Ovladač `EplDrv` systému REX
Uživatelská příručka

REX Controls s.r.o.

Verze 2.50.3
Plzeň
28.2.2017

Obsah

Kapitola 1

Ovladač `EplDrv` a systém REX

1.1 Úvod

Ovladač `EplDrv` přináší do systému REX podporu komunikace Ethernet POWERLINK. Ethernet POWERLINK (dále jen EPL) je otevřený protokol pro komunikaci v reálném čase po síti Ethernet.

Komunikace mezi stanicemi probíhá v sítí EPL cyklicky, dosažitelná délka nejkratšího cyklu je udávána 200 mikrosekund. Do sítě lze připojit až 240 stanic (uzlů). V síti je vždy přítomen jeden řídicí uzel (Managing Node - MN) a jeden nebo více řízených uzlů (Controlled Node - CN). Ovladač `EplDrv` implementuje oba typy uzlů.

Ovladač využívá openPOWERLINK[1], otevřenou implementaci protokolu EPL pod licencí BSD.

1.2 Požadavky na systém

Hlavním požadavkem na cílové zařízení je přítomnost podporovaného čipu Ethernet. Podporovány jsou následující čipy:

- RealTek 8139C/C+/D
- RealTek 8169
- VT6105M (Via Rhine)

Podporovány jsou platformy GNU/Linux a GNU/Linux s rozšířením reálného času Xenomai. Tuto platformu je vhodné zvolit v případě požadavku na krátký komunikační cyklus a deterministické odezvy.

Na vývojové zařízení nejsou kladeny žádné zvláštní nároky. Postačí jakékoliv PC s operačním systémem Windows Vista/7/8/10.

[1] http://sourceforge.net/projects/openpowerlink/

1.3 Instalace ovladače

Ovladač `EplDrv` se instaluje jako součást instalace řídicího systému REX. Pro nainstalování ovladače je nutné v instalačním programu systému REX zaškrtnout volbu `Ovladač protokolu Ethernet POWERLINK`. 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 na vývojové stanici zkopíroval soubor `EplDrv_H.dll`, který zajišťuje začlenění protokolu EPL do vyvíjené aplikace a také konfiguraci protokolu.

Na cílové stanici se instalace provede zkopírováním souboru `EplDrv_T.so` do adresáře `/usr/lib`. Soubor je taktéž součástí instalace systému REX pro systém GNU/Linux (+Xenomai).

Na cílové stanici je dále nutné nainstalovat jaderný modul `eplkrnl.ko` a příslušný ovladač síťové karty. Taktéž je nutné zakázat inicializaci síťové karty, která je určena pro komunikaci EPL, z automatické inicializace při startu jádra. Více informací o instalaci prostředí GNU/Linux pro systém REX je k dispozici v [1].

Kapitola 2

Základní principy fungování sítě

V této kapitole jsou stručně popsány základní funkce a vlastnosti sítě Ethernet POWERLINK. Pro hlubší porozumění se doporučuje nahlédnout do dokumentace [2].

2.1 Architektura

EPL využívá standarní fyzickou vrstvu Fast Ethernet s propustností 100Mb/s. Je podporováno připojení až 240 uzlů.

Při instalaci se doporučuje používat kabel UTP nebo STP ve variantě alespoň 5e. Při návrhu sítě by měla být zachována pravidla, která jsou popsána v dokumentaci [2]. Nejčastější topologií sítě je had. Uzly mají obvykle 2 Ethernet porty a fugují jako hub. Uzly lze zapojit také v topologii hvězda za použití rozbočovače. Jako rozbočovač by měl být použit výhradně hub. Switche jsou možné, ale nelze je doporučit.

Uzly jsou v síti identifikovány tzv. „číslem uzlu/stanice“ (node ID). V síti je vždy přítomen jeden řídicí uzel (Managing Node - MN) a jeden nebo více řízených uzlů (Controlled Node - CN). Řídicí uzel má vždy číslo 240, řízené uzly jsou pak číslovány od 1 do 239.

2.2 Datová struktura

Datová struktura vychází ze standardu *CANopen*. Data poskytována uzly jsou uložena v tzv. „objektech“. Všechny objekty tvoří tzv „slovník objektů“. Objekty jsou identifikovány dvěma čísly. Prvním číslem je 16-bitový index a druhým číslem je 8-bitový sub-index.

Slovník objektů je rozdělen na oblasti, kterým standard přiděluje určitý význam. Objekty v rozsahu 0 až 0x0FFF slouží pro identifikaci typů, rozsah 0x1000 až 0x1FFF je určen pro nastavení komunikace. Rozsah 0x2000 až 0x5FFF je určen pro objekty definované výrobcem a objekty s indexem nad 0x6000 jsou vyhrazené pro datové objekty definované ve standardním profilu zařízení. Při konfiguraci systému REX by měla být tato konvence dodržena.

Objekty lze nejčastěji chápat jako proměnné. Typ proměnné může být jednoduchý (VAR – BOOLEAN, INTEGER8, REAL32 apod.) nebo složený (RECORD, ARRAY). Jednoduchý typ je identifikován indexem i sub-indexem nebo může být identifikován pouze indexem. V takovém případě se automaticky předpokládá hodnota sub-indexu 0. Složený typ je identifikován indexem. Subindex pak slouží pro identifikaci jeho prvků. Sub-index 0 by měl být vždy typu UNSIGNED8 a měl by udávat počet prvků složeného typu. Speciálním typem je DOMAIN, který je chápán jako jednoduchá proměnná libovolné, a v určitých případech i proměnné, délky. Je pak na výrobci, aby význam této proměnné nebo jejích částí popsal.

2.3 Mechanismy výměny dat

Komunikace v síti probíhá cyklicky. Délka cyklu je udávána v mikrosekundách. Pro přenos dat se využívají dva druhy mechanismů. První zajišťuje výměnu dat na žádost aplikace v asynchronní (nedeterministické) části cyklu. Objekty, jejichž data se komunikují tímto mechanismem jsou tzv. „servisní datové objekty“ (SDO) a tento mechanismus je označen jako tzv. „služba SDO“. Všechny objekty ve slovníku by měly být takto přístupné v závislosti na nastavených přístupových právech.

