód CST_ERR_TIMEOUT. Časový limit pro provedení operace lze nastavit pomocí metody **CoSetTimeouts**.

7.1.2.5. Metoda CoBind

Metoda **CoBind** slouží k inicializaci a nastavení lokální adresy zařízení (stanice), alokování prostředků a přepnutí zařízení do aktivního stavu.

```
function  CoBind( AAddress: PCoAddress; AFlags: Word;
            AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

AAddress	Lokální adresa zařízení. Pokud se zde uvede hodnota **nil**, použije se implicitní adresa zařízení (pokud to konkrétní zařízení umožňuje)
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Lze zde použít také standardní konstanty: COF_PROPAGATE – Rekurzivně provést operaci Bind i ve všech nižších vrstvách. COF_QUIET – CoBind nevrací chybu (CST_ERR_ISBOUND), pokud je již při volání metody zařízení v aktivním stavu.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby příjmu vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_ISBOUND	Zařízení je již v aktivním stavu.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen.
CST_ERR_PROTOREG	Stejný typ zařízení nebo protokolu byl již u nižší vrstvy zaregistrován. Tato chyba vznikne např. v případě, pokud se budeme snažit vytvořit dvě IPv4 zařízení nad zařízením ETH01.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Metoda **CoBind** slouží k nastavení lokální adresy zařízení. Dokud není zavolána tato metoda není možné pomocí zařízení vysílat ani přijímat žádna data. Kromě přiřazení lokální adresy, metoda obvykle provádí veškerou inicializaci a alokování prostředků (jak v software tak i v hardware, pokud je potřeba) podle parametrů nastavených metodami **CoSetOption** příp. **CoSetOptionString**.

Pro uvolnění prostředků a vrácení zařízení do stavu před voláním **CoBind** slouží metoda **CoUnbind**.

7.1.2.6. Metoda CoUnbind

Metoda **CoUnbind** slouží k zrušení lokální adresy zařízení, uvolnění alokovaných prostředků a přepnutí zařízení do neaktivního stavu.

```
function  CoUnbind( AFlags: Word; AOverlapped: POverlapped ):
            TCoStatus;
```

Parametry:

AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Lze zde použít také standardní konstanty:
	COF_PROPAGATE — Rekurzivně provést operaci unbind i ve všech nižších vrstvách.
	COF_QUIET — CoUnbind nevrací chybu (CST_ERR_NOTBOUND), pokud je již při volání metody zařízení v neaktivním stavu.
	COF_IFNOTUSED — Provést operaci Unbind jen tehdy, pokud zařízení není používáno jiným zařízením. Interní příznak, není potřeba uvádět.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby příjmu vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTBOUND	Zařízení není v aktivním stavu.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

7.1.2.7. Metoda CoConnect

Metoda **CoConnect** slouží k navázání spojení s jinou stanicí (či uzlem).

```
function  CoConnect( AAddress: PCoAddress; AFlags: Word;
            AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

AAddress	Adresa stanice(uzlu), se kterou se navazuje spojení. Pokud se zde uvede hodnota **nil**, použije se implicitní adresa cílové stanice (pokud to konkrétní zařízení umožňuje)
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Lze zde použít také standardní konstanty: COF_PROPAGATE – Rekurzivně provést operaci Connect i ve všech nižších vrstvách. COF_QUIET – CoConnect nevrací chybu (CST_ERR_ISCONN), pokud je již při volání metody spojení navázáno.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby příjmu vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTBOUND	Zařízení není v aktivním stavu.
CST_ERR_ISCONN	Spojení již bylo navázáno.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen.
CST_ERR_HOSTUNREACH	Cílová stanice je nedostupná.
CST_ERR_NETUNREACH	Síť cílové stanice je nedostupná.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Při navázaném spojení se pro vysílání a příjem dat používají metody **CoSendBuffer** a **CoRecvBuffer**. Pro ukončení spojení se použije metoda **CoDisconnect**.

