Ovladač `EtcDrv` systému REX
Uživatelská příručka

REX Controls s.r.o.

Verze 2.50.4
Plzeň
17.5.2017

Obsah

Kapitola 1

Ovladač EtcDrv a systém REX

1.1 Úvod

V této příručce je popsáno používání ovladače EtcDrv pro připojení technických prostředků využívajících protokol EtherCAT [?] k řídicímu systému REX pro linuxové platformy. Ovladač umožňuje získávat vstupy a nastavovat výstupy z tzv. „process image“ a „CANopen over EtherCAT“. Ovladač byl vyvinut firmou REX Controls.

1.2 Požadavky na systém

Obecně lze říci, že pro použití ovladače EtcDrv musí být dodrženy minimální požadavky nutné k provozování řídicího systému REX. Pro konfiguraci ovladače postačuje běžný počítač PC (případně v průmyslovém provedení). Pro provozování ovladače na cílovém zařízení je potřeba vyhradit pro EtherCATjednu ethernetovou zásuvku. Ovladač není zatím implementován pro Windows (vyžaduje ovladač do jádra Windows, který umožní předávat přímo ethernetové packety aplikaci - v Linuxu je tato podpora automaticky).

Aby bylo možno ovladač využívat, musí být na vývojovém (konfiguračním) počítači a na cílovém zařízení (počítači) nainstalováno programové vybavení:

Vývojový počítač	
Operační systém	jeden ze systémů: Windows 7/8/10
Řídicí systém REX	verze pro operační systémy Windows
Cílové zařízení	
Řídicí systém REX	verze pro zvolené cílové zařízení s jedním z podporovaných operačních systémů: Linux Debian/XENOMAI/openWRT

V případě, že vývojový počítač je přímo cílovým zařízením (řídicí systém REX bude provozován v jedné z variant Windows), instaluje se pouze jedna kopie řídicího systému REX.

1.3 Instalace ovladače

Pro operační systém Windows se ovladač EtcDrv instaluje jako součást instalace řídicího systému REX. Pro nainstalování ovladače je nutné v instalačním programu systému REX zaškrtnout volbu Ovladač protokolu EtherCAT. Po typické instalaci se řídicí systém REX nainstaluje do cílového adresáře C:\Program Files\REX Controls\REX_<version>, kde <version> označuje verzi systému REX.

Po úspěšné instalaci se do cílového adresáře zkopírují soubory:

EtcDrv_H.dll – Konfigurační část ovladače EtcDrv.

EtcDrv_T.dll – Cílová část ovladače EtcDrv spouštěná exekutivou RexCore. Tato verze se používá pokud na cílovém zařízení běží operační systém Windows 7/8/10. Pro jinou cílovou platformu je na ni třeba nainstalovat příslušnou verzi systému REX.

DOC\EtcDrv_CZ.pdf – Tato uživatelská příručka.

Pro operační systém Linux je potreba nainstalovat balíček rex-etcdrvt. Pro variantu Linux/XENOMAI též balíčky kmod-edrv-mgr, kmod-edrv-dev a ovladač použité síťové karty pro XENOMAI (tj. jeden z kmod-edrv-via6105 kmod-edrv-r8169 kmod-edrv-r8139 - jiné nejsou zatím podpořeny). Dále je nutné označit zvolenou ethernetovou kartu, aby ji jádro linuxu ignorovalo a mohl ji využít subsystém XENOMAI v parametrech bootování. V openWRT image již toto je nastaveno pro eth1.

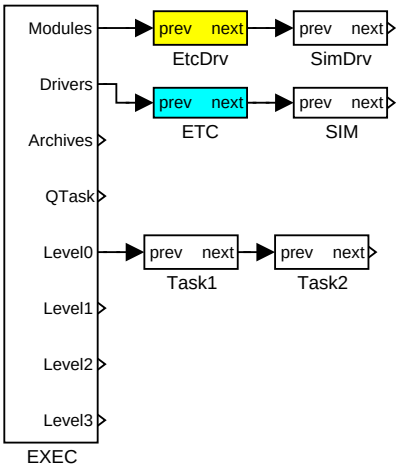
Kapitola 2

Zařazení ovladače do projektu aplikace

Zařazení ovladače do projektu aplikace spočívá v přidání ovladače do hlavního souboru projektu a z připojení vstupů a výstupů ovladače v řídicích algoritmech.

2.1 Přidání ovladače `EtcDrv` do projektu

Přidání ovladače `EtcDrv` do hlavního souboru projektu je znázorněno na obr. 2.1.



Obrázek 2.1: Příklad zařazení ovladače `EtcDrv` do projektu aplikace

Pro zařazení ovladače do projektu slouží dva zvýrazněné bloky. Nejprve je na výstup `Modules` bloku exekutivy `EXEC` připojen blok typu `MODULE` s názvem `EtcDrv`, který nemá žádné další parametry.