Druhým mechanismem je periodická komunikace dat v každém cyklu v jeho izochronní fázi. Tento mechanismus zajišťuje deterministické doručování dat. Objekty, které je možné takto komunikovat jsou tzv. „procesní datové objekty“ a mechanismus je označen jako tzv. „služba PDO“. Pouze objekty, které mají nastavenou podporu pro PDO, lze komunikovat pomocí služby PDO. Nedisponuje-li objekt touto vlastností, je nutné k němu přistupovat výhradně pomocí služeb SDO.

Kapitola 3

Zařazení ovladače do aplikace

V této kapitole je popsán způsob začlenění komunikace Ethernet POWERLINK do algoritmu pro řídicí algoritmus REX. V první části je popsán postup při začleňování, v další části je pak popsána konvence pojmenování bloků `FROM` a `GOTO`, které slouží v systému REX mimo jiné k předávání dat mezi ovladačem a algoritmem. Podrobný popis těchto bloků lze nalézt v [3].

3.1 Konfigurace exekutivy

Přidání ovladače `EplDrv` do hlavního souboru projektu (exekutivy) je znázorněno na obr. 3.1.

Pro zařazení ovladače do projektu slouží dva bloky. První je blok typu `MODULE`, který je připojen na výstup `Modules` bloku exekutivy. Název bloku se shoduje s názvem sdílené knihovny, ve které je tento modul uložen. Sdílená knihovna ovladače EPL se jmenuje `EplDrv`.

Druhý blok je typu `TIODRV` a je připojen na výstup `Drivers`. Parametry bloku mají následující význam:

`module` - název modulu s ovladačem, vždy `EplDrv`,

`classname` - název ovladače v modulu, vždy `EplDrv`,

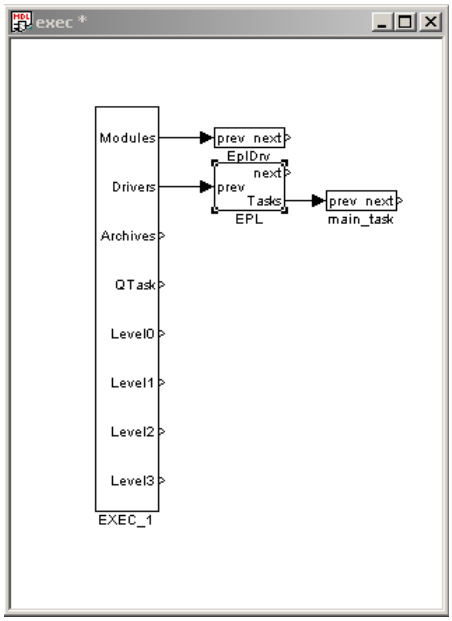
`cfgname` - název konfiguračního souboru ovladače, viz dále,

`factor` - v případě ovladače EPL bez významu

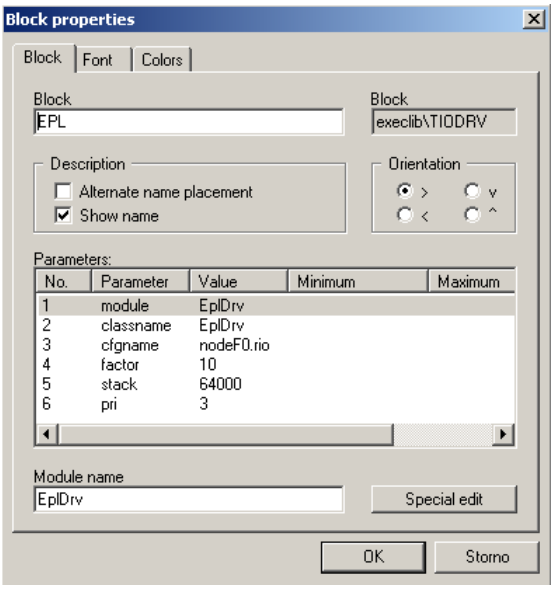
`stack` - velikost zásobníku, měla by být vhodně zvolena, minimálně se doporučuje 64kB,

`pri` - v případě ovladače EPL bez významu.

Vhodně vyplněné parametry ovladače jsou na obrázku 3.2.
Blok ovladače by měl být vždy pojmenován `EPL`.



Obrázek 3.1: Příklad zařazení ovladače `EplDrv` do projektu aplikace



Obrázek 3.2: Příklad nastavení parametrů ovladače `EplDrv` aplikace

3.2 Časování

Časovač ovladače EPL je zcela nezávislý na časovači exekutivy. V případě řídicího uzlu je využíván samostatný přesný časovač v jádře. V případě řízeného uzlu je časovače řízen rámcem *SoC*, který je vyslán řídicím uzlem na začátku každého cyklu. V obou případech je tik časovače synchronní s komunikačním cyklem sítě EPL.

Ovladač umožňuje synchronizovat vykonávání algoritmu s cyklem sítě. Algoritmus, který má být takto synchronizován, se připojuje přímo na výstup `Tasks` bloku ovladače. Synchronních algoritmů může být více. V takovém případě se algoritmy připojují sériově. Algoritmy jsou pak vykonávány postupně.

3.3 Přístup k datovým objektům

K datovým objektům ve slovníku objektů protokolu EPL se z algoritmu přistupuje pomocí bloků `FROM` a `GOTO`. Parametr `GotoTag` těchto bloků pak jednoznačně identifikuje objekt.

K objektům lze přistupovat pomocí služeb PDO i SDO. Pomocí služeb PDO lze přistupovat pouze k objektům, které tuto službu podporují. Jedná se o parametr *Pdo mapping*.

Je důležité říci, že pomocí služeb PDO lze z algoritmu přistupovat výhradně k lokálnímu slovníku objektů. Požaduje-li se komunikace dat se vzdáleným uzlem, je nutné objekty tzv. „namapovat“. Mapování objektů se provádí zpravidla při startu sítě během konfigurace síťových uzlů. Mapování je tedy zpravidla součástí konfiguračního souboru sítě EPL. Pokud jsou objekty PDO namapovány, jsou jejich data komunikována v každém cyklu sítě, aktualizace hodnot je tedy zajištěna v každém cyklu.[1]

Pomocí služeb SDO lze přistupovat k lokálním i vzdáleným objektům. Přístup pomocí služeb SDO je asynchronní a v porovnání se službami PDO pomalejší. V žádném případě nelze očekávat aktualizaci v každém cyklu.

