České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Komunikace Profinet IO

Petr Matiášek

Katedra řídicí techniky

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 28.5.2006

podpis

i

Poděkování

Především děkuji Ing. Pavlu Burgetovi za vedení mé diplomové práce a za cenné rady, které k jejímu vzniku významně přispěly. Mé poděkování patří také mojí rodině, přítelkyni Zuzaně a firmě Esonic a.s., kteří mě podporovali nejen při tvorbě této práce, ale i během celého studia.

Abstrakt

Tato práce se zabývá studiem vlastností nového průmyslového standardu v komunikačních technologiích nazvaném Profinet IO. Prezentuje způsob realizace řídicí jednotky sítě (IO-Controlleru) s využitím klasického PC a standardního ethernetového adaptéru. Implementovaná aplikace IO-Controlleru je založena na knihovně firmy Siemens, která poskytuje efektivní řešení. Aplikace je rozšířena o diagnostické funkce, které umožňují sledovat a ověřovat provoz komunikace. Toto řešení není vázáno na platformu Windows, pro kterou bylo primárně vyvíjeno, ale u umožňuje snadnou přenositelnost. Pro demonstraci byla aplikace zprovozněna na systému Linux.

Abstract

The thesis concerns a new communication concept for industrial distributed applications that is called Profinet IO. It presents realization of a control network unit (IO-Controller) using a common PC and a standard Ethernet adapter. The IO-Controller application is based on the Siemens library that provides an effective solution. The application is extended by diagnostic functions which enable to monitor and validate the communication process. This solution is not necessarily to be run o the Windows platform that has been originally developed for but it is easily portable. To demonstrate the portability the application has been set going on the Linux platform.

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky *Školní rok:* 2004/2005

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Petr Matiášek

Obor: Technická kybernetika

Název tématu: Komunikace Profinet IO

Zásady pro vypracování:

1. Seznamte se s protokolem Profinet IO. Nastudujte možnosti konfigurace sítě s komerčními zařízeními.
2. Realizujte aplikaci Profinet IO controller pod operačním systémem Windows 2000 s využitím knihovny od firmy Siemens. Zvažte přenos aplikace pro real-time rozšíření RTX.
3. Důkladně analyzujte a zdokumentujte komunikaci Profinet IO.

Seznam odborné literatury:

[1] Popp, M.; Weber, K.: *The Rapid Way to Profinet*. Profibus International, Karlsruhe 2004
[2] Profibus International: *Profinet Specification*. Profibus International, Karlsruhe 2004

Vedoucí diplomové práce: Ing. Pavel Burget

Termín zadání diplomové práce: zimní semestr 2004/2005 (změna zadání: 25.10.2005)

Termín odevzdání diplomové práce: květen 2006

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

v zast.

prof. Ing. Vladimír Kučera, DrSc.
děkan

V Praze dne 02.11.2005

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Průmyslová síť Profinet IO byla vyvinuta sdružením PROFIBUS International(PI) v roce 2004. Je určena pro integraci decentralizovaných periférií. Na rozdíl od komponentové verze Profinet CBA si drží klasický přístup známý z konceptu Profibus DP. Založena je však na Industrial Ethernetu.

Za cíl si Profinet IO klade rozšířit řady průmyslových zařízení, vybavených komunikačními rozhraními. Díky rozšířenosti, a tedy i nízké ceně standardních Ethernetových rozhraní, má dobrou šanci na úspěch. Zajímavá je i varianta IRT(Isochronous Real-Time), určená pro synchronizaci přesných soustav pohonů, která zajišťuje zcela deterministickou komunikaci ve vyhrazeném pásmu.

Tato práce se zabývá zkoumáním vlastností a funkcí tohoto nového standardu. Jejím hlavním cílem je analýza a dokumentace komunikačního protokolu Profinet IO a dále realizace aplikace Profinet IO-Controlleru(řídicí účastník sítě) pod operačním systémem Windows 2000 s využitím knihovny IO Base Programming Interface od firmy Siemens AG. V průběhu řešení práce bylo přistoupeno k rozšíření použitelnosti aplikace na systém Linux. Z tohoto důvodu byla vybrána knihova Qt firmy Trolltech AS, která podporuje tvorbu přenositelných aplikací. Zadání práce vyzývá ke zvážení možnosti použití real-time rozšíření Windows RTX. Vzhledem k tomu, že software RTX nebyl v době řešení dostupný, bylo od původního záměru upuštěno.

Aplikace IO-Controller byla pojata jako testovací nástroj, který umožňuje ověřování teoretických poznatků a analýzu komunikace. Důraz byl kladen na možnosti podpory při vývoji nových zařízení. Nástroj navíc disponuje funkcemi odchytávání a rozboru ethernetových rámců. Je určen pro klasické PC, vybavené standardním ethernetovým adaptérem nebo speciální komunikační kartou jako např. CP 1616.

Práce předpokládá u čtenáře základní znalosti v oblasti počítačových sítí a programování v jazyce C++. V textu je použito množství zkratek a odborných termínů, které jsou vysvětlovány při prvním výskytu, jejich kompletní přehled lze nalézt v dodatku A Použité zkratky.

Celá práce je rozdělena do třech hlavních kapitol. První část (1) podrobně popisuje vlastnosti a funkce Profinetu IO. Vychází zejména ze zdrojů [1] a [2]. Druhá část (2) je věnována prostředkům použitým k realizaci testovacího nástroje (knihovna Siemens, CP 1616, libpcap, Qt). Proces tvorby a výsledný testovací nástroj jsou zdokumentovány ve třetí kapitole (3). V závěru práce jsou shrnuty hlavní poznatky a výsledky, kterých bylo dosaženo a je zde zhodnoceno naplnění stanovených cílů.

Kapitola 1

Profinet IO

Profinet IO[1] je otevřený komunikační standard od organizace PROFIBUS International (PI) určený pro integraci jednoduchých distribuovaných zařízení a časově náročných aplikací. Nepřímo navazuje na úspěšný Profibus DP,a proto je možné nalézt některé jejich společné rysy, což mimo jiné usnadňuje přechod na tento nový systém.

Profinet IO nabízí řešení pro velké i malé systémy, složené z jednoho nebo více řídicích členů IO-Controller a téměř libovolného počtu podřízených zařízení IO-Device. Data jednoho IO-Device může sdílet více IO-Controllerů. Podporována jsou redundantní spojení i rekonfigurace systému za chodu.

Profinet IO definuje několik úrovní přenosu dat lišících se v rychlosti. Ten nejvýkonnější je schopen provádět cyklickou výměnu více než 1000 vstupů a výstupů ve 32 zařízeních za méně než 1 ms. Pro distribuci cyklických dat je použit model Provider/Consumer, který umožňuje distribuci dat více uživatelům současně.

Tato kapitola popisuje vlastnosti a funkce standardu Profinet IO. Pro přehlednost a srozumitelnost zde nejsou uvedeny všechny možnosti tohoto standardu. Důraz je kladen zejména na vlastnosti, které jsou povinné pro všechna zařízení. Podrobné informace, které nebyly ze zmíněného důvodu zahrnuty do této práce lze nalézt ve specifikaci [1].

1.1 Model Profinetu IO

Na obrázku obr. 1.1 je zobrazen model Profinetu IO, který slouží ke znázornění rolí účastníků komunikace a vazeb mezi nimi. Profinet IO se snaží zachovat úspěšné a časem prověřené vlastnosti svého předchůdce, Profibusu DP, a to například právě v pohledu na účastníky sítě, které dělí do těchto tříd:

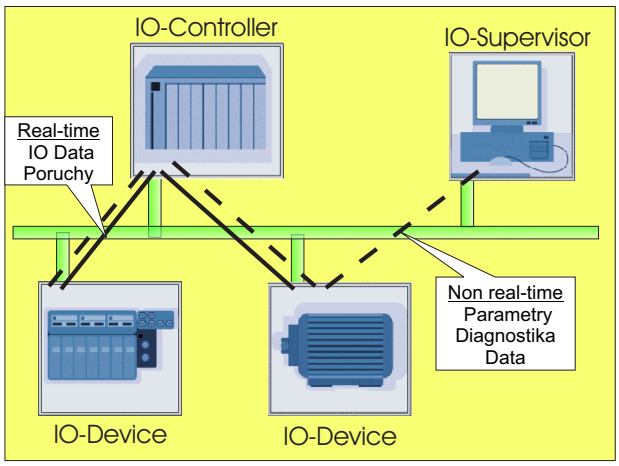
- **IO-Controller :**
  Typicky se jedná o PLC[2], ve kterém je prováděn program pro řízení automatizovaného zařízení. Tento účastník svými funkcemi odpovídá masterovi třídy DPV 1 v Profibusu DP.

- **IO-Supervisor :**
  Toto může být programovací stanice, PC nebo jiné zařízení sloužící k diagnostice sítě a k

[1] IO je zkratka pro Input/Output, tedy česky vstup/výstup.
[2] *Programmable Logic Controller*- programovatelný logický automat

jejímu zprovoznění. V Profibusu DP by se jednalo o master třídy DPV 2.

- **IO-Device :**
  Tato zařízení představují převážně decentralizované periférie. Může se jednat i o jiná zařízení se vstupně-výstupními(I/O) daty, která jsou schopná komunikovat v síti Profinet IO. Tento typ svým postavením a funkcemi odpovídá zařízením typu slave známým z Profibusu DP.



Obrázek 1.1: Model Profinetu IO

Každý IO systém, tak se nazývá jedna podsíť Profinetu IO, obsahuje minimálně jeden IO-Controller a několik zařízení IO-Device. Jedno zařízení IO-Device si může vyměňovat data s více IO-Controllery. IO-Supervisor bývá zpravidla k síti připojen pouze dočasně, a to při zprovozňování sítě nebo při hledání závad systému.

Profinet IO zahrnuje skupinu protokolů, které svými službami poskytují následující funkce:

- Cyklický přenos I/O dat definovaných adresním prostorem IO-Controlleru.
- Acyklický přenos poruchových hlášení, jejichž příjem je potvrzován.
- Acyklický přenos dat jako jsou parametry, diagnostické údaje, atd.

1.2 Datové objekty Profinetu IO

V této kapitole jsou popsány datové objekty, které se přenášejí mezi jednotlivými zařízeními. Napomůže to k vytvoření lepší představy o možnostech a funkcích systému. Objekty jsou zde chápány jako datové struktury sloužící k uchování a přenosu dat v rámci komunikačního systému. Popisované objekty korespondují s modelem uvedeným v předchozí kapitole.

1.2.1 I/O datové objekty

I/O data, která jsou nejčastěji reprezentována vstupními a výstupními daty řízených procesů, se cyklicky přenášejí mezi jednotlivými účastníky. Tato komunikace je založena na modelu Provider/Consumer (poskytovatel/konzument). To znamená, že účastník, který data poskytuje,

je odesílá bez vyžádání účastníkům, se kterými předtím navázal logické spojení. Konzument data přijímá, ale již je nepotvrzuje.

I/O data každého IO-Device jsou členěna do tzv. slotů a subslotů, které obsahují jednotlivé datové položky (viz kapitola 1.4). V datovém rámci může být přenášen libovolný výběr těchto položek, přičemž každý rámec může být navíc doplněn o informaci o stavu poskytovatele(IOPS) a konzumenta(IOCS) I/O dat. Tento způsob označování dat umožňuje rychle a efektivně bez dalších diagnostických informací rozpoznat neplatná data a reagovat na ně. V případě, že konzument přijme data označená poskytovatelem jako neplatná, měl by použít implicitní, předchozí nebo substituční hodnoty. Informace o stavu konzumenta, obsažená v rámci, říká poskytovateli, jestli byla jím naposledy odeslaná data srozumitelná.

Pro každé logické spojení mezi IO-Device a IO-Controlllerem je na začátku pevně nastavena perioda výměny I/O dat. Poskytovatel k odesílaným datům přidá stav cyklického čítače, který příjemci umožňuje kontrolovat stáří a posloupnost rámců. Ze dvou po sobě jdoucích rámců lze snadno určit periodu komunikace. Konzument trvale sleduje skutečnou periodu příjmu I/O dat, pokud je překročen nastavený monitorovací čas, obvykle se jedná o trojnásobek periody komunikace, oznámí tuto událost nadřazené aplikaci a dále čeká na data.

1.2.2 Objekty datových záznamů

Čtením a zápisem datových záznamů lze do IO-Device přenášet konfigurační data, nastavovat jejich parametry a číst jejich stavové informace. Tato výměna dat probíhá acyklicky, tzn. stylem žádostí a odpovědí.

Dále je definováno pět typů datových záznamů, které se vyskytují v každém IO-Device. Kromě nich mohou jednotliví výrobci specifikovat ještě další typy datových záznamů.

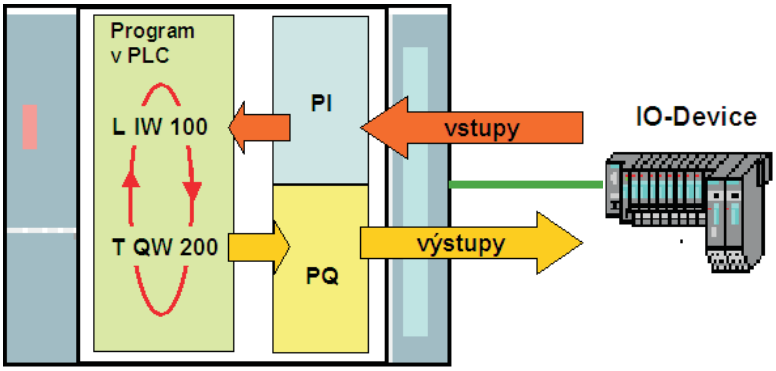
- **Diagnostické informace** - obsahují dat informující o stavu IO-Device a jeho jednotlivých částí a mohou být kdykoli přečteny uživatelem z každého zařízení.
- **Záznamy událostí** (Logbook entries) - záznamy o poruchách a chybách s časovými značkami. (je požadována kapacita minimálně pro 16 záznamů)
- **Identifikační informace** - informace pro identifikaci a údržbu zařízení blíže popsané ve specifikaci PI "I&M Functions"[**?**]
- **Informace o struktuře** - obsahuje informace o skutečné a požadované struktuře IO-Device
- **I/O datové objekty** - obsahují I/O dat příslušného IO-Device, která mohou být touto cestou přenášena acyklicky bez nutnosti realizace cyklické výměny dat.

1.2.3 Datové objekty poruch

Datové objekty poruch slouží k přenosu informací o vzniklých událostech v zařízeních. Jedná se jak o systémové události (např. vyjmutí a zasunutí modulu), tak o události definované výrobcem, které nastaly v zařízení nebo v provozu(např. přetížení kanálu nebo vysoká teplota). Každé zařízení, které přijímá poruchy, zejména tedy IO-Controller, musí mít vytvořenu frontu pro každý typ poruchy. Přijmutí každé poruchy je potvrzeno.

1.3 IO-Controller

Jak již bylo uvedeno, IO-Controller je řídicím účastníkem sítě. V průběhu konfigurace systému jsou do něho uloženy parametry a informace o zařízeních IO-Device, která jsou mu podřízena. IO-Controller se nejčastěji vyskytuje v podobě PLC nebo jiného počítače pro řízení. Obecná struktura tohoto zařízení je znázorněna na obrázku obr. 1.2. Program v PLC, který je prováděn ve smyčce, čte vstupní data a zapisuje výstupní data do tzv. obrazu procesních dat(PI,PQ). Jedná se o buffer, který obsahuje aktuální hodnoty dat vstupů a výstupů. IO-Controller do tohoto bufferu zapisuje vstupní data přijatá od svých IO-Device, kterým zároveň odesílá výstupní data přečtená z bufferu. Tento způsob předávání dat uživatelské aplikaci PLC nebo jiného počítače pro řízení (dále jen uživatelská aplikace IO-Controlleru) umožňuje nezávislý běh obou systémů.



Obrázek 1.2: Struktura IO-Controlleru

IO-Controller plní v Profinetu IO následující funkce, které jsou dále popsány v příslušných kapitolách:

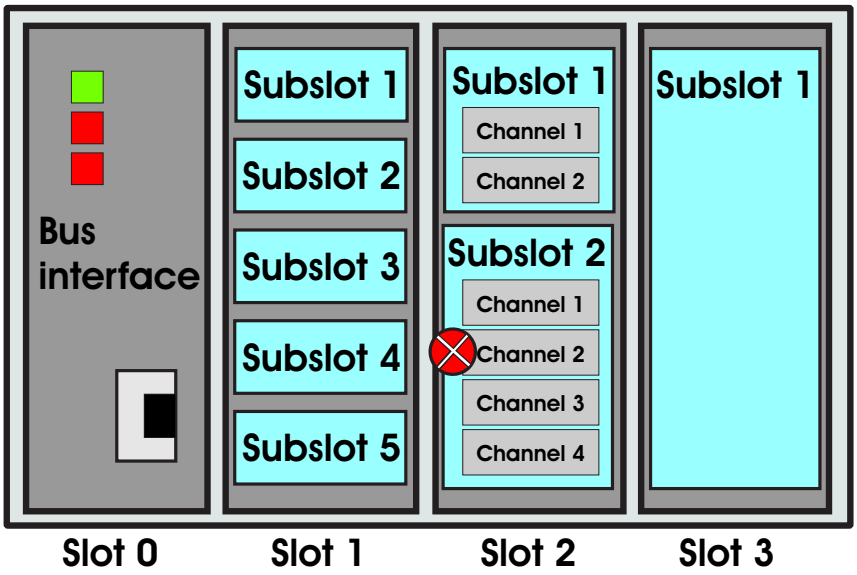
- **Identifikace IO-Device** - před vytvořením spojení s IO-Device provádí IO-Controller identifikaci tohoto zařízení prostřednictvím jeho jména, které mu bylo přiděleno při konfiguraci systému.
- **Přidělování IP adres** - po úspěšné identifikaci IO-Device je mu následně přidělena IP adresa
- **Navazování spojení s IO-Device** - před zahájením výměny dat musí IO-Controller vytvořit logické spojení s IO-Device
- **Parametrizace submodulů IO-Device** - před spuštěním cyklické komunikace provede IO-Controller nastavení parametrů jednotlivých datových položek IO-Device
- **Cyklická výměna I/O dat** - odesílání a příjem domluvených I/O dat
- **Příjem a zpracování poruchových hlášení** - přijaté poruchy předává uživatelské aplikaci IO-Controlleru a odesílá potvrzení jejich příjmu do IO-Device

- **Čtení a zápis datových záznamů** - umožňuje uživatelské aplikaci IO-Controlleru číst a zapisovat datové záznamy IO-device

1.4 IO-Device

Data IO-Device jsou z pohledu Profinetu IO logicky členěna ve dvou úrovních, do slotů a subslotů (viz obr. 1.3). Zjednodušeně si lze slot představit jako jakousi "přihrátku", která dále obsahuje jednotlivé subsloty. Každý subslot reprezentuje jednu I/O datovou položku určitého typu a dále pak skupinu různých datových záznamů, které jsou označovány indexy. V některých případech, zejména u modulárních systému, mohou subsloty obsahovat několik tzv. kanálů. Tyto jsou využívány v případě poruch zařízení k lokalizaci zdroje poruchy. Adresace I/O dat se provádí pomocí slotů a subslotů. Pro adresaci datového záznamu je ještě navíc použit příslušný index. Každé zařízení musí mít minimálně jeden slot, který obsahuje nejméně jeden subslot. Volba struktury zařízení závisí pouze na přístupu výrobce. Pro modulární systémy vstupů a výstupů se nabízí přidělit každému modulu jeden slot a dále jeho submoduly označit subsloty. Pro nemodulární zařízení může toto členění reprezentovat logickou strukturu dat.

Všechny možnosti a varianty konfigurace konkrétního IO-Device jsou popsány v tzv. GSD souboru, který dodává výrobce zařízení. Tyto soubory jsou blíže popsány v kapitole kapitola 1.15.



Obrázek 1.3: Struktura IO-Device

IO-Device plní v Profinetu IO následující funkce:

- **Identifikace** - v případě, že IO-Device odpovídá identifikačním parametrům uvedeným v dotazu IO-Controlleru, odešle mu své identifikační údaje a nastavení ethrenetového rozhraní.

- **Navazování spojení s IO-Controller** - po přijetí požadavku na vytvoření spojení od IO-Controlleru ověří IO-Device, zda může požadavku vyhovět a následně odešle odpověď.

- **Zápis přijatých parametrů submodulů** - v případě, že je vytvořeno spojení s IO-Controllerem, přijímá a ukládá IO-Device parametry pro jednotlivé submoduly v podobě datových záznamů.
- **Cyklická výměna I/O dat** - odesílání a příjem domluvených I/O dat
- **Hlášení poruchových událostí** - v případě, že dojde v IO-Device k události, které je reprezentována v Profinetu IO poruchovým hlášení, dojde k jeho odeslání IO-Controlleru
- **Čtení a zápis datových záznamů** - po vytvoření spojení s IO-Controllerem se IO-Device chová jako server pro četní a zápis datových záznamů a plní požadavky IO-Controlleru.

1.5 Princip komunikace mezi účastníky

Při popisu komunikace mezi účastníky jsou vždy chápána vstupní a výstupní data z pohledu IO-Controlleru. IO-Device poskytuje vstupní data z řízeného procesu do IO-Controlleru. Výstupní data, která od IO-Controlleru obdrží, používá dále k řízení procesu.

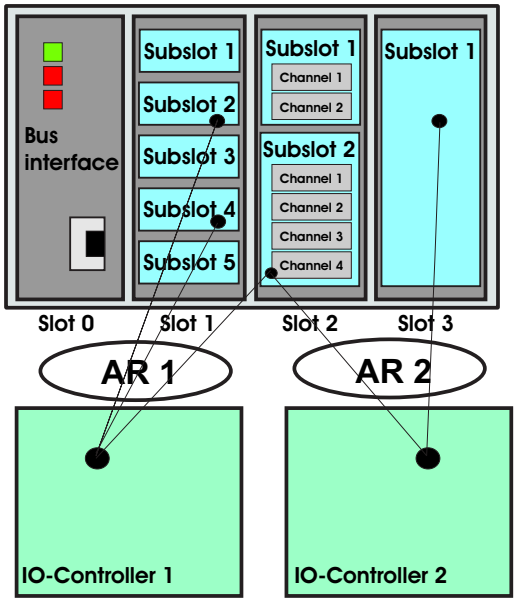
Komunikační spojení mezi účastníky je založeno na tzv. aplikačním vztahu (AR) a tzv. komunikačních vztazích (CR). Aplikační vztah je komunikační kanál vytvořený mezi účastníky ihned na začátku spojení. V rámci tohoto vztahu vznikají další komunikační vztahy, které jsou zaměřené na výměnu určitého typu dat. Obrázek obr. 1.4 zobrazuje aplikační vztahy mezi IO-Device a více IO-Controllery. Právě možnost sdílení jednoho IO-Device mezi více IO-Controllery je jedna z významných pozitivních vlastností komunikačního protokolu Profinetu IO. Jeden aplikační vztah je vždy vytvářen pro konkrétní skupinu subslotů IO-Device. Pokud se jedná o subsloty vstupních dat, mohou být sdíleny ve více AR. Subsloty výstupních dat jsou vždy rezervovány prvním IO-Controllerem, který požádá o jejich zahrnutí do AR. Na základě popsaného principu je možné do komunikace zahrnout pouze data, která IO-Controller opravdu potřebuje.

1.6 Aplikační vztah (AR)

Aplikační vztah (AR) je vazba mezi dvěma nebo více účastníky, která se realizuje za účelem spolupráce a výměny dat. K jeho vytvoření dojde tím, že IO-Controller nebo IO-Supervisor vyšle směrem k IO-Device požadavek o připojení. AR jsou navazovány s těmi IO-Device, která jsou uvedena v konfiguračních datech IO-Controlleru nebo IO-Supervisor. Bez navázání aplikačního vztahu pro výměnu I/O dat je možné provádět pouze speciální konfigurační funkce a číst data pomocí tzv. implicitního AR. Implicitní AR je vztah, který má standardně definované parametry, proto ho není nutné předem vytvářet. Je ho možné tedy v každém zařízení ihned použít. Umožňuje však pouze čtení vybraných datových záznamů. Žádná jiná datová výměna bez vytvoření regulérního AR v Profinetu IO není možná.

Využití AR umožní IO-Controlleru nebo IO-Supervisoru přistupovat k IO-Device za účelem provádění těchto činností:

- čtení vstupních dat



Obrázek 1.4: Aplikační vztahy mezi účastníky

- zápis výstupních dat
- přijímání poruch
- čtení datových záznamů
- zápis datových záznamů

IO-Device je při navazování AR pasivním účastníkem. Čeká tedy, až s ním některý IO-Controller nebo IO-Supervisor naváží spojení. Obecně může jedno IO-Device podporovat libovolný počet AR. Každý vytvořený AR však zabírá určité systémové prostředky IO-Device, které mohou být značně omezené. Povinně musí IO-Device umožňovat vytvoření dvou AR. Jedno pro IO-Controller a jedno pro IO-Supervisor. Tento počet bude v budoucnu ještě rozšířen o jedno povinné AR pro připojení redundantního IO-Controlleru pro případ výpadku hlavního. Skutečnost, že počet povinně podporovaný AR je takto nízký, představuje omezení pro sítě s více IO-Controllery, ve kterých by bylo možné, aby sdílely jedno IO-Device. Záleží tedy pouze na výrobcích, kolik AR budou jejich zařízení podporovat.

1.6.1 Vytvoření aplikačního vztahu

K vytvoření aplikačního vztahu mezi IO-Controllerem/IO-Supervisorem a IO-Device dochází na začátku spuštění systému, ihned poté co IO-Device obdrží IP adresu. IO-Controller/IO-Supervisor používají k vytvoření AR během spuštění připojovací rámec "Connect-Call", který obsahuje následující data pro IO-Device:

- obecné komunikační parametry AR
- informace a data, potřebná k vytvoření komunikačních vztahů (CR) pro přenos I/O dat
- předpokládanou strukturu zařízení

- parametry pro vytvoření CR pro poruchy

IO-Device si přijatá data zkontroluje a v případě, že je schopen vyhovět požadavkům, vytvoří příslušná CR. Pokud dojde v IO-Device během vyhodnocování přijatého rámce k některé z následujících událostí, zařízení odešle negativní potvrzení, v němž je uveden důvod odmítnutí. Jedná se o tyto možné události:

- IO-Device nemá prostředky pro vytvoření požadovaného AR nebo některého z CR
- typ požadovaného AR není možné vytvořit (např. redundantní, pokud neexistuje hlavní)
- nejsou podporovány požadované časové parametry CR pro výměnu I/O dat

Výměna I/O dat začíná tím, že zařízení IO-Device odešle pozitivní potvrzení požadavku o vytvoření připojení. Přenášená I/O data jsou však ještě neplatná, protože IO-Controller zatím neprovedl zápis parametrů do jednotlivých subslotů IO-Device zahrnutých do AR.

Po rámci "Connect-Call" následuje přenos inicializačních dat(rámec "Write-Call") do IO-Device pomocí CR pro přenos datových záznamů. IO-Controller tento rámec odešle pro každý subslot, který si předtím rezervoval rámcem "Connect-Call". Konec zápisu parametrů do subslotů signalizuje IO-Controller odesláním rámce "EndOfParametrization(DControl)". Pokud IO-Device na něj odpoví pozitivním potvrzením pomocí rámce "ApplicationReady", je AR úspěšně vytvořen.

Schématicky je aplikační vztah spolu se svými komunikačními vztahy zobrazen na obrázku obr. 1.5.

Během vytváření AR mohou nastat ještě další poruchové stavy:

- **IO-Device neexistuje, nebo neodpovídá konfiguračním údajům**
  V tomto případě IO-Controller nemůže vytvořit žádné CR s požadovaným IO-Device. V definovaném intervalu opakuje pokusy o připojení.
- **Slot nebo subslot, zahrnutý v AR, v IO-Device neexistuje**
  V tomto případě IO-Controller nemůže vytvořit žádné CR s požadovaným IO-Device. V definovaném intervalu opakuje pokusy o připojení.
- **(Sub)modul chybí nebo jeho typ neodpovídá**
  V tomto případě IO-Controller může s IO-Device navázat CR. Informace o chybných nebo chybějících modulech jsou odeslána v podobě poruch. Jejich I/O data jsou neplatná.

1.6.2 Zrušení aplikačního vztahu

Při zrušení AR dojde i ke zrušení všech CR s ním souvisejících. Způsob ošetření nastavení hodnot výstupních dat IO-Device do bezpečného stavu po zrušení AR určuje výrobce.