Druhý blok `ETC` typu `IODRV`, připojený na výstup `Drivers` exekutivy má parametry:

`modul` – jméno třídy ovladače, které se pro tento ovladač zadává: `EtcDrv`

`classname` – jméno třídy ovladače, které se pro tento ovladač zadává: `EtcDrv`POZOR! Jméno rozlišuje velká a malá písmena!

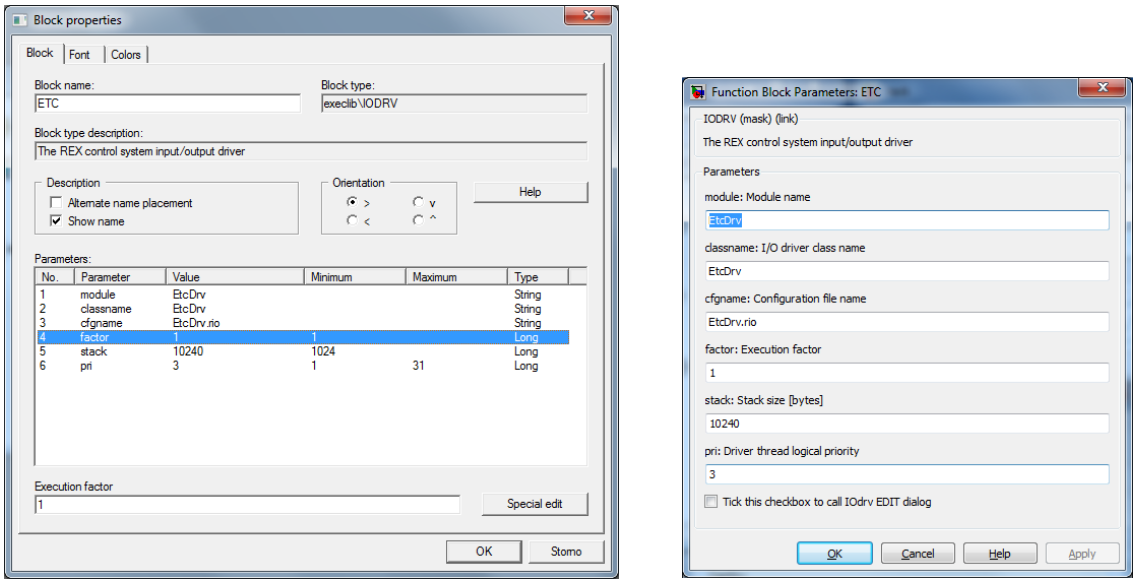
`cfgname` – jméno konfiguračního souboru ovladače. Vytváření konfiguračního souboru je popsáno v kapitole 3. Doporučeno je zadávat jej ve tvaru `<jméno_třídy>.rio`, kde přípona `.rio` (Rex Input Output) byla zavedena pro tento účel.

Další parametry slouží pro speciální případy a vyhovuje původní nastavení.

Jménem tohoto bloku, na obr. 2.1 zadaným jako `ETC`, začínají názvy všech vstupních a výstupních signálů připojených k tomuto ovladači.

Ovladač `EtcDrv` podporuje i úlohy běžící synchronně s komunikací. To se provede tak, že místo bloku typu `IODRV` se použije blok typu `TIODRV` (který má stejné parametry jako `IODRV`) a na jeho výstup `Tasks` přpipojíme blok typu `IOTASK` (má analogické parametry i význam jako blok typu `TASK`). Ovladač potom funguje tak, že přečte všechny „process image“, spustí algoritmus definovaný blokem `IOTASK` nastaví všechny „process image“ a čeká na další periodu.

Právě popsané parametry bloku `IODRV` se konfigurují v programu RexDraw v dialogovém okně, jak je patrno z obr. 2.2 a). Konfigurační dialog ovladače `EtcDrv`, popsaný v kapitole 3, se aktivuje po stisku tlačítka `Special Edit`.



a) b)

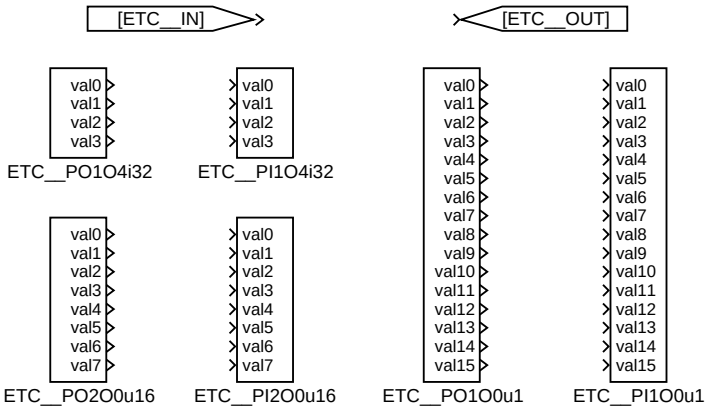
Obrázek 2.2: Konfigurace parametrů ovladače