3.4 Konvence pojmenování vstupů/výstupů

Parametr `GotoTag` bloku `FROM` a `GOTO` jednoznačně identifikuje proměnnou. Konvence pro tento parametr je v případě ovladače EPL následující:

```
EPL__NazevObjektu
```

kde `EPL` se odkazuje na blok `IODRV`, který je pojmenovaný `EPL`, jedná se tedy o odkaz na protokol Ethernet POWERLINK. `NazevObjektu` identifikuje objekt ve slovníku objektů (Object Dictionary, OD).

Identifikace objektu podle `NazevObjektu` probíhá v ovladači následovně:

[1] samozřejmě pouze pokud je vzdálený uzel připojen a komunikuje v každém cyklu, tzn. není zařazen do multiplexu

1. hledá se EplObjekt podle názvu v lokálním OD
2. hledá se EplObjekt podle aliasu v lokálním OD. Pokud je objekt nalezen:
   (a) podporuje-li objekt služby PDO, přistupuje se k němu pomocí služby PDO a data jsou uložena v *Process Image (PI)*, který je komunikován z/do jádra v každém EPL cyklu
   (b) nepodporuje-li objekt služby PDO, přistupuje se k němu pomocí služeb SDO
3. pokud je z řetězce EplObjektu jasné, že se jedná o SDO (např. je definován trigger, perioda), přistupuje se k objektu automaticky přes SDO. Podle aliasu je vyhledán objekt v OD všech uzlů v síti(viz EplConfig - aliases). Je-li objekt nalezen, přistupuje se k němu pomocí SDO, není-li objekt nalezen, je uživateli hlášena chyba.

3.5 Přístup pomocí PDO

Data objektů, které jsou přístupné službami PDO, jsou uložena v *Process Image* (PI), který je komunikován v každém EPL cyklu mezi ovladačem a jádrem. PI je vytvářen automaticky a dynamicky. Přítomnost v PI ještě neznamená, že data budou v síti skutečně komunikována - to záleží na konfiguraci objektů PDO. Mezi algoritmem a jádrem však komunikace (výměna dat) probíhá vždy, nezávisle na konfiguraci sítě.

3.6 Přístup pomocí SDO

Data objektů jsou v tomto případě komunikována asynchronně nedeterministicky pomocí SDO služeb. Způsob výměny dat pomocí SDO lze dále upřesnit pomocí vlajky, která má následující tvar: `{EPL__SdoObjekt_Par`, kde Par může mít následujíc tvar:

- `O`: „Once“ - objekt je čten/zapsán při přechodu do režimu Operational a při každé změně (výchozí tj. `EPL__SdoObjekt_O` je stejné jako `EPL__SdoObjekt`)
- `P`: „Periodic“ - objekt je čten/zapsán s periodou, která je udána číselně za parametrem v milisekundách
- `W`: „Written“ - jen pro zápisové SDO, udává skutečnou hodnotu, která byla do SDO naposledy úspěšně zapsána.
- `T`: „Trigger“ - objekt je čten/zapsán pouze pokud je nastaven tento trigger. Objekt je čten/zapsán při nastaveném triggeru jednou nebo periodicky podle toho, zda je definován jako `O` nebo `P`.
- `S`: „Status“ - poslední návratová hodnota služby SDO (viz openPowerlink stack)

Pokud je stejný SDO objekt čten i zapisován, je nutné trigger a status rozslišit:

- `X`: „Trigger“ pro čtení

- `Z`: „Status“ pro čtení
- `V`: „Trigger“ pro zápis
- `Y`: „Status“ pro zápis

Omezení:

- není možné přistupovat k SDO s různou periodou, doporučuje se periodu specifikovat pouze u jednoho `Goto`/`From` tagu
- nedoporučuje se kombinovat *Once* a *Periodic*
- pokud se kombinuje *Once* a *Periodic* a/nebo čtení a zápis, doporučuje se použít triggery
- nedoporučuje se kombinovat SDO a PDO přístup pro stejný objekt
- minimální perioda je 1 ms, avšak při velkém počtu SDO objektů se skutečná perioda může výrazně prodloužit

3.7 Speciální vstupy/výstupy

EPL driver definuje sadu speciálních vstupů/výstupů, které nejsou přímo objekty v OD, ale vztahují se k parametrům a stavům EPL driveru.

- `NodeOk` (R): Má hodnotu TRUE, pokud je uzel v režimu *operational*. Následuje-li `_cislo`, vztahuje se údaj k uzlu s odpovídajícím číslem.
- `NodeStatus` (R): Udává stav automatu NMT daného uzlu (viz openPowerlink stack). Následuje-li `_cislo`, vztahuje se údaj k uzlu s tímto číslem.
- `Error` (R): Hodnota odpovídá číslu poslední chyby EPL stacku (viz openPowerlink stack).
- `ErrorSource` (R): Hodnota určuje zdroj poslední chyby EPL stacku (viz openPowerlink stack).
- `BootEvent_cislo` (R): Hodnota určuje událost startu uzlu EPL (viz openPowerlink stack).
- `BootError_cislo` (R): Hodnota odpovídá číslu chyby startu uzlu EPL (viz openPowerlink stack).
- `Terminate` (R): Tento příznak indikuje požadavek na ukončení činnosti EPL stacku/driveru.
- `AllowTerminate` (R/W): Tento příznak povoluje ukončení činnosti EPL stacku/driveru. Tento příznak nastaví algoritmus. K ukončení činnosti dojde až po nastavení tohoto příznaku nebo uplynutí určité doby. Pokud se `AllowTerminate` v algoritmu nepoužije, je komunikace ukončena okamžitě.

Kapitola 4

Konfigurace sítě

V této kapitole je popsán způsob uvedení sítě Ethernet POWERLINK do provozu. Pro hlubší porozumění principům konfigurace uzlů se doporučuje nahlédnout do [2].

