www.rexcontrols.cz/rex

Ovladač `CanDrv` systému REX
Uživatelská příručka

REX Controls s.r.o.

Verze 2.50.1
Plzeň
7.11.2016

Obsah

Kapitola 1

Ovladač CanDrv a systém REX

1.1 Úvod

V této příručce je popsáno používání ovladače CanDrv pro připojení technických prostředků využívajících protokol CAN a CANopen k řídicímu systému REX pro Windows, Linux, Linux/XENOMAI. Je podporována varianta CAN 1.0 i CAN 2.0 (tj. Message ID 11 i 29 bitů). Ovladač umožňuje získávat vstupy a nastavovat výstupy a to jak v režimu PDO tak i SDO. Ovladač byl vyvinut firmou REX Controls.

Ačkoliv CANopen má architekturu producent-konzument, některé funkce (například download konfigurace, přepínání stavu sítě) řídí jen jedna stanice v síti a v dalším textu bude označována Master zatímco ostatní stanice budou označovány Slave.

1.2 Požadavky na systém

Obecně lze říci, že pro použití ovladače CanDrv 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 speciální komunikační karta.

Ovladač vyžaduje komunikační kartu firmy PEAK (varianta pro sběrnici USB, PCI, miniPCI, PCIexpress). Komunikační karty jiných výrobců nejsou podpořeny, ale v případě požadavku je možné jejich podporu doimplementovat.

Pro systém Windows a Linux/Debian je potřeba nainstalovat ovladač komunikační karty do jádra systému (viz webové stránky dodavatele nebo instalační CD dodávané s komunikační kartou). Pro systém Linux/openWRT je nutno nainstalovat balíček `kmod-peak-linux-driver`.

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 Vista/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: Windows Vista/7/8/10, 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 `CanDrv` instaluje jako součást instalace řídicího systému REX. Pro nainstalování ovladače je nutné v instalačním programu systému REX zaškrtnout volbu `Ovladač protokolu CAN`. 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:

`CanDrv_H.dll` – Konfigurační část ovladače `CanDrv`.

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

`DOC\CanDrv_CZ.pdf` – Tato uživatelská příručka.

Pro operační systém Linux je potreba nainstalovat balíček `rex-candrvt`.

Ve všech případech je na cílový počítač potřeba nainstalovat ovladač komunikační karty (viz výše).

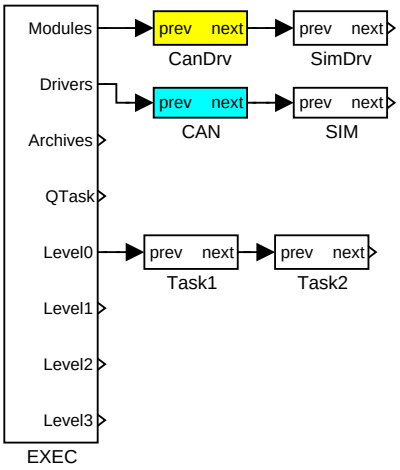
Kapitola 2

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 `CanDrv` do projektu

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



Obrázek 2.1: Příklad zařazení ovladače `CanDrv` 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 `CanDrv`, který nemá žádné další parametry.