1.7 Komunikační vztah (CR)

Jeden AR zahrnuje jeden nebo více CR, které vykonávají výměnu dat určitého typu. Všech CR vznikají v průběhu vytváření AR. Hlavním důvodem používání CR je logické členění

datových toků mezi účastníky sítě. Data jednotlivých typů CR jsou rozpoznatelná z dat ethernetového rámce pomocí položek *EtherType* případně *FrameID*(viz kapitola 1.9).

Každý koncový bod CR, tedy každý účastník navázaného CR, je tvořen bufferem nebo frontou. Data přenášená přes frontové CR jsou zpracována v pořadí, v kterém byla doručena. CR bufferového ty fungují tak, že se data vkládají do odesílacího bufferu, kde nahradí dříve uložené hodnoty. Buffer je trvale dostupný a obsahuje pouze aktuální data. Příjemce má také buffer, kde udržuje poslední přijatá data a poskytuje je dalším službám. Lze realizovat i kombinace těchto dvou způsobů výměny dat. Jeden účastník CR tedy může data odesílat z fronty a druhý je přijímat do bufferu nebo naopak.

Dále rozlišujeme CR podle těchto kritérii:

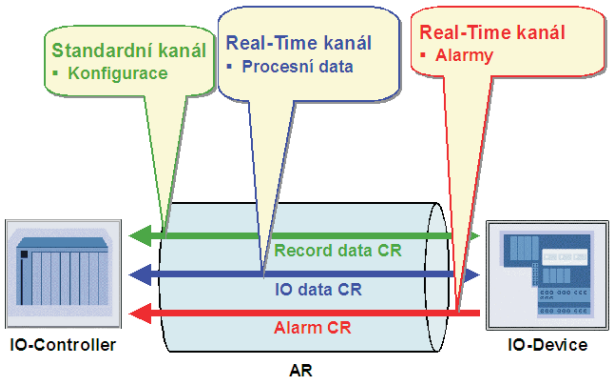
- **Typ spojení**
  - Orientované na spojení - je vytvořeno logické spojení službami účastníků. Dělí se na tři fáze (navázání,přenos dat, uvolnění).
  - Bez spojení - je stále ve stavu pro přenos dat(žádné navázání a uvolnění)
- **Typ služby**
  - Potvrzovaná - odesílatel vyžaduje potvrzení příjmu každého rámce
  - Nepotvrzovaná - odesílatel nevyžaduje potvrzení příjmu rámce
- **Směr přenosu**
  - Obousměrný - data jsou přenášena oběma směry
  - Jednosměrný - data jsou přenášena pouze jedním směrem
- **Vztah účastníků**
  - Client/Server - účastník typu server plní požadavky účastníka typu client
  - Producer/Consumer - účastník typu poskytovatel odesílá bez vyžádání data konzumentům tohoto CR

Standardní AR pro přenos I/O dat obsahuje následující typy CR:

- minimálně jeden nepotvrzovaný jednosměrný CR typu buffer pro přenos vstupních dat spolu se stavem poskytovatele (IOPS) z IO-Device do IO-Controlleru a/nebo pro přenos stavu konzumenta (IOCS) pro výstupní data z IO-Device do IO-Controlleru.
- minimálně jeden nepotvrzovaný jednosměrný CR typu buffer pro přenos výstupních dat spolu s IOPS z IO-Controlleru do IO-Device a/nebo pro přenos IOCS pro vstupní data z IO-Controlleru do IO-Device.
- jeden potvrzovaný obousměrný CR bez spojení typu fronta pro přenos poruch z IO-Device do IO-Controlleru a potvrzení v opačném směru. Možný je i přenos poruch opačným směrem.

- jeden potvrzovaný obousměrný CR se spojením typu fronta pro přístup k datovým záznamům (diagnostika, identifikace, adt.)

O vytvoření CR se stará tzv. kontextový management (CM), který je jak v IO-Controlleru, tak v IO-Device (viz kapitola 1.10.1).



Obrázek 1.5: Komunikační vztahy mezi zařízeními

1.7.1 Komunikační vztah pro objekty datových záznamů

Data v rámci CR datových záznamů se přenášejí acyklicky. K přenosu dat je použit protokol RPC (Remote Procedure Call), který je založen na vztahu Client/Server. Komunikaci inicializuje klient, nejčastěji IO-Controller, který odesílá serveru, jenž je tvořen nejčastěji zařízením IO-Device, požadavek o data. Server na tuto žádost odpovídá odesláním požadovaných dat nebo chybového hlášení.

CR datových záznamů slouží ke čtení a zápisu těchto dat:

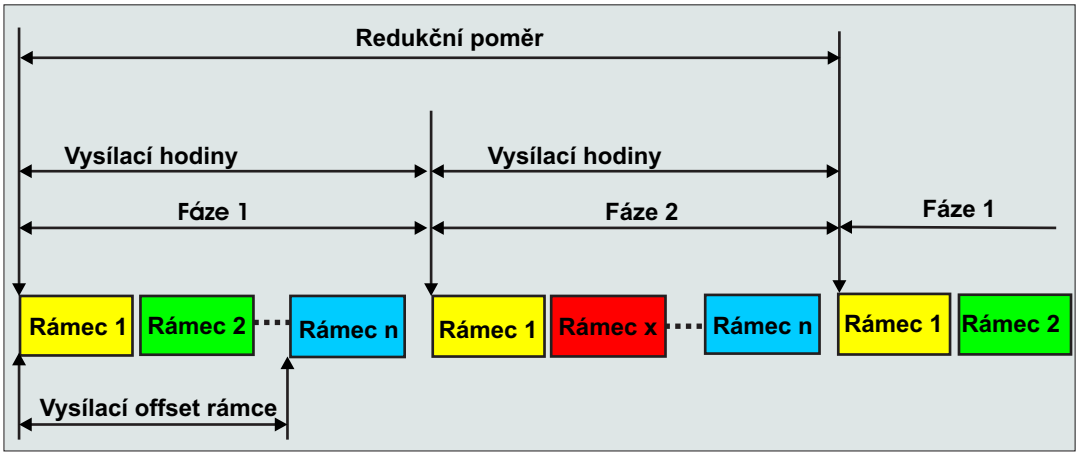
- Diagnostických datových objektů
- I/O datových objektů
- I/O datových objektů substitučních hodnot
- Identifikačních datových objektů (I&M funkce)
- Parametrů AR
- Záznamů o událostech (Logů)
- Dat fyzického zařízení (parametry komunikačního rozhraní, IP adresy, atd.)
- Konfiguračních dat
- Rozdíly mezi očekávanými a skutečnými moduly IO-Device
- Ostatních datových záznamů definovaných výrobcem

1.7.2 Komunikační vztah pro I/O datové objekty

Data I/O datových objektů jsou přenášena cyklicky. Parametry tohoto datového přenosu jsou definovány při navázání spojení IO-Controlleru s IO-Device pomocí rámce "Connect-Call". Jedná se zejména o pořadí dat a periodu jejich odesílání. Konfigurační data dále určují, kolik bude vytvořeno CR pro přenos I/O dat. Data jsou odesílána bez informací o jejich formátu, jsou však doplněna o informace o stavu každého submodulu.

I/O CR má tyto vlastnosti:

- Přenos mezi buffery: obě strany mají data trvale dostupná a aktualizovaná.
- Provider/Consumer: konzument monitoruje skutečný cyklus komunikace pomocí čítače v každém rámci.
- Nepotvrzovaný CR: příjemce neposílá potvrzení příjetí každého rámce. V případě, že konzumentovi nejsou data doručena v požadovaném intervalu, nebo jsou chybná, změní se hodnota IOCS cyklicky odesílaný poskytovateli.
- Cyklický CR: I/O data jsou odesílána v pevném cyklu. Je možné definovat různé doby cyklu pro různá zařízení. Stejně tak se mohou vzájemně lišit intervaly odesílání vstupních a výstupních dat. Pomocí parametrů komunikace lze ovlivnit periodu a pořadí odesílání datového rámce jednoho I/O CR (viz obr. 1.6).



Obrázek 1.6: Rozvrhování rámců jednotlivých I/O CR

Tento variabilní způsob odesílání dat umožňuje přizpůsobit výměnu dat vlastnostem a požadavkům skutečného procesu, kde je systém nasazen. Není totiž vždy požadavkem přenášet data co nejrychleji. Typickým příkladem jsou hodnoty teplot, které nepodléhají tak rychlým změnám jako například hodnoty tlaků, proto ani nemusí být tak často přenášeny. Pro využití této vlastnosti je nutné I/O data vhodně rozdělit do různých CR.

Základním parametrem k ovlivnění četností přenosu dat je tzv. redukční poměr, který je během konfigurace definován pro každý I/O CR. Tento parametr určuje, v kterém vysílacím cyklu budou odeslány I/O data zahrnutá v I/O CR. V rámci jedné fáze vysílání je definováno přesné pořadí rámců jednotlivých I/O CR pomocí vysílacího offsetu rámce.

Vysvětlení pojmů použitých v obrázku:

- **Vysílací hodiny:** určují cyklus vysílání. Hodnota udává čas v násobcích 31.25 μs, tedy pro cyklus 1 ms je hodnota tohoto parametru 32.
- **Redukční poměr:**určuje, jak často jsou odesílána data I/O CR. Skutečný cyklus odesílání dat je určen násobením vysílacích hodin a redukčního poměru.
- **Vysílací offset rámce:**určuje relativní časový posun vysílání rámce od začátku vysílacího cyklu, ve kterém je odesílán.
- **Fáze:** určuje pořadové číslo vysílacího cyklu, ve kterém je rámec odesílán.

I/O CR směřuje od poskytovatele dat k jejich konzumentovi. Pokud si IO-Controller vyměňuje s IO-Device data vstupů i výstupů, jsou vytvořeny dva komunikační vztahy. Jeden pro vstupy, kde je IO-Controller konzumentem a IO-Device poskytovatelem a druhý je určen pro přenos výstupů, v kterém mají opačné role. Poskytovatel vždy používá pro přenos jedné definované skupiny I/O dat právě jedno CR. Zároveň je v jednom směru možné vytvářet mezi účastníky více CR pro přenos I/O dat. Tato data jsou po přijetí vždy přiřazována submodulům, proto musí být pro výměnu I/O dat definován minimálně jeden submodul.

Nastavení parametrů I/O CR probíhá při spuštění systému. IO-Controller odesílá pro tento účel v rámci "Connect-Call" tato data:

- Seznam I/O datových objektů, zahrnutých v CR.
- Pořadí I/O datových objektů v rámci.
- Pozice a velikost IOPS a IOCS v rámci pro každý I/O datový objekt.
- Vysílací interval (vysílací hodiny, redukční poměr, fáze a vysílací offset rámce)
- Interval monitorování doručení dat

1.7.3 Komunikační vztah pro datové objekty poruch

Poruchová hlášení jsou přenášena acyklicky a jejich doručení musí být vždy potvrzeno. Odesílatel(zdroj poruchy) obdrží buď potvrzení, že data byla doručena v pořádku, nebo chybové hlášení. Příjemce poruch může požádat o zasílání více poruch najednou, ty budou následně potvrzovány jako celek. Data poruch jsou vždy odesílána v jednom rámci, segmentace dat do více rámců není pro acyklický kanál poruch podporována.

Při navazování CR pro poruchy se určí, na kterém konci je zdroj poruch a na kterém příjemce. Principiálně mohou jako zdroj poruch fungovat jak IO-Controller tak IO-Device. Běžně je však zdrojem pouze IO-Device. Tabulka tab. 1.1 popisuje typy poruch definovaných v Profinetu IO. Profinet IO definuje dvě úrovně priorit poruch, *High* a *Low*. Poruchy priority *High* by měly být zpracovávány přednostně jak v IO-Controlleru tak v IO-Device.

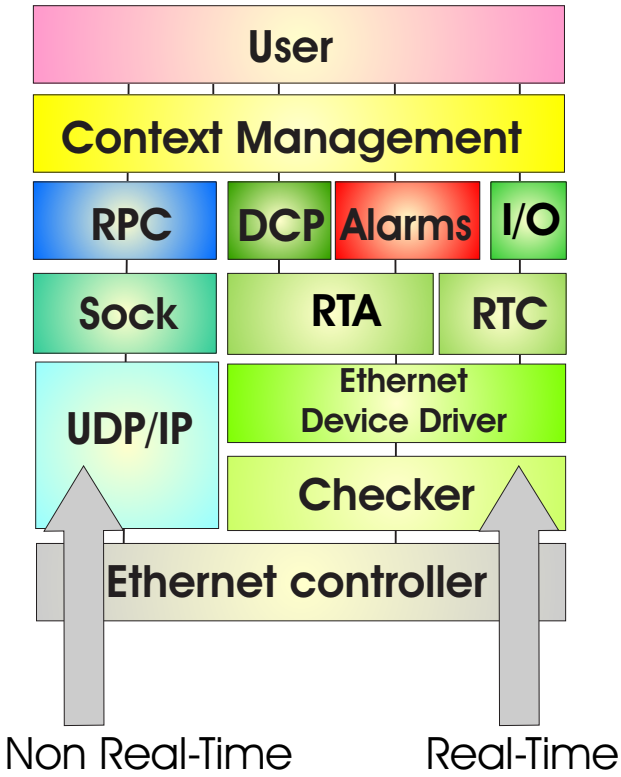
Typ poruchy	Význam a použití
Diagnosis Appears	Signalizuje událost v submodulu, jako jsou např. zkrat nebo překročení limitní hodnoty. Detailní informace lze získat přečtením diagnostických dat submodulu.
Process	Označuje poruchu z řízeného procesu.
Pull	Informuje o tom, že byl vytažen používaný (sub)modul.
Plug	Informuje o tom, že byl zastrčen zpět používaný (sub)modul.
Status	Označuje změnu stavu uvnitř (sub)modulu.
Update	Označuje změnu parametrů pro (sub)modul.
Redundancy	Upozorňuje záložní IO-Controller na výpadek hlavního.
Controlled	Upozorňuje na zamčení používaného (sub)modulu IO-Supervisorem.
Released	Upozorňuje na uvolnění používaného (sub)modulu IO-Supervisorem.
Plug Wrong Submodule	Informuje o tom, že byl zastrčen jiný, než nakonfigurovaný (sub)modul.
Diagnosis Disappears	Signalizuje konec diagnostické události v (sub)modulu.
Return Of Submodule	Slot signalizuje, že submodul je připraven začít znovu fungovat bez potřeby parametrizace.
Multicast Provider Communication Stopped	Konzument dat multicastového CR oznamuje, že vypršel monitorovací čas cyklu komunikace.
Multicast Provider Communication Running	Konzument dat multicastového CR oznamuje, že byla obnovena komunikace.
Port Data Change Notification	Oznamuje změnu dat komunikačního portu(např. připojení nebo odpojení).
Sync Data Changed Notification	Signalizuje změnu hodinové synchronizace.
Isochronous Mode Problem Notification	Signalizuje problém isochronní aplikace (např. byla spuštěna pozdě).
Manufacturere Specific	Význam těchto poruch určuje výrobce.

Tabulka 1.1: Typy poruch a hlášení v Profinetu IO

1.8 Komunikační cesty v rámci Profinetu IO

Všem datovým výměnám v Profinetu IO jsou přiřazeny určité komunikační protokoly, které definují způsob zpracování a cestu datových rámců komunikačním zásobníkem zařízení. Ten je tvořen jednotlivými protokoly uspořádanými v několika navazujících vrstvách. Takto strukturovaný zásobník obsahuje každé zařízení Profinetu IO. Jak je vidět z obrázku obr. 1.7, jeden konec zásobníku tvoří síťové rozhraní a druhý uživatelská aplikace zařízení. Mezi těmito vrstvami jsou uspořádány jednotlivé komunikační protokoly. Komunikační zásobník je vertikálně rozdělen na dvě základní části, část pro real-time komunikaci a část pro časově méně kritické přenosy(non-real-time). Real-time komunikace se dále dělí na cyklickou a acyklickou. Non-real-time komunikace zahrnuje protokoly IP/UDP, RPC a rozhraní sock pro správu socketů. Real-time komunikace je založena na čisté ethernetové komunikaci, nad kterou je definována rodina RT protokolů Profinetu IO (RTC,RTA,DCP). Kontextový management řídí celou komunikaci a

zajišťuje rozdělení dat.



Obrázek 1.7: Komunikační cesty v IO-Device

1.8.1 Non-real-time komunikace

Non-real-time komunikace slouží k přenosu dat datových záznamů. To zahrnuje jednak čtení a zápis těchto záznamů a dále pak požadavky na vytvoření spojení, které provádí kontextový management.

1.8.2 Cyklická real-time komunikace

Cyklická real-time (RTC) komunikace slouží k přenosu I/O datových objektů a monitorování stavu CR pomocí výměny informací o stavu účastníků. Cyklická data protokolu RTC jsou zpracovávána resp. vysílána stavovým automatem konzumenta resp. poskytovatele dat.

1.8.3 Acyklická real-time komunikace

Acyklická real-time (RTA) komunikace slouží k přenosu těchto dat:

- Poruch a diagnostických událostí
- Základních řídicích funkcí, které musí fungovat bez funkčního IP protokolu. Jedná se zejména o přiřazení jmen zařízení, identifikaci a nastavení parametrů IP protokolu.(protokol DCP)

- Časová synchronizace
- Protokoly pro řízení redundance komunikačních médii.

1.8.4 Další protokoly v Profinetu IO

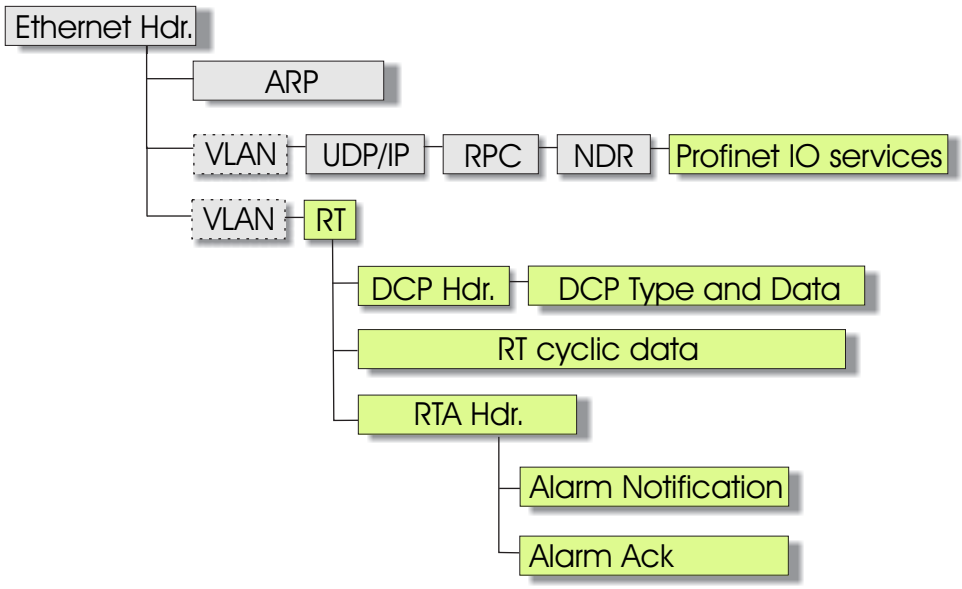
Kromě uvedených protokolů využívá Profinet IO ještě tyto standardní protokoly:

- **DHCP** (Dynamic Host Configuration Protocol) pro přiřazování IP adres a parametrů, pokud je podporován.
- **DNS** (Domain Name Service) pro správu logických jmen.
- **SNMP** (Simple Networ Management Protocol) pro čtení stavových a statistických informací a pro zjišťování komunikačních chyb.
- **ARP** (Addrss Resolution Protocol) pro převod mezi IP adresami a ethernetovými MAC adresami.
- **ICMP** (Internet Control Message Protocol) pro odesílání chybových informací.

Tyto protokoly jsou definovány v příslušných volně dostupných RFC specifikacích.

1.9 Rámce v Profinetu IO

Proto, aby bylo možné lépe porozumět struktuře a obsahu jednotlivých datových rámců přenášených mezi účastníky za účelem plnění funkcí Profinetu IO, jsou v této kapitole detailněji popsány hlavní datové položky standardních protokolů a jejich význam v Profinetu IO. Struktura a rozdělení rámců je zobrazena na obrázku obr. 1.8.



Obrázek 1.8: Přehled možných rámců v Profinetu IO

Každý řádek v obrázku reprezentuje jeden typ datového rámce v Profinetu IO. Všechny rámce začínají hlavičkou ethernetového protokolu, který je základním linkovým protokolem Profinetu IO. Po ní v rámci následují data dalších protokolů, a to buď protokolu ARP, IP/UDP nebo RT. Čárkovaně označené pole VLAN může, ale nemusí být vždy v rámcích obsaženo. Protokol RT má dále několik variant rámců, které budou popsány dále.

Ethernet Hlavička protokolu Ethernet kromě cílové a zdrojové MAC adresy obsahuje také typ nadřazeného protokolu, nazývaný *EtherType*. Pokud za ethernetovou hlavičkou následuje pole VLAN, které má svůj EtherType(0x8100), je typ následujícího protokolu uveden v poli VLAN. RT protokol Profinetu IO má přidělen *Ethertype* 0x8892.

Byte	0 - 7	8 - 13	14 - 19	20 - 21
	Preamble	Destination MAC Address	Source MAC Address	Ether Type/ Length

Obrázek 1.9: Hlavička protokolu ARP

ARP (Address Resolution Protocol) slouží v Profinetu IO k ověřování jedinečnosti IP adres. Před přidělením IP adresy IO-Device je vyslán dotaz, zda již tato adresa není přiřazena.

Byte	0 - 1	2 - 3	4	5	6 - 7	8 - 13	14 - 17	18 - 23	24 - 27
	HW Type.	Protoc Type.	HW Size.	Protoc. Size	Operat Code	HW Sender Addr.	Protocol Sender Addr.	HW Target	Protocol Target

Obrázek 1.10: Hlavička protokolu ARP

VLAN rozšiřuje ethernetový rámec o pole, definované standardem IEEE 802.1Q. Primárním účelem tohoto standardu je vytváření virtuálních sítí. Profinet IO si jej vypůjčil pro možnost priorizovat rámce způsobem, který je standardně podporován běžně používanými síťovými prvky. V Profinetu IO jsou využity tyto priority:

- Priority 8 a 7 - jsou určeny pro protokoly IRT komunikace
- Priorita 6 - je určena rámcům real-time cyklické komunikace a poruchovým hlášením s prioritou "*High*", přenášeným real-time acyklickou komunikací.
- Priorita 5 - je určena poruchovým hlášením s prioritou "*Low*", přenášeným real-time acyklickou komunikací.
- Priorita 0 - slouží rámcům všech ostatních protokolů (DCP,RPC). Jedná se ve směs o čtení a zápis parametrů a datových záznamů.

Podrobně je toto popsáno v [1].

Zařízení musí být připravena přijmout rámce s i bez pole VLAN, jelikož některá síťová zařízení, jako jsou bridge, tuto část rámce mohou před doručením do cíle vyjmout.

Bit	0-3	4	5 - 16	17 - 32
	Priority	Canon. Form.	VLAN ID	Type

Obrázek 1.11: Hlavička protokolu ARP

IP/UDP obsahuje IP adresy zdroje a cíle rámce. Dále pak čísla portů UDP a délku následujících dat v rámci. V Profinetu IO plní dvě funkce. Jednak může být použit k real-time komunikaci, jako UDP varianta protokolu RT třídy 1. V tomto případě není RT protokol v rámci umístěn za VLAN pole, ale až za UDP hlavičku. Dále slouží UDP/IP protokoly protokolu RPC, který je na nich postaven.

Bit	0 - 3	4 - 7	8 - 15	16 - 18	19 - 31
0	Version	Header Length	Type of Service	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options				
192	UDP Source Port			UDP Destination Port	
224	Length			Checksum	

Obrázek 1.12: Hlavičky protokolů IP a UDP

RPC (Remote Procedure Call) je obdobou lokálního volání procedury nebo funkce, kde je volání služeb *request* a *response* zabaleno do volání vzdálené procedury s předáváním parametrů. V Profinetu IO slouží tento protokol ke čtení a zápisu datových záznamů. Typy služeb jsou rozlišovány operačním číslem(opnum), obsaženým v hlavičce RPC (viztab. 1.2). Dále je v hlavičce určeno, zda se jedná o žádost nebo odpověď. Rozhraní a objekty jsou v Profinetu IO označovány speciálními UUID. Lze podle nich například rozlišit, mezi rozhraním IO-Device a IO-Controlleru. Pro zachování kompatibility mezi systémy s různou reprezentací dat je v hlavičce obsažena informace o způsobu jejich kódování.

Hodnota	Služba
0	Connect
1	Release
2	Read
3	Write
4	Control
5	Read Implicit

Tabulka 1.2: Typy služeb Profinetu IO označené v RPC opnum

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Version	Packet Type	Flag	00	Data Format			Serial High								
8	**Object UUID**															
24	**Interface UUID**															
40	**Activity UUID**															
56	Server Boot Time				Interface Version				Sequence Nr.				Operation Nr.		Interface Hint	
72	Activity Hint		Length of Body		Fragment Nr.		Serial Low									

Obrázek 1.13: Hlavičky protokolu RPC

1.10 Spuštění systému Profinet IO

Na začátku IO-Controller kontroluje dostupnost IO-Device prostřednictvím ověřování jména/adresy protokolem DCP/ARP. Následně je IO-Device přiřazena IP adresa. Když má IO-Device již přiřazenu IP adresu, je připraveno k datové výměně.

Spuštění systému vždy probíhá pomocí non-real-time komunikace s využitím RPC protokolu. Aktivní IO-Controller inicializuje svou žádostí pasivní IO-Device, které na každou žádost odpoví podle situace kladně nebo záporně.

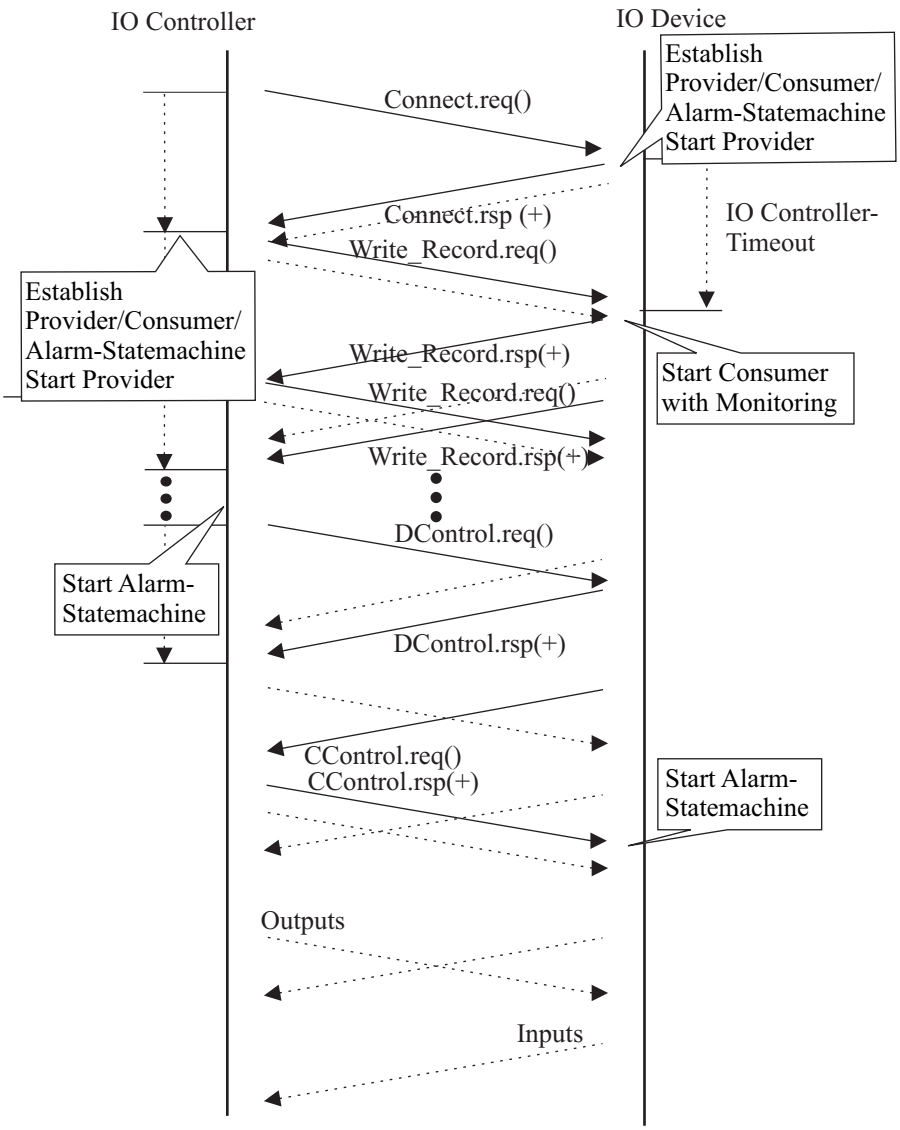
Navázání spojení je inicializováno vysláním rámce "Connect-Call" určujícím, která data budou cyklicky přenášena. Následně pomocí rámců "Write-Call" proběhne nastavení parametrů jednotlivých submodulů. V době, kdy IO-Device obdrží rámec s parametry prvního submodulu, je spuštěna cyklická komunikace mezi účastníky. Má však zatím význam pouze pro udržení vytvořených I/O CR. I/O data, obsažená v rámcích cyklické komunikace jsou ještě neplatná. Na základě přijetí každého "Write-Call" rámce provede IO-Device zápis parametrů do příslušného submodulu. IO-Controller oznámí konec této procedury vysláním rámce "EndOfParametrization(DControl)". IO-Device rámcem "Application "Ready(CControl)" oznámí, že na jeho straně je již připojení vytvořeno. Nyní jsou oba účastnící připraveni na datovou výměnu skutečných dat a poruch.