4.1 Mechanismy konfigurace

Předtím, než je uzel sítě EPL uveden do provozu, je ve většině případů potřeba provést jeho konfiguraci, tj. nastavit vhodné hodnoty v daných objektech. Příkladem může být například uzel s konfigurovatelnými vstupy a výstupy. Před uvedením uzlu do režimu *operational* bude zřejmě nutné zvolit správné režimy a rozsahy. V této sekci je popsán obecný mechanismus konfigurace uzlů v síti EPL.

Součástí standardu protokolu EPL je i specifikace způsobu konfigurace síťových uzlů. Aby bylo možné uzel nakonfigurovat, je potřeba znát jeho objekty ve slovníku objektů a jejich význam. K tomu slouží dva druhy popisných souborů:

`XDD` - tzv. „Device Description“ je popisný soubor všech objektů daného uzlu.

`XDC` - tzv. „Device Configuration“ obsahuje hodnoty objektů pro konkrétní aplikaci. Tento soubor je možné chápat jako počáteční konfiguraci, která uvede uzel do požadovaného režimu.

Tyto dva typy souborů jsou si navzájem podobné. Jsou uloženy ve formátu XML v odpovídajícím schématu. Liší se příponou a informacemi, které poskytují. Soubory mohou být uloženy v paměti uzlu nebo by měly být distribuovány výrobcem. Praxe je zatím taková, že výrobce často nedává k dispozici ani jeden z těchto souborů. Pokud jsou však k dispozici informace např. v textové podobě (uživatelský manuál apod.), je možné si soubor XDC vytvořit. Pokud nejsou o objektech zařízení informace ani v textové podobě, nelze většinou takové zařízení připojit k systému REX.

Systém REX podporuje tzv. „stručnou“ konfiguraci. Ta je vytvořena na základě souboru XDC a obsahuje pouze nezbytně nutné informace pro konfiguraci uzlu. Stručná konfigurace je uložena pro každý uzel v síti v binárním formátu na uzlu MN. Při startu sítě nebo po připojení nového (avšak známého) uzlu do sítě je porovnána aktuální verze

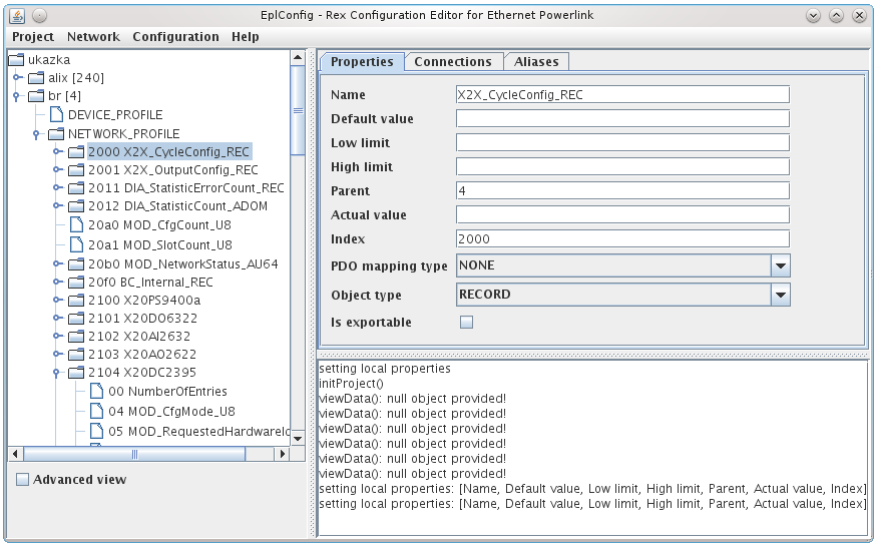
konfigurace uzlu vůči verzi na uzlu MN. Pokud se verze liší, postará se uzel MN o konfiguraci uzlu podle nové verze. K tomu se využívá služeb SDO. Poté je uzel restartován. Po novém startu by již měly verze souhlasit. Pokud jsou verze shodné, je uzel uveden do operačního režimu.

4.2 EplConfig– konfigurátor sítě

Pro konfiguraci a začlenění systému REX do sítě EPL slouží nástroj EplConfig. Tento nástroj zajišťuje následující funkce:

- **vytváření a editaci** souboru XDC,
- **generování** PDO mapování,
- **generování** „stručné“ binární konfigurace,
- **konfiguraci** ovladače `EplDrv`.

Konfigurátor je samostatná aplikace vyžadující pro svůj běh prostředí Java. Spouští se z prostředí RexDraw tlačítkem `Configure` v dialogu ovladače `EplDrv`. Je také možné spustit přímo soubor `eplconfig.jar`. Hlavní okno konfigurátoru je zachyceno na obrázku 4.1.



Obrázek 4.1: Hlavní okno konfigurátoru EplConfig

Podrobnější informace jsou k dispozici v uživatelském manuálu k programu EplConfig.

4.3 Jak postupovat

Vývoj algoritmů pro všechny uzly se systémem REX, které mají být zapojeny do sítě EPL, by měl probíhat v jediném adresáři.

Systém REX může v síti EPL zastávat roli uzlu MN i CN. Konfigurace protokolu EPL se pro tyto uzly částečně liší. V obou případech je však společný hlavní cíl – definovat uživatelskou část slovníku objektů. Tuto funkci (kromě dalších) zastává nástroj EplConfig.

Obecný postup konfigurace je následující. Jediným vstupem do procesu konfigurace sítě nástrojem EplConfig jsou popisné soubory uzlů `.xdc`. Uživatel vytvoří nový projekt a načte do něj tyto soubory. Poté provádí potřebné úpravy. Na základě jeho úprav pak nástroj EplConfig vytvoří několik souborů, které jsou dále využívány při kompilaci konfigurace. Následující soubory tvoří výstup konfigurátoru:

`nodeXX.rio` je popisný soubor slovníku objektů pro uzel s číslem `XX` v hexadecimálním tvaru. Soubor slouží pro potřeby ovladače `EplDrv` a musí být k dispozici při překladu.

`mnobd.txt` je soubor se „stručnou“ konfigurací sítě v textovém formátu. Tento soubor není dále používán, ale může sloužit např. ke kontrole uživatelem vytvořené konfigurace.