7.1.2.8. Metoda CoDisconnect

Metoda **CoDisconnect** slouží k ukončení navázaného spojení.

```
function  CoDisconnect( AFlags: Word;
            AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

AFlags — Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci.
Lze zde použít také standardní konstanty:

COF_PROPAGATE	Rekurzivně provést operaci Disconnect i ve všech nižších vrstvách.
COF_QUIET	CoDisconnect nevrací chybu (CST_ERR_NOTBOUND), pokud je již při volání metody zařízení v neaktivním stavu.
COF_IFNOTUSED	Provést operaci Disconnect jen tehdy, pokud zařízení není používáno jiným zařízením. Interní příznak, není potřeba uvádět.

AOverlapped — Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby příjmu vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTCONN	Spojení není navázáno.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen.
CST_ERR_HOSTUNREACH	Cílová stanice je nedostupná.
CST_ERR_NETUNREACH	Síť cílové stanice je nedostupná.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

7.1.2.9. Metoda CoListen

Metoda **CoListen** přepíná zařízení do stavu pasivního čekání na příchozí spojení.

```
function  CoListen( ABackLog: Word; AFlags: Word;
            AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

ABackLog	Maximální počet žadatelů čekajících ve frontě na potvrzení přijetí spojení.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTBOUND	Zařízení není v aktivním stavu.
CST_ERR_ISCONN	Spojení již bylo navázáno.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

7.1.2.10. Metoda CoAccept

Metoda **CoAccept** slouží pro přijetí příchozího požadavku na navázání spojení.

```
function  CoAccept( AAddress: PCoAddress; var ASocket: PCoDevice;
            AFilter: TCoAcceptFilterProc; AFlags: Word;
            AOverlapped: POverlapped): TCoStatus;
```

Parametry:

AAddress	Ukazatel na blok v paměti, kam bude překopírována adresa stanice se kterou bylo spojení navázáno.
ASocket	Proměnná, která po přijetí spojení obdrží ukazatel na nové zařízení, které bude reprezentovat vzniklé spojení.
AFilter	Ukazatel na „callback“ proceduru provádějící filtraci příchozích požadavků na spojení.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_NOTBOUND	Zařízení není v aktivním stavu.
CST_ERR_ISCONN	Spojení již bylo navázáno.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl překročen.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

7.1.2.11. Metoda CoIoctl

Metoda **CoIoctl** slouží k provádění nestandardních operací se zařízeními v řetězci zařízení.

```
function  CoIoctl( AClassId: Word; AFlags: Word; ACode: Word;
            ABuffer: Pointer; ABuffSize: Word;
            AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

AClassId	Identifikátor třídy zařízení, na kterém se požadovaná operace má provést.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
ACode	Identifikátor operace IOCTL_. Tyto identifikátory definuje konkrétní zařízení.
ABufer	Blok paměti vyhrazený pro předání dat. Struktura a délka dat je poplatná konkrétní operaci a třídě zařízení.
ABuffSize	Velikost bloku paměti, na který ukazuje parametr ABuffer.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl vyčerpán.
CST_ERR_INVCLASS	V řetězci zařízení nebylo nalezeno zařízení třídy **AClassId**.
CST_ERR_INVOPT	Daná třída zařízení nepodporuje uvedený identifikátor operace **ACode**.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Do dokončení operace nesmí aplikace měnit obsah předaných dat.

7.1.2.12. Metoda CoSetOption

Metoda **CoSetOption** slouží k nastavení jednoho parametru zařízení v řetězci zařízení.

```
function  CoSetOption( AClassId: Word; AOption: Word;
            ABuffer: Pointer; ABuffSize: Word; AFlags: Word;
            AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

AClassId	Identifikátor třídy zařízení, jehož parametr se má nastavit. Tj. konstanta DEV_CLASS_.
AOption	Identifikátor parametru, který se má u daného zařízení nastavit. Tj. konstanta COPT_.
ABuffer	Blok operační paměti ve kterém je uložena hodnota parametru. Struktura tohoto bloku je poplatná konkrétnímu parametru a třídě zařízení.
ABuffSize	Velikost bloku paměti, na který ukazuje parametr ABuffer.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby vrací metoda jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl vyčerpán.
CST_ERR_INVCLASS	V řetězci zařízení nebylo nalezeno zařízení třídy **AClassId**.
CST_ERR_INVOPT	Daná třída zařízení nepodporuje uvedený identifikátor parametru **AOption**.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Do dokončení operace nesmí aplikace měnit obsah předaných dat !