1.10.1 Úloha Kontext managementu

Hlavním úkolem Kontext managementu (CM) je vytvořit aplikační vztah mezi IO-Controllerem a IO-Device nebo IO-Supervisorem a IO-Device, založený na přijatých konfiguračních datech. Dále CM plní tyto funkce

- Vytvoření komunikačních vztahů (CR) (viz kapitola 1.7)
- Přiřazení příslušných komunikačních parametrů CR (viz kapitola 1.7.2)
- Jednoznačná identifikace přijatých dat - provádí rozdělování, patrících jednotlivým CR
- Distribuce parametrů popsaných v GSD souboru (viz kapitola 1.15)- CM IO-Controlleru tyto parametry zapisuje do jednotlivých IO-Device v podobě rámců "Write-Call"

Kontext managementu využívá k adresaci model slotů a subslotů IO-Device.



Obrázek 1.14: Spuštění systému Profinet IO (Zdroj:[2])

1.10.2 Connect Request (="Connect-Call")

Pomocí tohoto rámce IO-Controller (nebo IO-Supervisor) zahajuje vytvoření spojení s IO-Device. Obsahuje zejména parametry pro vytvoření požadovaných AR a CR.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	AR block	CR Input	CR Outp.	Alarm	Exp SubM	...	Exp SubM	FCS
6	6	4	2	28	80	20	58 + name	56+..	56+..	26	36/42		36/42	4

VLAN - je pouze volitelná část. Pro tento typ komunikace nemá zvláštní význam.

EtherType - v tomto případě se jedná o protokol IP (0x0800).

RPC - mimo jiné obsahuje typ služby connect(opnum=0).

NDR (Network Data Representation) - obsahuje informace o velikosti následujícího datového bloku rozděleného na jednotlivé bloky Profinetu IO.

AR blok - obsahuje jedinečný identifikátor (UUID) tohoto spojení, jeho typ (viz tab. 1.3) a vlastnosti, MAC adresu IO-Controlleru a jméno jeho stanice. Dále pak obsahuje monitorovací čas pro kontrolu výpadku při navazování spojení.

Hodnota	Typ	Popis
0x0001	IOCARSingle	základní typ AR
0x0003	IOCARCIR	AR určené k rekonfiguraci systému za chodu
0x0004	IOCAR_IOControllerRedundant	záložní AR
0x0005	IOCAR_IODeviceRedundant	záložní AR
0x0006	IOSAR	AR určený pro připojení IO-Supervisor

Tabulka 1.3: Typy aplikačních vztahů Profinetu IO

CR Input/Output blok - obsahuje informace o tom, v jakém cyklu a jak často bude RT rámec tohoto CR odesílán, které I/O datové objekty a stavy (IOPS,IOCS) v něm budou obsaženy a jaká bude jejich pozice v rámci. Součástí jsou i parametry pro monitorování komunikace na straně konzumenta dat.

Alarm CR Block - obsahuje maximální délku dat poruchového rámce, kterou je IO-Controller schopen přijmout. Dále obsahuje parametry vysílání poruch, jako jsou počet opakování a monitorovací čas.

Expected Submodule blok - počet těchto bloků v rámci odpovídá počtu submodulů, zahrnutých do vytvářeného AR. Kromě identifikace očekávaného modulu obsahuje blok informace o velikosti datové položky a položky stavových informací (IOPS,IOCS) příslušících tomuto submodulu.

1.10.3 Connect Response

Tímto rámcem IO-Device odpovídá na požadavek o připojení ("Connect-Call") poté, co provedl kontrolu proveditelnosti připojení. Odpověď může být kladná nebo záporná.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	AR block	CR Input	CR Outp.	Alarm	Modul Diff.	FCS
6	6	4	2	28	80	20	58 + name	12	12	12	14 + 18*Slot	4

VLAN - má stejný význam jako v předchozím rámci.

Ethertype - má stejný význam jako v předchozím rámci (= IP (0x0800)).

RPC - mimo jiné obsahuje typ služby connect(opnum=0).

NDR - kromě informace o velikosti následujícího datového bloku obsahuje stav vyřízení žádosti o připojení. Pokud byla žádost vyřízena bez chyby jsou všechny čtyři chybové byty nulové. V případě negativní odpovědi mají první dva byty, které určují typ poruchy, hodnoty 0xDB a 0x81. Následující byte ErrorCode1 určuje chybný blok, další byte ErrorCode2 označuje chybný parametr. Pokud je chybných parametrů více, je označen pouze první.

AR blok - kromě jedinečného identifikátoru (UUID) tohoto spojení obsahuje MAC adresu IO-Device.

CR Input/Output blok - obsahuje tzv. FrameID. Jedná se o označení, které bude uváděno v rámcích RT komunikace tohoto CR. Slouží k identifikaci rámců a jejich přiřazení správnému CR. (viz tab. 1.4)

Alarm CR Block - obsahuje referenční číslo sloužící k identifikaci poruchových hlášení. Dále obsahuje maximální velikost uživatelských dat v poruchovém rámci.

Module Difference blok - tento blok obsahuje seznam submodulů, které nemohou být zahrnuty do tohoto spojení. Jsou označena příslušnými adresami, tedy sloty a subsloty. Dále je uveden stav submodulu, který informuje o důvodu chyby. Důvodem může být například to, že je modul již zabrán jiným IO-Controllerem nebo že neodpovídá očekávanému.

1.10.4 Write Request

Touto žádostí provádí IO-Controller nastavení parametrů jednotlivých submodulů, zahrnutých ve spojení. Parametry submodulů jsou uvedeny v GSD souboru zařízení. Pro každý submodul je odeslán zvláštní rámec. Tento proces je prováděn pouze za předpokladu, že IO-Device předtím odeslal kladnou odpověď na rámec "Connect Call".

Struktura rámce vypadá takto:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	Write Block	Write Data	FCS
6	6	4	2	28	80	20	64	...	4

VLAN - má stejný význam jako v "Connect Call".

Ethertype - má stejnou hodnotu jako v rámci "Connect Call" (= IP (0x0800)).

RPC - mimo jiné obsahuje typ služby write(opnum=3).

NDR - obsahuje informace o velikosti datového bloku.

Write blok - obsahuje jedinečný identifikátor (UUID) tohoto spojení a adresu vybraného submodulu spolu s indexem datového záznamu parametrů.

Write data - obsahuje data parametrů adresovaného submodulu.

1.10.5 Write Response

Pomocí tohoto rámce IO-Device potvrzuje přijetí parametrů submodulu. IO-Controlleru tím oznamuje, že může pokračovat v procesu navázání spojení.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	Write Block	FCS
6	6	4	2	28	80	20	64	4

VLAN - má stejný význam jako v "Connect Call".

Ethertype - má stejnou hodnotu jako v rámci "Connect Call" (= IP (0x0800)).

RPC - mimo jiné obsahuje typ služby write(opnum=3).

NDR - kromě informace o velikosti následujícího datového bloku obsahuje stav vyřízení žádosti. Pokud byla žádost vyřízena bez chyby jsou všechny čtyři chybové byty nulové. V případě negativní odpovědi mají první dva byty, určující typ poruchy, hodnoty 0xDF a 0x80. Následující byte ErrorCode1 je rozdělen na dvě části ErrorClass a ErrorCode[3].

Write blok - je téměř totožný s blokem v žádosti odeslané IO-Controllerem. Je však ještě doplněn o dvě přídavná slova, která v případě negativní odpovědi obsahují dodatečné informace o chybě. Jejich význam definuje výrobce. V případě kladné odpovědi mají nulové hodnoty.

1.10.6 DControl Request

Pomocí DControl žádosti IO-Controller signalizuje dokončení nastavování parametrů submodulů.

Struktura rámce vypadá takto:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	Control Block	FCS
6	6	4	2	28	80	20	32	4

VLAN - má stejný význam jako v "Connect Call".

[3] Podrobně jsou tyto poruchy popsány v [1] na str.367-371

Ethertype - má stejnou hodnotu jako v rámci "Connect Call" (= IP (0x0800)).

RPC - mimo jiné obsahuje typ služby control(opnum=4).

NDR - obsahuje informace o velikosti datového bloku.

Control blok - obsahuje jedinečný identifikátor (UUID) tohoto spojení a údaj o typu řídicího příkazu nastavený na hodnotu "*Parameter End*".

1.10.7 DControle Response

Tímto rámcem IO-Device potvrzuje předchozí DControl žádost. Tím je vytvoření spojení dokončeno.

Struktura rámce vypadá takto:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	Control Block	FCS
6	6	4	2	28	80	20	32	4

VLAN - má stejný význam jako v "Connect Call".

Ethertype - má stejnou hodnotu jako v rámci "Connect Call" (= IP (0x0800)).

RPC - mimo jiné obsahuje typ služby control(opnum=4).

NDR - kromě informace o velikosti následujícího datového bloku obsahuje stav vyřízení žádosti o připojení. Pokud byla žádost vyřízena bez chyby jsou všechny čtyři chybové byty nulové. V případě negativní odpovědi mají první dva byty, určující typ poruchy, hodnoty 0xDD a 0x81. Následující byte ErrorCode1 určuje chybný blok, další byte ErrorCode2 označuje chybný parametr. Pokud je chyb více, je označen pouze první parametr.

Control blok - obsahuje jedinečný identifikátor (UUID) tohoto spojení a údaj o typu řídicího příkazu nastavený na hodnotu "*Done*".

1.10.8 CControl Request

Tímto rámcem IO-Device oznamuje, že je spojení v provozu.

Struktura rámce vypadá takto:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	Control Block	Modul Diff.	FCS
6	6	4	2	28	80	20	32	14 + 18*Slot	4

VLAN - má stejný význam jako v "Connect Call".

Ethertype - má stejnou hodnotu jako v rámci "Connect Call" (= IP (0x0800)).

RPC - mimo jiné obsahuje typ služby control(opnum=4).

NDR - obsahuje informace o velikosti datového bloku.

Control blok obsahuje jedinečný identifikátor (UUID) tohoto spojení a údaj o typu řídicího příkazu nastavený na hodnotu "*Application Ready*".

1.10.9 CControl Response

Tímto rámcem IO-Controller potvrzuje předchozí CControl žádost. Nyní končí fáze spuštění systému a začíná uživatelská výměna I/O dat a poruch, případně čtení nebo zápis datových záznamů.

Struktura rámce vypadá takto:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	Control Block	FCS
6	6	4	2	28	80	20	32	4

VLAN - má stejný význam jako v "Connect Call".

Ethertype - má stejnou hodnotu jako v rámci "Connect Call" (= IP (0x0800)).

RPC - mimo jiné obsahuje typ služby control(opnum=4).

NDR - kromě informace o velikosti následujícího datového bloku obsahuje stav vyřízení žádosti o připojení. Pokud byla žádost vyřízena bez chyby jsou všechny čtyři chybové byty nulové. V případě negativní odpovědi mají první dva byty, určující typ poruchy, hodnoty 0xDD a 0x81. Následující byte ErrorCode1 určuje chybný blok, další byte ErrorCode2 označuje chybný parametr. Pokud je chyb více, je označen pouze první parametr.

Control blok - obsahuje jedinečný identifikátor (UUID) tohoto spojení a údaj o typu řídicího příkazu nastavený na hodnotu "*Done*".

1.11 Uživatelská výměna dat

Přestože cyklická výměna I/O dat má význam až po provedení všech potřebných nastavení a vytvoření spojení, je spuštěna velice brzy po startu systému. V této době slouží k monitorování spojení mezi IO-Controllerem a IO-Device. V době spouštění systému toto monitorování provádí pouze IO-Controller. IO-Device sleduje spojení pouze v době mezi žádostí o připojení a zápisem parametrů subslotů.

Rámec pro cyklickou výměnu dat má následující strukturu:

Dest Addr.	Scr Addr.	Ether Type + VLAN	Ether Type	Frame ID	Data	IOPS	...	IOCS	...	Cycle	Data Sts	X Sts	FCS
6	6	4	2	2	1..	1		1		2	1	1	4

Nejmenší definovaná délka rámce je 64 bytů. Datová oblast mezi FrameID a cyklem má proto minimální délku 40 bytů. V případě, že není tato minimální kapacita využita, jsou volná místa vyplněna nulovými hodnotami. Maximální délka datové oblasti je 1440 bytů. Tato hodnota vychází z maximální velikosti ethernetového rámce. RT komunikace Profinetu IO totiž nepodporuje segmentaci dat do více rámců.

K cyklickému přenosu I/O dat může být volitelně použit i UDP protokol. V tom případě jsou mezi VLAN pole a FrameID ještě vloženy hlavičky protokolů IP a UDP(viz obr. 1.12). Zařízení jsou pak adresována IP adresami, které jim byly na začátku přiděleny.

VLAN - slouží v tomto případě k určení priority rámce.

EtherType - má hodnotu 0x8892, která je vyhrazena RT protokolu.

FrameID - slouží k identifikaci rámce. Jeho hodnota jednak vypovídá o třídě RT komunikace a dále také určuje I/O CR, kterému rámec patří. Podrobně je tento údaj popsán v tab. 1.4.

Hodnota	Význam
0x0000 - 0x00FF	Časová synchronizace (IRT)
0x0100 - 0x7FFF	Rámce RT třídy 3 unicast i multicast(IRT)
0x8000 - 0xBEFF	Rámce RT třídy 2 unicast (RT)
0xBF00 - 0xBFFF	Rámce RT třídy 2 multicast (RT)
0xC000 - 0xFAFF	Rámce RT třídy 1 unicast (RT)
0xFB00 - 0xFBFF	Rámce RT třídy 1 multicast (RT)
0xFC01	Poruchy priority "high"
0xFE01	Poruchy priority "low"
0xFEFE - 0xFEFF	DCP protokol

Tabulka 1.4: Význam položky FrameID protokolu RT

Data - zahrnují hodnoty I/O datových objektů, jejich stavy (IOPS) a stav konzumenta (IOCS). Velikost a pozice jednotlivých údajů v datové oblasti je definována v rámci "*Connect Call*", popsaném v kapitola 1.10.2. Stav konzumenta může být definován pro každý I/O datový objekt zvlášť. Konzument dat se jím vyjadřuje k datům, které naposledy přijal od poskytovatele.

Cycle - obsahuje stav cyklického čítače komunikace. Jeho hodnota udává čas v násobcích 31.25 μs. Slouží ke kontrole komunikace. Umožňuje určit periodu komunikace, pořadí rámců a jejich stáří.

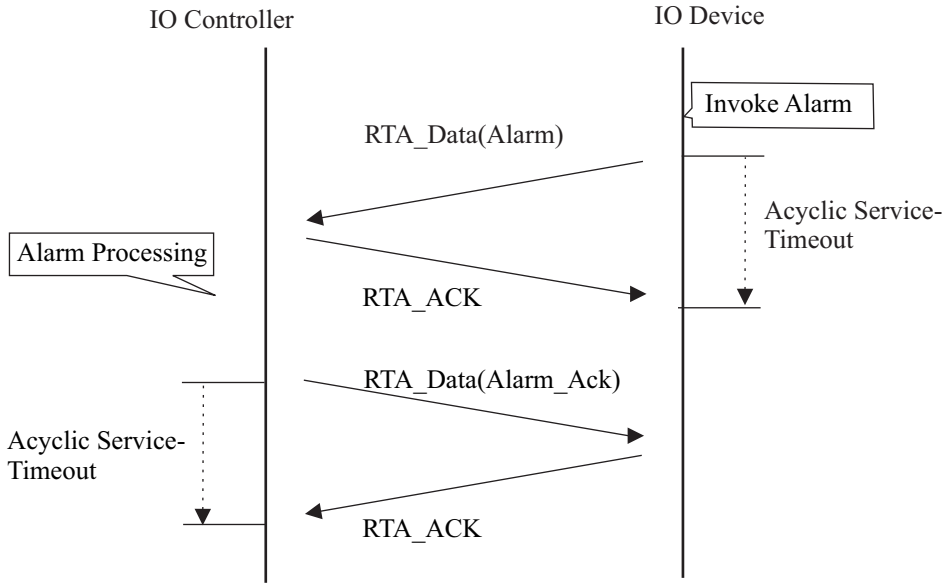
Data Status - vypovídá o celkovém stavu a významu dat, obsažených v tomto rámci. Určuje:

- zda se jedná o data primárního nebo záložního CR
- zda jsou data platná či ne
- zda byl poskytovatel v době odesílání rámce v režimu "*Run*" nebo "*Stop*"
- zda jsou v IO-Device dostupná nějaká diagnostická data

Transfer Status - má význam pouze pro IRT komunikaci. Pro RT komunikaci má trvale nulovou hodnotu.

1.12 Poruchová hlášení

Všechny události v Profinetu IO jsou přenášeny ve formě poruchových hlášení pomocí RT komunikace. Ta umožňuje těmto zprávám dát vyšší prioritu a urychlit tak jejich doručení. Všechny poruchy musí být potvrzeny, a to na dvou úrovních. IO-Controller musí nejprve potvrdit příjem poruchy. Po jejím převzetí uživatelskou aplikací musí ještě odeslat potvrzení jejího zpracování. Doručení tohoto potvrzení je následně ještě potvrzeno zařízením IO-Device.



Obrázek 1.15: Přenos poruchového hlášení (Zdroj:[2])

1.12.1 Použití poruchových hlášení v Profinetu IO

V Profinetu IO platí pro užívání poruchových hlášení následující pravidla:

- Hlášení jsou odesílány v době, kdy nastala. Oznamuje se i ukončení chybového stavu či události. V případě, že je doba trvání poruchy velmi malá, stačí odeslat oznámení o jejím ukončení.

- Procesní poruchy jsou odesílány v případě, že dojde k nějaké události v procesu definovaném výrobcem. Jedná se například o překročení limitní hodnoty měřené veličiny.
- Diagnostické události slouží k ohlašování interních poruch IO-Device, jako jsou zkraty nebo ztráty spojení.
- Poruchy vyjmutí a zasunutí (Pull and Plug) nastávají v modulárních IO-Device, když dojde k vyjmutí nebo vložení (sub)modulu do (sub)slotu. Po zasunutí (sub)modulu zpět do IO-Device a odeslání příslušného poruchového hlášení dojde k zařazení (sub)modulu do spojení. Tento proces je téměř totožný s procesem vytvoření spojení mezi IO-Controllerem a IO-Device. Probíhá znovu zápis parametrů do submodulu a následně výměna rámců "CControl" a "DControl". V případě, že je do slotu zasunut jiný než očekávaný submodul, je o tom IO-Controller informován vysláním poruchy "plug wrong".
- Informace o změnách v redundantních spojeních jsou oznamována speciálními hlášeními. Dochází k nim v případě přechodu na redundantní spojení, případně při návratu k hlavnímu spojení.
- Tzv. "return alarm", neboli hlášení o návratu, slouží k oznámení, že submodul, který byl původně v chybovém stavu, je opět připraven fungovat bez potřeby nastavení parametrů.
- Převzetí kontroly nad submodulem IO-Supervisorem a následné uvolnění submodulu jsou IO-Controlleru oznamována příslušnými hlášeními. Po uvolnění submodulu se IO-Controller chová stejně, jako by byl submodul znovu zasunut.

1.12.2 Oznámení poruchy

Pomocí rámce RTA_Data(Alarm) oznamuje IO-Device IO-Controlleru, že nastal začátek nebo konec stavu oznamovaného poruchovými hlášeními.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	Ether Type + VLAN	Ether Type	Frame ID	RTA Header	Alarm.. Notification	FCS
6	6	4	2	2	12	64	4

VLAN slouží v tomto případě k určení priority rámce.

EtherType má hodnotu 0x8892, která je vyhrazena RT protokolu.

FrameID určuje kanál poruch. Jsou definovány dva kanály: Alarm High (0xFC01) a Alarm Low (0xFE01) (viz tab. 1.4).

RTA Header primárně obsahuje označení poruchového CR zdroje a cíle hlášení. Důležitou položkou je typ rámce, který říká, že se jedná o data nové poruchy. Dále pak obsahuje informace o velikosti následujícího bloku a sekvenční číslo rámce.

Alarm Notification obsahuje označení typu poruchy a adresu objektu, ve kterém porucha nastala. Může se jednat o kanál, submodul, nebo dokonce celý modul. Rámec může být doplněn o data obsahující diagnostické informace.

1.12.3 Potvrzení oznámení poruchy na přenosové úrovni

Pomocí rámce RTA_ACK potvrzuje IO-Controller, že poruchové hlášení přijal. Oznamuje tím, že je připraven na příjem nových poruch. Dále zabrání opakovanému vysílání poruchy, které by nastalo, pokud by včas neodpověděl.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	Frame ID	RTA Header	FCS
6	6	4	2	2	12	4

VLAN slouží v tomto případě k určení priority rámce.

EtherType má hodnotu 0x8892, která je vyhrazena RT protokolu.

FrameID určuje kanál poruch. Jsou definovány dva kanály: Alarm High (0xFC01) a Alarm Low (0xFE01) (viz tab. 1.4). Je nastaven na stejnou hodnotu, jako v rámci oznamujícím poruchu.

RTA Header primárně obsahuje označení poruchového CR zdroje a cíle hlášení. Důležitou položkou je typ rámce, který říká, že se jedná o potvrzení přijetí poruchy. Dále pak obsahuje sekvenční číslo rámce.

1.12.4 Potvrzení převzetí poruchy na uživatelské úrovni

Po odeslání rámce RTA_Data(Alarm_Ack) si může být IO-Device jistý, že porucha byla převzata uživatelskou aplikací IO-Controlleru.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	Frame ID	RTA Header	Alarm Ack	FCS
6	6	4	2	2	12	64	4

VLAN slouží v tomto případě k určení priority rámce.

EtherType má hodnotu 0x8892, která je vyhrazena RT protokolu.

FrameID určuje kanál poruch. Jsou definovány dva kanály: Alarm High (0xFC01) a Alarm Low (0xFE01) (viz tab. 1.4). Je nastaven na stejnou hodnotu jako v rámci oznamujícím poruchu.

RTA Header primárně obsahuje označení poruchového CR zdroje a cíle hlášení. Důležitou položkou je typ rámce, který říká, že se jedná o datový rámec(Data-RTA). Dále pak obsahuje informace o velikosti následujícího bloku a sekvenční číslo rámce.

Alarm Ack obsahuje typ poruchy a adresu objektu, který poruchu vyvolal. Dále následuje stav IO-Controlleru, který obsahuje informace o tom, zda je daný typ poruchy podporován.

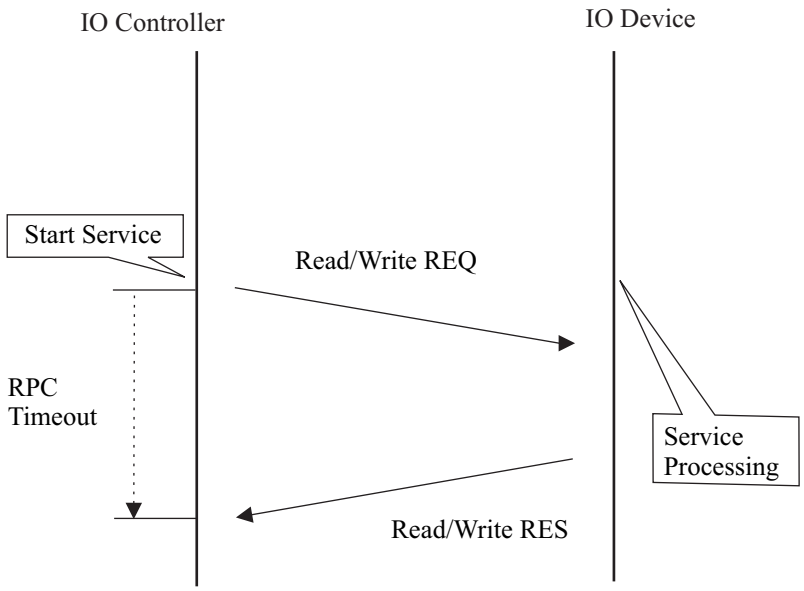
1.12.5 Potvrzení převzetí potvrzení o zpracování proruchy

Pomocí rámce RTA_ACK potvrzuje IO-Device převzetí potvrzení RTA_Data(Alarm_Ack). Oznamuje tím, že není potřeba opakovat vysílání předchozího rámce.

Jedná se o rámec se stejnou strukturou a obsahem jako má rámec uvedený v kapitole 1.12.3.

1.13 Služby pro čtení a zápis datových záznamů

Funkce čtení a zápisu datových záznamů využívají stejný komunikační vztah jako kontextový management při navazování spojení s IO-Device. Jsou tedy také založeny na RPC protokolu, který je v Profinetu-IO určen pro přenosy, které nejsou časově kritické, ale spíše vyžadují větší komfort přenosu dat. Stejně jako při navazování spojení je výměna dat založena na principu žádost-odpověď.



Obrázek 1.16: Čtení a zápis datových záznamů (Zdroj:[2])

Zápis datových záznamů je totožný s tím, který je uveden v kapitole 1.10.4, proto je nyní detailněji popsáno pouze čtení datových záznamů.

1.13.1 Read Request

Tato žádost slouží ke čtení datových záznamů z IO-Device.

Struktura rámce vypadá takto:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	Read Block	FCS
6	6	4	2	28	80	20	64	4

VLAN - je pouze volitelná část. Pro tento typ komunikace nemá zvláštní význam.

EtherType - v tomto případě se jedná o protokol IP (0x0800).

RPC - mimo jiné obsahuje typ služby read(opnum=2).

NDR - obsahuje informace o velikosti datového bloku.

Read blok obsahuje jedinečný identifikátor (UUID) spojení a adresu vybraného submodulu spolu s indexem datového záznamu.

1.13.2 Read Response

Pomocí tohoto rámce jsou přenášena požadovaná data do IO-Controlleru.
Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	IP UDP	RPC	NDR	Read Block	Read Data	FCS
6	6	4	2	28	80	20	64	...	4

VLAN - je pouze volitelná část. Pro tento typ komunikace nemá zvláštní význam.

EtherType - v tomto případě se jedná o protokol IP (0x0800).

RPC - mimo jiné obsahuje typ služby read(opnum=2).

NDR - kromě informace o velikosti následujícího datového bloku obsahuje stav vyřízení žádosti o připojení. Pokud byla žádost vyřízena bez chyby jsou všechny čtyři chybové byty nulové. V případě negativní odpovědi mají první dva byty, určující typ poruchy, hodnoty 0xDF a 0x80. Následující byte ErrorCode1 je rozdělen na dvě části ErrorClass a ErrorCode[4].

Read blok - je téměř totožný s blokem v žádosti, která byla odeslané IO-Controllerem. Je však ještě doplněn o dvě přídavná slova, která v případě negativní odpovědi obsahují dodatečné informace o chybě. Jejich význam definuje výrobce. V případě kladné odpovědi mají nulové hodnoty.

Read data - obsahuje uživatelská data.