`mnobd.cdc` je soubor se „stručnou“ konfigurací sítě v binárním formátu. Soubor slouží pro potřeby ovladače `EplDrv` a musí být k dispozici při překladu algoritmu uzlu MN.

4.3.1 REX v roli řídicího uzlu

Má-li být systém REX připojen do sítě EPL jako uzel MN, je nutné mít k dispozici konfigurační soubory XDC pro všechny uzly sítě. Tyto soubory by měly být dodány spolu se zařízením. Např. u zařízení od firmy B&R se využívá program „Fieldbus Designer“, který na základě sestavy sběrnice X2X vytvoří soubor XDC. Dále je potřeba mít k dispozici popisný soubor XDD pro systém rex. Ten je dodáván spolu s ovladačem `EplDrv` a jmenuje se `REX_profile.xdd`. V tomto souboru je definována komunikační část slovníku objektů tj. objekty *0x1000* až *0x1FFF*.

Při konfiguraci se postupuje následovně:

1. Vytvořte nový projekt ve stejném adresáři, v jakém budete vyvíjet algoritmy systému REX.

2. Do projektu importujte soubory XDC pro všechny CN uzly třetích stran.

3. Do projektu importujte soubor `REX_profile.xdd` pro všechny CN uzly se systémem REX a taktéž pro pro uzel MN. Pro uzly se systémem REX lze také využít již existující soubor XDD/XDC např. z předchozích projektů.

4. Definujte uživatelskou část slovníku objektů (tj. objekty s indexem *0x2000* až *0x3FFF* u všech uzlů se systémem REX.

5. Propojte všechny proměnné, které mají být komunikovány službou PDO. K tomuto účelu slouží tabulka „connections“. Každá řádka tabluky reprezentuje jeden PDO kanál. Proměnné se do kanálu zařadí přetáhnutím objektu ze stromu objektů.

6. Definujte přiřazení aliasů jednotlivým objektům. Aliasy slouží pro další volitelné pojmenování objektů. Pokud potřebujete v algoritmu přistupovat k objektům vzdáleného uzlu přes SDO, musí být pro tento objekt alias definován.

7. Na základě vytvořených PDO kanálů vytvořte příslušná PDO mapování příkazem „Configuration–Generate PDO“.

8. Vytvořte výstupní soubory pro systém REX příkazem „Configuration–Generate ouptuts for REX“.

Nyní je síť EPL nakonfigurována. Od této chvíle již nemusíte nástroj EplConfig používat a můžete se věnovat pouze vývoji algoritmů. Pokud budete algoritmy přenášet, nezapomeňte kromě souborů `.mdl` zkopírovat také všechny soubory `.rio` a soubor `mnobd.cdc`.

4.3.2 REX v roli řízeného uzlu

Pokud má být systém REX provozován jako uzel CN, mohou nastat dvě varianty:

1. MN sítě EPL je uzel se systémem REX,

2. MN sítě EPL je uzel s jiným systémem.

Pokud je MN systém REX, postupujte podle předchozí kapitoly. Pokud je MN systém třetí strany, musíte pravděpodobně do tohoto systému importovat konfigurační soubor XDC systému REX. Tento soubor vytvoříte nástrojem EplConfig následovně:

1. Vytvořte nový projekt ve stejném adresáři, v jakém budete vyvíjet algoritmus systému REX.

2. Do projektu importujte soubor `REX_profile.xdd`.

3. Definujte uživatelskou část slovníku objektů (tj. objekty s indexem *0x2000* až *0x3FFF*.

4. Vytvořte výstupní soubory pro systém REX příkazem „Configuration–Generate outputs for REX“.

V tomto okamžiku jsou k dispozici konfigurační soubory pro systém REX. V adresáři s projektem je také uložen konfigurační soubor `XDC`. Např. má-li algoritmus systému REX běžet na uzlu číslo 12 a má-li např. tento uzel název *REXmain*, bude se konfigurační soubor jmenovat `010_REXmain.cdc`. Tento soubor může být importován do systému na uzlu MN.

Kapitola 5

Ukázkový příklad

Tato kapitola provede uživatele procesem konfigurace sítě EPL krok po kroku. Cílem je vytvořit jednoduchou síť EPL se dvěma uzly. Systém REX na platformě Alix[1] zajišťuje v síti úlohu řídicího uzlu. Řízený uzel číslo 3 je tvořen kontrolérem BC0083, ke kterému je připojen modul analogových vstupů AI4632 a modul analogových výstupů AO2632. Cílem je sestavit funkční síť EPL včetně konfigurace objektů PDO tak, aby byly hodnoty vstupů i výstupů komunikovány v každém cyklu.

1. krok
Pomocí aplikace „FieldbusDESIGNER" vytvořte konfigurační soubor XDC pro kontrolér BC0083 s moduly AI4632 a AO2632. Nejprve založte nový projekt, jako MN zvolte *Generic POWERLINK MN*. Do projektu vložte kontrolér a k němu připojte moduly.

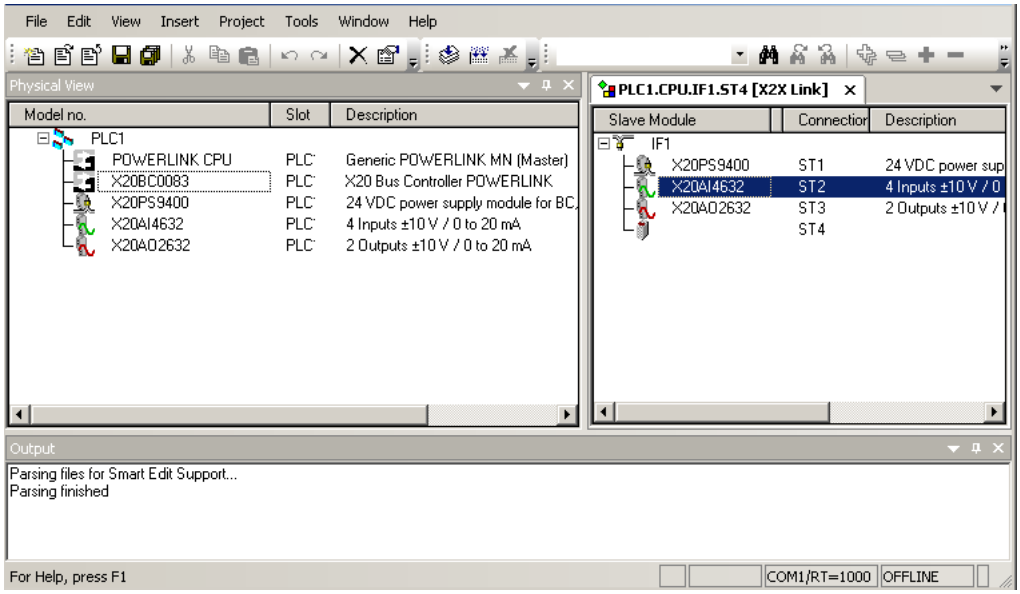
Parametry modulů nakonfigurujte pomocí *Open I/O configuration*. Nakonec přeložte celý projekt příkazem *Project–build configuration*. Pokud je projekt přeložen bez chyb, máte v adresáři `Output` uložen konfigurační soubor `0100006C_X20BC0083_4.xdc`, který budete potřebovat v následujícím kroku.