V programovém systému Matlab Simulink se parametry bloku `IODRV` zadávají v parametrickém dialogu znázorněném na obrázku 2.2 b). Poslední parametr slouží k volání konfiguračního dialogu ovladače přímo z prostředí programu Matlab Simulink. Pokud při

editaci parametrů je invertováno zaškrtnutí tohoto parametru, bude po stisku tlačítek `OK` nebo `Apply` zavolán konfigurační dialog ovladače `EtcDrv`, popsaný v kap. 3.

2.2 Připojení vstupů a výstupů do řídicího algoritmu

Vstupy a výstupy z ovladačů se připojují do souborů s příponou `.mdl` jednotlivých úloh. V hlavním souboru projektu jsou soubory úloh uvedeny pouze odkazem v blocích typu `QTASK` nebo `TASK` nebo `IOTASK` připojovaných na výstupy `QTask`, `Level0`,..., `Level3` exekutivy. Pro připojení vstupů a výstupů z ovladače `EtcDrv` do řídicího systému REX lze použít bloky, znázorněné na obr. 2.3.



Obrázek 2.3: Příklady použití vstupně-výstupních bloků s ovladačem `EtcDrv`

Blok typu `From` sloužící pro připojení jednoho vstupu má parametr `Goto tag` roven `ETC__IN`, blok typu `Goto` používaný pro připojení jednoho výstupu má hodnotu parametru `Tag` (v programu RexDraw parametru `GotoTag`) rovnu `ETC__OUT`. Ostatní bloky mají přímo na začátku svého jména prefix `ETC` následovaný dvěma znaky `_` (podtržítko).

Přesněji řečeno, daný vstupně výstupní blok je považován systémem REX za blok připojený k ovladači `EtcDrv`, pokud jeho jméno (či, v případě bloků typu `From` a `Goto`, parametry `Goto tag` a `Tag`) začíná jménem bloku typu `IODRV` popisujícího daný ovladač (na obr. 2.1 to byl právě blok `ETC`). Začátek jména vstupního nebo výstupního bloku je od zbytku jména vždy oddělen dvěma znaky `_`.

Kdyby byl např. blok `ETC` z obr. 2.1 přejmenován na `XY`, začínala by jména všech vstupně výstupních bloků připojených k ovladači `EtcDrv` znaky `XY__`. Z praktických důvodů je však rozumnější volit prefix mnemotechnicky blízký názvu ovladače.

Zbytek jména vstupně výstupního bloku má jednu z následujících struktur:

`PO<slave addr>O<offset><value type>` – Čtení hodnoty typu <value type> z „process image“ ze Slave stanice s adresou <slave addr>. Hodnota se nečte ze začátku „process image“, ale její začátek je posunut o <offset> bajtů (v případě logické hodnoty <offset> bitů). Typ hodnoty může být:

- `u1` logická hodnota
- `u8` celé číslo bez znaménka o velikosti 8 bitů (tj. může nabývat hodnot 0 až 255)
- `u16` celé číslo bez znaménka o velikosti 16 bitů (tj. může nabývat hodnot 0 až 65535)
- `i16` celé číslo se znaménkem o velikosti 16 bitů (tj. může nabývat hodnot -32768 až 32767)
- `u32` celé číslo bez znaménka o velikosti 32 bitů (tj. může nabývat hodnot 0 až 4294967295)
- `i32` celé číslo se znaménkem o velikosti 32 bitů (tj. může nabývat hodnot -2147483648 až 2147483647)
- `f32` desetinné číslo o velikosti 32bitů (dle standardu IEEE 754)
- `f64` desetinné číslo o velikosti 64bitů (dle standardu IEEE 754)

`PI<slave addr>O<offset><value type>` – Jako předchozí případ, ale hodnota je zapisována.

`<item name>` – Čtení nebo zápis objektu CANopen metodou SDO. <item name> je jméno položky zadávané při vytváření konfigurace.

`MASTER__Status` – Stav stanice Master, tj. ovladače `EtcDrv`. Možnosti jsou:

- `1` init(po zapnutí napájení),
- `2` preop(lze komunikovat přes mailbox, tj. CANopen příkazy , ale „process image“ aplikace Slave stanice neaktualizuje ani neakceptuje),
- `4` safeop(lze komunikovat přes mailbox, tj. CANopen příkazy , „process image“ lze číst, ale zapsané hodnoty aplikace Slave stanice neakceptuje - používá bezpečnou náhradní hodnotu (obvykle 0)),
- `8` operational(stanice plně funkční)

`S<slave addr>_Status` – Stav stanice Slave s adresou <slave addr>. Možnosti jsou stejné jako pro stanici Master.