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

`modul` – jméno modulu ovladače, které se pro tento ovladač zadává: `CanDrv`

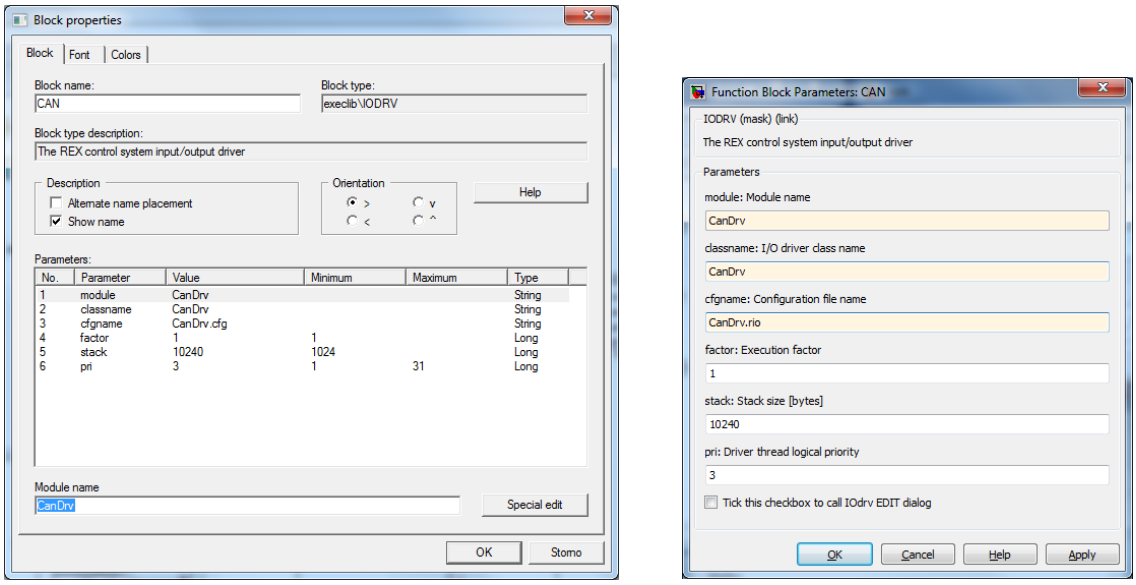
`classname` – jméno třídy ovladače, které se pro tento ovladač zadává: `CanDrv`POZOR! Jméno rozlišuje velká a malá písmena!

`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.

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

Ovladač `CanDrv` podporuje i úlohy běžící synchroně 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` připojíme blok typu `IOTASK` (má analogické parametry i význam jako blok typu `TASK`). Ovladač potom funguje tak, že odešle `SYNC` zprávu/packet (popř. čeká na přijetí `SYNC` zprávy/packetu), spustí algoritmus definovaný blokem `IOTASK`, odešle všechny synchronní PDO 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 `CanDrv`, popsaný v kapitole 3, se aktivuje po stisku tlačítka `Special Edit`.



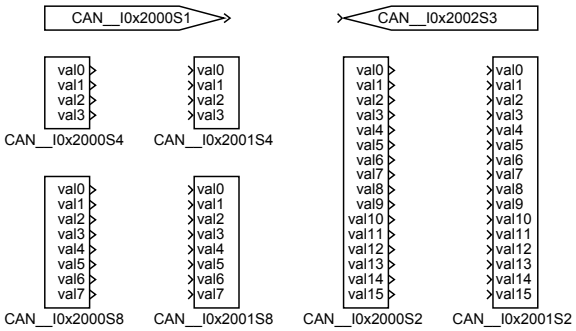
a) b)

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. Okamžitě po zaškrtnutí tohoto políčka bude zavolán konfigurační dialog ovladače `CanDrv` 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`, popř. `IOTASK` připojovaných na výstupy `QTask`, `Level0`,..., `Level3` exekutivy. Pro připojení vstupů a výstupů z ovladače `CanDrv` do řídicího systému REX lze použít bloky, znázorněné na obr. 2.3.



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

Blok typu `From` sloužící pro připojení jednoho vstupu má parametr `GotoTag` roven `CAN__<IN>`, blok typu `Goto` používaný pro připojení jednoho výstupu má tento parametr nastaven na `CAN__<OUT>`, kde `<IN>` a `<OUT>` jsou řetězce odkazující na `object dictionary` (viz dále). Všechny řetězce používané jako odkazy na data poskytovaná a přijímaná ovladačem `CanDrv` mají přímo na svém začátku prefix `CAN` povinně následovaný dvěma znaky `_` (podtržítko).

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

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

Zbytek jména vstupního nebo výstupního bloku je odkaz do `object dictionary` (viz dále) a má následující strukturu:

`I<index>S<subindex>`

kde `<index>` a `<subindex>` jsou čísla definující objekt v `object dictionary`, jehož hodnotu čteme/zapisujeme. Je možné číst/zapisovat další pomocné signály k danému objektu. To se provede přidáním přípony do názvu. Možnosti jsou:

`_RE` – povolení čtení po sběrnici CAN.

`_WE` – povolení zápisu po sběrnici CAN.

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

Dále existují speciální symboly:

`Status` – Stav stanice. Možné hodnoty jsou:

- `0` neexistující stanice (není v konfiguraci),
- `1` neznámý stav (stanice neodpovídá),
- `2` init (po zapnutí napájení),
- `3` preop (lze posílat SDO, ale PDO se neposílají a neakceptují),
- `4` stop (jako stav PREOP, ale aplikace může reagovat jinak),
- `5` operational(stanice plně funkční)

`Node<nodeID>` – Stav vzdálené stanice, jejíž číslo je `<nodeID>`. Hodnoty jsou stejné jako v předchozím případě.