2. krok
Spusťte aplikaci EplConfig a vytvořte nový projekt. Příkazem *Network–Add node* importujte soubor `REX_profile.xdd`. Tento soubor představuje řídicí uzel se systémem REX. Zaškrtněte volbu *It's a REX node*. Uzel pojmenujte *Alix* a přiřaďte mu číslo 240.

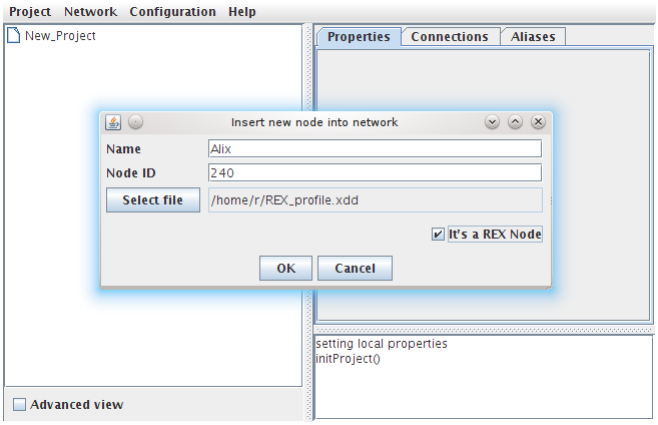
Stejným způsobem proveďte import řízeného uzlu reprezentovaného souborem `0100006C_X20BC0083_4.xdc` z FieldbusDESIGNERu. V tomto případě nezaškrtávejte volbu *It's a REX node*. Uzel pojmenujte *BR* a přiřaďte mu číslo 3.

3. krok
Nyní si prohlédněte slovník objektů. Na pozicích s indexem od 2100 se nachází konfigurace pro jednotlivé moduly. Na sub-indexu 1E je umístěn tzv. *Input Image* tj. aktuální hodnoty vstupních veličin v kompaktní binární podobě. Zkontrolujte, zda je sub-index 1E typu *DOMAIN* a pokud tomu tak není, nastavte to.

[1] http://www.pcengines.ch/



Obrázek 5.1: Hlavní okno programu FieldbusDESIGNER

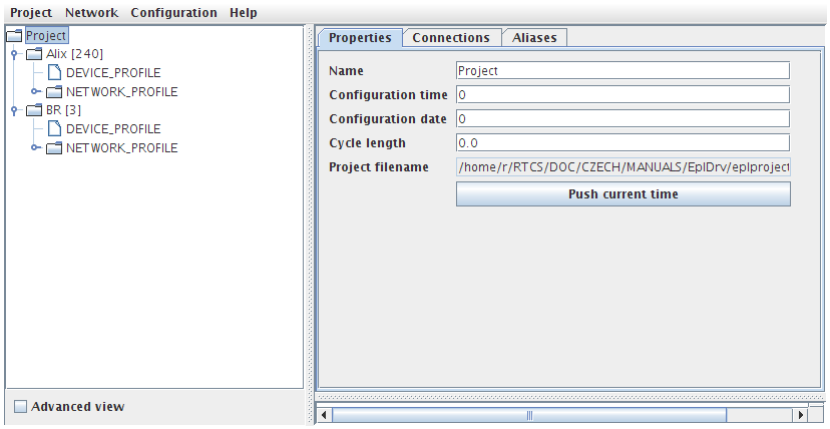


Obrázek 5.2: Import řídicího uzlu

Na subindexu 20 je umístěn tzv. *Output Image* tj. aktuální hodnoty výstupních veličin v kompaktní binární podobě. Zkontrolujte, zda je sub-index 20 typu *DOMAIN* a pokud tomu tak není, změňte nastavení na tuto hodnotu.

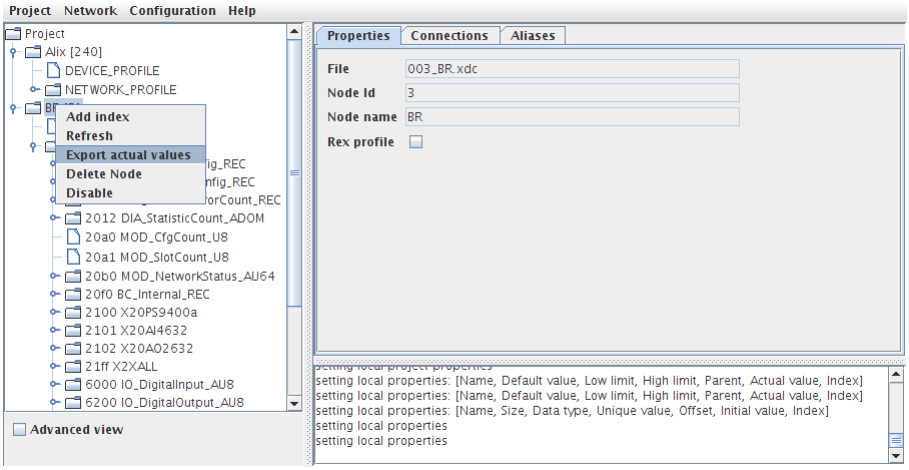
Pokud je objekt nastavený na typ *DOMAIN* a jsou-li k tomuto objektu příslušné informace, lze objekt „rozbalit“ až na jeho jednotlivé položky. Tak by tomu mělo být pro i sub-indexy 1E a 20.