`<slave name>_Status` – Stav stanice Slave se jménem <slave name> (zadává se při konfiguraci ovladače `EtcDrv`). Možnosti jsou stejné jako v předchozím případě.

`<slave name>_StatusE` – Pokud Slave stanice nepřejde do požadovaného režimu (PREOP, SAFEOP, OPERATIONAL), tak zde je kod chyby (v některých případech jen jeden tick ovladače `EtcDrv`).

Pro signály, které se komunikují pomocí CANopen, je možné číst/zapisovat další atributy daného objektu. To se provede přidáním přípony do názvu. Možnosti jsou:

`_Value` – hodnota objektu čtená/zapisovaná po sběrnici EtherCAT(tj. stejná hodnota, jako bez přípony)

`_RE` – povolení čtení po sběrnici EtherCAT.

`_WE` – povolení zápisu po sběrnici EtherCAT.

`_Fresh` – udává počet sekund od poslední změny hodnoty (resp. kdy naposledy přišla hodnota po sběrnici EtherCAT - hodnota se nemusela změnit).

`_Avi` – přepsání typu položky z konfigurace (0x1000 = bool, 0x2000 = byte, ...).

`_Slave` – přepsání adresy slave z konfigurace

`_Index` – přepsání parametru Index z konfigurace

`_Subindex` – přepsání parametru Subindex z konfigurace

Použití bloků `From` a `Goto` pro vstup a výstup jednoho signálu do/z řídicího algoritmu umožňuje snadno přecházet ze simulační verze algoritmu testované v systému Matlab Simulink do systému reálného času REX. V systému Simulink je možno k blokům `From` a `Goto` přiřadit „protikusy“, kterými bude připojen simulační model procesu, po otestování může být model procesu z projektu odstraněn. Při překladu modelu nahradí díky zavedené a právě popsané konvenci systém REX zbylé bloky `From` a `Goto` vstupními a výstupními bloky.

Protože ovladač umožňuje pod jedním symbolickým jménem získávat několik vstupů či nastavovat několik výstupů, lze s výhodou používat bloky čtyřnásobných, osminásobných a šestnáctinásobných vstupů a výstupů (`INQUAD`, `OUTQUAD`, `INOCT`, `OUTOCT` a `INHEXD`, `OUTHEXD`), viz obr. 2.3. V tomto případě je v názvu bloku odkaz na první požadovaný objekt a v následujících signálech jsou následující subindexy. Výhodou takového užití je zvýšení rychlosti a částečně i přehlednosti algoritmů. Přechod od simulační verze je však v takovém případě trochu pracnější. Podrobný popis vícenásobných vstupů a výstupů lze nalézt v příručce [1].

Kapitola 3

Konfigurace ovladače

Konfigurace ovladače spočívá ve vytvoření nastavení parametrů všech Slave stanic a v definování položek, které se čtou asynchronně pomocí SDO příkazů protokolu CANopen.

Obecný popis konfiguračního dialogového okna a postup při konfiguraci jednotlivých typů objektů je uveden v následujících sekcích této kapitoly.

3.1 Konfigurační dialogové okno

Zatím není implementováno. Lze pouze vygenerovat implicitní konfiguraci (což doporučujeme, protože se tím vytvoří základní struktura souboru a je potřeba jen změnit parametry). Dále je nutné editovat přímo *.rio soubor v textovém editoru (viz 5).

Kapitola 4

Stručný popis sběrnice EtherCAT

EtherCAT je protokol, který pro přenos zpráv/packetů používá ethernet. Lze použít také UDP protokol, což se používá hlavně pro Master - Master komunikaci. Stanice Master je běžná ethernetová karta, ale stanice Slave využívají speciální řadič, se dvěma (popř. až čtyřmi) ethernetovými porty/konektory. Každý byte dat přijatých po jednom ethernetu je okamžitě analyzován a buď přímo přeposílán nebo nahrazen vlastními daty. Data jsou přeposílána buď na druhý ethernetový port/konektor nebo (pokud je druhý port nezapojen) zpět na port, po kterém přišly (vzhledem k full-duplex kabelu je to možné). Tento princip umožňuje Master stanici obsloužit všechny Slave stanice jediným ethernetovým packetem (za předpokladu, že dat je málo, což obvykle platí) a navíc Slave stanice mohou být synchronizovány s přesností do 20ns, tj. kompenzuje se i konečná rychlost světla (šíření signálu po kabelu).