7.1.2.13. Metoda CoGetOption

Metoda **CoGetOption** slouží ke čtení jednoho parametru zařízení v řetězci zařízení.

```
function  CoGetOption( AClassId: Word; AOption: Word;
            ABuffer: Pointer; ABuffSize: Word; AFlags: Word;
            AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

AClassId	Identifikátor třídy zařízení, jehož parametr se má nastavit. Tj. konstanta DEV_CLASS_.
AOption	Identifikátor parametru, který se má u daného zařízení nastavit. Tj. konstanta COPT_.
ABuffer	Blok operační paměti, ve kterém bude po provedení metody uložena hodnota parametru. Struktura tohoto bloku je poplatná konkrétnímu parametru a třídě zařízení.
ABuffSize	Velikost bloku paměti, na který ukazuje parametr ABuffer.
AFlags	Doplňující příznaky COF_. Tyto příznaky definuje konkrétní zařízení a jsou uvedeny v jeho dokumentaci. Pro standardní chování metody zde uveďte nulovou hodnotu.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

V případě chyby metoda vrací jednu z následujících hodnot:

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl vyčerpán.
CST_ERR_INVCLASS	V řetězci zařízení nebylo nalezeno zařízení třídy **AClassId**.

CST_ERR_INVOPT	Daná třída zařízení nepodporuje uvedený identifikátor parametru **AOption**.
CST_ERR_INVRANGE	Hodnota parametru není v platném rozsahu.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

7.1.2.14. Metoda CoSetOptionString

Metoda **CoSetOptionString** umožní nastavit jeden nebo více parametrů zařízení zformátovaných do konfiguračního textového řetězce.

```
function  CoSetOptionString( AClassId: Word; const AString: string;
            AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

AClassId	Identifikátor třídy zařízení, jehož parametr se má nastavit. Tj. konstanta DEV_CLASS_.
AString	Konfigurační textový řetězec reprezentující nastavení jednoho nebo více parametrů zařízení.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.
CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl vyčerpán.
CST_ERR_INVCLASS	V řetězci zařízení nebylo nalezeno zařízení třídy **AClassId**.
CST_ERR_SYNTAX	Chyba syntaxe konfiguračního řetězce **AString**.
CST_ERR_INVRANGE	Hodnota některého z parametrů konfiguračního řetězce není v platném rozsahu.

Mimo výše uvedené chybové kódy, může metoda vrátit jiný chybový kód specifický pro dané zařízení.

Poznámky:

Do dokončení operace nesmí aplikace měnit obsah předaného konfiguračního řetězce!

Syntaxe textového řetězce AString se řídí podle následujících pravidel:

PARAMSTR -> EQUATION {EQUATION}
EQUATION -> PARNAME '=' PARVALUE
PARNAME -> IDENTIFIER
PARVALUE -> VALUE | STRING | IDENTIFIER
IDENTIFIER -> ('A'..'Z','_') {'A'..'Z','_','0'..'9'}
STRING -> '"' any characters '"' | ''' any characters '''
VALUE -> decimal number | '$' hexadecimal number

x | y je varianta x nebo y
{ x } opakování x 0-krát nebo vícekrát

Příklady:

```
1. IOBASE=$300 MODE=CSMACD
2. IPADDR="192.168.1.1" IPMASK="255.255.255.0" ARPTMO=60000
```

7.1.2.15. Metoda CoGetOptionString

Metoda **CoGetOptionString** slouží k získání úplného konfiguračního řetězce zařízení.

```
function  CoGetOptionString( AClassId: Word; AString: PString;
            AMaxLength: Byte; AOverlapped: POverlapped ): TCoStatus;
```

Parametry:

AClassId	Identifikátor třídy zařízení, jehož parametr se má nastavit. Tj. konstanta DEV_CLASS_.
AString	Blok v paměti pro uložení konfiguračního řetězce zařízení.
AMaxLength	Velikost bloku na který ukazuje parametr AString.
AOverlapped	Ukazatel na strukturu TOverlapped nebo hodnota **nil**, pokud má operace proběhnout jako blokující.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS. Pokud je metoda volána s parametrem AOverlapped různým od **nil**, metoda může vrátit hodnotu CST_PENDING, což znamená, že operace se provádí a bude dokončena později. Pro zjištění stavu operace je v takovém případě potřeba použít metodu **CoGetIOResult**.

CST_ERR_NOTSUPP	Metoda není daným zařízením podporována.
CST_ERR_INVALID	Jeden z parametrů nemá platnou hodnotu.
CST_ERR_MEMOVF	Nedostatek operační paměti pro provedení operace.