4. krok



Obrázek 5.3: Projekt s dvěma uzly

Exportujte všechny aktuální hodnoty příkazem *Export actual values* z kontextové nabídky uzlu *BR* (kliknutím pravým tlačítkem ve stromu objektů na uzel *BR*).



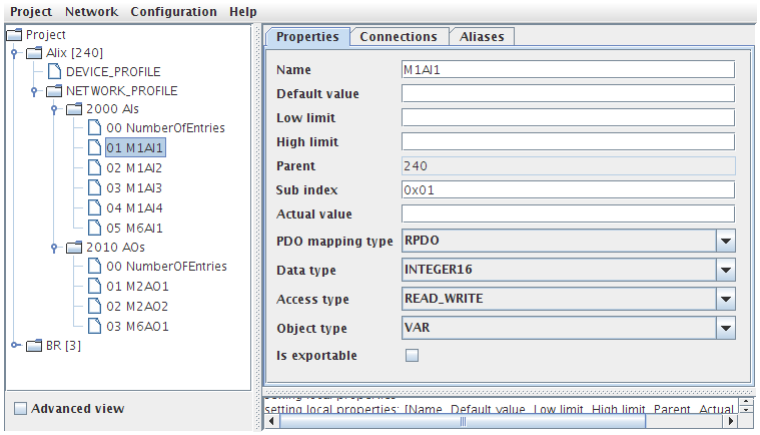
Obrázek 5.4: Export aktuálních hodnot

5. krok

Nyní je potřeba vytvořit slovník objektů v řídicím uzlu. Vytvořené objekty umístěte výhradně do oblasti vyhrazené pro výrobce tj. na indexy 2000 a výš.

Vytvořte pole analogových vstupů. Pole může být reprezentováno objektem *RECORD* nebo *ARRAY*. Pole pojmenujte např. *AI*. Vytvořte 4 objekty se subindexy 01 až 04. Parametry objektu nastavte následovně:

Object type – `VAR`



Obrázek 5.5: Objekty pro hodnoty vstupů

Access type – `READ_WRITE`
: Poznámka: *Access type* určuje práva přístupu k objektu z pohledu sítě EPL. Vytváříme objekty na řídicím uzlu pro analogové vstupy tj. data do těchto objektů budou přes síť zapisována. Právo čtení můžeme také dovolit.

Data type – `INTEGER16`
: Poznámka: data vstupního modulu jsou na kontroléru reprezentována celočíselně na 16-bitech se znaménkem.

Pdo mapping – `RPDO`
: Poznámka: *Pdo mapping* určuje jakým směrem mohou „téct“ data z pohledu uzlu. Data do těchto objektů budou přijímána ze sítě EPL tj. nastavíme RPDO (Receive-PDO).

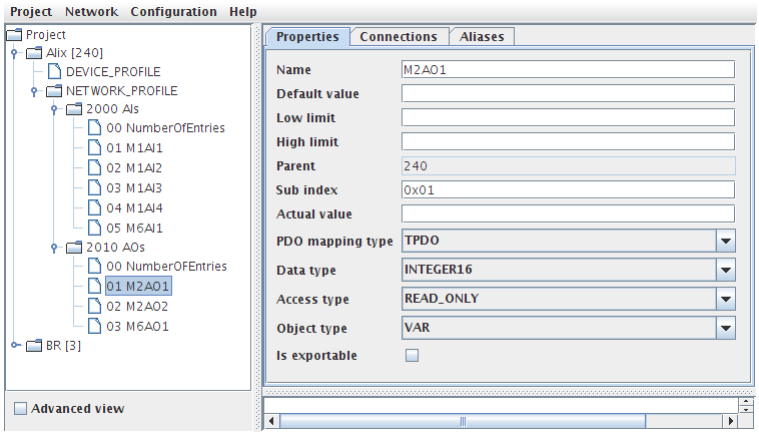
Is exportable – `prázdné`
: Poznámka: tyto objekty nejsou součástí konfigurace, proto nemá smysl, aby byly aktuální hodnoty součástí konfiguračních dat.

Analogicky vytvořte pole analogových výstupů. Pole může být reprezentováno objektem *RECORD* nebo *ARRAY*. Pole pojmenujte např. *AO*. Vytvořte 4 objekty se sub-indexy 01 až 02. Parametry objektu nastavte následovně:

Object type – `VAR`

Access type – `READ_ONLY`
: Poznámka: *Access type* určuje práva přístupu k objektu z pohledu sítě EPL. Vytváříme objekty na řídicím uzlu pro analogové výstupy tj. data z těchto objektů budou ze sítě čtena. Zápis nelze povolit.

Data type – `INTEGER16`
: Poznámka: data výstupního modulu jsou na kontroléru reprezentována celočíselně na 16-bitech se znaménkem.



Obrázek 5.6: Objekty pro hodnoty výstupů

Pdo mapping – `TPDO`
: Poznámka: *Pdo mapping* určuje jakým směrem mohou „téct“ data z pohledu uzlu. Data z těchto objektů budou vysílána do sítě EPL tj. nastavíme TPDO (Transmit-PDO).

Is exportable – `prázdné`
: Poznámka: tyto objekty nejsou součástí konfigurace, proto nemá smysl, aby byly aktuální hodnoty součástí konfiguračních dat.

6. krok

V tomto kroku definujte tok dat mezi procesními datovými objekty. Otevřete záložku *Connections* s tabulkou datových toků a z levého panelu se stromovou hierarchií projektu přetáhněte do tabulky ty objekty PDO, jejichž data mají být komunikována. Do sloupce *Source* přetáhněte poskytovatele dat tj. objekt typu *TPDO*. Do sloupců *Destination* pak přetáhněte konzumenty těchto dat tj. objekty typu *RPDO*. Dbejte na to, aby byly všechny objekty v jedné řádce stejného typu.

7. krok

Na základě definovaných datových toků vygenerujte *mapovaní objektů PDO* příkazem *Configuration–generate PDO*. Tento krok opakujte při každé změně v tabulce datových toků.