Každá Slave stanice obsahuje až 64kB dat, přičemž první 4kB jsou vyhrazeny na speciální registry a ve zbytku jsou vlastní data aplikace. Speciální registry obsahují různé identifikační údaje (adresa, sériové číslo, výrobce, podporované režimy), konfigurační údaje (kde je namapován tzv. mailbox a process image), ale jsou zde i registry pro synchronizaci času, stav aplikace (INIT, PREOP, SAFEOP, OPERATIONAL), čtení a zápis konfigurační EEPROM a další.

Data lze adresovat několika způsoby

- adresa stanice + offset v datech stanice
- pořadí stanice na kabelu + offset v datech stanice
- ve stanici se nastaví mapovací registry, které umožní namapovat libovolný kus (v některých případech až po jednotlivých bitech) adresního prostoru/paměti stanice do 32bitového adresového prostoru a oblast se pak adresuje touto globální 32-bitovou adresou
- příznak všech stanic (tzv. broadcast) + offset v datech stanice
- lze kombinovat cteni a zapis

V datové oblasti lze definovat tzv. process image. Je to oblast dat, u které je mechanismy protokolu EtherCAT a implementací linkové vrstvy Slave stanice zajištěna konzistence, tj. nemůže se stát, že aplikace ve Slave stanici zapíše polovinu oblasti a Master stanice tento nekonzistentní stav přečte. Obdobně v opačném směru.

V případě „process image“ může být perioda zápisu jiná, než perioda čtení a vždy se přečtou poslední konzistentně zapsaná data. V případě „mailboxu“ je navíc zajištěno, že obsah paměti není přepsán, dokud jej adresát (tj. aplikace ve slave stanici nebo Master stanice - podle směru přenosu) nepřečte.

Je obvyklé, že velikost a adresu „mailbox“ a „process image“ musí nastavit Master, nicméně aplikace ve Slave stanici očekává nějaké hodnoty, a pokud jsou jiné, tak se vůbec nerozběhne (nepřejde do stavu SAFEOP ani OPERATIONAL). Dále je obvyklé, že kromě speciálních registrů a mailboxů a „process image“ Slave stanice jinou oblast paměti nemá a při pokusu ji číst/zapisovat se vrací chyba. Oblastí mailbox a process image může být více, ale Slave stanice obvykle podporují jen jednu pro každý směr.

Pro data v mailboxu existuje několik standardizovaných formátů. Jsou to:

1	AoE	protokol ADS - proprietární protokol firmy Beckhoff (původně pro sériovou linku)
2	EoE	přenos ethernethových packetů po EtherCAT
3	CoE	přenos CANopen zpráv po EtherCAT
4	FoE	přenos souborů
-	VoE	nestandartizované (vendor specific) formáty

Ovladač `EtcDrv` podporuje pouze formát CoE.

Ačkoliv EtherCAT explicitně nezakazuje jiný ethernetový/IP provoz a použití switchů/hubů, dopručuje se pro Master - Slave komunikaci používat oddělenou kabeláž s liniovou strukturou (popř. EtherCAT Slave zařízení s více porty místo switche). Problémy jsou:

- EtherCAT packety jsou ethernetový broadcast, tj. switche je posílají do všech stanic v síti. Navíc perioda komunikace je často v jednotkách milisekund nebo i kratší, takže síť se zahlcuje. Tento problém lze řešit konfigurací VLAN.
- Switche zavádí do signálové cesty časové zpoždění, které je navíc pokaždé jiné. To zhoršuje časové parametry realtime packetů. Huby zase způsobují kolize a následnou ztrátu packetů, což má opět za následek zhoršení časových parametrů raltime packetů.
- Již jsme si řekli, že Slave stanice packet doplní o vlastní data a přeposílá dále, takže se nakonec dostane zpět k Master stanici. Při tomto přeposílání, ale nevyměňují MAC adresu odesílatele, což zmate switche (z pohledu switche to vypadá, že jedna MAC adresa je na dvou místech/portech, což je špatně a může to detekovat jako útok a packety zahazovat).
- Pokud jsou Slave stanice v různých liniích, Master stanice dostane na jeden dotaz více odpovědí, přičemž v každé z nich to vypadá, že některé stanice nefungují, což může algoritmus špatně vyhodnotit. Mezi různými liniemi také nelze zajistit časovou synchronizaci. Dále nelze použít adresování pořadím na kabelu (což se často využívá při inicializaci).

- Na konec řetězce/linie Slave stanic se nesmí připojit jiná stanice, než EtherCAT Slave. V opačném případě by Master stanice nedostala odpověď.

