ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ŘÍDICÍ TECHNIKY

DIPLOMOVÁ PRÁCE

Implementace protokolu LonWorks

2009 **Zdeněk Vít**

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Zdeněk Vít**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Implementace protokolu LonWorks**

Pokyny pro vypracování:

1. Seznamte se s principy komunikace na sběrnici LonWorks, seznamte se základními rysy operačního systému reálného času On-Time.
2. S využitím dodávaných driverů pro PC104 kartu zprovozněte pod operačním systémem Windows komunikaci LonWorks. Karta dohromady s PC bude v režimu standardního nódu. Karta obsahuje firmware MIP/P50 v režimu Network Interface Selection, tj. komunikace maximálně 62 síťových proměnných.
3. Upravte software tak, aby bylo možno využívat až 4096 síťových proměnných, tj. karta bude pracovat v tzv. Host Selection módu. Nadřazená aplikace musí obsahovat správu proměnných a propojení mezi nódy.
4. Proveďte implementaci nejnižšího driveru karty pod operační systém reálného času On-Time. Nad tímto driverem otestujte aplikaci napsanou v bodě 2 a 3.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Pavel Burget

Platnost zadání: do konce zimního semestru 2008/2009

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 10. 9. 2007

Abstrakt

V této práci se zabývám implementaci nejnižšího ovladače PC104 kartu pro síť LonWorks a vytvořením *Host Application* pro tuto kartu. Jak ovladač, tak Host application je implementována pro operační systém Windows a pro operační systém reálného času On-time.
Karta dohromady s počítačem je v režimu standardního nódu, nechová se tedy jako manažer sítě. Je zde rozebrán *Network Interface Selection* mód, tj. komunikace maximálně 62 proměnných. Karta obsahuje firmware MIP/P50 v režimu *Host Selection*, tj. rozšíření množství proměnných až na maximální počet 4096.
Pro přístup na hardware karty pod Windows se užívá dodávaná DLL knihovna wldv32.dll, která implementuje funkce ldv_open, ldv_close, ldv_read, ldv_write. Dalším úkolem bylo napsat ovladač pro On-time a nahradit knihovnu wldv32.dll vlastní implementací funkcí ldv_open, ldv_close, ldv_read, ldv_write.
Software je založen na vlastní implementaci prezentační a aplikační vrstvy protokolu LonTalk, proto se rozborem těchto vrstev zabývá podstatná část této práce. Výsledkem je jen jedna *Host Application* a podmíněným překladem jsou vytvořeny verze pro oba operační systémy.

Abstract

This final-year dissertation deals with implementing the lowest driver for PC104 card for the LonWorks and creating a host application for this card. Both the driver and the Host Application are implemented for the Windows operating system and for the real–time operating system On-time.

The card is, together with the computer, in the based node regime, and does not behave as a manager of the network. A *Network Interface Selection* mode, which means communication of the maximum of 62 variables. The card is equipped with the MIP/P50 firmware in the *host selection* mode is then analysed, i.e. the expansion of the number of variables up to the maximum number of 4096.

For the access to the hardware card under Windows the DLL library wldv32.dll is used, which implements functions ldv_open, ldv_close, ldv_read, ldv_write. The following task was to create a driver for On-time and replace the wldv32.dll library with a new implementation of functions ldv_open, ldv_close, ldv_read, ldv_write.

Because the software is based on my own implementation of the presentation and application layers of the LonTalk protocol, a large part of this work is devoted to the analysis of these layers. The result is only one Host Application suitable for both systems, with the needed version being generated by the conditioned compilation.

Poděkování

Na tomto místě bych rád poděkoval svým rodičům, přítelkyni a příbuzným za vytrvalou podporu při studiu. Rád bych také poděkoval Ing. Ondřeji Netíkovi za zodpovězení dotazů týkajících se této práce a za poskytnutí cenných informací při tvorbě aplikace. Dále děkuji Ing. Pavlu Burgetovi, Ph.d. za odborné vedení diplomové práce a za poskytnutí všech potřebných materiálů.