[4] Podrobně jsou tyto poruchy popsány v [1] na str.367-371

1.14 Přidělování adres a identifikace zařízení

V Profinetu IO existují dvě varianty přidělování IP adres zařízením IO-Device. Všechna zařízení musí povinně podporovat přiřazování adres pomocí DCP protokolu(Discovery and Configuration Protocol). Volitelně lze využít IT standard DHCP. Základní rozdíl v přístupu těchto dvou protokolů je v chování IO-Device, pokud ještě nemá přidělenu IP adresu. V případě použití DCP protokolu IO-Device pasivně čeká, až se ho některý IO-Controller dotáže. DHCP protokol vyžaduje, aby IO-Device samo kontaktovalo DHCP server. Tento server nemusí být součástí IO-Controlleru, ale může fungovat jako samostatná aplikace.

Většina zařízení dostupných v současné době podporuje pouze povinnou variantu DCP, která je v následujícím textu podrobněji popsána. Nejprve si ukážeme jak je způsob přidělování adres řešen v konfiguračním nástroji SIMATIC Step 7. Proces probíhá v následujících krocích:

- Pomocí informací z GSD souborů jednotlivých zařízení je sestaven I/O systém.
- Uživatel každému IO-Device přiřadí logické jméno.
- Konfigurační nástroj sám naplánuje přidělí IP adresy každému IO-Device z rozsahu odpovídajícímu nastavení sítě.
- Nyní pomocí funkce pro přidělování jmen IO-Device provede nástroj prohledání sítě a vypíše dostupná zařízení a jejich MAC adresy.
- Uživatel každému nalezenému IO-Device přiřadí jméno ze seznamu, který byl vytvořen ze jmen, dříve naplánovaných pro jednotlivá zařízením.
- Jména jsou nyní zapsána a uložena do IO-Device.

Hardwarová konfigurace uložená v IO-Controlleru obsahuje IP adresu pro každé IO-Device. IO-Controller identifikuje IO-Device podle jména, které mu bylo přiřazeno. Následně mu přidělí jeho IP adresu.

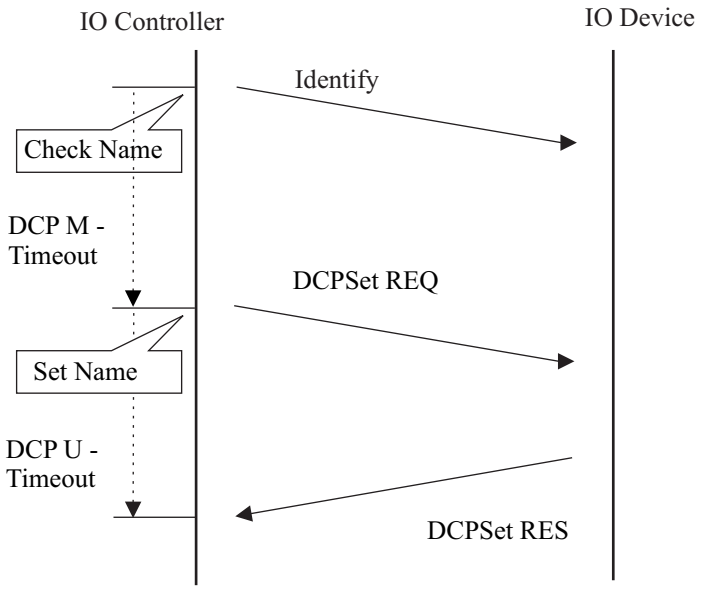
1.14.1 Přidělení jména IO-Device

Identifikace IO-Device probíhá v Profinetu IO prostřednictvím jedinečných jmen. Jména, na rozdíl např. od čísel, umožňují používat taková označení, která souvisí s významem či umístěním IO-Device v řízeném procesu. MAC adresy neslouží k prvotní identifikaci IO-Device zejména proto, aby bez nutnosti změny konfigurace sítě mohla být zařízení v případě poruchy jednoduše vyměněna za nová.

Přiřazování jmen probíhá tak, že konfigurační zařízení nejprve ověří, zda jméno, které se chystá přiřadit, už není používáno některým IO-Device v síti. Pokud je jméno volné, dojde k vyslání žádosti o jeho přidělení IO-Device, které převzetí jména potvrdí.

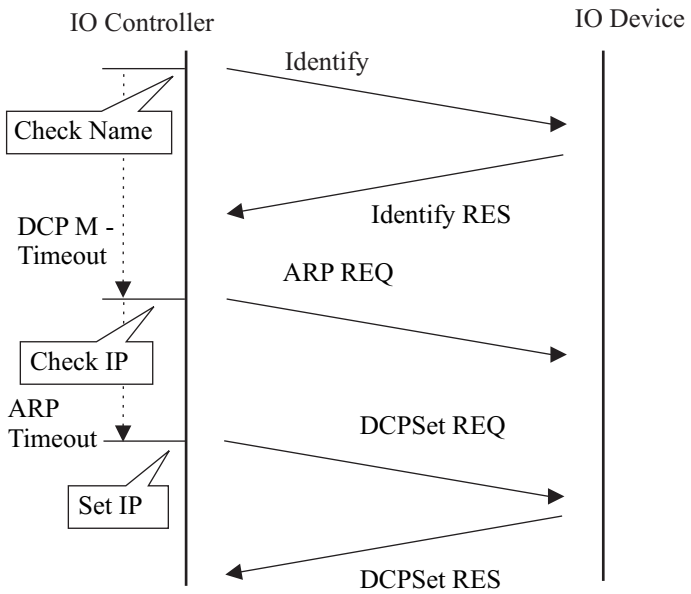
1.14.2 Postup pro přidělení IP adresy

Při navazování spojení s IO-Device je nutné, aby v té době již mělo IP adresu. Ta však v něm nebývá permanentně uložena. Proto musí nejprve dojít k jejímu nastavení. Tento úkol plní IO-Controller, který nejprve zjistí podle jména, zda je požadované IO-Device dostupné. Pokud



Obrázek 1.17: Přidělení jména IO-Device (Zdroj:[2])

ano, tak zkontroluje, zda již má přiřazenu IP adresu. V případě, že se IP adresa shoduje s tou, která je uvedena v konfiguračních datech, proces přidělování končí. Jinak IO-Controller pomocí ARP protokolu ověří, zda je IP adresa, kterou se chystá přiřadit volná, a provede její zápis do IO-Device.



Obrázek 1.18: Přidělení IP adresy IO-Device (Zdroj:[2])

1.14.3 Rámce protokolu DCP

Identify Request

Obecně může být tato služba použita k vyhledávání zařízení na síti podle různých kritérií. V procesu přidělování jmen a IP adres slouží k vyhledání IO-Device podle jména.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	Frame ID	DCP Hdr	DCP Data	...	DCP Data	FCS
6	6	4	2	2	10	4+n		4+n	4

DestAddr a SrcAddr obsahují adresy zdroje a cíle rámce. Cíl není stanoven, a proto je použita cílová adresa 01-0E-CF-00-00-00, určená pro multicasting.

VLAN je pouze volitelný.

EtherType má hodnotu 0x8892, která je vyhrazena RT protokolu.

FrameID obsahuje označení DCP multicast rámce, tedy hodnotu 0xFEFE.

DCP HDR obsahuje označení služby (viz tab. 1.5) a její typ (Request). Dále je v hlavičce obsaženo číslo transakce, které slouží k identifikaci žádosti. Jelikož se jedná o žádost typu multicast, je možné, že odesílateli přijde více odpovědí. DCP protokol z tohoto důvodu umožňuje umístit do hlavičky položku *ResponseDelay*, ze které IO-Device spolu s hodnotou části své MAC adresy vypočte dobu zpoždění odpovědi. Tím se zaručí, že odesílatel nedostane nekontrolovatelné množství odpovědí v krátkém časovém úseku.

Hodnota	Služba
3	Get
4	Set
5	Identify

Tabulka 1.5: Služby DCP protokolu

DCP Data obecně se jedná o blok dat, který obsahuje na svém začátku typ bloku a délku datové části. V tomto případě je v datové části obsaženo jméno, které ma být přiřazeno.

Identify Response

Pomocí Identify Response IO-Controller obdrží odpověď od IO-Device, které vyhovují odeslanému filtru.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	Frame ID	DCP Hdr	DCP Response	...	DCP Response	FCS
6	6	4	2	2	8	20		4..n	4

DestAddr a SrcAddr obsahují adresy zdroje a cíle rámce.

VLAN je pouze volitelný.

EtherType má hodnotu 0x8892, která je vyhrazena RT protokolu.

FrameID obsahuje označení DCP multicast rámce, tedy hodnotu 0xFEFE.

DCP HDR obsahuje označení služby (viz tab. 1.5) a její typ (Response). Dále je v hlavičce obsaženo číslo transakce, které se shoduje s číslem uvedeným v žádosti.

DCP Response obsahuje označení bloku spolu s jeho délkou a stavem parametrů.

Set Request

Tato služba slouží k zápisu různých parametrů IO-Device souvisejících s nastavením komunikace. V proceduře přiřazení jména se využívá k jeho zápisu do IO-Device. V případě přiřazování IP adresy obsahuje parametry pro ethernetové rozhraní IO-Device, jako jsou IP adresa, maska sítě a adresa výchozí brány.

Rámec má následující strukturu:

Dest Addr.	Scr Addr.	VLAN	Ether Type	Frame ID	DCP Hdr	DCP Data	...	DCP Data	FCS
6	6	4	2	2	8	6..n		6..n	4

DestAddr a SrcAddr obsahují adresy zdroje a cíle rámce.

VLAN je pouze volitelný.

EtherType má hodnotu 0x8892, která je vyhrazena RT protokolu.

FrameID obsahuje označení DCP multicast rámce, tedy hodnotu 0xFEFE.

DCP HDR obsahuje označení služby (viz tab. 1.5) a její typ (Response). Dále je v hlavičce obsaženo číslo transakce, které slouží k identifikaci žádosti.

DCP Data obsahuje označení bloku, jeho délku a příslušná data.

Set Response

Tento rámec slouží pro potvrzení požadavku na nastavení parametrů IO-Device.

Dest Addr.	Scr Addr.	VLAN	Ether Type	Frame ID	DCP Hdr	DCP Response	...	DCP Response	FCS
6	6	4	2	2	8	6..n		6..n	4

DestAddr a SrcAddr obsahují adresy zdroje a cíle rámce.

VLAN je pouze volitelný.

EtherType má hodnotu 0x8892, která je vyhrazena RT protokolu.

FrameID obsahuje označení DCP unicast rámce, tedy hodnotu 0xFEFD.

DCP HDR obsahuje označení služby (viz tab. 1.5) a její typ (Set). Dále je v hlavičce obsaženo číslo transakce, které se shoduje s číslem uvedeným v žádosti.

DCP Response obsahuje označení bloku spolu s jeho délkou a stavem parametrů, které byly nastaveny.

1.15 GSD soubory - popis zařízení Profinetu IO

Každé zařízení Profinetu IO je popsáno tzv. GSD souborem. Na rozdíl od GSD souborů zařízení Profibusu DP, která jsou tvořena obyčejnými textovými soubory s položkami označenými klíčovými slovy, je popis profinetových zařízení zaznamenán v podobě XML dokumentu. Tento rozšířený standard umožňuje data v dokumentu lépe strukturovat a efektivněji číst.

Každý výrobce dodává profinetová zařízení s příslušnými GSD soubory, které byly testovány a certifikovány spolu se zařízením. Organizace PI poskytuje XML schéma pro popis Profinet IO zařízení, které usnadňuje vytváření a kontrolu GSD souborů.

Podrobný popis GSD souborů pro Profinet IO je uveden ve specifikaci [5].

1.15.1 Pojmenovávání GSD souborů

Pojmenovávání GSD souborů je standardizováno následujícím způsobem:

```
GSD-[verze GSD schématu]-[výrobce]-[označení zařízení]-[datum].xml
```

1.15.2 Struktura GSD souboru

GSD soubory jsou založeny na standardu ISO 15745, určeném pro popis zařízení. Soubor má dvě základní části **Profile Header** a **Profile Body**.

Profile Header

Tato část je pro všechny soubory stejná a měla by vypadat takto:

```
<ProfileHeader>
   <ProfileIdentification>PROFINET Device Profile</ProfileIdentification>
   <ProfileRevision>1.00</ProfileRevision>
   <ProfileName>Device Profile for PROFINET Devices</ProfileName>
   <ProfileSource>PROFIBUS Nutzerorganisation e. V. (PNO)</ProfileSource>
   <ProfileClassID>Device</ProfileClassID>
   <ISO15745Reference>
      <ISO15745Part>4</ISO15745Part>
      <ISO15745Edition>1</ISO15745Edition>
      <ProfileTechnology>GSDML</ProfileTechnology>
      </ISO15745Reference>
</ProfileHeader>
```

Profile Body

Zde je uveden popis Profinet IO zařízení. Tato část se dále dělí podle ISO 15745 na tyto části:

- **DeviceIdentity** - udává jedinečné ID výrobce a zařízení, sloužící k jeho identifikaci
- **DeviceFunction** - určuje třídu zařízení a jméno řady výrobce
- **ApplicationProcess** - jedné se o hlavní část obsahující konfigurační data jednoho aplikačního procesu zařízení

Nejdůležitější části bloku ApplicationProcess jsou:

DeviceAccessPointList: obsahuje seznam rozhraní zařízení a jejich parametry. Je zde například uvedeno, jakou maximální rychlost cyklické RT komunikace rozhraní podporuje nebo zda podporuje přidělování adres pomocí DHCP. Dále zde lze zjistit maximální velikost bloku I/O dat či velikost stavových informací IOPS a IOCS.

ModuleList: uvádí popis jednotlivých modulů, které mohou být obsaženy v zařízení. Jsou zde mimo jiné popsány možné submoduly každého modulu a jejich datové položky. Důležitou částí je **ParameterRecordDataItem**, která obsahuje popis datového záznamu parametrů každého submodulu, které jsou nastavovány při navázání spojení s IO-Device.

ValueList: obsahuje seznam textů pro hodnoty různých parametrů.

ChannelDiagList: obsahuje texty pro uvedené typy poruch kanálů.

GraphicList: odkazuje na grafické soubory zařízení, která jsou určené pro konfigurační nástroj.

CategoryList: obsahuje seznam a název kategorií, do kterých jsou rozděleny jednotlivé moduly.

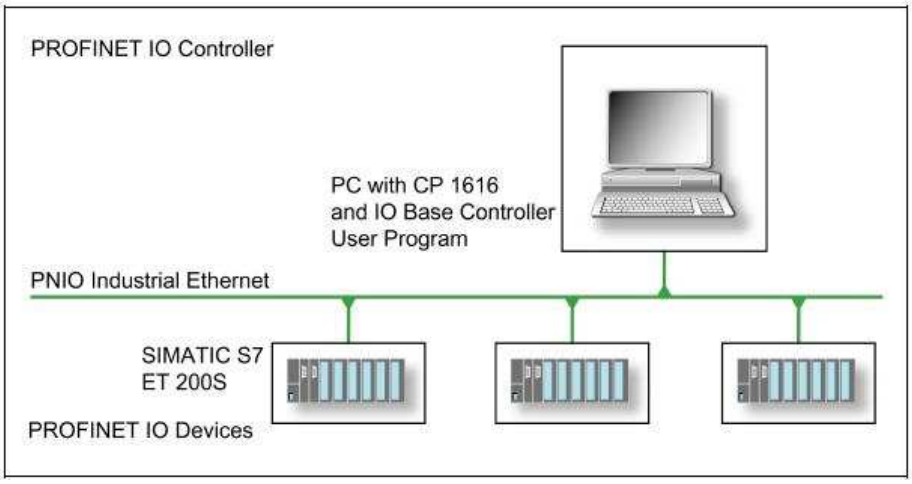
ExternalTextList: obsahuje texty, na které se mohou odkazovat jednotlivé položky dokumentu. Texty lze uvádět ve více jazycích.

Kapitola 2

Použité systémové prostředky

2.1 Uživatelské rozhraní IO Base

IO Base User Programming Interface (dále jen IO Base) je knihovna jazyka C, která je součástí programového balíku Simatic NET firmy Siemens. Toto rozhraní je určeno pro realizace jednodušších automatizačních úloh, využívajících standardní PC. Díky funkcím, zaměřeným na čtení diagnostických informací a příjem poruch, je toto řešení vhodné i jako základ aplikace pro diagnostiku sítě a jejich účastníků. Jako síťové rozhraní lze využít jak standardní ethernetový adaptér, tak výkonný komunikační procesor CP 1616, který má základní služby a protokoly Profinetu IO implementovány přímo ve svém firmware. Příklad typické aplikace je zobrazen na obrázku obr. 2.1.



Obrázek 2.1: Příklad aplikace IO Base rozhraní (Zdroj:[3])

Knihovna je určena jak k realizaci aplikace IO Controlleru, tak IO Device. Programové rozhraní nabízí funkce potřebné k ovládání obou těchto účastníků sítě. Tato práce se zaměřuje pouze na část související s IO Controllerem. Kompletní přehled a popis funkcí lze získat z dokumentu [3].

Tato kapitola popisuje principy a funkce knihovny IO Base, které byly použity k realizaci aplikace IO-Controlleru.

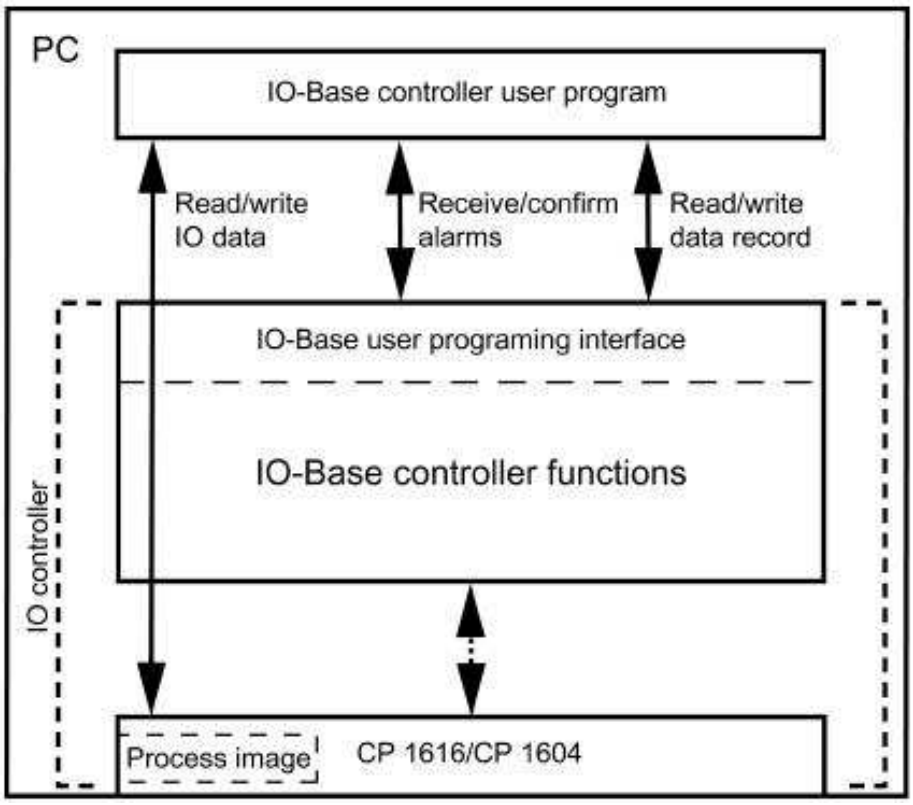
2.1.1 Softwarová architektura aplikace

Popis architektury aplikace s využitím CP 1616

IO Base je propojeno s knihovnou PNIO, která komunikuje pomocí ovladače přímo s kartou CP 1616. Knihovna PNIO si s CP 1616 pouze vyměňuje data pomocí několika jejích specializovaných obousměrných kanálů. Karta se sama stará o komunikaci pomocí standardních služeb a protokolů. CP 1616 je podrobněji popsána v kapitole 2.2.

IO Base ve spojení s CP 1616 poskytuje všechny funkce, které uživatelský program potřebuje ke komunikaci se zařízeními IO Device.

Na obrázku obr. 2.2 je zobrazena struktura aplikace s vyznačenými funkcemi pro čtení a zápis I/O dat, odesílání a příjem poruch a čtení a zápis datových záznamů. Aplikace se připojuje k procesu pomocí Profinet IO komunikačního procesoru. Nastavení komunikace se provádí pomocí nástrojů Step 7 nebo NCM PC. Výsledná konfigurace je následně odeslána do CP 1616 buď lokálně, to je možné jen pro aplikaci na platformě Windows, nebo pomocí TCP/IP protokolů přímo do karty na jiném PC.



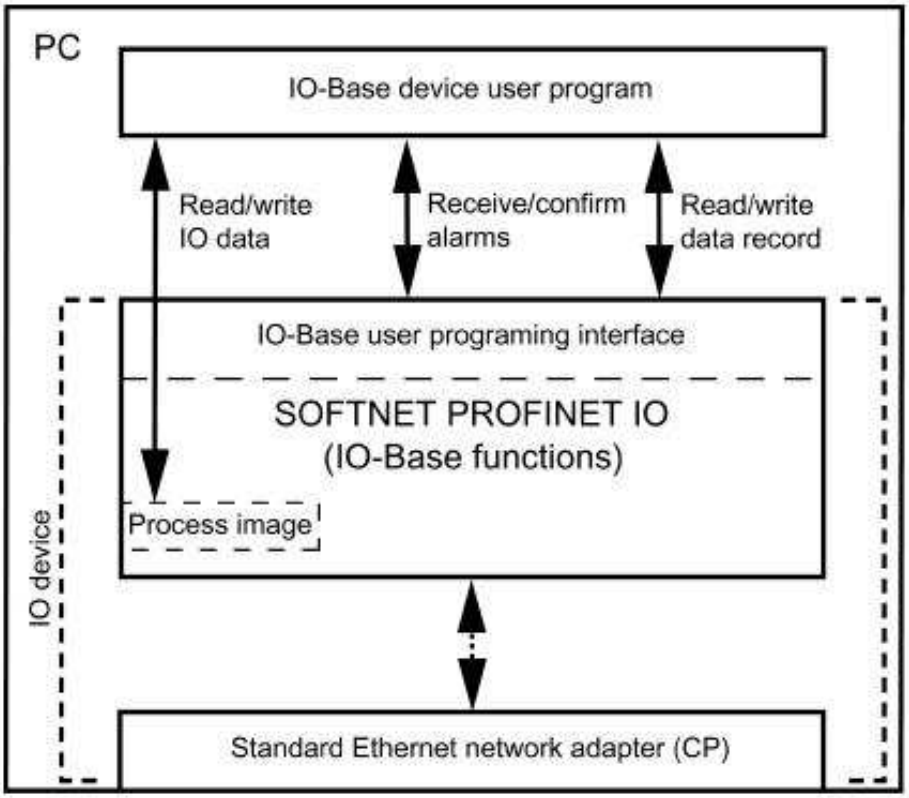
Obrázek 2.2: Struktura IO Controlleru s CP 1616 a IO Base (Zdroj:[3])

Popis architektury SOFTNET Profinet IO

Tato varianta je určena pro aplikace využívající standardní ethernetový adaptér. V tomto případě je použita knihovna PNIO, která obsahuje kompletní softwarový zásobník Profinetu IO, nazývaná SOFTNET Profinet IO.

Toto řešení má svá omezení. Zejména je určeno pouze aplikacím typu IO Controller. Pro IO Device je nutné použít předchozí řešení. Obraz vstupů a výstupů z celého procesu není v tomto případě uložen na kartě, ale v operační paměti PC. Nelze proto použít rychlý a optimalizovaný způsob čtení a zápisu IO dat pomocí funkcí využívajících buffer. Další omezení se týkají funkcí hlídání běhu nadřazené uživatelské aplikace, které jsou dostupné pouze s CP 1616.

Na obrázku obr. 2.3 je zobrazena struktura aplikace s vyznačenými funkcemi pro čtení a zápis I/O dat, odesílání a příjem poruch a čtení a zápis datových záznamů. Aplikace se připojuje k síti pomocí standardního ethernetového adaptéru. Konfigurace spojení se zařízeními IO Device se provádí pomocí nástrojů Simatic Manager nebo NCM PC.



Obrázek 2.3: Struktura IO Controlleru se SOFTNET Profinet IO (Zdroj:[3])

2.1.2 Fáze chování IO Controlleru s IO Base

Typický způsob chování uživatelského programu realizující IO Controller pomocí IO Base může být rozdělen do tří fází:

- Inicializační fáze
- Produktivní fáze
- Dokončovací fáze

Inicializační fáze

Tato fáze je rozdělena do čtyřech kroků:

1. `PNIO_controller_open( )` - zaregistruje aplikaci IO Controlleru v IO Base a dále zaregistruje callback funkce pro datové záznamy a popruchy.
2. `PNIO_register_cbf( )` - zaregistruje ostatní callback funkce. Například pro změny režimu.
3. `PNIO_set_mode( )` - odešle požadavek o nastavení IO Controlleru do režimu *OPERATE*
4. **Čekání na změnu režimu**

Systém callback funkcí je blíže popsán v kapitole 2.1.5.

Produktivní fáze

Během produktivní fáze probíhá výměna dat se zařízeními IO Device. Jedná se o tyto data:

- Čtení a zápis IO dat
- Příjem a potvrzování poruch
- Čtení a zápis datových záznamů

Čtení a zápis IO dat Následující funkce slouží ke čtení a zápisu procesních dat:

- `PNIO_data_read( )`
- `PNIO_data_read_cache_refresh( )`
- `PNIO_data_read_cache( )`
- `PNIO_data_write( )`
- `PNIO_data_write_cache( )`
- `PNIO_data_write_cache_flush( )`

Tyto funkce přistupují k I/O datům, uloženým v obraze procesních dat. IO Base nezávisle a samostatně provádí cyklickou komunikaci se zařízeními IO Device. Nastavení časových parametrů jednotlivých spojení bylo provedeno v průběhu konfigurace systému v Simatic Manageru.

Příjem a potvrzování poruch Příjem a zpracování poruch probíhá pomocí callback funkce pro příjem poruch (PNIO_CBE_ALARM_IND), která je zavolána po příchodu nové poruchy. Všechna potvrzení příjmu poruchy provádí knihovna sama.

Čtení a zápis datových záznamů Následující funkce umožňují čtení a zápis datových záznamů IO Device:

- `PNIO_rec_read_req( )`
- `PNIO_rec_write_req( )`

Požadované datové záznamy a potvrzení zápisu záznamů je řešeno prostřednictvím callback funkcí, které požadovaná data a odpovědi předávají uživatelské aplikaci.

Postup čtení datového záznamu

1. `PNIO_rec_read_req( )` - odešle požadavek na čtení datového záznamu
2. Callback událost `PNIO_CBE_REC_READ_CONF` - signalizuje potvrzení žádosti a doručení dat. Zároveň předá ukazatel na doručený datový záznam.

Postup zápisu datového záznamu

1. `PNIO_rec_write_req( )` - odešle požadavek na zápis datového záznamu
2. Callback událost `PNIO_CBE_REC_WRITE_CONF` - předá výsledek operace.

Systém callback funkcí je blíže popsán v kapitole 2.1.5.

Dokončovací fáze

Tato fáze zahrnuje tři kroky:

1. `PNIO_set_mode( )` - odešle požadavek o nastavení IO Controlleru do režimu *OFFLINE*
2. **Čekání na změnu režimu** - změna je signalizována voláním callback funkce. Po změně stavu je již veškerá komunikace ukončena.
3. `PNIO_controller_close( )` - zruší všechny registrace IO Controlleru v knihovně IO Base.

2.1.3 Cyklická výměna dat

Princip

Pro cyklický datový přenos jsou dostupné dva způsoby:

- Přímý přístup do obrazu procesních dat
- Rychlý přístup do obrazu procesních dat pomocí bufferu

Stav IO dat

Kvalita IO dat je popisována stavem dat, který může nabývat hodnot GOOD (dobrý) nebo BAD (špatný). Všechna vyměňovaná data jsou vždy ohodnocená dvěma stavovými údaji:

- Local status - stav aplikace IO Controlleru
- Remote status - stav komunikačního partnera

Při výměně IO dat slouží *Local status* k informování komunikačního partnera o stavu odesílatele. V případě stavu BAD nemusí příjemce data převzít a může je nahradit např. implicitními hodnotami. *Remote status* říká odesílateli dat, jestli příjemce data přijal v pořádku.