CST_ERR_ABORTED	Operace byla zrušena voláním metody **CoCancelIO**. *Připadá v úvahu pouze u neblokujících operací.*
CST_ERR_TIMEOUT	Časový limit pro provedení operace byl vyčerpán.
CST_ERR_INVCLASS	V řetězci zařízení nebylo nalezeno zařízení třídy **AClassId**.

Poznámky:

Pokud je hodnota AMaxLength menší než celková délka konfiguračního řetězce, je kopírovaný řetězec bezpodmínečně oříznut na délku AMaxLength.

7.1.2.16. Metoda CoGetIOResult

Metoda **CoGetIOResult** slouží ke zjištění stavu a výsledku neblokující operace.

```
function  CoGetIOResult( AOverlapped: POverlapped;
            AFlags: Word ): TCoStatus;
```

Parametry:

AOverlapped	Ukazatel na strukturu TOverlapped identifikující neblokující operaci.
AFlags	Upřesňující příznaky, pro standardní chování zde uveďte hodnotu 0. Jinak je možné použít jednu z následujících konstant:
	COF_WAIT — Metoda **CoGetIOResult** se nevrátí do té doby než bude neblokující operace dokončena.
	COF_NOTICK — Metoda **CoGetIOResult** neprovádí krok automatu zařízení. (Pouze pro interní účely.)

Návratové hodnoty:

CST_SUCCESS	Neblokující operace byla úspěšně dokončena.
CST_PENDING	Neblokující operace je v rozpracovaném stavu a bude dokončena později.

V případě chyby metoda **CoGetIOResult** vrací jednu z konstant CST_ERR_. Tyto chybové návratové kódy jsou definovány u příslušné metody CoXXX.

Poznámky:

Metoda **CoGetIOResult** (stejně tak i ostatní metody CoXXX) nevracejí přímo konstanty CST_ERR_. Chybový kód je obsažen v nižším bajtu návratové hodnoty. Ve vyšším bajtu je uveden identifikátor třídy zařízení, ve kterém chyba nastala. Pouze konstanty CST_SUCCESS a CST_PENDING jsou vraceny přímo (vyšší bajt je

nastaven vždy na nulu). Zpracování chyb po neblokující operaci může vypadat následovně.

```
var
  Device : PCoDevice;
  Ovlp   : TOverlapped;
  Status : TCoStatus;
.
.
.
Status := Device^.CoXXX( , , , @Ovlp );

case Status of
  CST_SUCCESS:  { Operace dokončena }
  CST_PENDING:  { Operace probíha }
else
  { Chyba operace }
end;
.
.
.
Status := Device^.CoGetIOResult( @Ovlp, 0 );

case Status of
  CST_SUCCESS:  { Operace dokončena }
  CST_PENDING:  { Operace probíhá }
else
  { Chyba operace }
end;
.
.
.
```

7.1.2.17. Metoda CoCancelIO

Metoda **CoCancelIO** slouží k předčasnému ukončení probíhající neblokující operace.

```
function  CoCancelIO( AOverlapped: POverlapped; AFlags: Word ):
            TCoStatus;
```

AOverlapped — Ukazatel na strukturu TOverlapped identifikující neblokující operaci.

AFlags — Upřesňující příznaky, pro standardní chování zde uveďte hodnotu 0. Jinak je možné použít jednu z následujících konstant:

- COF_WAIT — Metoda **CoCancelIO** se nevrátí do té doby než bude neblokující operace dokončena.

Návratové hodnoty:

Návratové hodnoty metody jsou identické s návratovými hodnotami metody **CoGetIOResult**. Pokud bude operace předčasně ukončena, metoda **CoGetCancelIO** nebo některé z dalších volání **CoGetIOResult** vrátí návratový kód CST_ABORTED.

Poznámky:

Metoda **CoCancelIO** nezaručuje okamžité zrušení neblokující operace. To jestli a za jak dlouho bude operace zrušena rozhodují implementace zařízení v řetězci zařízení. *V současné době mají všechna implementovaná zařízení a protokoly zabudované mechanismy, které umožní jakoukoli operaci ukončit prakticky ihned.*

Po zavolání metody CoCancelIO není možné ihned uvolnit strukturu TOverlapped. Je třeba nejdříve vyčkat na dokončení operace pomocí metody **CoGetIOResult** příp. zavolat **CoCancelIO** s příznakem COF_WAIT.