`RecvMsg` – V režimu `CAN` (tj. nikoliv `CANopen` obsahuje celou přečtenou zprávu. Je potřeba použít blok `INQUAD` a potom:

- `y0` message ID
- `y1` délka dat v byte(tj. 0 až 8; -1 značí žádnou příchozí zprávu),
- `y2` první 4 byte dat (tj. 1. až 4. byte),
- `y3` druhé 4 byte dat (tj. 5. až 8. byte),

Pro příjem více zpráv zároveň lze použít symboly `RecvMsg1`, `RecvMsg2`, atd.

`SendMsg` – V režimu `CAN` (tj. nikoliv `CANopen` obsahuje celou odesílanou zprávu. Je potřeba použít blok `OUTQUAD` a potom:

- `u0` message ID
- `u1` délka dat v byte(tj. 0 až 8; -1 značí žádná odesílaná zpráva),
- `u2` první 4 byte dat (tj. 1. až 4. byte),
- `u3` druhé 4 byte dat (tj. 5. až 8. byte),

Pro odeslání více zpráv zároveň lze použít symboly `SendMsg1`, `SendMsg2`, atd.

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 tzv. „object dictionary“. Specifikace `CANopen` definuje, že všechny parametry a předávané hodnoty jsou v tomto „object dictionary“. Musíme tedy definovat, které objekty naše zařízení obsahuje, jakou mají počáteční hodnotu a zda je lze číst/zapisovat z algoritmu systému REX a zda je lze číst/zapisovat po sběrnici CAN.

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 všechny povinné objekty). Dále je nutné editovat přímo *.rio soubor v textovém editoru (viz 6).

Kapitola 4

Stručný popis sběrnice CAN

Sběrnice CAN je dvouvodičová sériová sběrnice na fyzické vrstvě podobná s dobře známou RS-485. Budiče jsou s tzv. otevřeným kolektorem, takže logická 0 „přetlačí“ logickou 1. Dále platí pravidlo, že stanice, která zjistí, že vysílá logickou 1 a na sběrnici je logická 0 musí okamžitě přestat vysílat a celou zprávu se pokusí vyslat znova po ukončení vysílání aktuální zprávy.

Vzhledem ke konečné rychlosti světla (šíření signálu v kabelu) a k požadavku kontroly kolize na každém bitu dostáváme omezení na celkovou délku kabelu. Pro zamezení odrazu signálu na konci vedení musí být kabel na obou koncích zakončen odporem rovnajícím se impedanci vedení (obvykle kolem 120ohm). Vzhledem k maximálnímu proudu budičů je omezen počet zařízení na jednom kabelu na 32. Detailní informace o kabelech a konektorech jsou uvedeny ve specifikaci `CAN` (soubor 303_1v01070001.pdf).

Celá zpráva/packet posílaný po sběrnici CAN obsahuje číslo zprávy (tzv. `Message ID` někdy též označované `COB-ID`) a vlastní data, kterých může být 0 až 8 byte. Zpráva obsahuje ještě několik dalších bitů, které nejsou pro další výklad podstatné. Z výše uvedeného vyplývá, že zprávy s nižším `Message ID` mají vyšší prioritu. Pokud dále zajistíme, že každé `Message ID` vysílá nejvýše jedna stanice, nemůže dojít ke ztrátě dat z důvodu kolize na sběrnici.

Stanice dále může vyslat paket žádající o vyslání určité `Message ID` (to se ovšem v `CanDrv` nevyužívá).

Původní standard CAN1.0 zavádí 11-bitové `Message ID`, pozdější revize CAN2.0 dovoluje 11 i 29-bitové `Message ID`. 29-bitové `Message ID` lze použít jen pokud jej podporují všechny zařízení na sběrnici/kabelu.

Sběrnice CAN má architekturu producent-konzument, tj, každý packet/zprávu přijímají všechny stanice. Stanice však může mít zapnutý filtr (ovladač `CanDrv` to nevyužívá) a některé zprávy pak nepřijímá (resp. nepředává nadřízeným vrstvám). Na sběrnici existuje mechanismus potvrzování, takže vysílající stanice pozná, že zprávu nikdo nepřijal.