Přímý přístup

S tímto přístupem jsou IO dat vždy zapisována přímo do paměti obrazu procesních dat, umístěné přímo v komunikačním procesoru nebo v paměti PC (záleží na použitém komunikačním rozhraní).

Pro tento přístup jsou určeny tyto funkce:

- `PNIO_data_read( )` - okamžité načtení vstupních dat do obrazu procesních dat
- `PNIO_data_write( )` - okamžitý zápis výstupních dat do obrazu procesních dat

Tyto funkce jsou vhodné při zápisu malého množství dat.

Rychlý bufferový přístup

Tento přístup vždy používá vyrovnávací paměť hostitelského počítače. Vždy je přenášen celý obraz procesních dat. Přenos dat je optimalizovaný pomocí některých mechanismům. K obsluze rychlého přístupu slouží tyto funkce:

Čtení dat zahrnuje tyto kroky

1. `PNIO_data_read_cache_refresh( )` - přenese celý obraz procesních dat do vyrovnávací paměti pro čtení
2. `PNIO_data_read_cache( )` - čte vstupní data příslušného submodulu ze vstupní vyrovnávací paměti

Zápis dat zahrnuje tyto kroky

1. `PNIO_data_write_cache( )` - zapíše výstupní data pro příslušný submodul do vyrovnávací paměti pro zápis
2. `PNIO_data_write_cache_flush( )` - přenese všechna výstupní data z vyrovnávací paměti pro zápis do obrazu procesních dat

Bufferový přístup je určen pouze pro aplikace se speciálním komunikačním rozhraním, jako je CP 1616.

Výměna dat s IO Device

Jak při zápisu, tak při čtení IO dat přistupují všechny funkce IO Base pouze do obrazu procesních dat nikoliv přímo do zařízeních IO Device. Výměna dat mezi obrazem procesních dat a zařízeními IO Device je prováděna automaticky a cyklicky funkcemi na nižší úrovni.

2.1.4 Způsob adresování IO Device

V rámci jednoho Profinet IO systému existuje pouze jeden adresní prostor vstupních a výstupních dat, který je společný pro všechna zařízení. Přiřazení logických adres adresám podle specifikace Profinetu IO (modul/submodul/IO Device) probíhá během konfigurace systému.

2.1.5 Callback systém

Systém callback funkcí umožňuje IO Base volat uživatelsky definované funkce jako reakci na interní události. Callback funkce jsou standardní funkce definovaného typu, implementované v uživatelském programu.

Callback událost je asynchronní událost vyvolaná IO Base rozhraním, která přeruší chod uživatelské aplikace a zavolá příslušnou zaregistrovanou callback funkci ve zvláštním vláknu. Proto musí tyto funkce obsahovat synchronizační techniky pro přístup ke sdíleným datům a funkcím.

Typy callback funkcí IO Base pro aplikace IO Controlleru jsou uvedeny v následující tabulce. Tabulka také obsahuje informace o tom, jak se příslušná funkce registruje a co způsobuje její vyvolání.

Callback událost	**Typ callback události**	**Registrováno...**	**Vyvoláno...**
Přijetí poruchy	`PNIO_CBE_-ALARM_IND`	`PNIO_controller_open()`	IO Device
Přijetí výsledku čtení datového záznamu	`PNIO_CBE_REC_-READ_CONF`	`PNIO_controller_open()`	`PNIO_rec_read_req()`
Přijetí výsledku zápisu datového záznamu	`PNIO_CBE_REC_-WRITE_CONF`	`PNIO_controller_open()`	`PNIO_rec_write_req()`
Změna lokálního režimu	`PNIO_CBE_MODE_IND`	PNIO_register_cbf()	`PNIO_set_mode()`
Změna stavu spojení s IO Device	`PNIO_CBE_-DEV_ACT_CONF`	`PNIO_register_cbf()`	`PNIO_device_activate()`

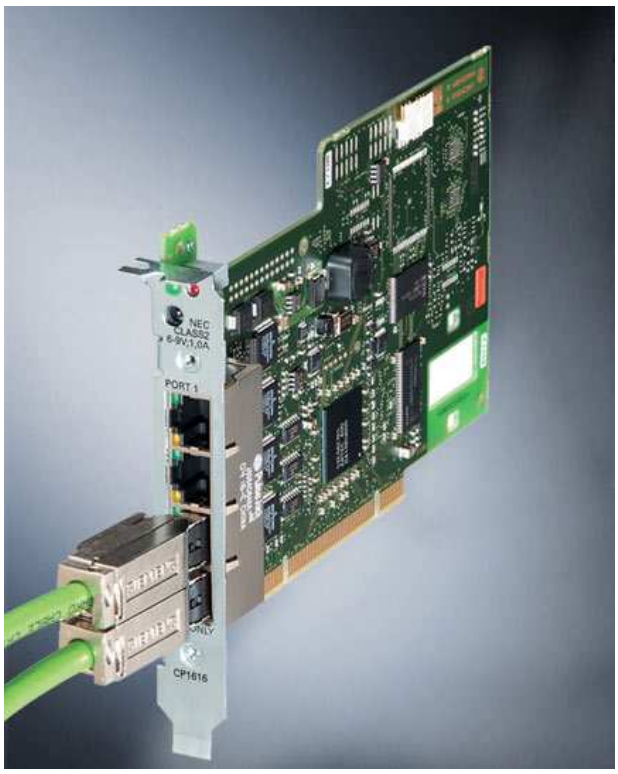
Tabulka 2.1: Callback funkce IO Base pro IO Controller

2.2 CP 1616

CP 1616 je PCI modul, který umožňuje připojení standardního PC do Profinetu IO. Na rozdíl od běžných ethernetových adaptéru obsahuje obvod ASICS ERTEC 400, který je přímo určený pro real-time komunikaci po Ethernetu. Tento obvod umožňuje 4-portový real-time switch, který je také na kartě integrován. Switch podporuje funkci AutoCrossover, která umožňuje používat jak křížená, tak nekřížené kabely. Karta je schopna se chovat jako IO-Controller i IO-Device. Podporuje také obě varianty real-time komunikace, RT i IRT. Pro aplikace, vyžadující trvalý a bezpečný provoz sítě lze využít možnost externího napájení karty, které při výpadku napájení PC zachová funkci switche.

CP 1616 může být provozována na různých platformách. Standardně jsou dodávány zdrojové kódy ovladače pro Linux, které mohou být převedeny i na jiné operační systémy. Ve Windows zajišťuje přístup ke kartě softwarový balík Simatic Net.

Ovladač zpracovává přerušení karty, mapuje obraz procesních dat karty pro knihovnu PNIO a zpracovává komunikační úkoly mezi knihovnou PNIO a firmware karty. Obrázek obr. 2.5 zobrazuje základní strukturu ovladače a karty. Šipky označují komunikační kanály mezi ovladačem a firmware karty. Kanály jsou realizovány paměťovou oblastí v CP 1616. Každý kanál obsahuje dva kruhové buffery, jeden pro data od ovladače pro firmware a jeden pro opačný směr.



Obrázek 2.4: Komunikační karta CP 1616 (Zdroj:[13])

Každý kanál karty je vyhrazeny pro jiný typ dat. V operačním systému Linux je každý kanál tvořen zvláštním souborem zařízení, který umožňuje přímí přístup do paměti karty, vyhrazené pro příslušný kanál. Jednotlivé kanály mají tento význam:

sync komunikační kanál pro synchronní komunikaci

alarm komunikační kanál pro asynchronní komunikaci

modind komunikační kanál pro asynchronní změny stavu protokolu

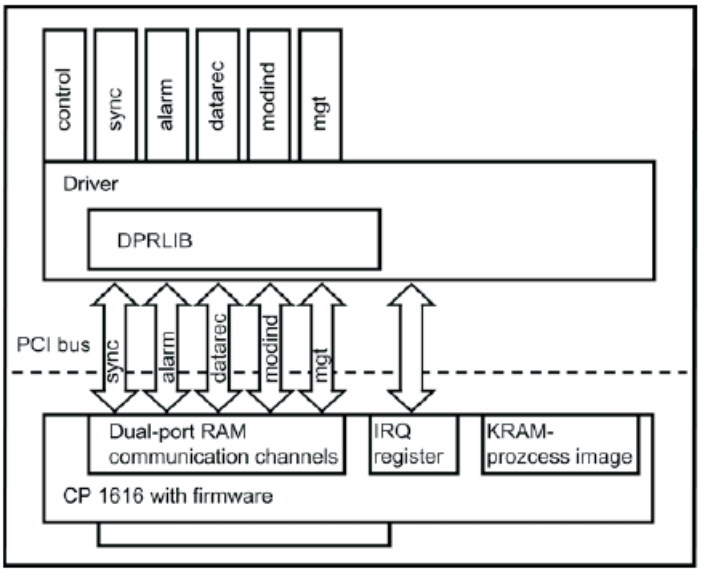
datarec komunikační kanál pro přenos dat datových záznamů

mgt komunikační kanál pro správu watchdogu aplikace

control není skutečným kanálem komunikační karty. Slouží k řízení vztahu ovladače a aplikace

2.3 Monitorování síťového provozu

Existuje více způsobů, jak monitorovat provoz sítě typu Ethernet. Asi nejrozšířenějším je knihovna libpcap, která umožňuje přímí přístup k linkové vrstvě v operačního systému. Tuto knihovnu používají renomované síťové analyzátory, jako je např. Ethereal nebo TCPDUMP. Kromě odchytávání rámců ze síťového adaptéru umožňuje i jejich filtrování a další zpracování. Jednou z hlavních výhod, plynoucích z jejího použití, je snadná přenositelnost uživatelských programů mezi platformami Windows a Linux. Kromě původní Linuxové verze libpcap existuje totiž i její verze pro Windows, nazvaná WinPCap.



Obrázek 2.5: Přístup k paměti CP 1616 (Zdroj:[4])

2.3.1 Popis knihovny WinPCap

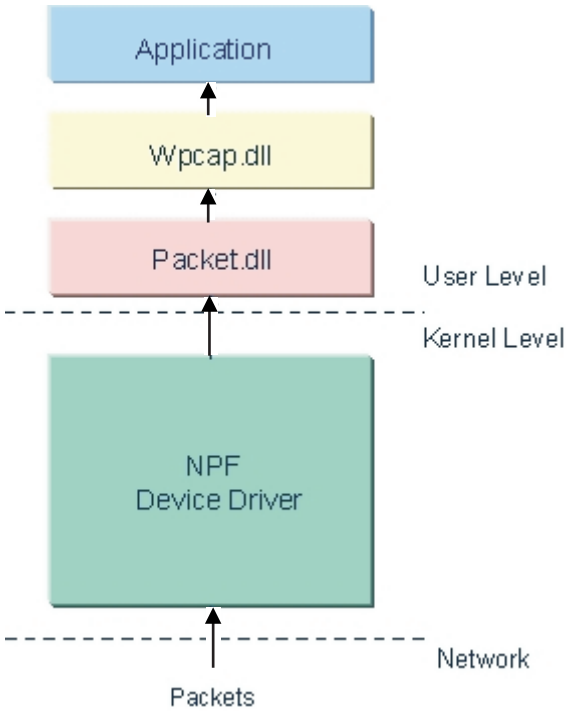
WinPCap je architektura určená k odchytávání rámců a síťovou analýzu v operačních systémech Win32. Obsahuje vnitřní filtr rámců, dynamickou knihovnu pro nízkoúrovňové funkce (*packet.dll*) a systémově nezávislou dynamickou knihovnu pro funkce na vyšší úrovni (*wpcap.dll*). Pro odchytávání rámců je nutné úzce spolupracovat se síťovým adaptérem a funkcemi operačního systému. Struktura architektury WinPCap je zobrazena na obrázku obr. 2.6.

Odchytávací systém musí nejprve obejít protokolový zásobník operačního systému, aby získal přístup k nezpracovaným datům ze sítě. To vyžaduje část programu, běžícího uvnitř jádra operačního systému, která spolupracuje přímo s ovladači síťových adaptérů. Tato část je značně závislá na systému. WinPCap toto řeší použitím ovladače nazývaného Netgroup Packet Filter (NPF). Pro každou verzi Windows je poskytována zvláštní verze tohoto ovladače. Ovladač nabízí jak základní funkce pro odchytávání a odesílání taketů, tak programovatelný filtrační systém a monitorovací nástroj. Filtrační systém umožňuje definovat skupinu rámců, které budou odchytávány. Monitorovací systém nabízí jednoduchý a výkonný způsob získání statistických údajů o síťovém provozu.

Pro snadné použití funkcí NPF obsahuje WinPCap také rozhraní pro uživatelské programy. Toto rozhraní je již nezávislé na verzi operačního systému. Skládá se ze dvou knihoven:

- *packet.dll* - přístup k funkcím ovladače NPF
- *wpcap.dll* - skupina výkonnějších funkcí vyšší úrovně, kompatibilních s libpcap (Linuxová verze). Tyto funkce umožňují přístup nezávislý na hardware i operačním systému.

Vzhledem k cíli této práce, vytvořit aplikaci přenositelnou mezi systémem Windows a Linux, se budeme zabývat pouze těmi funkcemi, které jsou nezávislé na operačním systému. Tedy funkcemi z knihovny *wpcap.dll*.



Obrázek 2.6: Architektura WinPCap (Zdroj:[10])

2.3.2 Získání seznamu síťových adaptérů

První věc, kterou je nutné provést při monitorování síťového provozu, je výběr síťového rozhraní, aktuálně dostupného v systému. WinPCap i libcap obsahují funkci `pcap_findalldevs()`, která zjistí všechny dostupné adaptéry. Tato funkce vrací lineární seznam struktur `pcap_if`, každá z nich obsahuje kompletní informace o příslušném adaptéru. Položky *name* a *description* obsahují srozumitelný název a popis adaptéru.

```
pcap_if_t *alldevs;
pcap_if_t *d;
int i=0;
char errbuf[PCAP_ERRBUF_SIZE];
/* Retrieve the device list from the local machine */
if (pcap_findalldevs(&alldevs, errbuf) == -1)
{
    fprintf(stderr,"Error in pcap_findalldevs_ex: %s\n", errbuf);
    exit(1);
}

/* Print the list */
for(d= alldevs; d != NULL; d= d->next)
{
    printf("%d. %s", ++i, d->name);
    if (d->description)
```

```
17          printf(" (%s)\n", d->description);
18      else
19          printf(" (No description available)\n");
20  }
21
22  /* We don't need any more the device list. Free it */
23  pcap_freealldevs(alldevs);
```

Některé operační systémy neposkytují popis síťových adaptérů, proto musí být u přenositelných aplikací ošetřen stav, kdy proměnná *description* nabývá hodnoty NULL.

Každá struktura `pcap_if` v seznamu síťových adaptérů obsahuje seznam struktur `pcap_addr`, každá struktura má tyto položky:

- seznam IP adres adaptéru
- seznam síťových masek (korespondující se seznamem IP adres)
- seznam broadcast adres (korespondující se seznamem IP adres)
- seznam cílových adres (korespondující se seznamem IP adres)

Po skončení práce se síťovými adaptéry se načtený seznam uvolní pomocí funkce `pcap_freealldevs()`.

2.3.3 Odchytávání rámců

Když jsme si zvolili požadovaný síťový adaptér, můžeme začít s jeho monitorování. Funkce, která zpřístupní zvolený adaptér, se nazývá `pcap_open_live()`. Jejími parametry jsou:

snaplen : určuje část rámců, která bude zaznamenávána. V některých operačních systémech (jako xBSD a Win32) může být ovladač nastaven tak, aby omezoval maximální velikost zaznamenávané části paketů, to umožňuje snížit množství dat přenášených aplikaci, a tím zvýšit efektivitu odchytávání. Implicitně je přednastavena hodnota 65536, která je větší než maximální možné MTU(Maximum Transmission Unit) délka. Tím je zaručeno, že aplikace implicitně přijímá všechna data.

promisc : umožňuje přepnout adaptér do promiskuitního režimu. Během normálního provozu adaptér ze sítě přijímá pouze rámce jemu určené. Rámce určené jiným účastníkům jsou ignorovány. Pokud je adaptér v promiskuitním režimu, přijímá všechny rámce. Toto však platí pouze pro sítě bez switchů (přepínačů), které rámce směrují cílovým účastníkům.

to_ms : udává timeout pro čtení v milisekundách. Funkce čtoucí data z adaptéru (například `pcap_dispatch()` nebo `pcap_next_ex()`) se vždy navrátí do to_ms milisekund, i když neobdrží žádná data. Nastavením parametru to_ms na hodnotu 0 způsobí, že funkce budou na data čekat tak dlouho, dokud nějaká neobdrží. Hodnota -1 naopak způsobí vždy okamžitý návrat funkcí.

```
/* Open the device */
if ( (adhandle= pcap_open_live(
        d->name,                    // name of the device
        65536,                      // portion of the packet to capture
        PCAP_OPENFLAG_PROMISCUOUS,  // promiscuous mode
        100,                       // read timeout
        errbuf                      // error buffer
    ) ) == NULL)
{
    fprintf(stderr,"\nUnable to open the adapter. ");
    /* Free the device list */
    pcap_freealldevs(alldevs);
    return -1;
}
```

Po otevření adaptéru může začít vlastní odchytávání rámců. Existuje více způsobů, pro naše potřeby se však nejvíce hodí funkce `pcap_dispatch()`, která po zavolání čeká na příjem paketů maximálně dobu omezenou parametrem *to_ms*. Omezen je i počet paketů, které funkce převezme při jednom zavolání. Při dosažení tohoto počtu ještě před nastaveným časovým limitem dojde k návratu funkce. Tato funkce se stará pouze o příjem rámců, ty jsou dále předávány ke zpracování prostřednictvím uživatelské callback funkce, která je uvedena jako parametr funkce `pcap_dispatch()`. Callback funkce je vždy volána pro každý přijatý rámec zvlášť.

```
/* start the capture */
pcap_dispatch (adhandle, 10, packet_handler, NULL);

/* Callback function invoked by libpcap for every incoming packet */
void packet_handler(u_char *param, const struct pcap_pkthdr *header,
const u_char *pkt_data) {
struct tm *ltime;
char timestr[16];

/* convert the timestamp to readable format */
ltime=localtime(&header->ts.tv_sec);
strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);

printf("%s,%.6d len:%d\n", timestr, header->ts.tv_usec, header->len);
```

2.3.4 Filtrování provozu

Mezi nejvýznamnější vlastnosti knihovny WinPCap patří právě možnost nastavení filtru rámců, které působí již na úrovni ovladače v jádru operačního systému. Tento způsob poskytuje výkonný nástroj, který umožňuje efektivní odchytávání rámců. Při dobrém nastavení filtru

jsou potom aplikaci předávány pouze ty rámce, které jí opravdu zajímají. Nedochází tedy ke zbytečnému zatěžování systému zpracováváním nežádoucích dat.

Nastavení filtru se provádí po otevření síťového adaptéru ještě před spuštěním odchytávání. Výběr rámců se provádí v textové podobě pomocí definovaných označení protokolů a některých významných částí rámců, které lze různě kombinovat pomocí logických operátorů a závorek. Pokud je textová podoba filtru syntakticky správná, je filtr pomocí funkce `pcap_compile()` přeložen do binární podoby. Následně je funkcí `pcap_setfilter()` přiřazen otevřenému síťovému adaptéru.

```
1  //compile the filter
2  if (pcap_compile(adhandle, &fcode, "ip and tcp", 1, netmask) < 0)
3  {
4      fprintf(stderr,"\nUnable to compile the packet filter.");
5      return -1;
6  }
7
8  //set the filter
9  if (pcap_setfilter(adhandle, &fcode) < 0)
10 {
11     fprintf(stderr,"\nError setting the filter.\n");
12     return -1;
13 }
```

Podrobný popis vytváření filtrů je uveden v dokumentaci knihovny [10] v kapitole "Filtering expression syntax". Zde si uvedeme pouze základní pravidla a důležitá klíčová slova používaná v této práci.

2.3.5 Interpretace rámců

Registrovaná callback funkce pro zpracování rámců přebírá rámec v podobě odkazu na hlavičku záznamu a odkazu na čistá data rámce. Hlavička obsahuje časovou známku, přidělenou ovladačem v době přijetí rámce. Dále obsahuje informace o délce rámce, tedy i o délce datové oblasti. Data rámce jsou uložena v souvislé paměťové oblasti, začínající na adrese, na kterou ukazuje uvedený ukazatel. Požadované části rámce lze z dat získat pomocí ukazatele a známé pozice části v rámci. Data Ethernetových rámců neobsahují synchronizační pole,označení začátku rámce ani kontrolní součet. Tyto části jsou zpracovávány na nižší úrovni, než na které působí ovladač knihovny.

2.3.6 Ukládání a načítání přijatých paketů

Součástí knihovny jsou i funkce pro ukládání a načítání dat rámců do souboru. Tyto funkce zavádějí standard formátu těchto souborů, který je společný pro všechny aplikace využívající tuto rozšířenou knihovnu. Ve vlastní nově vytvořené aplikaci lze tedy bez problémů zpracovávat data uložená např. používaným programem Ethereal a naopak.

Načítání dat rámců ze souboru je velmi podobné odchytávání přímo ze síťového adaptéru. Pro otevření souboru slouží funkce `pcap_dump_open()`. Další postup je již totožný. Pomocí funkce pcap_dispatch() se data načítají a zpracovávají stejnou uživatelskou callback funkcí jako aktuální rámce.

Ukládání rámců do souboru je po otevření souboru prováděno postupným voláním funkce `pcap_dump()` pro zápis dat jednoho rámce.

2.4 Knihovna Qt

Qt je knihovna C++ pro tvorbu GUI aplikací nezávislých na platformě. Aplikace jsou portovatelné na operační systémy Microsoft Windows, Mac OS X, Linux, všechny hlavní komerční verze Unixu a embedded Linux. To je také hlavní důvod, proč byla tato knihovna vybrána jako základ vyvíjené aplikace IO-Controlleru. Qt je plně objektově orientovaná. Rozsah komponent, které knihovna nabízí, pokrývá nejen potřeby pro tvorbu grafického rozhraní, ale i práce s XML a vlákny.

V této kapitole jsou uvedeny některé důležité vlastnosti knihovny a dále jsou popsány vybrané komponenty, které byly použity ve vyvíjené aplikaci. Informace zde uvedené jsou platné pro knihovnu verze Qt 4.1, která také byla použita pro vývoj aplikace.

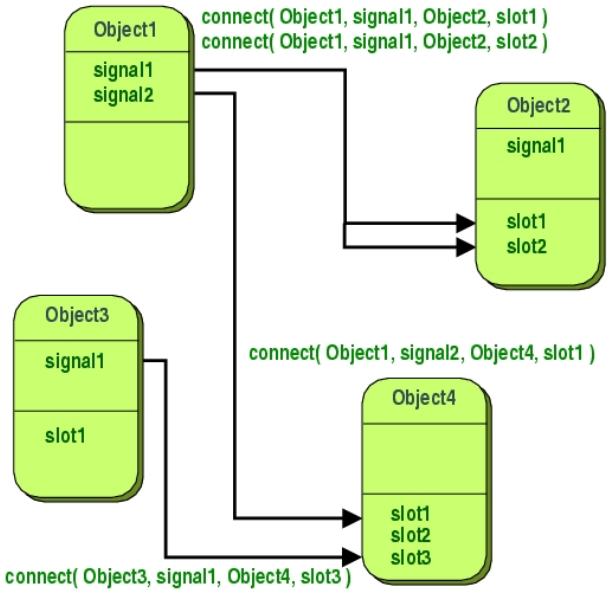
2.4.1 Signály a sloty

Signály a sloty slouží ke komunikaci mezi objekty. Tato vlastnost je základním mechanismem všech komponent knihovny. Všichni potomci(přímý i nepřímý) třídy QObject, která je předkem všech komponent knihovny, mohou využívat signály a sloty. Signály jsou vysílány objekty, při změně jejich stavu, jenž může být zajímavý pro "okolní svět". Vysílající objekty neví o tom, zda nějaký jiný objekt či objekty přijímají jimi vyslaný signál. Tento fakt zajišťuje to, že objekty mohou být použity jako programové komponenty.

Signály mohou vysílat pouze třídy (a jejich podtřídy) ve kterých jsou definovány. Signály mohou slotům předávat parametry, nesmí však mít návratovou hodnotu. Pro univerzální použití objektů by neměly být parametry signálů speciálních typů, aby mohly být spojovány se sloty objektů různých tříd. Toto však není nutná podmínka pro správnou funkci mechanizmu.

Slot je jen jiný název pro handler signálu. Jsou to obyčejné C++ funkce a mohou být stejně jako funkce volány. Jejich speciální vlastností je, že k nim může být připojen signál. Parametry slotů nemohou mít defaultní hodnoty. Sloty jako běžné členské funkce, mají též přístupová práva, určující, kdo se k nim může připojit. Aby bylo možné propojit slot s určitým signálem, musí se typy jejich parametrů shodovat. Počet parametrů slotu může být menší, pokud některé parametry nejsou potřeba.

V příkladu uvedeném níže je ukázáno použití uvedeného způsobu komunikace. Propojení signálů se sloty se provádí prostřednictvím metody `connect()` objektu, který je potomkem třídy QObject. Správu nad prováděním komunikačního mechanismu signálů a slotů provádí za chodu programu objekt třídy QMetaObject, který je součástí každého objekt třídy QObject, tedy i jeho potomků. QMetaObject se stará pouze o spojení signálů a slotů, která jsou zaregistrována u jeho objektu třídy odvozených z QObject.



Obrázek 2.7: Komunikace mezi objekty pomocí signálů a slotů (Zdroj:[8])

```
1
2   //Deklarace třídy potomka QObject
3   #include <QObject>
4
5   class Counter : public QObject
6   {
7       Q_OBJECT
8
9   public:
10      Counter() { m_value = 0; }
11
12      int value() const { return m_value; }
13
14  public slots:
15      void setValue(int value);  //Definice slotu
16
17  signals:
18      void valueChanged(int newValue);  //Definice signálu
19
20  private:
21      int m_value;
22  };
23
24  //Implementace metody setValue, která emituje signál o změně hodnoty
25  void Counter::setValue(int value)
```

```
26  {
27      if (value != m_value) {
28          m_value = value;
29          emit valueChanged(value); //Vyslání signálu
30      }
31  }
32
33  //Příklad komunikace prostřednictvím signálů a slotů
34  void main()
35  {
36  Counter a, b;
37  //Propojení signálu valueChanged objektu a se slotem setValue objektu b
38  QObject::connect(&a, SIGNAL(valueChanged(int)),
39                   &b, SLOT(setValue(int)));
40
41  a.setValue(12);     // a.value() == 12, b.value() == 12
42  b.setValue(48);     // a.value() == 12, b.value() == 48
43  }
44
```

V aplikaci IO-Controlleru jsou sloty a signály využívány nejen ke komunikaci s komponentami knihovny, mimo to tvoří významnou roli v předávání dat mezi nově vytvořenými objekty. Využívají se například k předávání textových hlášení objektu třídy CReport, který zajišťuje výpis událostí v programu uživateli. Použity jsou i při předávání nově příchozích rámců ke zpracovávání.

2.4.2 Správa paměti

Qt používá velmi jednoduchý, nicméně dobře fungující model správy paměti. Objekty tříd odvozených z QObject jsou organizovány do stromových struktur. Při vytvoření nového objektu se zadává jeho rodič jako parametr konstruktoru. Když je 0, vytvoří se kořen nového stromu. Při zrušení objektu se Qt postará o automatické zrušení všech jeho potomků ve stromu objektů. Ve třídě QObject jsou definovány metody pro manipulaci se stromy objektů. Metoda objectTrees vrací seznam kořenů všech existujících stromů. Rodičovský a synovské objekty se dají zjistit voláním metod parent a children. Ke změně stromu slouží metody insertChild a removeChild.

Widgety, tj. objekty odvozené od základní třídy grafický objektů QWidget navíc vytvářejí stromy widgetů. Ty určují, jak budou jednotlivé widgety do sebe vnořeny na obrazovce. Kořeny stromů widgetů jsou top-level okna. Obvykle strom widgetů koresponduje se stromem objektů, takže při zrušení widgetu se automaticky zruší všechny widgety, které jsou v něm vizuálně obsaženy. Při vytvoření objektu se rodič zadaný jako parametr konstruktoru použije pro zapojení jak do stromu objektů, tak i do stromu widgetů. Změny ve stromě widgetů je možné provádět metodou QWidget::reparent. Je třeba dávat si pozor na to, že přesun widgetu v rámci stromů objektů nemění pozici widgetu ve stromě widgetů. Pokud má být zachována korespondence obou struktur, je nutné spolu s QObject::insertChild nebo QObject::removeChild volat také

QWidget::reparent.