Prohlášení

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 17.1.2009 …….

podpis

OBSAH

1. Úvod

Ve svých začátcích byla oblast průmyslové automatizace vždy doménou hrstky inženýrů, kteří se snažili pomoci v řízení značně složitých technologií, na které člověk svými schopnostmi již nestačil. S rostoucí složitostí velkých technologií obsluha už nebyla schopna pojmout obrovské množství údajů, v reálném čase je zpracovat a orientovat technologii tím či oním směrem, aby výsledky výrobního procesu odpovídaly požadavkům zákazníka. V té době nastal „zlatý věk“ konvenčních centralizovaných řídicích systémů. Pod pojmem konvenční řídicí systém si můžeme představit robustní průmyslový počítač, který je schopen přijmout velké množství údajů ze senzorů, zpracovat je a dát zpětně povel akčním členům ke změně. Inženýři měli tu výhodu, že v době, kdy na svět přišly numerické počítače, které byly schopny zvládnout tyto operace, mohli již stavět na teorii automatizovaného řízení, jež v té době byla již velmi dobře zpracována a nabízela celou řadu nástrojů, které stačilo jen uchopit a použít. Pojmy jako fuzzy nebo LQG regulace, které jsou v současné době moderní, byly již dávno na světě. Na první pohled by se mohlo zdát, že i současné požadavky kladené na automatizaci by konvenční řídicí systém byl schopen zvládnout. Čas však ukázal, že s centralizovanou inteligencí bychom dlouho nevystačili. V první řadě to byl požadavek bezpečnosti a modularity, který přispěl k tomu, aby se jeden centralizovaný systém rozdělil na několik menších, navzájem mezi sebou komunikujících. Přestože řídicí vlastnosti byly téměř stejné, zejména větší bezpečnost při poruchách, nižší náklady na opravy i instalaci a menší komplikovanost a lepší celková přehlednost celého systému byly důvody, proč distribuovaná inteligence nastoupila svou vítěznou cestu. A ta neskončila u formálního rozdělení jednoho celku na více menších částí.

Na konci osmdesátých let již byl rozvoj mikroelektroniky natolik daleko, že konečně bylo možné zpracovat signál mikroprocesorem přímo u senzoru a dále ho posílat jako digitální. V tu chvíli začaly ztrácet důležitost jednotlivé moduly automatizace a do popředí se dostala komunikace mezi nimi. Metoda decentralizované automatizace zpočátku narážela na vysoké ceny inteligentních přístrojů. Tento problém se podařilo vyřešit velmi elegantně. To, že vývoj půjde stále dopředu a výrobní cena se bude neustále snižovat, bylo předem jasné. Obchodní blokádu cen výrobků se však podařilo v zájmu zákazníka prorazit tím, že firmy své komunikační systémy standardizovaly a jako otevřené je poskytly libovolnému zájemci, který chtěl vyrábět kompatibilní přístroje. Na

scénu tak vstoupil hlavní fenomén automatizace současnosti, kterým jsou inteligentní otevřené sběrnice.

Komunikační sběrnicí se nazývá fyzické propojení několika zařízení podle komunikačního modelu ISO/OSI (International standard organization/Open system interconnection). Tento komunikační model definuje sedm vrstev, od fyzické (definice přenosového média) až po aplikační (vrstva aplikačních programů), ve kterých je sběrnice standardizována, a tak přístupná všem výrobcům. Jednou ze světově nejvíce rozšířených sběrnic je systém s názvem LonWorks, vyvinutý americkou firmou Echelon.

Sběrnice LonWorks je průmyslová sběrnice navržená s ohledem na specifické potřeby automatizace budov. Lze ji využít k ovládání technických zařízení nejen v jednotlivých obytných nebo účelových budovách, ale i v rozsáhlých komplexech průmyslových podniků, skladišť, letišť, nádraží nebo sportovních areálů. Může zde ovládat osvětlení, vytápění, větrání, klimatizaci, rovněž tak např. pohyblivá schodiště nebo výtahy.