Kapitola 5

Stručný popis protokolu CANopen

`CANopen` definuje objekty, které jsou přístupné nadřízené vrstvě (obvykle cílové aplikaci). Objekty se adresují čísly 0 až 65535(0xFFFF). Jednotlivé objekty mohou být logická hodnota, celé i desetiné číslo, text nebo obecné pole bajtů (tzv. DOMAIN). Dále objekt může být pole nebo struktura výše uvedených typů. K jednotlivým prvkům se potom přistupuje pomocí subindexu, přičemž subindex 0 udává počet prvků. Tato struktura se nazývá `Object Dictionary` a platí následující pravidla:

0x0000 ... 0x0FFF	Reservováno pro definici typů; při komunikaci se nepoužívá
0x1000 ... 0x1FFF	Mají přesně daný význam a definují zejména, jak se data (hodnoty objektů v `Object Dictionary`) předávají po sběrnici CAN mezi jednotlivými stanicemi.
0x2000 ... 0x5FFF	mohou se libovolně použít aplikací
0x6000 ... 0xFFFF	jsou definovány aplikačním profilem; pokud například zařízení podporuje profil DS402(servozesilovače, řízení motorů) pak je 0x6040 řídící slovo(s přesně daným významem jednotlivých bitů), 0x6063 aktuální poloha, atd.

Povinná část obsahuje následující objekty(jde je o základní sadu; pozdější rozšíření specifikace doplňuje například objekty pro multiplexed-PDO nebo konzoli operačního systému):

Index (hex)	Object type	Name	Data type	Acc
1000	VAR	device type	UNSIGNED32	ro
1001	VAR	error register	UNSIGNED8	ro
1002	VAR	manufacturer status register	UNSIGNED32	ro
1003	ARRAY	pre-defined error field	UNSIGNED32	ro
1004	-	reserved for compatibility reasons	-	-
1005	VAR	MESSAGE-ID SYNC	UNSIGNED32	rw
1006	VAR	communication cycle period	UNSIGNED32	rw
1007	VAR	synchronous window length	UNSIGNED32	rw
1008	VAR	manufacturer device name	Vis-String	const
1009	VAR	manufacturer hardware version	Vis-String	const
100A	VAR	manufacturer software version	Vis-String	const
100B	-	reserved for compatibility reasons	-	-
100C	VAR	guard time	UNSIGNED16	rw
100D	VAR	life time factor	UNSIGNED8	rw
100E	-	reserved for compatibility reasons	-	-
100F	-	reserved for compatibility reasons	-	-
1010	ARRAY	store parameters	UNSIGNED32	rw
1011	ARRAY	restore default parameters	UNSIGNED32	rw
1012	VAR	MESSAGE-ID TIME	UNSIGNED32	rw
1013	VAR	high resolution time stamp	UNSIGNED32	rw
1014	VAR	MESSAGE-ID EMCY	UNSIGNED32	rw
1015	VAR	Inhibit Time EMCY	UNSIGNED16	rw
1016	ARRAY	Consumer heartbeat time	UNSIGNED32	rw
1017	VAR	Producer heartbeat time	UNSIGNED16	rw
1018	RECORD	Identity Object	Identity(23h)	ro
1019 .. 11FF	-	reserved for future extension	-	-
1200 .. 127F	RECORD	1st to 128th Server SDO parameter	SDO Parameter(22h)	ro
1280 .. 12FF	RECORD	1st to 128th Client SDO parameter	SDO Parameter(22h)	ro
1300 .. 13FF	-	reserved for future extension	-	-
1400 .. 15FF	RECORD	1st to 512th receive PDO Parameter	PDO CommPar(20h)	rw
1600 .. 17FF	RECORD	1st to 512th receive PDO mapping	PDO Mapping(21h)	rw
1800 .. 19FF	RECORD	1st to 512th transmit PDO Parameter	PDO CommPar(20h)	rw
1A00 .. 1BFF	RECORD	1st to 512th transmit PDO mapping	PDO Mapping(21h)	rw

použité struktury mají následující prvky:

- PDO CommPar(20h)

1	UNSIGNED32	MessageID	pokud je nahozen bit29 jde o 29bitové MessageID, pokud je nahozen bit31, MessageID je neplatné
2	UNSIGNED8	transmision type	1 až 240 posílá se cyklicky a synchroně se SYNC packetem, číslo značí po kolika SYNC packetech jdou data, ostatní jsou necyklické režimy (v `CanDrv` některé nefungují)
3	UNSIGNED16	inhibit time	doba ve 100us po kterou je zablokováno vyslání PDO od jeho předchozího vyslání (tj. minimální perioda v necyklickém režimu)
4	UNSIGNED8	reserved	-
5	UNSIGNED16	event timer	v `CanDrv` se nepoužívá