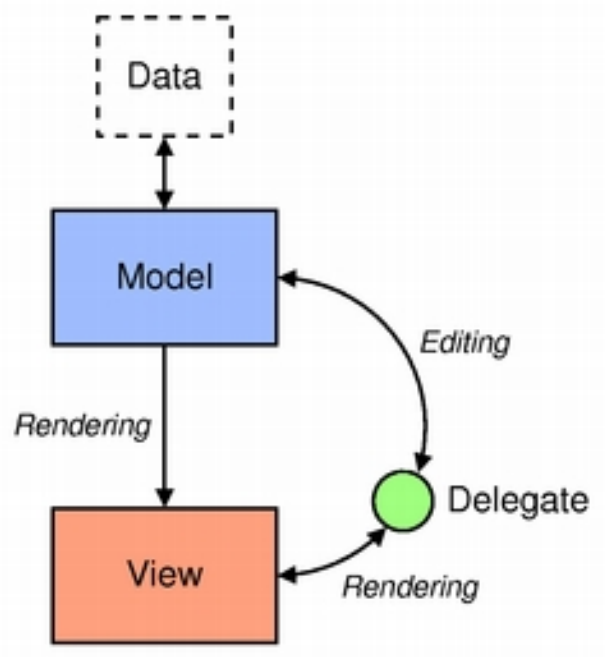
Při zrušení objektu mohou na různých místech programu zbýt ukazatele na již neexistující objekt. Jejich dereferencování typicky způsobí havárii programu. Qt proto definuje šablonu QPointer, což je ukazatel, který se automaticky nastaví na 0 při zrušení objektu, na nějž ukazuje.

2.4.3 Způsob zobrazování datových položek

Qt používá pro zobrazování datových položek speciální architekturu model/view (model/pohled), která odděluje uložení dat od způsobu jejich zobrazení. Tento přístup poskytuje velkou flexibilitu při úpravě zobrazování datových položek. Vzhledem k tomu, že zobrazování datových položek v různých podobách tvoří významnou část aplikace IO-Controlleru, je v následujícím textu tento přístup vysvětlen.

Architektura model/view

Tento přístup k zobrazování dat v uživatelských rozhraních je založen na třech skupinách komponent: modelech, pohledech a delegátech (viz obr. 2.4.3). Model je ve spojení se zdrojem dat a tvoří jeho rozhraní pro ostatní komponenty. Podstata výměny dat mezi modelem a zdrojem závisí na způsobu, jak jsou naimplementovány metody modelu.



Obrázek 2.8: Architektura model/pohled (Zdroj:[8])

Jednotlivé datové položky modelu jsou označovány indexy. Toto označení je společné pro všechny komponenty a slouží k jednoznačné identifikaci a rozdělení dat. Pohled vždy ve svém požadavku posílá index a dostává od modelu data. Indexu také přísluší určitá pozice v zobrazení pohledu (tabulce,seznamu či stromu). Ve standardní pohledech delegáti poskytují editory pro úpravu datové položky určené indexem.

Všechny tyto komponenty jsou definované jako abstraktní třídy, které slouží k odvození

nových tříd, odpovídajících jejich účelu. Uvedené komponenty mezi sebou komunikují prostřednictvím signálů a slotů:

- Signály z modelu informují pohled o změnách jemu příslušejících dat z datového zdroje
- Signály z pohledu poskytují informace o zásahu uživatele do zobrazených dat
- Signály z delegáta jsou používány v době editace datové položky a informují model a pohled o stavu editoru

2.4.4 Popis tříd použitých komponent

QString

Při předávání řetězce typu char* funkci musí být jasné, zda si funkce udělá kopii řetězce, nebo ne. A když ne, zda funkce řetězec dealokuje, nebo zda ho má dealokovat volající. Dalším problémem při používání řetězců je, že často je potřeba předávat parametry funkcí hodnotou, aby následně funkce i volající mohli datovou strukturu měnit a změny provedené uvnitř funkce se neprojevily vně a naopak. Qt odstraňuje oba problémy pomocí datových struktur s implicitně sdílenými daty. Objekt QString je ve skutečnosti velmi malá struktura, která obsahuje ukazatel na data uložená v samostatně alokovaném bloku paměti. Používá se zde počítání referencí a copy-on-write[1]. V objektu QString je poznamenáno, kolik objektů ho používá. Když tento počet klesne na 0, je objekt uvolněn z paměti. Jestliže je počet odkazů větší než jedna a některý objekt chce data změnit, nejprve si automaticky vytvoří privátní kopii a změna dat se provede v ní. V Qt je tedy možné předávat řetězce (typu QString) hodnotou bez obav, že bude program zpomalen zbytečným kopírováním.

Třída QString dále nabízí operátory pro jednoduché slučování řetězců a jejich porovnávání. Umožňuje také vkládat do textu číselné údaje v různých číselných soustavách.

```
1 QString status = QString("Processing file")+
2                  QString(" %1 of %2: %3")
3                       .arg(i)         // current file's number
4                       .arg(total)     // number of files to process
5                       .arg(fileName); // current file's name
```

QString je využíván všemi komponentami knihovny, které pracují s textem. Zejména grafickými komponentami, které zobrazují textové údaje. V uživatelsky definovaných objektech aplikace IO-Controlleru představují objekty této třídy většinu textových proměnných. Klasické řetězce tvořené poly znaků jsou použity jen v případech, kdy to vyžadují systémové funkce, či funkce jiných knihoven.

[1] Princip Copy-on-write spočívá v tom, že v okamžiku, kdy je vydán příkaz k pořízení kopie dat, se ve skutečnosti fyzická kopie nevytvoří, a aplikaci je předán jiný odkaz na již existující data. Skutečná kopie je vytvořena teprve ve chvíli, kdy jedna z aplikacích sdílejících společnou kopii vydá pokyn k zápisu dat.

QThread

Třída QThread poskytuje vlákna, nezávislá na platformě. Pro použití vlákna je nutné odvodit od QThread novou třídu a v ní naimplementovat čistě virtuální metodu `run()`. Kód obsažený v této metodě bude spuštěn ve zvláštním vláknu procesu. Pro spuštění vlákna slouží metoda `start()`. Provádění vlákna končí po návratu z metody `run()`. O spuštění a ukončení vlákna informují objekty této třídy okolní objekty signály `started()` a `finished()`. Pro ukončení vlákna zvenčí slouží metoda `terminate()`, ta je však nebezpečná a měly by být použita pouze ve výmečných případech. Může totiž přerušit kód vlákna v libovolném místě, například při modifikaci dat, a tím uvést program do nedefinovaného stavu. Každému vláknu lze při jeho spuštění metodou `start()` nebo za chodu metodou `setPriority()` nastavit prioritu.

Aplikace IO-Controlleru využívá vlákna knihovny Qt pro příjem rámců ze síťového rozhraní, dále pak pro časově náročná zpracování dat, jako je například načítání dat ze souboru.

QMutex

Synchronizační objekty třídy QMutex, dále jen mutex, se používají pro zabránění nebezpečnému souběhu vláken. Mutex je speciální typ zámku, který v daný okamžik může být uzamčen pouze jediným vláknem. Pokud jedno vlákno mutex uzamkne a pak se druhé vlákno pokusí také mutex uzamknout, toto druhé vlákno se zablokuje. Teprve až první vlákno odemkne mutex, druhé se odblokuje a může dále pokračovat v činnosti. Systémem je garantováno, že nemůže dojít k současnému uzamčení mutexu více vlákny. Vždy pouze jedno vlákno dostane možnost mutex uzamknout a ostatní budou zablokována.

Pro použití mutexu se vytvoří instance třídy QMutex. V metodě či funkci, kde je potřeba zabránit souběžnému volání části kódu více vlákny je na začátku této oblasti zavolána metoda mutexu s názvem `lock()`. Tato metoda uzamkne následující část kódu funkce či metody pro vlákno, které zavolalo metodu `lock()` jako první. Následující vlákna se při volání `lock()` zastaví, dokud není oblast odemčená vláknem, které ji předtím uzamklo. Pro odemčení zabezpečené části kódu se na jejím konci zavolá metoda mutexu `unlock()`.

```
1  void method2()
2  {
3      mutex.lock();
4      number *= 3;
5      number /= 2;
6      mutex.unlock();
7  }
```

Při použití popsaného způsobu zamykání kritických částí kódu je nutné hlídat, aby se funkce náhle nenavrátila uvnitř kritické části bez předchozího zavolání metody `unlock()`. Potom by totiž mohla tato část kódu zůstat trvale zamčena jedním vláknem, což by nejspíše způsobilo zablokování celého programu. Knihovna Qt nabízí způsob, jak se tomuto problému vyhnout. Řešením je třída QMutexLocker, která ve svém konstruktoru provede zamčení mutexu, jehož ukazatel je parametrem tohoto konstruktoru. Destruktor třídy QMutexLocker provede uvolnění dříve zamčeného mutexu. Jak je uvedeno v následujícím příkladu, je nutné pro správné využití této třídy vytvořit její instanci jako lokální proměnou v rozsahu(scope), který chceme zabezpečit.

```
1  void method2()
2  {
3      QMutexLocker locker(&mutex);
4      number *= 3;
5      number /= 2;
6  }
```

Pomocí mutexů jsou v aplikaci IO-Controlleru ošetřeny všechny funkce, které mohou být volány ve vláknech. To znamená i ty, které jsou volány v callback funkcích IO-Controlleru, registrovaných v knihovně IO Base. Každá callback funkce IO-Controlleru je totiž volána ve zvláštním vláknu.

QWidget

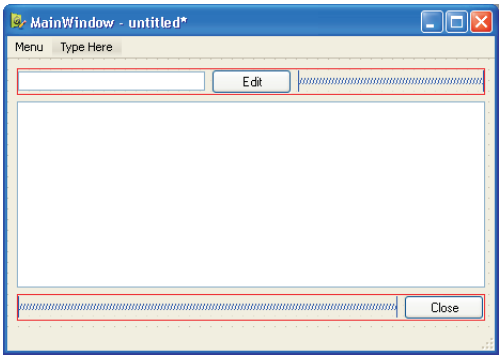
Třída QWidget je základní třídou všech grafických objektů uživatelského rozhraní. Objekty třídy QWidget a od ní odvozených tříd jsou do sebe vnořovány přesně tak, jako je tomu v grafické podobě uživatelského rozhraní. Základním objektem je objekt třídy QMainWindow nebo QDialog, které tvoří hlavní okno s příslušnými funkcemi pro změnu velikosti a ukončení okna. Do nich jsou dále umisťovány další potomci třídy QWidget v podobě tlačítek, popisů, tabulek a jiných běžně pooužívaných grafických prvků. Příklad takovéhoto formuláře je uveden na obrázku obr. 2.9. Výsledný formulář je tedy tvořen stromem objektů tříd odvozených od QWidget.

Uživatelské rozhraní lze vytvářet dynamicky za chodu programu nebo pomocí nástroje zvaného Designer, který umožňuje vytvářet grafickou podobu rozhraní již v době návrhu. Výsledný návrh obsahu hlavního okna je uložen do souboru *.ui v podobě XML dokumetu, který je při kompilaci projektu pomocí prekompileru *qmake* převeden na hlavičkový soubor C++, obsahující deklarace všech objektů formuláře, jejich nastavení a to vše v definici jedné třídy. Název hlavičkového souboru grafického rozhraní je odvozen od názvu souboru *.ui a to tak, že pokud máme soubor [*jméno*].ui bude se jeho hlavičkový soubor jmenovat ui_[*jméno*].h. Následně je potřeba deklarovat objekt této třídy, například v definici uživatelské třídy hlavního okna, odvozené od QMainWindow,QDialog nebo čistě QWidget. V konstruktoru této třídy se provede zavolání metody `setupUi()` objektu grafického rozhraní, kterému se touto cestou předá ukazatel na objekt hlavního okna. V Designeru lze propojovat i signály a sloty jednotlivých grafických komponent.

Skupiny grafických komponent lze združovat do tzv. layoutů, ve kterých jsou podle typu layoutu komponenty uspořádány pod sebou, vedle sebe nebo v mřížce. V obrázku obr. 2.9 jsou tyto layouty zobrazeny červeně.

QTableView

Tato komponenta představuje jedno z možných řešení pohledu architektury model/view. Jedná se o komponentu, která umožňuje zobrazení dat v klasické tabulce. Data jsou rozdělena do sloupců a řádků, jejichž pořadová čísla tvoří index datových položek modelu dat. QTableView se používá ve spojení s komponentou QAbstractTableModel, která představuje model dat, vhodný



Obrázek 2.9: Příklad navrženého grafického rozhraní v Qt Designeru

právě pro zobrazení dat v tabulce. Objekt představující tento model se připojí ke komponentě QTableView popocí její metody `setModel()`. Jeho metoda `data()` provádí přiřazení položek ze zdroje dat indexům, které požaduje komponenta QTableView, či jiné objekty. Další metody modelu poskytují QTableView např. názvy sloupců a jejich počet, dále pak počet řádků. V případě, že jsou data za chodu chodu programu měněna, musí se v modelu naimplementovat ještě metody pro vkládání a odebírání řádků a pro úpravu dat položky.

V aplikaci IO-Controlleru jsou tabulky asi nejpoužívanějšími komponentami. Slouží jak k nastavování parametrů např. IO-Device či ke zobrazení přijatých rámců a výpisu hlášení o událostech.

QTreeView

Na rozdíl o předchozí komponenty, slouží tato ke zobrazování dat ve stromové struktuře. Index datové položky, používaný pro tento pohled, obsahuje navíc kromě čísla řádku a sloupce ještě ukazatel na svého předka ve stromě. Tato komponenta se používá dohromady s modelem QAbstractItemModel, který je obecnějším předkem modelu QAbstractTableModel, používaného v předchozím v případě. Metody modelu se od předchozího liší pouze v přístupu k datům, odvozeném od odlišné prezentace indexu.

Aplikace IO-Controlleru využívá zobrazení dat ve stromové struktuře k jedinému účelu a to je zobrazení rozloženého obsahu rámců.

QTextBrowser

Tato komponenta představuje jednoduchý prohlížeč textu. Její asi nejsilnější funkcí je možnost prezentace textu v html formátu, což umožňuje vypisovat data přehledným způsobem a využít všech možností tohoto standardu.

Tato komponenta slouží v realizované aplikaci k výpisu celého obsahu rámce v hexadecimální a ASCII podobě. Ještě je použita k výpisu výsledků analýzy rámců RTC komunikace.

2.4.5 Kompilace projektu

Vzhledem ktomu, že zdrojový kód programu, používajícího komponenty knihovny Qt obsahuje tzv. meta-object kód, který slouží k popisu speciálních funkcí knihovny, jako jsou například signály a sloty, je nutné před spuštěním kompilace kódu tyto nestandardní části

převést do podoby, ve které může být již zkompilován standardním kompilerem C++. V případě, že jsou uživatelská rozhraní aplikace vytvářena dopředu v nástroji Qt Designer, jsou uložena v podobě XML dokumentu a musí být před začleněním do kódu převedena na hlavičkové soubory. Oba tyto problémy řeší nástroj `qmake`, který navíc automaticky vytváří Makefile pro daný projekt.

Meta object compiler-MOC

MOC slouží pro „předkompilaci“ speciálních klíčových slov Qt knihovny. Vytvoří ze zadaného hlavičkového souboru, jenž obsahuje objekt, který je potomkem třídy QObject, zdrojový kód, provádějící inicializaci meta objektu a obsahující základní informace o objektu a jména všech signálů,slotů a ukazatele na tyto sloty resp. funkce. Tento zdrojový kód musí být zkompilován a slinkován společně s ostatními soubory v projektu. `qmake` nově vygenerované soubory se zdrojovým kódem přidá do výsledného souboru Makefile projektu.

Postup kompilace projektu

1. Nejprve se vytvoří projektový soubor, který obsahuje seznam souborů projektu a jeho parametry. V projektovém adresáři se spustí:

   ```
   qmake -project
   ```

2. Následuje vygenerování moc souborů a vytvoření souboru Makefile:

   ```
   qmake -makefile
   ```

3. Nyní je možné zkompilovat celý program:

   ```
   make
   ```

Kapitola 3

Realizace aplikace IO-Controlleru

3.1 Požadavky na funkčnost aplikace

Požadavky na aplikaci IO-Controlleru vycházejí z možností rozhraní IO Base knihovny PNIO, popsaném v kapitole 2.1. Tato knihovna má v sobě naimplementován softwarový zásobník Profinetu IO, určený pro standardní PC. Hlavním účelem aplikace je prezentace funkcí a možností Profinetu IO. Důraz je kladen na diagnostické a analytické schopnosti. Aplikace by měla také sloužit jako nástroj pro podporu při vývoji a testování nových zařízení typu IO-Device. PNIO i IO Base existují jak ve verzi pro Windows tak pro Linux. Tato skutečnost vedla na začátku této práce k rozhodnutí, realizovat celou aplikaci nezávislou na operačním systému tak, aby mohla být použita na obou uvedených systémech.

Pro aplikaci byly stanoveny následující požadavky:

- Realizovat plnohodnotný IO-Controller, umožňující:
  - zobrazovat přehled IO-Device patřících IO-Controlleru
  - číst a zapisovat I/O data
  - přijímat a zaznamenávat poruchová hlášení
  - číst a zapisovat libovolné datové záznamy IO-Device
- Integrovat do aplikace nástroj pro analýzu komunikace, umožňující tyto funkce:
  - zaznamenávání průběhu komunikace
  - důkladný rozbor zaznamenaných rámců
  - souhrnnou analýzu rámců cyklické komunikace

3.2 Návrh aplikace

Na základě stanovených požadavků byly zvoleny vhodné systémové prostředky, které umožnily aplikaci realizovat v dostatečné kvalitě a uživatelsky přijatelné podobě.

Jako základ aplikace byla zvolena knihovna Qt(viz kapitola 2.4), která nabízí základní prostředky pro tvorbu aplikací, nezávislých na platformě. Umožňuje realizovat nezávislá grafická

rozhraní ve standardní kvalitě. Q-toolkit, jak se knihovna také nazývá, je jednou ze dvou nejrozšířenějších knihoven, sloužících k tomuto účelu. Na rozdíl od svého konkurenta, knihovny GTK+, má Q-toolkit několik zásadních vlastností, které ji odlišují. Jedná se zejména o mechanismus komunikace mezi objekty pomocí signálů a slotů(viz kapitola 2.4.1), či způsob zobrazování datových položek prostřednictvím architektury model/view (viz kapitola 2.4.3). Toto byly hlavní důvody, pro které byla vybrána.

Jelikož rozhraní IO Base nabízí přístup k Profinetu IO na mnohem vyšší úrovni, než na které dochází ke zpracování čistých ethernetových rámců, které probíhá uvnitř použité knihovny PNIO, bylo nutné přistoupit k použití další knihovny, která by umožňovala přístup k těmto rámcům. Vybrána byla knihovna libpcap, primárně určená pro operační systém Linux. Existuje však i verze WinPCap, určená pro operační systém Windows. Podrobně je tato knihovna popsána v kapitole 2.3. Libpcap je ucelený nástroj pro odchytávání, filtraci a další zpracování rámců, snímaných z linkové vrstvy síťového zásobníku operačního systému. Použitím této knihovny je z pohledu zpracování rámců zachována kompatibilita realizovaného nástroje s jinými, běžně používanými nástroji, jako je například Ethereal. Formát souborů, použitých k ukládání dat rámců je totiž díky funkcím knihovny libpcap společný. Aplikace tedy umožňuje načítat soubory s rámci, které byly pořízeny v jiných nástrojích a ty pak dále zpracovávat. Naopak soubory rámců uložené v aplikaci IO-Controlleru mohou být zpracovávány v jiných programech.

3.3 Implementace funkcí IO-Controlleru

3.3.1 Struktura

Část aplikace, zabývající se funkcemi Profinetu IO je řešena prostřednictvím objektů, reprezentujících jednotlivé prvky tohoto standardu. Objekty tvoří hierarchickou strukturu od submodulu až po IO-Controller. Každý objekt poskytuje metody, odpovídající jeho úrovni v hierarchické struktuře. Všechny uvedené objekty sdílejí několik dalších objektů jiných tříd, zaměřených například na zaznamenání událostí a jejich výpis, či obraz procesních dat.

CIOController

Objekt třídy CIOController se v aplikaci vyskytuje pouze jednou. Představuje nejvyšší úroveň hierarchie objektů, určených pro řízení Profinetu IO. Jeho metody jsou úzce spojeny s funkcemi rozhraní IO Base. Kromě přístupu k I/O datům, které je řešeno na úrovni modulů a submodulů, má na starosti veškeré řízení přístupu k Profinetu IO. Vlastní několik metod, které jsou při startu IO-Controlleru prostřednictvím jejich statických wrapperů [1] zaregistrovány u IO Base jako callback funkce (viz kapitola 2.1.5). Pomocí těchto metod, které jsou volány rozhraním IO Base, je aplikace informována o interních událostech IO-Controlleru v knihovně PNIO. Tímto způsobem se aplikaci předávají poruchová hlášení, změny stavu IO-Controlleru a jeho IO-Device, případně výsledky požadavků na čtení a zápis datových záznamů. Dále tento

[1] Wrapper metody - takto se označuje statická metoda, která slouží pouze k tomu, aby pomocí ukazatele na objekt zavolala některou jeho dynamickou metodu. Tohoto způsobu se využívá při předávání ukazatelů na metody objektů.

objekt poskytuje metody pro spuštění IO-Controlleru, jeho vypnutí, případně změnu jeho stavu. Obsahuje informace o svých IO-Device v podobě seznamu objektů, které je v této aplikaci reprezentují. Ostatním objektů nabízí metody pro vyhledávání objektů IO-Device podle různých parametrů.

CIODevice

Hlavním úkolem objektů třídy CIODevice je uchovávání a poskytování parametrů příslušného IO-Device a informací o jeho struktuře. Objekt IO-Controlleru vlastní tolik objektů této třídy, kolik je uvedeno IO-Device v hardwarovém projektu Step 7. Objekt IO-Device obsahuje parametry, sloužící převážně k jeho identifikaci. Jedná se o jméno zařízení, jeho IP a MAC adresu, případně identifikační číslo z projektu Step 7. Při předávání informací od IO Base prostřednictvím callback funkcí slouží k identifikaci IO-Device adresa některého z modulů, který je v něm umístěn. Obsahem tohoto objektu je i seznam objektů, reprezentujících moduly, které byly v konfiguračním nástroji. Poskytována je i metoda pro aktivaci a deaktivaci příslušného IO-Device.

CModule

Objekty této třídy se nacházejí v každém objektu třídy CIODevice a reprezentují jednotlivé moduly tohoto zařízení, uvedené v hardwarové konfiguraci projektu. Z pohledu Profinetu IO se jedná o jednotlivé datové položky IO-Device, umístěné v jeho substotech. V použitém systému firmy Siemens jsou jim však přiřazovány nové logické adresy, které již nemají z pohledu uživatele nic společného s původními adresami Profinetu IO - slot/subslot. Objekt třídy CModule obsahuje data parametrů příslušného modulu. Jedná se o logickou adresu, pocházející z projektu Step 7. Logická adresa se skládá z označení typu dat (vstupní/výstupní) a jejich pozice v adresním prostoru. Dalším parametrem je délka dat, příslušejících modulu na dané adrese. Posledním důležitým údajem je stav daného modulu. Tedy pro vstupní moduly se jedná o stav poskytovatele dat (IOPS), pro výstupní je to stav konzumenta dat (IOCS). Objekty této třídy poskytují metody pro přístup k rozhraní IO Base za účelem aktualizace I/O dat příslušejících danému modulu. Objekt třídy CModule by měl obsahovat minimálně jeden objekt třídy CSubmodule, aby bylo možné zobrazovat v aplikaci jeho data (viz níže).

CSubmodule

Úkolem objektů třídy CSubmodule je prezentace I/O dat modulů uživateli. Slouží k rozdělení dat modulu na datové položky konkrétního typu. Pokud by se například jednalo o modul se čtyřmi binárními signály, budou data modulu tvořena jedním bytem, ve kterém budou první čtyři bity reprezentovat příslušné kanály modulu. Přiřazením objektů třídy CSubmodule objektu reprezentujícímu tento modulu je možné z uživatelského pohledu data rozdělit na uvedené bity a dále k ním takto přistupovat. Kromě uvedeného logického datového typu jsou podporovány všechny číselné datové typy Profinetu IO.

CIOData

V aplikaci se nachází jedna instance třídy CIOData, které představuje obraz procesních dat vstupů a výstupů. Tento objekt pokrývá celý adresní prostor vstupů a výstupů, který

má velikost 16383 bytů pro každý směr. Přístup k těmto datům mají všechny objekty výše uvedených tříd. Pro zjednodušení práce s daty obrazu jsou poskytovány metody pro čtení a zápis všech podporovaných datových typů. V případě, že je uživatelem ponecháno nastavení cyklické aktualizace I/O dat, jsou data v tomto objektu aktuální.

CReport

Tato třída definuje objekt, který v aplikaci souží k výpisu událostí, které nastaly za chodu programu. Jedná se například o spuštění IO-Controlleru, přijetí poruchového hlášení atd. Třída CReport obsahuje slot, sloužící k přidání události do seznamu, který spravuje. Ostatní objekty aplikace se na tento slot připojují svými signály a předávají tak svá hlášení. Kromě textu hlášení je předáván i jeho typ, případně ještě ukazatel na objekt IODevice, se kterým hlášení souvisí. Při přijetí požadavku na přidání hlášení je událost doplněna o časovou známku a přidána do seznamu, který je zobrazen uživateli. Typ hlášení určuje barvu, kterou je vypisován text. Lze tak vizuálně odlišit např. poruchová hlášení od ostatních. Hlášení jsou na základě jeji typu propojována se zaznamenanými rámci. Rámce související s příslušným hlášením lze zobrazit samostatně.

3.3.2 Spuštění IO-Controlleru

Spuštění IO-Controlleru provádí uživatel stisknutím příslušného tlačítka v aplikaci. Tím dojde k zavolání metody `Run()` objektu třídy CIOController. Tato metoda zavolá všechny potřebné funkce, popsané v kapitole 2.1.2. Zaregistruje aplikaci a pomocí wrapperů také metody objektu jako callback funkce. Dále nastaví handler určený pro spojení s IO Base na hodnotu, kterou obdrží. Potom aplikace čeká na potvrzení změny režimu IO-Controlleru do stavu OPERATE. Následně je spuštěna cyklická aktualizace I/O dat.

3.3.3 Čtení a zápis I/O dat

Cyklická aktualizace vstupních a výstupních dat se provádí pro jednotlivé moduly, které jsou zadány v konfiguraci programu. Při aktualizaci se využívá hierarchické struktury objektů. Jednou za periody použitého časovače se zavolají metody objektu IOController `UpdateInputData()` a `UpdateOutputData()`, které provedou volání metod stejných jmén všech objektů IODevice, které má IOController zadány ve svém seznamu. Objekty IODevice zase provedou volání těchto metod u svých objektů třídy CModule, které již volají funkce IO Base pro přímí přístup k I/O datům knihovny PNIO. Funkce IO Base pro čtení a zápis I/O dat poskytují také stavy modulů (IOPS,IOCS), ty jsou zaznamenávány a následně zobrazovány uživateli.

Nastavování výstupních dat probíhá zásahem uživatele, po kterém se změna zapíše na příslušnou adresu do dat objektu IOData. Při změně stavu vstupních dat jsou pomocí signálu informovány příslušné komponenty grafického rozhraní, které se starají o zobrazení stavu dat.

3.3.4 Zpracování poruchových hlášení

Poruchová hlášení jsou přijímána objektem IOController prostřednictvím callback funkce, v tomto případě metody, která byla pro tento účel zaregistrována u rozhraní IO Base. Po zavolání této metody (rozhraním IO Base) jsou zpracována data z datové struktury poruchy,

na kterou ukazuje parametr metody. Nejprve je určeno IO-Device, kterému tato porucha patří. Tato identifikace se provádí pomocí logické adresy, uvedené v datech. Tato adresa se shoduje s logickou adresou některého z modulů, obsažených v hledaném IO-Device. Pro vyhledání IO-Device je použita metoda `FindDevice()`, které se tato adresa předá. Následně jsou informace o poruše spolu s určením nalezeného IO-Device předána objektu třídy CReport, který poruchu zaznamená a zobrazí uživateli.