Celá práce je založen na vlastní implementaci prezentační a aplikační vrstvy protokolu LonTalk. Proto rozboru LonTalk protokolu věnuji podstatnou část této práce. Podrobněji jsou rozebrány poslední dvě vrstvy ISO/OSI modelu. Rozbor je zaměřen hlavně na ty části, které jsou nezbytné pro vytvoření aplikace. V současné době neexistuje open source implementace celého LonTakl protokolu. Je k dispozici pouze referenční implementace napsaná v jazyce C, která je dostupná na stránkách firmy Echelon.

Dále se tato práce zabývá implementací nejnižšího ovladače PC104 karty pro síť LonWorks a vytvořením *Host Application* pro tuto kartu. Jak ovladač, tak Host application je implementována pro operační systém Windows a pro operační systém reálného času On-time. Karta dohromady s počítačem je v režimu standardního nódu (nechová se tedy jako manažer sítě). Karta obsahuje firmware MIP/P50 v režimu *Host Selection*, tj. komunikace maximálně 4096 proměnných. Rozborem implementace sítě LonWors se proto zabívá samostatná kapitola. Pro přístup na hardware karty pod Windows se užívá dodávaná DLL knihovna wldv32.dll, které je k volně ke stažení na stránkách výrobce karty, firmy Gesytec. Pro Operační systém On-time jsou nahrazeny funkce knihovny wldv32.dll vlastní implementací funkcí, které knihovna poskytuje.

2. Představení technologie Lonworks

2.1. Komunikační síť LonWorks

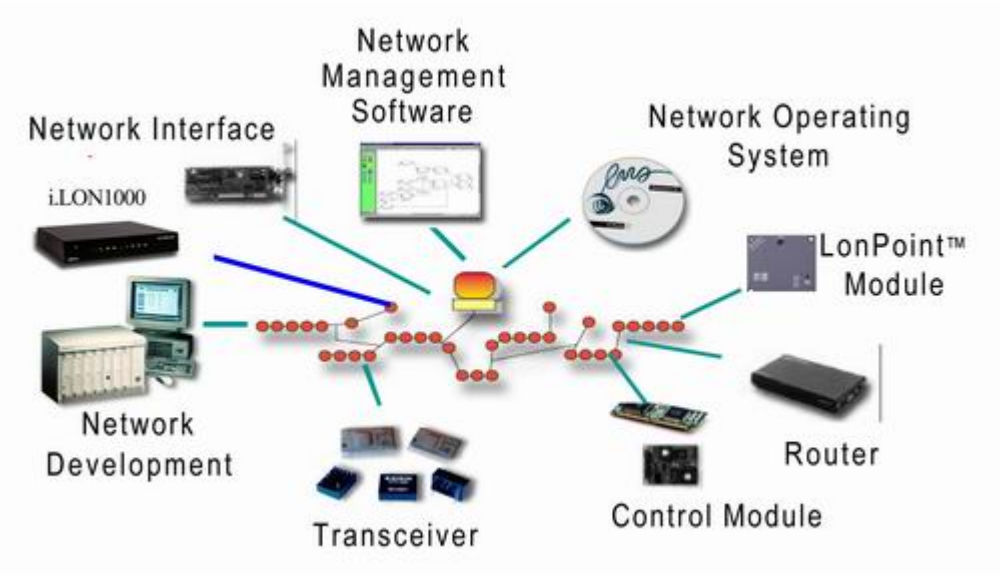
Technologie, která využívá sítě distribuované inteligence pro řídicí aplikace vytvořená americkou firmou Echelon – Local Operating Network (LON), na jejímž základě vznikl komunikační protokol LonWorks. Stěžejními uživateli této techniky se staly průmyslová automatizace a komplexní řešení automatizace technických zařízení budov pro tzv. inteligentní budovy. Ihned za nimi jsou doprava a „digitální“ domácnosti. Je zcela na místě hovořit o novém mezinárodním standardu. Technika LonWorks podporuje řešení řídicích systémů s distribuovanou inteligencí a z toho vyplývající až dosud netušené možnosti využití. Ve světě zaznamenává tato komunikační síť velmi podobný dynamický rozmach jako internet, Wi-Fi atd. Ke spojení mezi jednotlivými segmenty sítě LonWorks řešící danou aplikaci lze využít rozvod elektrické energie, kroucenou dvojlinku nebo např. internet (protokol TCP/IP). Protokol LON byl od prvopočátku určen k použití v řídicích systémech, kde důraz je kladen především na inteligenci každého prvku sítě – tzv. uzlu (node). Síť vytvořená na jeho základě tedy má vlastnosti sítě s distribuovanou inteligencí – podobně jako síť neuronová.