- PDO Mapping(21h)

1 .. 40	UNSIGNED32	1st to 64th object to be mapped	po řadě index(U16), subindex(U8), počet bitů v PDO(U8)

- PDO Parameter(22h)

1	UNSIGNED32	MessageID client->server	-
2	UNSIGNED32	MessageID server->client	-
3	UNSIGNED8	NodeID	číslo stanice (NodeID), se kterou se komunikuje
6-255	SIGNED64	SDOmapping	rozšíření `REX` - po rade perioda v milisekundách(U16), místní index(U16), vzdálený index(U16), místní subindex(U8), vzdálený index(U8)

Data mezi jednotlivými stanicemi se vyměňují buď mechanismem SDO(Service Data Object) nebo mechanismem PDO(Process Data Object). Mechanismus SDO funguje tak, že jedna strana (tzv. client) pošle dotaz, ve kterém je index a subindex objektu a pokud je to zápis, tak i zapisovaná hodnota. Druhá strana (tzv. server) přijme požadavek a odpoví požadovanou hodnotu (resp. zapíše hodnotu a pošle potvrzení) nebo chybový kód. Pokud se data nevejdou do jednoho packetu/zprávy (tj. pokud jsou delší než 4byte), rozdělí se na více zpráv. `Message ID` pro SDO zprávy definují objekty 0x1200 až 0x127F (každý objekt pro jednu stanici, tj. tímto způsobem lze komunikovat s až 128 stanicemi) pro server a 0x1280 až 0x12FF pro klienta.

Mechanismus PDO funguje tak, že data (opravdu jen vlastní data bez dalších údajů) z několika objektů jsou poskládána do jedné zprávy a odeslána. Přijímací strana pozná podle `Message ID` co je to za data a nastaví je do příslušných objektů (obecně i obvykle jsou to jiné objekty než na vysílací straně). Takovýchto PDO přenosů (vysílacích i přijímaných) může být definováno na každé stanici více (až 512 - viz popis object dictionary,

ale některá zařízení podporují méně nebo je mají nastaveny napevno). Pro nastavení `Message ID` ,periody a dalších parametrů vysílaných PDO slouží objekty/parametry 0x1800 až 0x19FF, přičemž pořadí hodnot ve zprávě (tj. hodnoty kterých objektů z object dictionary se posílají) určují objekty/parametry 0x1A00 až 0x1BFF, tj. 1.PDO má parametry v objektu 0x1800 a přiřazení hodnot v objektu 0x1A00, 2.PDO má parametry v objektu 0x1801 a přiřazení hodnot v objektu 0x1A01, atd. Pro přijímaná PDO se totéž definuje v objektech 0x1400 až 0x15FF a 0x1600 až 0x17FF.

V předchozím textu bylo ukázáno, jak se v `CANopen` definují různé zprávy. V zásadě lze pro každý typ zprávy definovat `Message ID` libovolně, jen je potřeba dodržet pravidlo, že dvě stanice nesmí vysílat stejné `Message ID`. Aby se toto usnadnilo, jsou některé hodnoty pro daný účel dopručené/implicitní a některé zakázané. Situaci shrnuje následující tabulka:

Typ zprávy	MessageID	poznámka
NMT	0	nelze změnit
-	1	rezervováno pro pozdější použití
SYNC	128(0x80)	lze změnit v objektu 0x1005
EMERGENCY	128(0x80)+<NodeID>	lze změnit v objektech 0x1014, 0x1015
TIMESTAMP	256(0x100)	lze změnit v objektech 0x1012, 0x1013
-	256(0x100)+<NodeID>	rezervováno pro pozdější použití
PDO1(tx)	384(0x180)+<NodeID>	nastavení viz text
PDO1(rx)	512(0x200)+<NodeID>	nastavení viz text
PDO2(tx)	640(0x280)+<NodeID>	nastavení viz text
PDO2(rx)	768(0x300)+<NodeID>	nastavení viz text
PDO3(tx)	896(0x380)+<NodeID>	nastavení viz text
PDO3(rx)	1024(0x400)+<NodeID>	nastavení viz text
PDO4(tx)	1152(0x480)+<NodeID>	nastavení viz text
PDO4(rx)	1280(0x500)+<NodeID>	nastavení viz text
SDO(tx)	1408(0x580)+<NodeID>	rezervováno; nesmí se používat k jiným účelům
SDO(rx)	1537(0x600)+<NodeID>	rezervováno; nesmí se používat k jiným účelům
-	1760(0x6E0)	rezervováno pro pozdější použití
NMT Error Control	1793(0x700)+<NodeID>	rezervováno; nesmí se používat k jiným účelům; lze změnit v objektech 0x1016, 0x1017
-	2020(0x780)	reservováno pro pozdější použití
-	2020(0x780)+<NodeID>	reservováno pro pozdější použití