Kapitola 5

Formát konfiguračního souboru

Soubor *.rio je textový, takže jej lze v případě potřeby prohlížet i upravovat v libovolném textovém editoru pracujícím s prostým textem (například Notepad). Struktura souboru je zřejmá z následujícího příkladu:

```
EtherCAT {
  #NetAdapter              "eth1"
  NetAdapter              "edev0"
  Timeout                 0.010
  TimeoutSdo              0.5
  MasterMode              2
  Slave {
    Name                    "LenzeDrive1"
    Flags                   0x0004F021
    Address                 1
    ReadMailboxAddress      4224
    WriteMailboxAddress     4096
    ReadProcessAddress      4952
    WriteProcessAddress     4352
    ReadMailboxSize         128
    WriteMailboxSize        128
    ReadProcessSize         16
    WriteProcessSize        16
  }
  Item {
    Name                    Active_RxPDO(C13484)
    SlaveAdr                1
    Index                   11091
    SubIndex                1
    Flags                   0x0000000A
    avi                     8192
    Value                   1
```

```
  }
  Item {
    Name                    Active_TxPDO(C13784)
    SlaveAdr                1
    Index                   10791
    SubIndex                1
    Flags                   0x0000000A
    avi                     8192
    Value                   1
  }
}
```

Platí, že parametry, jejichž název začíná znakem # jsou ignorovány a lze je tedy využít jako komentář. Sekce `Slave` se opakuje tolikrát, kolik je v síti Slave stanic, přičemž musí být uvedeny v pořadí, v jakém jsou „na kabelu“ (`EtcDrv` využívá při inicializaci adresování pořadím na kabelu). Obdobně sekce `Item` se opakuje pro každý prvek CANopen „object dictionary“, který chceme číst/zapisovat (podpořeny jsou pouze SDO příkazy). V názvech parametrů i sekcí se rozlišují velká a malá písmena.

Význam jednotlivých parametrů je následující:

`NetAdapter` – Název ethernethového adaptéru v operačním systému. V Linuxu je to obvykle `eth1` pro XENOMAI obvykle `edev0`.

`Timeout` – Čas v sekundách, jak dlouho se čeká na ethernetový packet s odpovědí.

`TimeoutSdo` – Čas v sekundách, jak dlouho se čeká na odpověď na SDO příkaz.

`DcJitter` – Časový rozdíl v sekundách, který se při synchronizaci hodin považuje za náhodnou nepřesnost (na straně Master , která je implementována sofwareově) a nekoriguje se.

`DcShift` – Časový rozdíl v sekundách, mezi začátkem periody Master stanice a synchronizačním pulsem (eventem) ve Slave stanicích (ve všech stejně).

`DcFactor` – Počet cyklů sběrnice (period EtcDrv), po kterých se kontroluje a koriguje odchylka hodin všech stanic a tím i synchronnost celé sítě.

`MasterMode` – Upravuje některé vlastnosti ovladače. Každý bit představuje/zapíná určitou vlastnost, přičemž:

`bit 0`	lowJitter	pro snížení časové neurčitosti se posílají stále stejné dotazy, tj. v některých režimech posílají i data, která nejsou momentálně potřeba
`bit 1`	singlePacket	všechny požadavky jsou soustředěny do jednoho packetu (tento režim je nutný pro měniče TG drives, hodí se i pro krátké periody a málo dat)
`bit 2`	lockData	synchronizace semaforem (lze pro urychlení vypnout, pokud všechny vstupy a výstupy do tohoto ovladače vedou jen z jemu přidruženému IOTASKu)

`Name` – Název Slave stanice; slouží pro identifikaci uživatelem - `EtcDrv` jej využívá jen pro alternativní formát vlajky (viz výše 2.2).

`SlaveAdr` – Adresa Slave stanice. Je to číslo od 1 do 65535 (0xFFFF). Musí být jednoznačná v celé síti (ve smyslu sítě připojené k ethernetovému portu identifikovaného parametrem `NetAdapter`) a to včetně tzv. alias adres.

`Flags` – Upravuje některé Slave stanice. Každý bit představuje/zapíná určitou vlastnost, přičemž:

`bit 0`	mandatory	povinný Slave; pokud se jej nepovede inicializovat, Master nepřejde do režimu OPERATIONAL ani SAFEOP
`bit 1`	swap bytes	prohodí se vyšší a nižší byte u dvoubajtových čísel (v datech)
`bit 2`	swap words	prohodí se vyšší a nižší word u čtyřbajtových čísel (v datech)
`bit 3`	use events	pro některé oblasti paměti Slave lze získat informaci o změně ze speciálního (event) pole v packetu; pokud je tento bit vypnut, tak se příslušné oblasti čtou trvale, při zapnutí jen při změně. Event režim nelze kombinovat s režimem `single packet` a `low jitter`
`bit 4`	large CoE	protože sběrnice CAN má datovou část zprávy omezenou na 8byte, CANopen nevyužívá delší packety/zprávy; pro EtherCAT toto omezení neplatí a tímto bitem se zapíná použití delších zpráv
`bit 5`	smart mailbox	mechanismus pro zajištění konzistence dat mailboxu vyžaduje zapsat/přečíst jeho první a poslední byte; pokud je tedy mailbox dlouhý a zpráva krátká, lze prostředek nezapisovat a nepřenášet tolik dat, což se zapíná tímto příznakem
`bit 6`	distributed clock	zapíná synchronizaci hodin pro tento Slave (podle specifikace je možné zapnout až dva eventy od hodin, naše implementace podporuje jen první)
`bit 12`	write mailbox event	aktivuje použití eventu/signálu při zápisu do mailboxu
`bit 13`	read mailbox event	aktivuje použití eventu/signálu při přečtení mailboxu
`bit 14`	write data event	aktivuje použití eventu/signálu při zápisu do „proces image“
`bit 15`	read data event	aktivuje použití eventu/signálu při přečtení „proces image“
`bit 16`	write mailbox watchdog	aktivuje watchdog pro zápisový mailbox (pokud do něj Master dlouho nezapíše, aktivuje se speciální událost)
`bit 17`	read mailbox watchdog	aktivuje watchdog pro čtecí mailbox (pokud do něj aplikace Slave dlouho nezapíše, aktivuje se speciální událost)
`bit 18`	write data watchdog	aktivuje watchdog pro „process data“ (pokud do oblasti Master dlouho nezapíše, aktivuje se speciální událost)
`bit 19`	read data watchdog	aktivuje watchdog pro „process data“ (pokud do oblasti aplikace Slave dlouho nezapíše, aktivuje se speciální událost)

`WriteMailboxAddress` – Adresa v adresním prostoru Slave stanice, kde začíná mailbox pro zápis Master stanicí. Může nabývat hodnot 0x1000 až 0xFFFF.

`WriteMailboxSize` – Délka (v bajtech) mailboxu pro zápis nebo 0 pokud se mailbox nepoužívá.

`ReadMailboxAddress` – Adresa v adresním prostoru Slave stanice, kde začíná mailbox pro čtení Master stanicí. Může nabývat hodnot 0x1000 až 0xFFFF.

`ReadMailboxSize` – Délka (v bajtech) mailboxu pro čtení.

`WriteProccessAddress` – Adresa v adresním prostoru Slave stanice, kde začíná „process image“ pro zápis Master stanicí. Může nabývat hodnot 0x1000 až 0xFFFF.

`WriteProcessSize` – Délka (v bajtech) „process image“ pro zápis nebo 0 pokud se „process image“ nepoužívá.

`ReadProcessAddress` – Adresa v adresním prostoru Slave stanice, kde začíná „process image“ pro čtení Master stanicí. Může nabývat hodnot 0x1000 až 0xFFFF.

`ReadProcessSize` – Délka (v bajtech) „process image“ pro čtení.

`SlaveAdr` – adresa slave stanice (její parametr `Address`) do které objekt patří.

`Index` – index objektu v CANopen.

`Subindex` – subindex objektu v CANopen.

`Flags` – Upravuje některé vlastnosti položky. Každý bit představuje/zapíná určitou vlastnost, přičemž:

`bit 0`	readable	stanice Master (ovladač EtherCAT) může číst objekt ze stanice Slave
`bit 1`	writeable	stanice Master (ovladač EtherCAT) může zapisovat hodnotu objektu ve stanici Slave
`bit 2`	no subindex	objekt nemá subindexy; hodnota představuje hodnotu celého objektu
`bit 3`	write in init	ovladač EtherCAT hodnotu zapíše v inicializační fázi; pokud se hodnotu nepodaří zapsat, síť nepřejde do stavu SAFEOP ani OPERATIONAL

`avi` – Typ hodnoty. Možnosti jsou:

`0x1000`	logická hodnota (on/off)
`0x2000`	BYTE/UNSIGNED8 - 8bitové číslo bez znaménka
`0x3000`	SHORT/SIGNED16 - 16bitové číslo se znaménkem
`0x4000`	LONG/SIGNED32 - 32bitové číslo se znaménkem
`0x5000`	WORD/UNSIGNED16 - 16bitové číslo bez znaménka
`0x6000`	DWORD/UNSIGNED32 - 32bitové číslo bez znaménka
`0x7000`	FLOAT/REAL32 - 4bajtové desetinné číslo (dle IEEE754)
`0x8000`	DOUBLE/REAL64 - bitové desetinné číslo (dle IEEE754)
`0xA000`	LARGE/SIGNED64 - 64bitové číslo se znaménkem
`0xC000`	STRING - text
`0xD000`	INTPTR/DOMAIN - obecné pole bajtů (zadává se do uvozovek jako číslo v hexadecimálním formátu)

`Value` – Vlastní (počáteční) hodnota subindexu. Formát musí odpovídat parametru `avi`.