7.1.2.18. Metoda CoGetTimeouts

Metoda **CoGetTimeouts** slouží ke zjištění nastavených časových limitů pro provádění některých operací se zařízením.

```
procedure CoGetTimeouts( var ATimeouts: TCoDeviceTimeouts );
```

Parametry:

ATimeouts	Struktura **TCoDeviceTimeouts**, která je po návratu z metody vyplněna aktuálními hodnotami nastavení časových limitů. Struktura **TCoDeviceTimeouts** je popsaná v kapitole 4.2.5.

7.1.2.19. Metoda CoSetTimeouts

Metoda **CoSetTimeouts** slouží k nastavení časových limitů pro provádění některých operací se zařízením.

```
procedure CoSetTimeouts( const ATimeouts: TCoDeviceTimeouts );
```

Parametry:

ATimeouts	Struktura **TCoDeviceTimeouts**, která musí být před voláním metody vyplněna požadovanými hodnotami nastavení časových limitů. Struktura **TCoDeviceTimeouts** je popsaná v kapitole 4.2.5.

Poznámky:

Provedená změna časového limitu se promítne až u nově spuštěných operací CoXXX. Rozpracované neblokující operace proběhnou s původním nastavením časových limitů.

7.1.2.20. Metoda CoAttachDevice

Metoda **CoAttachDevice** provádí propojení zařízení k nižší vrstvě (jinému zařízení).

```
function  CoAttachDevice( ADevice: PCoDevice ): TCoStatus;
```

Parametry:

ADevice	Zařízení nižší vrstvy, ke kterému se má zařízení připojit.

Návratové hodnoty:

V případě úspěchu vrací metoda návratový kód CST_SUCCESS.
V případě chyby odesílání vrací metoda jednu z následujících hodnot:

CST_ERR_INVALID	Tato operace je pro dané zařízení neplatná (např. zařízení síťového adaptéru)
CST_ERR_ISATTCHED	Zařízení již bylo dříve připojeno k nižší vrstvě.
CST_ERR_LISTFULL	Seznam zařízení nižší vrstvy je plný, zařízení nelze připojit.

Poznámky:

Dokud není zařízení připojené k nižší vrstvě, není s ním možné provádět žádné operace CoXXX. V běžných případech není nutné volat metodu **CoAttachDevice** přímo. Zařízení lze propojit jednoduše rovnou při jejich vytváření funkcí **CoCreateDevice**.

Pro rozpojení řetězce zařízení slouží metoda **CoDetachDevice**.

7.1.2.21. Metoda CoDetachDevice

Metoda **CoDetachDevice** provádí odpojení zařízení od nižší vrstvy(zařízení).

```
procedure  CoDetachDevice;
```

Poznámky:
Stav zařízení při odpojení zůstává nezměněn. Pokud je zařízení v aktivním stavu popř. ve stavu navázaného spojení, zůstane v tomto stavu i nadále. Všechny rozpracované neblokující operace, které již byly předány nižší vrstvě budou i po rozpojení dokončeny. Operace které jsou uloženy ve frontách zařízení a nebyly ještě předány nižší vrstvě budou dokončeny s chybovým kódem CST_ERR_TIMEOUT.

Až na naprosté výjimky není vhodné odpojovat zařízení v aktivním stavu. U odpojeného zařízení není totiž možné provést žádnou další operaci, tj. ani **CoUnbind**.

Metodu **CoDetachDevice** není ve většině případů nutné volat přímo. Rozpojení zařízení se provede automaticky uvolněním posledního ukazatele na zařízení funkcí **CoReleaseDevicePtr**.

7.1.2.22. Metoda CoStatusToStr

Funkce **CoStatusToStr** převádí numerickou hodnotu návratového kódu na srozumitelný textový řetězec.

```
function CoStatusToStr( AStatus: TCoStatus ): String;
```

Parametry:

AStatus	Návratový kód dříve volané funkce.

Návratové hodnoty:

Funkce vrací textový řetězec odpovídající návratovému kódu AStatus. Tento řetězec je ve tvaru: *(název třídy): ERR popis chyby*. Pokud třída nebo návratový kód nebyl registrován, je uveden v číselné podobě.

Poznámky:

Na rozdíl od globální funkce **CoStatusToStr**, která převádí do textové podoby pouze standardní návratové kódy definované v kapitole 4.1.1, metoda **CoStatusToStr** převádí i návratové kód specifické pro dané zařízení.

7.2. Blokující a neblokující volání metod

Uživatelské metody zařízení CoXXX (CoSendBuffer apod)., lze volat buď tzv. **blokujícím** nebo **neblokujícím** způsobem.