Detailní popis všech objektů, formát SDO packetů a pod. je ve specifikaci CANopen (v souboru 301_v04000201.pdf).

Kapitola 6

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:

```
CANopen {
  NetAdapter              "pcanpci0"
  #NetAdapter              "usb1"
  NodeID                  1
  BaudRate                1000000
  NodeMode                0x207
  TimeoutSdo              0.2
  Object {
    Index                   0x1000
    Count                   1
    Entry {
      Subindex                1
      Flags                   0x00000125
      avi                     0x6000
      Value                   301
    }
  }
  Object {
    Index                   0x1280
    Count                   3
    Entry {
      Subindex                1
      Flags                   0x0000000D
      avi                     0x6000
      Value                   0x602
    }
    Entry {
```

```
      Subindex                2
      Flags                   0x0000000D
      avi                     0x6000
      Value                   0x582
    }
    Entry {
      Subindex                3
      Flags                   0x0000000D
      avi                     0x2000
      Value                   2
    }
  }
}
```

Platí, že parametry, jejichž název začíná znakem # jsou ignorovány a lze je tedy využít jako komentář. Sekce `Object` se opakuje tolikrát, kolik definujeme objektů/indexů v „object dictionary“. Obdobně sekce `Entry` se opakuje pro každý subindex. V názvech parametrů i sekcí se rozlišují velká a malá písmena.

Význam jednotlivých parametrů je následující:

`NetAdapter` – Název komunikační karty v operačním systému. V Linuxu je to obvykle `pcanpci0` pro PCI kartu a `pcanusb0` pro USB kartu; ve Windows `usb1` pro USB kartu.

`NodeID` – Číslo stanice pro `CANopen`. Může nabývat hodnot 1 až 127.

`BaudRate` – Rychlost sběrnice v bitech za sekundu. Všechny stanice na jedné lince musí mít nastavenu stejnou.

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

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

- `bit 0` stanice má Master funkce (spouštění sítě, konfigurace stanic)
- `bit 1` synchronizace semaforem (lze pro urychlení vypnout, pokud všechny vstupy a výstupy do tohoto ovladače vedou jen z jemu přidruženému IOTASKu)
- `bit 2` Master stanice přejde do plného provozu i když nejsou k dispozici všechny nakonfigurované Slave stanice
- `bit 8` režim CAN (tj. bez CANopen vrstvy); celá konfigurace je ignorována a lze používat jen vstup `RecvMsg` a výstup `SendMsg`; v tomto režimu nelze použít IOTASK
- `bit 9` ve stavu preop se ignoruje, že Slave stanice neposílá stavové informace (tzv. heartbeat); odporuje to sice specifikaci `CANopen`, ale některá zařízení dokud nejsou nakonfigurována status neposílají

`Index` – Číslo objektu v „object dictionary“

`Count` – Počet subindexů objektu, tj. počet následujících sekcí `Entry`. Subindexy se nesmí vynechávat, takže je to současně nejvyšší subindex.

`Subindex` – Číslo subindexu, který definuje tato sekce `Entry`.

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

`bit 0`	hodnota/subindex může být čten systémem REX
`bit 1`	hodnota/subindex může být měněn/zapisován systémem REX
`bit 2`	hodnota/subindex může být čten po sběrnici CAN
`bit 3`	hodnota/subindex může být měněn/zapisován po sběrnici CAN
`bit 4`	hodnota/subindex může být mapován do PDO
`bit 5`	nastavuje se pokud, je jen jeden subindex a je považován za hodnotu celého objektu

`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 - 8bajtové 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 7

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.