Kapitola 6

Poznámky k implementaci

V této kapitole jsou soustředěny poznatky, které vznikly z dosavadních zkušeností. Některé položky v konfiguraci jsou často nesprávně pochopeny, ale podrobný popis výše by zhoršoval čitelnost textu. Proto jsou tyto postřehy uvedeny ve zvláštní kapitole.

- Každý event/událost má za následek, že se nastaví bit v nějakém registru Slave stanice, popřípadě se zvýší nějaký čítač (opět registr v adresním prostoru Slave stanice). Některé události jsou také kopírovány do jednotlivých bitů ve speciální oblasti EtherCAT packetu. Události určené pro aplikaci ve Slave stanici mohou generovat interrupt.

- Parametry pro konfiguraci Slave stanice je potřeba vyčíst z XML souboru dodaného se zařízením (resp. staženého z internetových stránek výrobce). Dokud nebude dokončena integrace do GUI, je potřeba v tomto souboru najít položky <SM> (POZOR jeden XML může být pro více zařízení - je potřeba najít správnou sekci <device>). Vypadá to nějak takto:

```
<Sm MinSize="64" MaxSize="512" DefaultSize="136" StartAddress="#x1c00"
    ControlByte="#x26" Enable="1">MBoxOut</Sm>
<Sm MinSize="64" MaxSize="512" DefaultSize="136" StartAddress="#x1e00"
    ControlByte="#x22" Enable="1">MBoxIn</Sm>
<Sm StartAddress="#x1000" ControlByte="#x24" Enable="1">Outputs</Sm>
<Sm DefaultSize="0" StartAddress="#x1600"
    ControlByte="#x00"Enable="1">Inputs</Sm>
```

  V parametru `ControlByte` jednotlivé bity znamenají:

bit0-1	typ dat	0=proces image, 2=mailbox
bit2-3	směr přenosu	0=Master čte, 1=Master zapisuje
bit5	povolení eventu	1=event pro aplikaci Slave povolen
bit6	povolení watchdogu	1=watchdog povolen

  význam ostatních parametrů je zřejmý. V uvedeném příkladu je velikost „process image“ 0 (resp. neuvedena). To je obvyklé pro modulární a/nebo konfigurovatelná

zařízení. V takovém případě je potřeba určit velikost dat jiným způsobem - buď je to zřejmé z konfigurace (speciálního konfiguračního nástroje od výrobce) nebo je to uvedeno na jiném místě XML souboru (soubor je textový a samopopisný - v podstatě anglický strukturovaný text). Někdy to je možné uhádnout (např. modul s 8 logickými vstupy bude velmi pravděpodobně zabírat 1byte dat), někdy stanice funguje i s jiným nastavením.

- podrobnější popis protokolu CANopen lze najít v popisu ovladače `CanDrv` nebo přímo v jeho specifikaci.
- Ačkoliv ovladač `EtcDrv` nenastavuje parametry watchdogu, většinou není problém, pokud se někde zapne sledování watchdogem, protože výchozí doba je 100ms. Naopak někdy je problém, pokud je v XML souboru watchdog zapnut a v konfiguraci se nezapne.

Kapitola 7

Co dělat při problémech

Nejčastější chyby jsou:

Konfigurace neodpovídá definičnímu XML souboru jednotlivých Slave stanic.

Implementace Slave má nějakou odchylku od specifikace. Například měniče TG Drives (nepochopitelně) vyžadují všechna data v jednom packetu. V opačném případě aplikace vyhodotí nekonzistenci periody a přejde do PREOP režimu. Některé implementace mohou vyžadovat synchronizaci času (což v součastnosti není podporováno).

Některé Slave stanice vyžadují dodatečnou konfiguraci. Například měniče LENZE vyžadují nastavit pomocí CoE, které PDO se mapuje do „process image“, tj. do objektů 11091(0x2B53) a 10791(0x2A27) zapsat hodnotu 1.

V případě, že daný ovladač `EtcDrv`funguje v jednoduchých testovacích příkladech správně a při potřebné konfiguraci nefunguje, prosíme o zaslání informace o problému (nejlépe elektronickou cestou) na adresu dodavatele. Pro co nejrychlejší vyřešení problému by informace by měla obsahovat:

- Identifikační údaje Vaší instalace – verzi, číslo sestavení (build), datum vytvoření instalace, licenční číslo.
- Stručný a výstižný popis problému.
- Co možná nejvíc zjednodušenou konfiguraci řídicího systému REX, ve které se problém ještě vyskytuje (ve formátu souboru s příponou `.mdl`).
- Konfigurační soubor ovladače `EtcDrv`.

Literatura

[1] REX Controls s.r.o.. *Funkční bloky systému REX – Referenční příručka*, 2017.