3.3.5 Čtení a zápis datových záznamů

Čtení datových záznamů se provádí na povel uživatele ve zvláštním okně aplikace, kde se také přijatá data zobrazují. Pro vyslání žádosti o čtení datového záznamu slouží zvláštní funkce IO Base, nazývaná `PNIO_rec_ read_ req()`. Parametry této funkce kromě handleru pro komunikaci s IO Base, který se používá ve všech funkcích tohoto rozhraní, jsou logická adresa modulu, ze kterého se bude datový záznam číst. Dále pak index požadovaného záznamu, referenční číslo žádosti, které bude uvedeno i v odpovědi a maximální délka dat, která mohou být přijata. Odpověď na tuto žádost je doručena prostřednictvím callback funkce, reprezentované příslušnou metodou objektu IOController. Při přijetí odpovědi, která obsahuje ukazatel na požadovaná dat, jsou tyto data přepsána do vyhrazeného bufferu, ze kterého jsou následně vypsána uživateli.

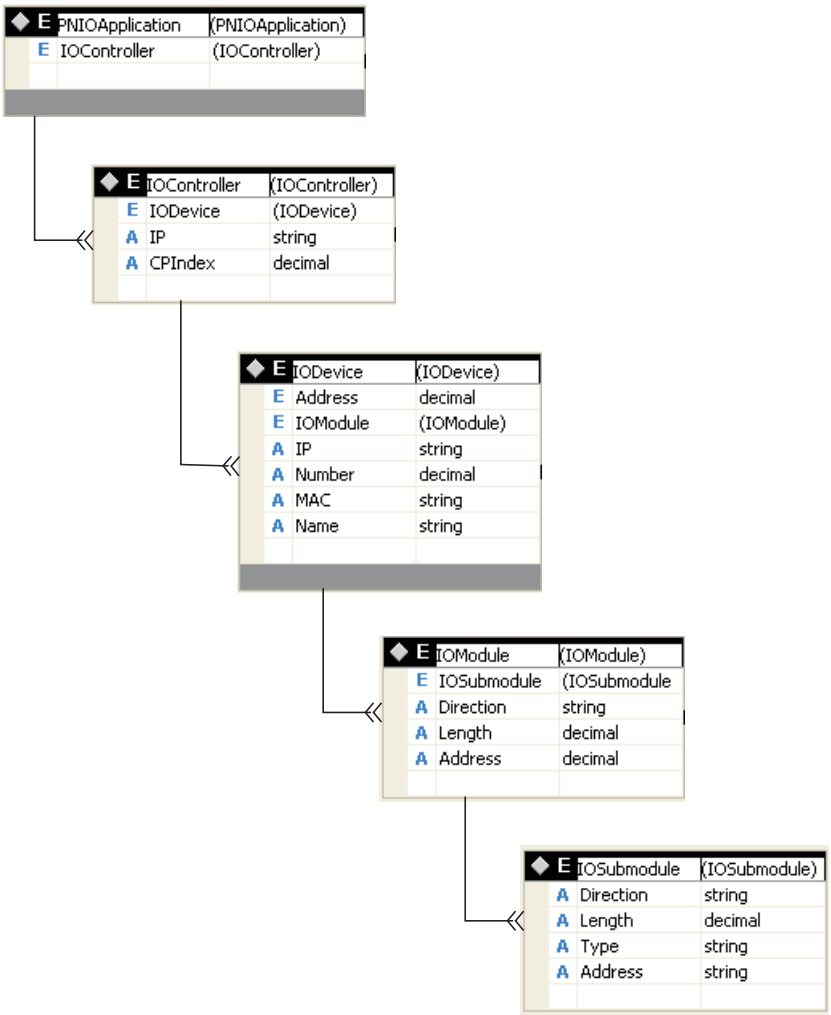
Zápis datových záznamů také probíhá ve zvláštním okně aplikace. K tomuto účelu slouží funkce `PNIO_rec_write_req()`, které se oproti funkci pro čtení datových záznamů předává ještě ukazatel na data, která mají být zapsána. Parametr délka dat má v tomto případě význam skutečné velikosti dat, uložených v předávané oblasti. Výsledek operace zápisu dat je aplikaci předán pomocí příslušné callback funkce, taktéž reprezentované metodou objektu IOController.

3.3.6 Konfigurační soubor

Popis IO systému, zahrnující všechny IO-Device, přidané do konfigurace systému spolu se všemi jejich moduly a submoduly a jejich parametry lze uložit do konfiguračního souboru. Z tohoto souboru lze při příštím spuštění programu znovu načíst celou konfiguraci systému. Během načítání se vytvoří všechny potřebné objekty a nastaví se jejich parametry. Konfigurační soubor je uložen v podobě XML dokumentu, jehož schéma je uvedeno na obrázku obr. 3.1.

3.4 Implementace analýzy komunikace

Analýza komunikace je založena na rozboru rámců, zaznamenaných pomocí knihovny libpcap resp. WinPCap. Pomocí filtru knihovny jsou přesně definovány rámce, které mají být předány aplikaci IO-Controlleru. Přijaté rámce jsou ukládány a případně ještě propojovány s jednotlivými hlášeními událostí. Pro prohlížení a rozbor obsahu rámců slouží nástroj, který kromě výpisu čistých dat rámce rozebírá jeho obsah na jednotlivé položky, které pak zobrazuje ve stromové struktuře i s jejich popisem. Pro analýzu rámců cyklické real-time komunikace Profinetu IO je určen speciální nástroj, který umožňuje zobrazit souhrnné informace o obsahu velkého množství těchto rámců.



Obrázek 3.1: XML schéma konfiguračního souboru aplikace)

3.4.1 Záznam rámců

Před spuštěním záznamu je uživatelem vybrán síťový adaptér, ze kterého budou data sbírána. Adaptér může být tedy zvolen nezávisle na tom, který je používán pro IO-Controller. V případě použití aplikace se standardním síťovým rozhraním je však nejlepším řešením zvolit stejný adaptér. Naopak při použití speciální komunikační karty CP 1616, která není v operačním systému zaregistrována jako běžné síťové rozhraní a tudíž ani není možné použít knihovnu libpcap pro snímání jejího provozu, je jediným možným způsobem použít pro analýzu komunikace jiný síťový adaptér. Ten by měl být připojen do stejné sítě jako CP 1616 pomocí zařízení pro odposlech ethernetové komunikace nebo by měl být zapojen do portu switche, který je zrcadlen s tím, ke kterému je připojen IO-Controller. Tuto funkci však neumožňují všechny switche.

Rámce jsou zaznamenávány pomocí funkce `pcap_dispatch()`, která je cyklicky spouštěna ve zvláštním vláknu aplikace. Při volání této funkce se knihovně předává ukazatel na callback funkci, která přijme a zpracuje data rámců. V callback funkci se od rámců oddělují ty, které obsahují data cyklické real-time komunikace (RTC), která je z pohledu záznamu rámců náročnější, jelikož perioda jejich vysílání je velmi krátká (až jednotky milisekund na jedno IO-Device). Tyto

rámce se na rozdíl od ostatních neuchovávají v operační paměti, ale jsou ukládány do zvláštního souboru, který je automaticky vytvořen při spuštění záznamu. Ostatní rámce, které jsou po dobu záznamu uchovávány v operační paměti, je možné ručně uložit do samostatného souboru.

3.4.2 Nastavení filtru

Knihovna libpcap poskytuje efektivní způsob filtrování rámců na úrovni jádra operačního systému. Tento filtr slouží k přesnému nastavení odchytávání rámců tak, aby byly vždy zaznamenávány pouze rámce, související s komunikací Profinet IO. Uživatel má možnost výběru záznamu konkrétního typu komunikace. Rozlišovány jsou tyto typy komunikace: RPC, RTA, RTC, DCP a ARP. Způsob sestavování filtrů je popsán v dokumentaci ke knihovně WinPCAp [10]. Níže jsou uvedeny filtry, které byly sestaveny a použity pro výběr jednotlivých typů rámců.

Filtr pro protokol RTA:

```
((ether proto 0x8892 and ether[14:2]>=0xfc00 and
  ether[14:2]<0xfe00) or (ether proto 0x8100 and ether[16:2]=0x8892
  and ether[18:2]>=0xfc00 and ether[18:2]<0xfe00))
```

Filtr pro protokol RTC:

```
((ether proto 0x8892 and ether[14:2]<0xfc00) or (ether proto 0x8100
  and ether[16:2]=0x8892  and ether[18:2]<0xfc00))
```

Filtr pro protokol DCP:

```
((ether proto 0x8892 and ether[14:2]>=0xfe00 and
  ether[14:2]<=0xfeff) or (ether proto 0x8100 and ether[16:2]=0x8892
  and ether[18:2]>=0xfe00 and ether[18:2]<=0xfeff))
```

Filtr pro odstranění všech rámců RT protokolu Profinetu IO:

```
((not ether proto 0x8892) and not(ether proto 0x8100 and
  ether[16:2]=0x8892))
```

Filtr pro protokol ARP:

```
(ether proto 0x0806)
```

Filtr pro protokol RPC:

```
(ip and udp)
```

Uvedené filtry jsou kombinovány pomocí logických spojek tak, aby výsledný filtr odpovídal požadavkům uživatele.

3.4.3 Rozbor rámců

V operační paměti jsou rámce uloženy v podobě objektů třídy CPacket. Tato třída poskytuje kompletní sadu metod pro identifikaci a rozbor všech typů rámců komunikace Profinet IO. Funkce pro rozbor rámců Profinetu IO byly převzaty ze zdrojového kódu nástroje Ethereal, kde byly uvedeny jako jeho doplněk. Nástroj Ethereal má vlastní způsob prezentace a zpracování rámců, který se se systémem vyvíjeného nástroje ne zcela shoduje. Proto byla z jeho funkcí využita zejména logická část rozboru a funkce byly přepsány tak, aby je bylo možné do nástroje integrovat.

V době, kdy je rámec přijat, je ihned po převedení do objektu proveden jeho rozklad. Během tohoto rozkladu je nejprve vyplněna struktura, obsahující základní informace o rámci, které zahrnují kód jeho typu, sloužící pro identifikaci při spojování s hlášením události a dále pak název protokolu a informační text o jeho obsahu. Tyto textové údaje slouží ke zobrazení v přehledové tabulce přijatých rámců. Po určení typu rámce je zavolána příslušná metoda, která provede rozbor dat rámce a výsledek uloží do paměti v podobě modelu stromu dat. Následně je vyslán signál objektu třídy CReport, který zjistí, zda je možné přijatý rámec přiřadit k některé události. V případě že ano, tak přidá ukazatel na rámec do seznamu rámců, spojených s hlášením.

3.4.4 Ověřování cyklických dat

Pro analýzu velkého množství rámců RTC komunikace, uložených v souboru, je určen zvláštní nástroj. Tento nástroj data zpracovává postupným načítáním jednotlivých rámců. U každého rámce určí pomocí adres odesílatele a příjemce a hodnoty položky FrameID, o který komunikační vztah se jedná. Následně provede rozbor dalšího obsahu rámce a výsledné údaje přidá k ostatním informacím o tomto CR. Pomocí porovnání stavu čítačů nově načteného a předešlého rámce je možné určit, zda nebyl přijat mimo pořadí a také jaká je perioda komunikace na straně odesílatele. Dále jsou kontrolovány jednotlivé bity v položce DataStatus (viz kapitola 1.11), které obsahují informace o celkovém stavu dat a jejich odesílatele. V případě, že je v této položce nalezena některá z chybových značek je rámec přidán do seznamu rámců s touto značkou. Tyto seznamy jsou vytvořeny pro každé CR zvlášť.

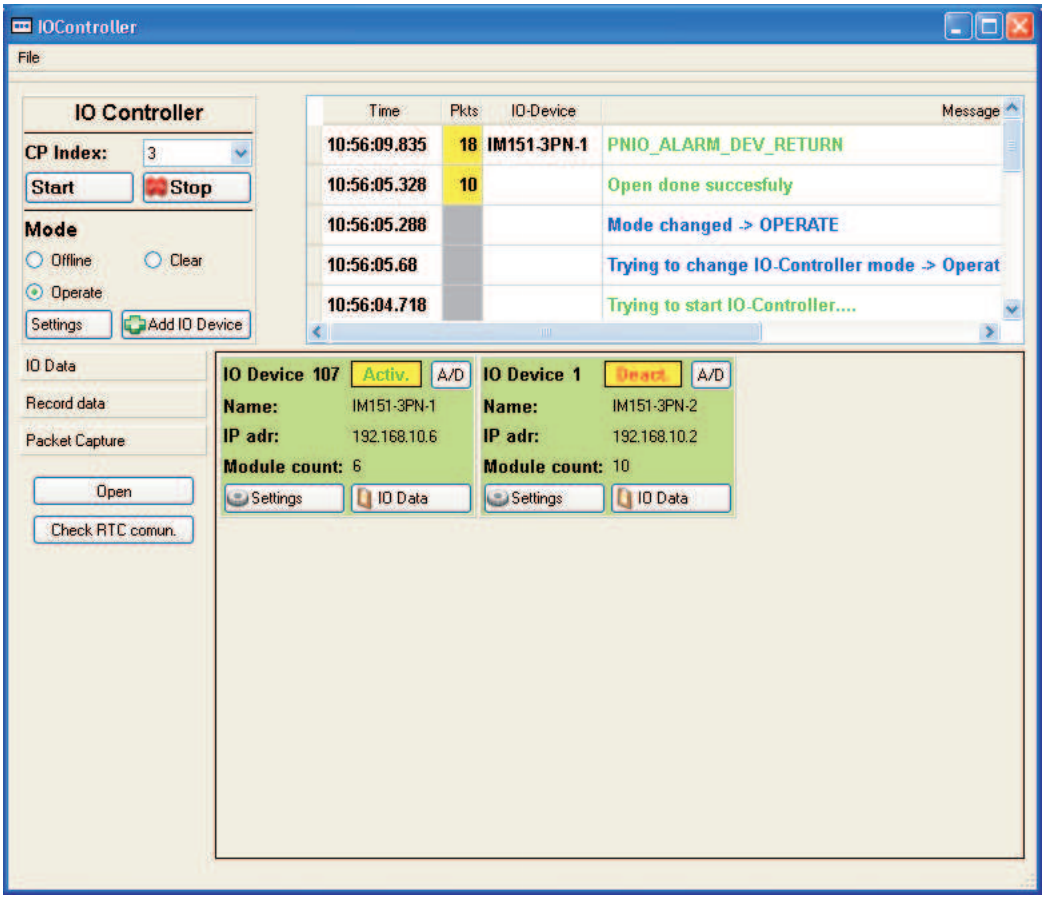
Po zpracování celého souboru uvedeným způsobem jsou ještě provedeny závěrečné výpočty. Pak je celá analýza zobrazena v textové podobě uživateli. Mezi závěrečné výpočty patří zejména určení průměrné periody vysílání poskytovatele, která je vypočtena ze sumy rozdílu stavů čítačů jednoho CR a počtu zpracovaných rámců tohoto CR. Z doby trvání komunikace a počtu přijatých rámců je určena průměrná perioda příjmu na straně konzumenta.

Popsaný nástroj umožňuje získat přehled o celé síti Profinet IO pouze ze ze záznamu komunikace. Tento záznam může být díky standardizovanému formátu souborů pořízen v libovolném programu pro záznam síťové komunikace, založeném na knihovně libpcap. Záznam nemusí obsahovat pouze rámce RTC komunikace, před zpracováním každého rámce je totiž provedena jeho identifikace. Tedy přesné určení, zda se jedná o protokol RT a zda hodnota položky FrameID odpovídá jedné ze tříd RTC komunikace.

3.5 Uživatelské rozhraní

Uživatelské rozhraní aplikace bylo vytvořeno pomocí komponent knihovny Qt. Skládá se z hlavního okna, které obsahuje základní informace o stavu sítě a jejich účastníků. Z tohoto okna je možné pomocí tlačítek zobrazovat jiná okna, zaměřená na speciální funkce programu či nastavení různých parametrů.

3.5.1 Hlavní okno aplikace



Obrázek 3.2: Hlavní okno aplikace IO-Controller

Hlavní okno aplikace, uvedené na obrázku obr. 3.2, je rozděleno do čtyřech hlavních částí. V levém horním rohu je umístěn panel s ovládacími prvky pro řízení IO-Controlleru. Je zde možné IO-Controller spustit a vypnout, případně změnit jeho režim. Před spuštěním je možné zvolit index komunikačního rozhraní v Simatic Configuration Manageru (viz kapitola 3.6), určeného pro IO Base. Nastavení IP IO-Controlleru adresy lze provést ve zvláštním okně, které se zobrazí po stisknutí tlačítka "Settings". Pomocí tlačítka "Add Device" jsou do konfiguračních dat IO-Controleru přidávána další zařízení IO-Device, která se zobrazují v poli, umístěném v pravé spodní části okna.

Jednotlivá IO-Device jsou v poli zobrazována v podobě panelů, které obsahují kromě základních informací o IO-Device několik tlačítek. Pomocí tlačítka v pravém horním rohu panelu

lze IO-Device deaktiovat, případně opět aktivovat. Deaktivovaná IO-Device jsou vyjmuta z komunikace, spojení s nimi navázané je tedy ukončeno. Pomocí tlačítka "IO Data" je zobrazeno okno se stavem I/O dat příslušného IO-Device (viz níže). Tlačítkem "Settings" lze vyvolat okno, kde je možné provést nastavení parametrů jako jsou IP adresa, MAC adresa, jméno zařízení a jeho struktura. Toto okno je také popsáno níže.

V pravém horním rohu hlavního okna aplikace je umístěna tabulka, která slouží k vypisování událostí, které nastaly za běhu programu. Události jsou řazeny od nejnovějších po ty nejstarší. Každý řádek tabulky obsahuje následující údaje:

- čas výskytu události
- počet rámců spojených s touto událostí
- jméno IO-Device, kterého se událost týká (nemusí být vždy uvedeno)
- popis události

Pokud je počet rámců, spojených s událostí nenulový, je možné pomocí dvojitého kliknutí myší na příslušný řádek události vyvolat zobrazení okna nástroje pro rozbor rámců (viz níže), které bude obsahovat pouze rámce, které souvisejí s touto událostí.

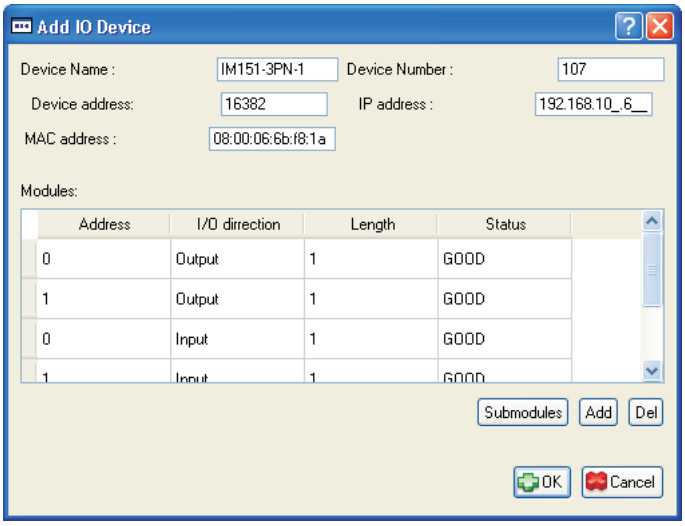
Poslední část hlavního okna, umístěná v levé spodní části, obsahuje tzv. toolbar, který má několik záložek, obsahujících tlačítka pro zobrazení např. oken nástrojů na analýzu rámců, či čtení a zápis datových záznamů. Dále jsou zde v záložce IO Data umístěny prvky pro nastavení stavu IO-Controlleru a způsobu aktualizace I/O dat v objektu IOData.

3.5.2 Nastavení IO-Device

Okno pro nastavení parametrů a struktury IO-Device (viz obr. 3.3) se zobrazí po stisknutí tlačítka "Settings" umístěného na panelu příslušného IO-Device. Toto okno obsahuje jednak políčka s jednotlivými parametry, které mohou být měněny a dále pak tabulku modulů IO-Device. Každý řádek obsahuje parametry jednoho modulu, které mohou být v této tabulce nastavovány. Je zde uvedena adresa a délka dat, ale i aktuální stav modulu, který slouží pouze ke čtení. Moduly lze do tabulky přidávat pomocí tlačítka "Add". K odebrání označeného modulu slouží tlačítko "Del". Při přidávání nového modulu se zobrazí další okno, velmi podobné tomuto, které slouží k zadání parametrů modula a zejména k vložení jeho submodulů, které představují datové položky, zobrazované v přehledu I/O dat zařízení.

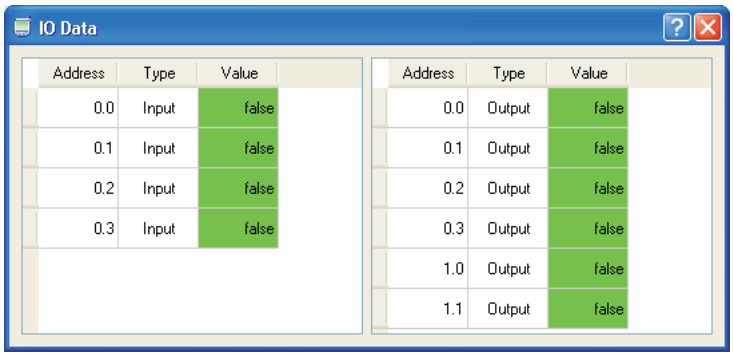
3.5.3 Okno I/O dat

Okno I/O dat (viz obr. 3.4), které se zobrazí po stisknutí příslušného tlačítka aplikace, umístěného na panelu IO-Device, obsahuje dvě tabulky. První tabulka (vlevo) je určena pro zobrazení stavu vstupních dat. Tabulka vpravo slouží k zobrazení a provádění změn výstupních dat. Každý řádek v obou tabulkách odpovídá jednomu submodu, uvedenému v nastavení IO-Device. Parametry každého submodulu určují směr jeho dat (vstu/výstup) a datový typ. Data v tabulce výstupů lze měnit poklepáním na příslušnou buňku s daty. Následně se zobrazí v buňce editor položky, určený pro konkrétní datový typ submodulu. Jak je vidět na obrázku, jsou všechny řádky v obou tabulkách označeny barevně. Barva řádku slouží ke zobrazení stavu



Obrázek 3.3: Okno pro nastavení parametrů a struktury IO-Device

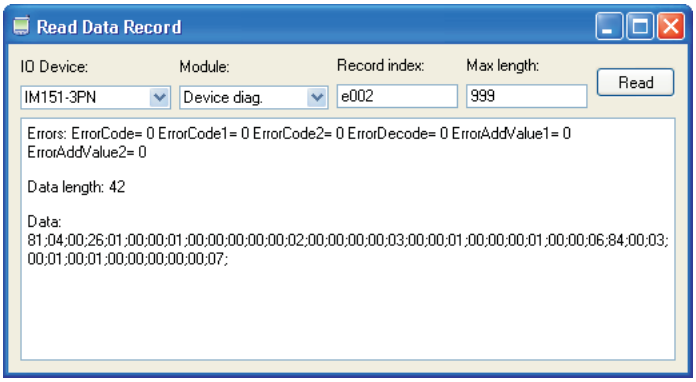
modulu, kterému data na řádku přísluší. Zelená barva je vyhrazena pro stav GOOD, červená pro stav BAD.



Obrázek 3.4: Okno pro zobrazení a ovládání I/O dat IO-Device

3.5.4 Čtení datových záznamů

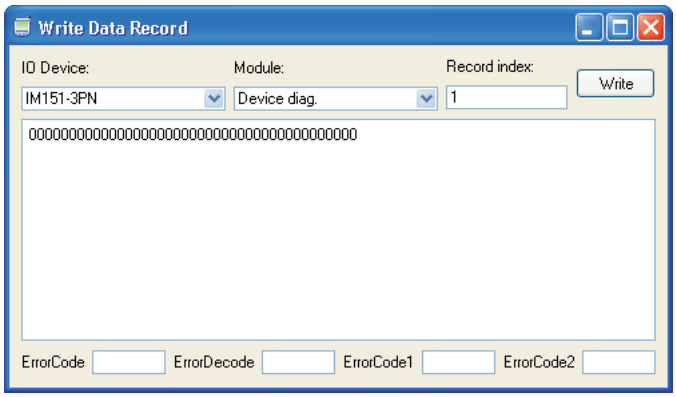
Čtení datových záznamů je možné provádět ve zvláštním okně (viz obr. 3.5), určeném jen k tomuto účelu. K jeho zobrazení dojde po stisknutí tlačítka "Read" v záložce "Record data" toolbaru umístěném v hlavním okně aplikace. Ve vrchní části okna vybere uživatel nejprve IO-Device a následně jeho modul, ze kterého ma být přečten datový záznam. V následujícím políčku vyplní číslo indexu požadovaného datového záznamu a nakonec ještě maximální možnou délku požadovaných dat. Po zadání všech těchto údajů je možné stisknout tlačítko "Read", které se nachází v pravém horním rohu okna. Tím dojde k vyslání požadavku o datový záznam. Po obdržení odpovědi je v textovém poli okna, nacházejícím se pod panelem pro výběr záznamu, nejprve vypsán stav chybových kódu a následně všechna přijatá data v hexadecimálním tvaru.



Obrázek 3.5: Okno pro čtení datových záznamů IO-Device

3.5.5 Zápis datových záznamů

Stejně jako okno pro čtení datových záznamů má okno pro jejich zápis (viz obr. 3.6) v horní části umístěny ovládací prvky pro výběr cílového záznamu. Jedná se tedy o výběr IO-Device, modulu a indexu. Níže je umístěno pole pro zadání dat, které mají být odeslány v žádosti o zápis do datového záznamu. Po provedení zápisu pomocí tlačítka v horní části se do příslušných políček ve spodní části okna vypíší chybové údaje, signalizující výsledk operace.

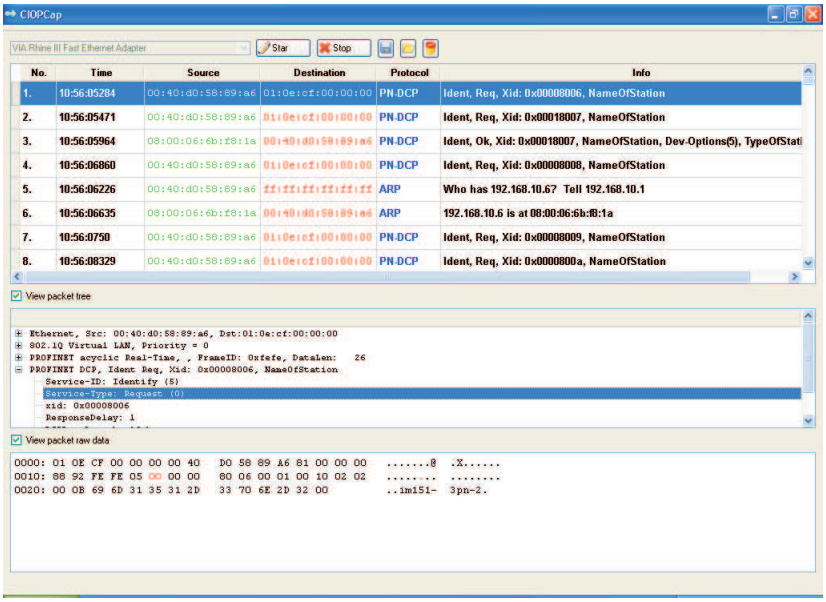


Obrázek 3.6: Okno pro zápis datových záznamů IO-Device

3.5.6 Prohlížeč ethernetových rámců

Pro spouštění záznamu rámců a jejich prohlížení je určeno zvláštní okno aplikace(viz obr. 3.7). Návrh tohoto okna byl inspirován způsobem zobrazování informací o rámcích, který používá nástroj Ethereal. Okno je rozděleno do několika částí. V horní části okna se nacházejí ovládací prvky sloužící k výběru síťového adaptéru, spuštění či vypnutí záznamu. Dále jsou zde umístěny tlačítka pro ukládání rámců do souboru či jejich načítání, případně vymazání zaznamenaných rámců. Po stisknutí tlačítka pro spuštění záznamu rámců se před vlastní spuštěním ještě zobrazí dialogové okno, určené pro nastavení filtru zaznamenávaných typů komunikace. Po provedení výběru a jeho potvrzení dojde ke spuštění záznamu.