- Někdy je potřeba číst/zapisovat hodnotu, která nejde namapovat do PDO. Protože zvolená koncepce umožňuje předávat do výkresu jen hodnoty z lokálního „object dictionary“ a nikoliv volat SDO, je potřeba hodnoty z jiné stanice nějak přečíst. Za tím účelem byly do struktury SDO client parameters (tj. do objektů 0x1280 až 0x12FF) přidány od subindexu 6 další parametry. Musí být typu UNSIGNED64 nebo SIGNED64 kde (od nejvyšších bitů):

UNSIGNED16	perioda čtení/zápisu v milisekundách
UNSIGNED16	index lokálního objektu
UNSIGNED16	index objektu na vzdálené stanici
UNSIGNED8	subindex lokálního objektu na vzdálené stanici
UNSIGNED8	subindex objektu na vzdálené stanici

  S každou stanicí lze takto vyměňovat až 250 objektů pomocí SDO. Perioda je vlastně „ihibit time“, tj. dotazy se nevysílají častěji. Pokud je perioda krátká a dotazů hodně, bude skutečná perioda delší. Formát objektu/subindexu na vzdálené stanici se předpokládá stejný jako v lokálním objektu/subindexu.

- Pokud je potřeba konfigurovat PDO po síti (tj. nastavovat objekty 0x1400 až 0x1BFF) je potřeba vždy nejprve stanici přepnout do PREOP režimu, pak zakázat PDO(tj. v MessageID nastavit bit31), dále nastavit délku pole (tj. subindex 0) na 0, a pak změnit další prvky objektu. Nakonec nastavit správnou délku objektů a MessageID. Postup se může mírně lišit podle výrobce, ale pokud do těchto prvků (případně i jiných) nejde zapisovat, tak příčina je pravděpodobně jedna z výše uvedených.

- Implementace `CANopen` v systému REX nepodporuje TIME_SYNC (tj. přesnější synchronizaci) a nepodporuje multiplexed-PDO. SYNC packet je vysílán v každé periodě ovladače `CanDrv` . Režim `Slave` je podpořen, ale synchronizace na SYNC

packet je jen přibližná (zprávy se vyčítají z komunikační karty s periodou, která je nastavena pro ovladač v systému REX a to je tedy i nepřesnost zasynchronizování).

- V případech, kde více objektů slouží ke stejnému účelu, se musí vždy použít první objekt z dané skupiny. Vždy tedy musí být použita dvojice 0x1800/0x1A00 pro odchozí PDO, 0x1400/0x1600 pro příchozí PDO, 0x1200 pro serverovská SDO a 0x1280 pro klientská SDO. Toto drobné omezení zjednodušuje implementaci.

- Zdá se, že pokud vyslanou zprávu žádná stanice nepřijme komunikační karta přejde do chybového stavu a za určitých okolností se již nevzpomatuje. Toto nastává pokud se připojují zařízení na sběrnici CAN „pod napětím“ popřípadě se každé zařízení zapíná a vypíná nezávisle. Podobná chyba také vzniká při různých komunikačních rychlostech. V takovém případě je nutné vše vypnout a zapnout pokud možno najednou nebo Master stanici jako poslední.

- V linuxu jsou podporovány 3 typy ovladačů:

raltime-char-device	je podporovan jen v jadru s rozsire
char-device	pokud je spravne nainstalovan, zob
net-device	pokud je spravne nainstalovan, zob

Kapitola 8

Co dělat při problémech

Nejčastější chyby jsou:

Nezapojený ukončovací odpor.

Rozdílná bitová rychlost u zařízení na jedné lince.

Pokud se používá 29-bitové `Message ID`, existují zřejmě různé implementace takže se někdy stává, že je obráceně pořadí bitů (nejnižších 11bitů je na nejvyšších bitech MessageID). Pokud tedy zprávy nechodí, je vhodné toto zkontrolovat.

Každý komunikační standard definuje, zda se pro přenos použije little-endian nebo big-endian formát. CANopen používá little-endian (tj. stejný jaký používají procesory Intel nebo ARM). Občas se stává, že na to vývojáři zapomenou a konverzi neprovádí (problém samozřejmě vzniká, pokud procesor je big-endian, tj. například Motorola nebo Siemens), takže vícebajtová čísla mají obráceně pořadí bajtů.

V případě, že daný ovladač `CanDrv`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 `CanDrv`.

Literatura

[1] REX Controls s.r.o.. *Funkční bloky systému REX – Referenční příručka*, 2016.

Referenční číslo dokumentace: 7567