8. krok

Vygenerujte popisný soubor pro ovladač systému REX příkazem *Configuration–generate outputs for REX*. V tomto případě bude vytvořen soubor `nodeF0.rio` pro řídicí uzel. Tento krok opakujte při každé změně slovníku objektů.

9. krok

Otevřete program RexDraw a vytvořte exekutivu řídicího uzlu podle následujícího schématu. Soubor pojmenujte `epl_exec.mdl`. Použijte bloky `EXEC`, `MODULE`, `TIODRV`, `TASK`.

Blok `MODULE` pojmenujte *EplDrv* a připojte jej na výstup exekutivy `Modules`. Blok `EXEC` pojmenujte *EPL* a připojte jej na výstup `Drivers`. Na výstup `Task` pak připojujte ty úlohy, jejichž vykonávání má být synchronizováno s každým komunikačním cyklem sítě EPL.

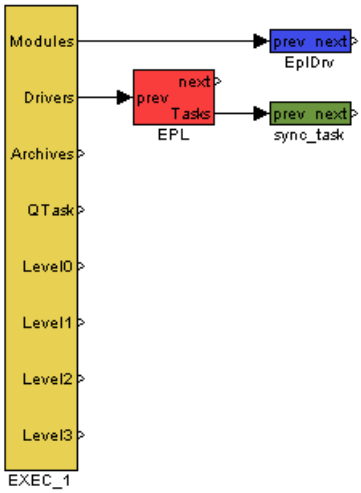
Parametry bloku `TIODRV` nastavte následovně:

`Module` – *EplDrv*

`classname` – *EplDrv*

`cfgname` – *nodeF0.rio*

stack – doporučuje se *64000*



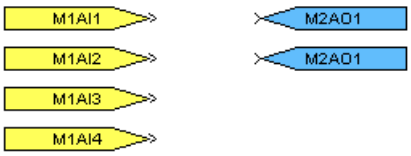
Obrázek 5.7: Schéma exekutivy

10. krok
Vytvořte požadovanou úlohu, soubor pojmenujte `sync_task.mdl`. K datům objektů PDO z lokálního slovníku objektů přistupujte pomocí bloků `FROM` a `GOTO`. Požadovaný objekt je určen podle pojmenování bloku. Pojmenování má tvar

```
EPL__NazevObjektu
```

11. krok
Nastavte schéma exekutivy jako hlavní schéma projektu příkazem *File–Set as main*. Zkompilujte projekt příkazem *Compiler–compile*. Zkontrolujte, zda překlad proběhl bez chyb.

12. krok
Nahrajte soubor s exekutivou `epl_exec.rex` do výkonného jádra systému REX na cílové platformě pomocí programu RexCore příkazem *Target–PC to target device*.



Obrázek 5.8: Schéma úlohy

13. krok
Počkejte, než dojde ke konfiguraci uzlu. To může trvat až několik desítek vteřin. Po úspěšné konfiguraci jsou uzly přepnuty do provozního režimu *operational*. V tomto režimu svítí všechny zelené diody na vstupních/výstupních modulech a data objektů PDO jsou komunikována v každém komunikačním cyklu sítě.

Pokud nedošlo k přepnutí do režimu *operational*, proveďte diagnostiku podle následující kapitoly.

Kapitola 6

Co dělat při problémech

Problém: u vstupních/výstupních modulů nesvítí dioda *r* zeleně

- Zkontrolujte, jestli probíhá komunikace v síti. Kontrolky rozhraní Ethernet blikají zeleně.
- Zkontrolujte, zda souhlasí číslo uzlu kontroléru BC0083 s číslem v konfiguraci projektu.
- Zkontrolujte, zda dochází ke konfiguraci uzlu. Pokud během startu kontroléru BC0083 zabliká červená dioda, je konfigurace chybná a je nutné ji opravit.

Problém: vstupy jsou čteny, ale výstupy se nedaří nastavit

- Pravděpodobně nemáte namapovány hodnoty všech výstupů a kontrolér nemá všechna data nutná pro nastavení výstupů. Namapujte všechny výstupy v tabulce *Connections*.

Problém: překladač hlásí *Error -205: Invalid identifier*

- Zkontrolujte, zda název bloku `FROM` a `GOTO` souhlasí s názvem objektu.
- Zkontrolujte, zda všechny objekty, ke kterým se z algoritmu přistupuje blokem `GOTO` mají nastavený *Access type* na *Read only*.
- Zkontrolujte, zda všechny objekty, ke kterým se z algoritmu přistupuje blokem `FROM` mají nastavený *Access type* na *Read/write*.

Problém: cílová platforma je nestabilní

- Zkontrolujte, zda máte nastavenou vhodnou hodnotu velikosti zásobníku. Doporučená hodnota je alespoň 32kB.

Problém: síť EPL je nestabilní

- Pravděpodobně máte nastavenou příliš krátkou hodnotu délky cyklu, kterou při dané konfiguraci nelze dosáhnout. Zvyšte délku komunikačního cyklu sítě EPL.

Literatura

[1] REX Controls s.r.o.. *Začínáme s řídicím systémem REX*, 2009.

[2] Ethernet POWERLINK Standardisation Group. *DS 301 V1.1.0 - Communication Profile Specification*. Ethernet POWERLINK Standardisation Group, POWERLINK-Office of the EPSG, Schaperstrasse 18, D-10719 Berlin, Germany, 2008.

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

Přílohy

Licenční ujednání

```
(c) SYSTEC electronic GmbH, D-07973 Greiz, August-Bebel-Str. 29
    www.systec-electronic.com

Project:      openPOWERLINK

License:

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:

  1. Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.

  2. Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.

  3. Neither the name of SYSTEC electronic GmbH nor the names of its
     contributors may be used to endorse or promote products derived
     from this software without prior written permission. For written
     permission, please contact info@systec-electronic.com.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
  COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
  INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
  BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
  LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
  ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  POSSIBILITY OF SUCH DAMAGE.

  Severability Clause:
```

If a provision of this License is or becomes illegal, invalid or unenforceable in any jurisdiction, that shall not affect:

1. the validity or enforceability in that jurisdiction of any other provision of this License; or
2. the validity or enforceability in other jurisdictions of that or any other provision of this License.