Blokující volání znamená, že volaná metoda se nevrátí do té doby, než je prováděná operace dokončena (ať už úspěšně či neúspěšně). Blokující volání se provede u metod CoXXX jednoduše tak, že za parametr AOverlapped dosadíme hodnotu **nil**. Pokud používáme o.s. ReTOs je nutné správně nastavit globální proměnnou **CoSleep** (viz. kapitola 5.1.1).

příklad:

```
var
 Status : TCoStatus;
 Device : PCoDevice;

.
.
.

 Status := Device^.CoSendBuffer( @Data, SizeOf( Data ), 0, nil );

{ V tomto okamžiku je operace dokončena a její výsledek je v
  proměnné Status }

 case Status of
   CST_SUCCESS: ; { Operace proběhla v pořádku }
 else
   { Nastala chyba }
   WriteLn( Device^.CoStatusToStr( Status ) );
```

```
   ..
   ..
 end;
```

Naproti tomu při **neblokujícím volání** se metody CoXXX navracejí okamžitě. Jejich návratová hodnota, je v drtivé většině případů CST_PENDING, což znamená, že požadovaná operace je v rozpracovaném stavu a bude dokončena později. Stav operace lze průběžně sledovat pomocí metody **CoGetIOResult**.

příklad:

```
var
  Ovlp: TOverlapped;
  ..
  ..

Status := Device^.CoSendBuffer( @Data, SizeOf( Data ), 0, @Ovlp );

{ V tomto okamžiku je operace s nejvyssi pravdepodobnosti v
  rozpracovanem stavu }

while (Status = CST_PENDING) do
begin
  Status := Device^.CoGetIOResult( @Ovlp, 0 );
end;

{ V tomto okamžiku je operace dokončena a její výsledek je v
  uložen proměnné Status }

case  Status of
   CST_SUCCESS: ; { Operace proběhla v pořádku }
else
   { Nastala chyba }
   WriteLn( Device^.CoStatusToStr( Status ) );
   ..
   ..
end;
```

Následující operace zařízení lze volat neblokujícím způsobem:
CoSendBuffer, CoSendBufferTo, CoRecvBuffer, CoRecvBufferFrom, CoBind, CoUnbind, CoConnect, CoDisconnect, CoIoctl, CoSetOption, CoGetOption, CoSetOptionString, CoGetOptionsString

Ke každé neblokující operace je potřeba vytvořit místo pro strukturu TOverlapped.

Struktura TOverlapped jednoznačně identifikuje rozpracovanou operaci a používá se ke zjištění stavu operace metodou **CoGetIOResult** příp. předčasnému stornování operace metodou **CoCancelIO**.

Po dobu provádění operace nesmí aplikace obsah struktury TOverlapped měnit ani použít pro jinou operaci. Operace je dokončena v okamžiku, kdy buď samotná neblokující metoda nebo metoda **CoGetIOResult** vrátí hodnotu různou od CST_PENDING.

Hlavní výhodou neblokujících operací je to, že jich může běžet v jednom okamžiku několik. Aplikace tak může například současně vysílat a přijímat nebo vyslat více paketů najednou a pak teprve čekat na dokončení vysílání.

*Metoda **CoGetIOResult** provede kromě zjištění stavu operace také jeden krok automatu zařízení.*

Rozpracovanou operaci lze předčasně stornovat pomocí metody **CoCancelIO**.

příklad:

```
var
  Ovlp : TOverlapped;
  ..
  ..

Status := Device^.CoSendBuffer( @Data, SizeOf( Data ), 0, @Ovlp );

{ V tomto okamžiku je operace s nejvyssi pravdepodobnosti v
  rozpracovanem stavu }

while (Status = CST_PENDING) do
begin

  if (...) then
  begin
    { Předčasné ukončení operace }
    Device^.CoCancelIO( @Ovlp, 0 );
  end;

  Status := Device^.CoGetIOResult( @Ovlp, 0 );
end;
```

Metoda **CoCancelIO** nezaručí okamžité ukončení prováděné operace. Jak rychle bude operace zrušena záleží pouze na implementaci konkrétního zařízení. V každém případě je nutné po zavolání metody **CoCancelIO** vyčkat na dokončení operace pomocí volání metody **CoGetIOResult**. Do té doby je struktura TOverlapped dané operace ve vlastnictví zařízení a aplikace ji nemůže použít k jinému účelu.