Ihned po uložení zaznamenaného rámce do paměti je v tabulce, nacházející se přímo pod ovládacími prvky, zobrazen nový řádek obsahující informace o tomto rámci. Je zde uveden



Obrázek 3.7: Okno pro záznam a rozbor ethernetových rámců

přesný čas provedení záznamu, jeho zdroj a cíl, jméno hlavního protokolu rámce a dále pak stručný popis obsahu rámce. Pod touto tabulkou jsou ještě umístěny dvě pole pro zobrazování podrobného obsahu rámce, vybraného v tabulce.

První pole slouží ke zobrazení a popisu jednotlivých položek rámce ve stromové struktuře. Každá položka kořenu stromu představuje větev údajů jednoho protokolu, obsaženého v rámci. Data těchto protokolů jsou dále rozložená podle jejich struktury. Některá jsou dále členěna do podstromů a jiná jsou již uvedena jen jako seznam položek. Uživatel má možnost prohlížet si obsah rámce rozbalováním jednotlivých částí stromuvé struktury.

Ve spodní části okna se nachází textové pole, určené k výpisu dat rámce v hexadecimální a textové podobě. Data jsou vypisována vždy po šestnácti bytech v jedno řádku. Textové pole je rozděleno na dva sloupce. První slouží pro zobrazení dat v hexadecimální podobě. Druhý sloupec obsahuje výpis dat v podobě ASCII znaků, která umožňuje v rámci snadno rozpoznat textové řetězce, jako je například jméno IO-Device.

Jednotlivé položky ve stromové struktuře jsou provázány s výpisem dat v textovém poli. Uživatel může označit vybranou položku ve stromové struktuře a data příslušející této položce se následně v textovém poli obsahujícím výpis dat zvýraznění.

Okno prohlížeče rámců může být zobrazeno i po poklepání na vybranou událost v hlavním okně. V tomto případě je v okně zobrazen pouze výběr rámců, souvisejících s danou událostí. Zobrazení výběru rámců je v horní části okna signalizováno zaškrtávacím políčkem, pomocí kterého může být výběr zrušen. Toto políčko je viditelné pouze při zobrazeném výběru dat.

3.5.7 Analýza RTC komunikace

V případě, že chce uživatel analyzovat soubor, obsahující rámce RTC komunikace Profinetu IO, použije k tomu speciální nástroj. Ten se zobrazí v novém okně aplikace (viz obr. 3.11) po stisknutí tlačítka "Check RTC comm.", umístěném na záložce "Packet Capture" v toolboxu

hlavního okna.

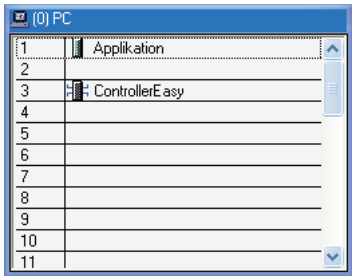
Okno tohoto nástroje se skládá z tlačítka pro výběr požadovaného souboru, tlačítka pro spuštění analýzy dat a textového pole pro výpis výsledku analýzy.

3.6 Konfigurace projektu Step 7

Před použitím realizované aplikace IO-Controlleru je nejprve nutné vytvořit nový projekt Simatic PC stanice v Simatic Manageru nebo Simatic NCM nástroji. V tomto projektu se provede konfigurace celého Profinet I/O systému, který má být řízen IO-Controllerem, realizovaným aplikací založenou na IO Base rozhraní. Tato konfigurace se následně nahraje do PC stanice. Postup vytváření nového projektu v Simatic Manageru je popsán níže. V nástroji Simatic NCM by se postupovalo obdobně.

Nový projekt

Nejprve se v Simatic Manageru založí nový projekt, do kterého se vloží nový objekt "Simatic PC Station". Ten reprezentuje PC, na kterém bude IO-Controller realizován. V konfiguraci PC stanice se na vybrané pozice postupně vloží objekt představující aplikaci "Application" a objekt síťového rozhraní, který muže být reprezentován standardním síťovým adaptérem "IE General" nebo speciální komunikační kartou Profinetu IO např. "CP1616". Při vkládání objektu komunikačního rozhraní se ještě provede nastavení síťových parametrů rozhraní (IP adresa,maska podsítě, případně adresa výchozí brány). Před potvrzením nastavených parametrů je ještě nutné vytvořit novou podsíť (subnet).



Obrázek 3.8: Nový projekt Simatic PC Station

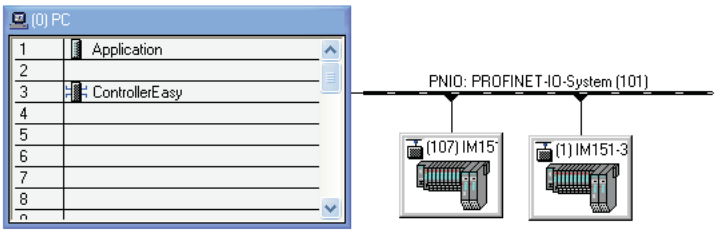
3.6.1 Založení nového Profinet IO systému

Po vložení příslušných objektů PC stanice je nyní nutné založit v projektu nový IO systém, ke kterému pak bude možné připojovat jednotlivá IO-Device. Nový IO systém se přiřadí síťovému rozhraní kliknutím na jeho objekt pravým tlačítkem a výběrem volby "Insert PROFINET IO System".

3.6.2 Vložení a nastavení všech IO-Device

K založenému IO systému je nyní možné připojovat IO-Device, které má IO-Controller řídit. Po výběru v katalogu zařízeních se IO-Device připojí pouhým přetažením do IO systému.

Každému vloženému IO-Device je nutné přiřadit jméno, pokud nevyhovuje to, které mu bylo přiděleno. Stejně je tomu tak i s IP adresou. Pokud se jedná o modulární IO-Device je ještě potřeba vložit z katalogu do zařízení příslušné moduly.



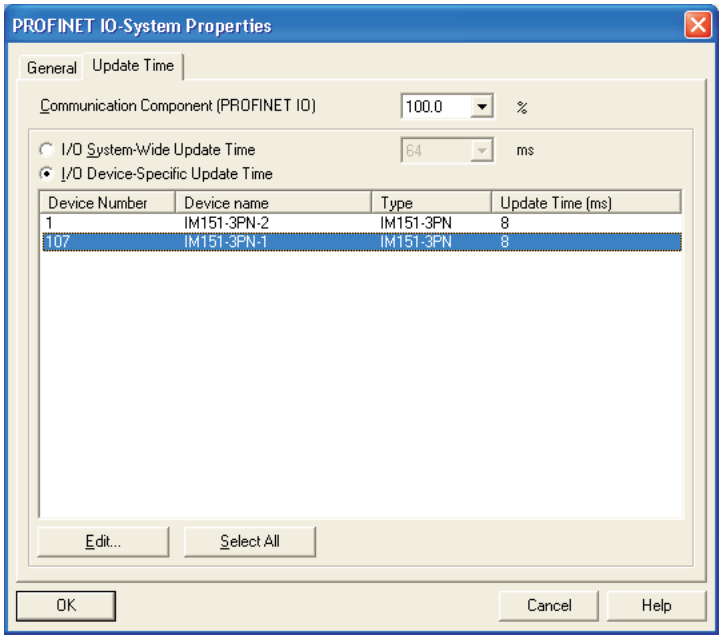
Obrázek 3.9: IO systém Profinetu IO s připojenými IO-Device

3.6.3 Nastavení period komunikace

Po vložení všech požadovaných IO-Device do projektu je potřeba provést nastavení požadovaných period komunikace mezi jednotlivými IO-Device a IO-Controllerem. Toto nastavení se provádí ve vlastnostech IO systému v záložce "Update Time". V tomto nastavení lze zvolit jednotnou periodu pro všechna spojení nebo definovat pro každé IO-Device jinou. Možné hodnoty periody u každého IO-Device korespondují s časy uvedenými v GSD souboru zařízení. Řešení firmy Siemens neumožňuje definovat jinou periodu pro vstupy a výstupy jednoho IO-Device, natož vytvářet více komunikačních spojení mezi IO-Device a IO-Controllerem v jednom směru. V nastavení IO systému je ještě uveden jeden parametr, nazvaný "Communication Component (PROFINET IO)", který umožňuje dopředu definovat kolik procent z celkového možného vytížení komunikačního rozhraní bude rezervováno pro Profinet IO. Tento údaj pak slouží jako pomocný parametr při rozvrhování rámců Profinetu IO. V případě, že by hodnota tohoto parametru byla příliš nízká, mohlo by dojí k tomu, že by nebyla všechna I/O data doručena v požadovaných časech.

3.6.4 Přiřazení jmen IO-Device

Před nahráním konfigurace do IO-Controlleru a spuštěním komunikace je nutné přiřadit skutečným IO-Device jejich jména, která jim byla přidělena během vytváření konfigurace projektu. Před tímto úkonem je nutné nejprve připojit konfigurační stanici do sítě, kde jsou i zařízení IO-Device a nastavit PG/PC Interface Simatic Manageru na TCP/IP protokol na používaném síťovém rozhraní. V hardwarové konfiguraci projektu je v menu PLC-¿Ethernet příkaz "Assign Device Name", který vyvolá okno pro nastavení jmen IO-Device. Nejprve dojde k prohledání sítě a v okně se následně vypíší dostupná IO-Device. U každého je uvedena jeho MAC adresa, případně jméno a IP adresa, pokud již má nějaké přiděleny. Pokud je zařízeních více a není jisté, která MAC adresa je čí, lze využít funkce Flash, která rozbliká LED diody na IO-Device se zvolenou MAC adresou. Tlačítkem "Assign" lze zvolenému IO-Device přiřadit nové jméno, vybrané ze seznamu všech jmen v projektu, umístěným v horní části okna.



Obrázek 3.10: Nastavení periody komunikace ve Stepu 7

3.6.5 Nahrání konfigurace do IO-Controlleru

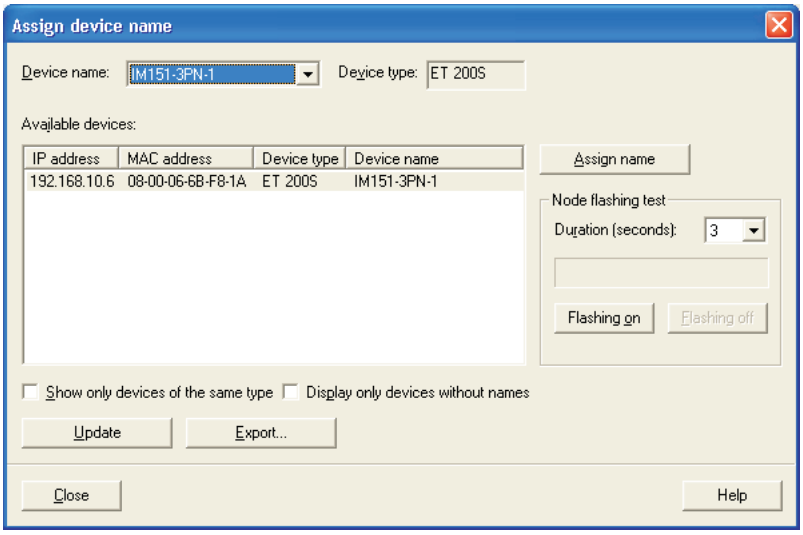
Posledním krokem před spuštěním systému je nahrání konfiguračních dat do IO-Controlleru. Popsaný konfigurační nástroj je dostupný pouze pro operační systém Windows. Konfiguraci je zde však nutné vytvořit i pro aplikace s CP 1616, které fungují na jiném operačním systému. V tom případě je ponecháno nastavení PG/PC Interface na TCP/IP protokol.

Pokud je konfigurace určena pro PC, na kterém je spuštěn konfigurační nástroj, musí se zvolit rozhraní "PC internal". V tomto případě je před nahráním ještě nutné v aplikaci "Station Configuration Editor" vložit objekt "Application" a objekt síťového rozhraní, na pozice shodující se s těmi, které jim byly přiřazeny v projektu. Při vkládání objektu síťového rozhraní se zvolí konkrétní ethernetový adaptér, dostupný v PC. Před jeho vložením je ještě nutné zkontrolovat jeho nastavení. Pokud by totiž jeho IP adresa neodpovídala té v projektu, nebylo by možné nahrání konfigurace provést.

Po provedení uvedených nastavení je možné přistoupit k vlastnímu nahrání konfigurace. To se provede v okně hardwarové konfigurace příslušným tlačítkem v toolbaru. Projekt však musí být nejprve uložen a zkompilován.

3.7 Převod aplikace na systém Linux

Aplikace IO-Controlleru byla převedena a zprovozněna také na operačním systému Linux Suse 9.2 s jádrem verze 2.6. Díky použitým knihovnám, které mají své verze i pro tento operační systém, byl proces převodu relativně nenáročný. V tomto operačním systému lze IO-Controller založený na IO Base rozhraní realizovat pouze s využitím speciální komunikační karty, jako je CP 1616, která byla použita v tomto případě. K této kartě jsou výrobcem dodávány zdrojové kódy ovladače a knihoven PNIO a IO Base, určené pro Linux. Ovladač a knihovny musejí



Obrázek 3.11: Přiřazení jména IO-Device ve Stepu 7

být nejprve přeloženy a nainstalovány. Pro správný chod aplikace je dále nutné nainstalovat knihovnu libpcap, určenou pro záznam datových rámců ze síťového rozhraní(není určena pro CP 1616) a knihovnu Qt verze 4.1, na které je celá aplikace založena. Při překladu aplikace IO-Controlleru k ní boudou všechny uvedené dynamické knihovny připojeny.

3.7.1 Odlišnosti verze pro Linux

Verze aplikace IO-Controlleru pro Windows a Linux se liší pouze v jednom hlavičkovém souboru, nazvaném `os.h`. Tento soubor slouží zejména k připojení standardních systémových knihoven, které se jsou na obou systémech odlišné. Jedná se například o knihovny pro práci s řetězci znaků, či knihovny se základními datovými typy. V hlavičkovém souboru, určeném pro Linux je navíc definováno makro pro převod instrukce `Sleep()` na `usleep()`. Tyto funkce slouží k přerušení činnosti vlákna na definovanou dobu. Parametrem instrukce `Sleep()` ve Windows je čas v milisekundách na rozdíl od Linuxu, kde parametr této funkce je zadáván v sekundách. V Linuxu však existuje uvedená instrukce `usleep()`, které se zadává čas v mikrosekundách. Řešení je následující:

```
#define Sleep(x) usleep(x*1000)
```

3.7.2 Instalace ovladače CP 1616

Zdrojové kódy ovladače, dodávané společně s kartou CP 1616 je nutné nejprve přeložit a nainstalovat. Ovladač karty se nejprve zkompiluje jako modul jádra, který se následně nainstaluje. Výrobce má vše připravené v souboru Makefile, dodávaném s ovladačem. Ten však dobře nefungoval, proto bylo přistoupeno k ruční instalaci. Pro překlad ovladače je nutné mít v adresáři `/usr/src/linux` umístěny zdrojové kódy jádra. Překlad a instalace knihoven PNIO a IO Base jsou dobře popsány v dokumentaci [4].

Postup pro překlad a instalaci ovladače CP 1616 je následující:

1. Soubor `host-20050512-2.8.6d.tar.gz`, který je umístěn na CD, dodávaném společně s kartou je nutné nejprve rozbalit do vybraného adresáře např. `/home/user/ProfinetIO`.

2. Pro překlad ovladače a jeho připojení k jádru jsou nutná práva uživatele `root`, proto je nutné v shellu zadat příkaz `su` a následně heslo uživatele `root`.

3. Nyní lze přistoupit k překladu ovladače jako modulu jádra pomocí příkazu:
   `make -C /usr/src/linux SUBDIRS=/home/user/pn/cp16xx/driver modules`

4. Zkompilovaný modul se nainstaluje pomocí příkazů:
   `make -C /usr/src/linux SUBDIRS=/home/user/pn/cp16xx/driver modules_install`
   `/sbin/depmod -aF /lib/modules/$(VERSION)`

5. Nyní již stačí zavést ovladač do paměti pomocí souboru Makefile, umístěném v adresáři ovladače:
   `cd /home/user/pn/cp16xx/driver`
   `make load`

Kapitola 4

Závěr

Tato diplomová práce představuje a podrobněji přibližuje nový komunikační standard určený především pro distribuované průmyslové aplikace. Teoretická část byla zaměřena na analýzu a dokumentaci tohoto komunikačního protokolu a poskytuje přehled hlavních funkcí a vlastností i detailnější rozbor rámců jednotlivých protokolů Profinetu IO.

Profinet IO má dobrý koncept, který vyhovuje potřebám systémů založených na distribuovaných periferiích. Drží se osvědčených vlastností svého předchůdce Profibusu DB. Specifikace tohoto standardu definuje několik vlastností jako je například možnost výměny dat mezi zařízeními stejné třídy(IO-Controller - IO-Controller, IO-Device - IO-Device), či možnost rozdělení I/O dat jednoho IO-Device do více skupin a ty pak přenášet ve více spojeních, která mohou mít různou periodu komunikace i cílového účastníka. Tyto vlastnosti však v praxi zatím nejsou podporovány. Zařízení dostupná v současné době poskytují většinou pouze povinné vlastnosti Profinetu IO, které se rozsahem funkcí příliš neliší od obdobných komunikačních standardů.

Real-time ethernet, na kterém je Profinet IO postaven, umožňuje využívat svoji síť i pro standardní IT komunikace (za předpokladu že jsou použity síťové prvky podporují priorizaci rámců pomocí VLAN). Je tedy možné integrovat do zařízeních Profinetu-IO například web servery, ke kterým je teoreticky možné přistupovat i z internetu. Tyto možnosti však přináší do oblasti řízení nové problémy týkající se zabezpečení sítí proti neoprávněnému vniknutí a útokům. Vzhledem k tomu, že na provoz výrobních zařízení, kde jsou průmyslové sítě nejčastěji nasazovány, jsou kladeny vysoké nároky z hlediska spolehlivosti a každá prodleva představuje pro podnik velké ztráty, zřejmě v dohledné době nedojde k většímu rozmachu těchto nových technologií. Real-time ethernet vyžaduje také nové přístupy k instalaci sítí. Topologie sběrnice, kterou používá například Profibus DP, je zde nahrazena topologií hvězdy či stromu, které vyžadují kromě jiného přístupu ke kabeláži i použití ethernetových přepínačů (switch). Výsledná instalace může být ve výsledku nákladnější než stejné provedení například s uvedeným Profibusem DP.

Pro praktickou demonstraci vlastností Profinetu-IO byla realizována aplikace IO-Controlleru. Tato aplikace je založena na knihovně firmy Siemens, která představuje kompletní softwarový zásobník Profinetu IO, rozšířený o rozhraní *IO Base User Programming Interface*. Toto rozhraní nabízí funkce pro snadný způsob realizace aplikací IO-Controlleru i IO-Device. Nevýhodou takovéto aplikace je její závislost na softwarovém balíku uvedené firmy. Díky omezenému rozsahu funkcí IO Base rozhraní, které jsou určeny pouze k běžnému provozu zařízení a nedovolují proniknout do interních stavů Profinetu-IO, bylo přistoupeno k rozšíření aplikace o knihovnu libpcap,

která umožňuje zaznamenávat průběh ethernetové komunikace a dále zpracovávat jednotlivé rámce. V praxi se použitá knihovna osvědčila i pro záznam cyklické real-time komunikace, jejíž rámce jsou přímo ukládány do souborů, což umožňuje provést záznam velkého množství rámců. Program dále nabízí nástroje pro analýzu jednotlivých rámců i pro hromadné vyhodnocení velkého množství rámců cyklické real-time komunikace. Díky kombinaci funkcí IO-Controlleru s nástroji pro analýzu komunikace by tento program mohl být vhodný například pro pomoc při vývoji a testování nových zařízení IO-Device například v některé z navazujících diplomových prací.

Při vývoji uvedené aplikace byl kladen důraz na přenositelnost na jiné operační systémy, než jsou Windows. Toto je možné díky tomu, že s komunikační kartou CP 1616 od firmy Siemens jsou dodávány zdrojové kódy knihovny se softwarovým zásobníkem Profinetu-IO. Tato knihovna má stejné rozhraní IO Base jako její verze pro Windows a může být převedena na libovolný operační systém. Vždy je však na rozdíl od operačního systému Windows nutné použít komunikační kartu jako je CP 1616 a jí podobné. Aplikace byla pro demonstraci zprovozněna na operačním systému Linux.

Aplikaci IO-Controlleru lze dále rozšiřovat zejména v oblasti analýzy komunikace. Pro podporu při testování IO-Device by bylo možné například doplnit program o nástroj na sestavování libovolných rámců Profinetu-IO, které by mohly být následně vysílány nezávisle na funkčním IO-Controlleru. Ty by sloužily například k umělému zatížení sítě či dokonce k ověřování konkrétních funkcí IO-Device.

Literatura

[1] PROFIBUS International (2005). *PROFINET IO - Application Layer Service Definition and Protokol Specification Version 2.0*.
PROFIBUS Nutzeorganization e.V.

[2] Popp, M.,Weber, K. (2004). *The Rapid Way to PROFINET*.
PROFIBUS Nutzeorganization e.V.

[3] Siemens AG (2005). *IO-Base User Programming Interface, Programming Manual*.
`Iobase_e.pdf`

[4] Siemens AG (2005). *DK-16xx PN IO Porting Instructions*.
`DK_16xx_PN_IO_en.pdf`

[5] PROFIBUS International (2005). *GSDML Specification for PROFINET IO Version 2.0*.
PROFIBUS Nutzeorganization e.V.

[6] IEC (2005). *Real-time Ethernet PROFINET IO*.
`http://www.iec.ch`.

[7] Popp, M. (2004). *Das PROFINET IO-Buch. Grundlagen und Tipps für Anwender*.
Hüthig Verlag Heidelberg

[8] Trolltech AS (2006). *Qt Reference Documentation 4.1*.
`http://doc.trolltech.com`.

[9] M. Beran (2006). *Programování pro X Window System*.
`http://www.ms.mff.cuni.cz/~beran/vyuka/X/`.

[10] The WinPcap Team (2005). *WinPcap Documentation 3.1*.
`http://www.winpcap.org`.

[11] U. Lamping, R. Sharpe, E. Warnicke (2005). *Ethereal User's Guide*.
`http://www.ethereal.com`

[12] R. Bednář (2006). *Latex manuál*.
`http://www.cstug.cz`.

[13] Siemens AG (2006). *Siemens Image Database*.
`http://www.automation.siemens.com/bilddb/`.

[14] Siemens AG (2006). *Siemens - Automation and Drives - Service & Support*. `http://support.automation.siemens.com/`.

Dodatek A

Použité zkratky

AR Application Relationship - je vazba mezi dvěma nebo více účastníky, která se realizuje za účelem spolupráce a výměny dat.

ARP Address Resolution Protocol - slouží k určení MAC adresy spojené s určitou IP adresou

CM Context Management - kontext definuje platformu na které dvě zařízení spolupracují.

CR Communication Relationship - komunikační kanál do IO-Device založený na určitém protokolu

DCP Discovery and Configuration Protocol - slouží ke čtení či nastavování jmen, IP adres a dalších parametrů. Služba Identify identifikuje stanice v síti.

DHCP Dynamic Host Configuration Protocol - používá se k dynamickému přidělování IP adres.

DNS Domain Name System - je hierarchický systém doménových jmen, který je realizován servery DNS a protokolem stejného jména, kterým si vyměňují informace. Systém zajišťuje mapování doménových jmén na IP adresy uzlů sítě.

ERTEC Enhanced Real-Time Ethernet Controller - obvod s integrovanými funkcemi Profinetu IO.

GSDML Generic Station Description Markup Language - popisovací jazyk pro vytváření GSD souborů.

GUI Graphical User Interface - grafické rozhraní aplikace.

I&M Identification and Maintenance - služby, poskytující podporu během sprovozňování a údržby.

ICMP Internet Control Message Protocol - mechanizmus pro odesílání chybových informací

IEC International Electrotechnical Commission

I/O Input-Output - vstupy a výstupy (chápáno z pohledu IO-Controlleru)

IOCS I/O Consumer Status - používáno konzumentem (příjemcem) I/O dat pro signalizaci svého stavu

IOPS I/O Provider Status - používáno poskytovatelem (odesílatelem) I/O dat pro signalizaci svého stavu

IRT Isochronous Real-time - komunikační kanál Profinetu IO, určený pro deterministickou komunikaci ve vyhrazeném pásmu.

MAC Media Access Control - určuje přístup stanice k médiu. Ve full duplexním ethernetu může kdykoli vysílat kterákoli stanice.

NDR Network Data Representation - 20 bytová hlavička určená k definování pravidel pro reprezentaci dat v RPC blocích.

PLC Programmable Logic Controller - programovatelný logický automat

PI PROFIBUS International - organizace, která vyvinula Profinet IO

RPC Remote Procedure Call - protokol pro vzdálené volání procedur. Profinet IO používá variantu Open Source Foundation C706,CAE Specification DCE11:Remote Procedure Call.

RT Real-time - jméno real-time protokolu, založeného na čistém ethernetu bez specielních úprav.

RTA Real-time Acyclic Protocol - používá se k přenosu poruch.

RTC Real-time Cyclic Protocol - používá se k přenosu I/O dat.

UDP/IP User Datagram Protocol/Internet Protocol - nezabezpečený přenos dat

UUID Universal Unique Identifier - slouží k jedinečné identifikaci v části funkcí Profinetu IO

XML Extensible Markup Language - je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovací jazyků pro různé účely a široké spektrum různých typů dat.

Dodatek B

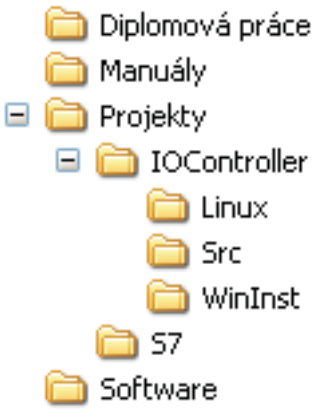
Technická data CP 1616

Technical data	**CP 1616**
Transmission rate	10/100 Mbit/s autosensing
Interfaces 10BaseT, 100 BaseTX	 4 x RJ45, Autocrossing
Connection to programming device/PC	PCI 2.2 and PCI-X compatible, 32 bit, executable in 64-bitPCI-X slots, 33 MHz/66 MHz,3.3 V/5 V universal key
External supply for switch from standard plug-type power supply unit (optional)	Low-voltage socket for hollow plug 3.5 mm (-) / 1.3 mm (+)
Power supply	
Internal	5 V DC through PCI
External	Optional for switch with PC power-down
- Electrical isolation	Required
- Rated voltage range	6 - 9 V DC
Current consumption	
Internal	Max. 800 mA from PCI 5 V
External	Optional for switch with PC power-down
- At 6 V DC (optional)	max. 650 mA
- At 9 V DC (optional)	max. 450 mA
Power loss	
In normal mode (PC on; without external supply)	Approx. 4 W
In switch mode (PC power down and external supply)	Approx. 3.9 - 4.1 W
Perm. ambient conditions Operating temperature	 +5 °C to +55 °C
Transport/storage temperature	-20 °C to +60 °C
Relative humidity	Max. 95% at +25 °C
Construction	
Module format	PCI card, short
Dimensions (W x D) in mm	107 x 167
Weight	Approx. 110 g
Space requirements	1 x PCI slot (32-bit; 3.3 V/5 V)
Processor	ERTEC 400 / ARM9

Performance data	
PN IO-Controller performance data	
Number of operable PN IO-Devices	Max. 64
Size of IO data areas overall	
I/O input area	Max. 32 KB
I/O output area	Max. 32 KB
Size of I/O data areas per connected PN IO-Device	
I/O input area	Max. 1472 byte
I/O output area	Max. 1472 byte

Dodatek C

Struktura přiloženého CD ROM



Přiložený CD ROM v jednotlivých složkách obsahuje :

Diplomová práce – Tato diplomová práce ve formátu LaTeX

Manuály – Dokumentace k používanému hardwaru a softwaru ve formátu PDF.

Projekty\S7 – Projekty Stepu 7 pro CP 1616 i Softnet

Projekty\IOController\Src – Zdrojový kód aplikace IO-Controlleru

Projekty\IOController\WinInst – Spustitelná verze IO-Controlleru se všemi knihovnami pro operační systém Windows

Projekty\IOController\Linux – Instalace aplikace IO-Controlleru pro operační systém Linux

Software – Knihovna Qt 4.1 pro Windows i Linux, knihovny WinPCap a libpcap .