Komunikační síť LonWorks nyní představuje ucelené technické řešení, jež bylo vyvinuto s cílem umožnit realizaci cenově dostupných, inteligentních distribuovaných řídicích systémů, které mohou být tvořeny dvěma až 32 000 zařízeními. Její koncepce, reprezentující dynamický trend, je velmi odlišná od vžitých schémat systémů řízení založených na centrálních řídicích jednotkách. Je možné ji použít v mnoha aplikacích: od řízení technických zařízení budov, přes kopírky a kávovary po dopravní prostředky a průmyslovou automatizaci. Řešení LonWorks je, na rozdíl od většiny jiných systémů, skutečně komplexní, postihující návrh, tvorbu, instalaci i údržbu sítí distribuované inteligence. Technika LonWorks byla vyvinuta proto, aby bylo možné navzájem integrovat zařízení od různých výrobců, aniž by tito museli vědět jeden o druhém. Na rozdíl od jiných síťových systémů je koncipována tak, aby její cena byla nízká a inteligence dostatečná k tomu, aby mohla být distribuována až do každého spínače osvětlení, do každého snímače teploty apod.

2.2. *Protokol LonWorks*

Technologie LonWorks podporuje široké spektrum přenosových médií. To umožňuje konstruktérovi optimalizovat přenos z hlediska maximální propustnosti kanálu při minimální ceně a co nejjednodušší instalaci. V současnosti jsou protokolem a sortimentem směrovačů podporována tato média: kroucený pár vodičů pro přenos dat linkou RS-485, rádiový kanál, elektrorozvodná síť pro napětí 230 V až 10 kV AC, optický kabel, infračervený přenos, napájecí vedení 48 V DC při přenosu napájení i dat po týchž vodičích.

Síť LonWorks (viz. obrázek 2.1) je svojí hierarchií adres podobná sítím IP. Veškeré adresy v síti jsou logické a definují se v okamžiku instalace sítě.



Obr. 2.1 Příklad možností sítě LonWorks

2.3. *Architektura, otevřenost, řídicí systém*

Základní myšlenkou metody LON je „rozložení“ dosavadního jediného zařízení či systému na skupinu inteligentních prvků (uzlů), které po propojení komunikačním médiem vytvoří síť. Síť LonWorks ke své činnosti nevyžaduje centrální prvek ani nevyužívá architekturu řídicí-řízený (master-slave). Inteligentní uzly komunikují navzájem mezi sebou, a jde tedy o síť typu peer-to-peer. Typický uzel sítě uskutečňuje

pouze jedinou jednoduchou úlohu, přičemž zařízení zapojená v síti jsou samostatné prvky. Komplexní řídicí činnost vykonává teprve síť jako celek. Správného chodu takto decentralizované sítě lze dosáhnout pouze vestavbou „inteligence„ přímo do uzlu. Jádrem každého uzlu je proto tzv. neuronový chip. Jde o tříprocesorový mikropočítač s firmwarem, který zajišťuje síťovou komunikaci komunikačním protokolem LonTalk a základní chod aplikačního programu.

Technologie LonWorks poskytuje téměř neomezený počet stupňů volnosti a systém zcela otevřený pro budoucí rozšiřování. Například libovolný další prvek přidaný v budoucnu bude moci využívat již nyní realizované měřicí body, a to pro libovolné budoucí řídicí funkce.

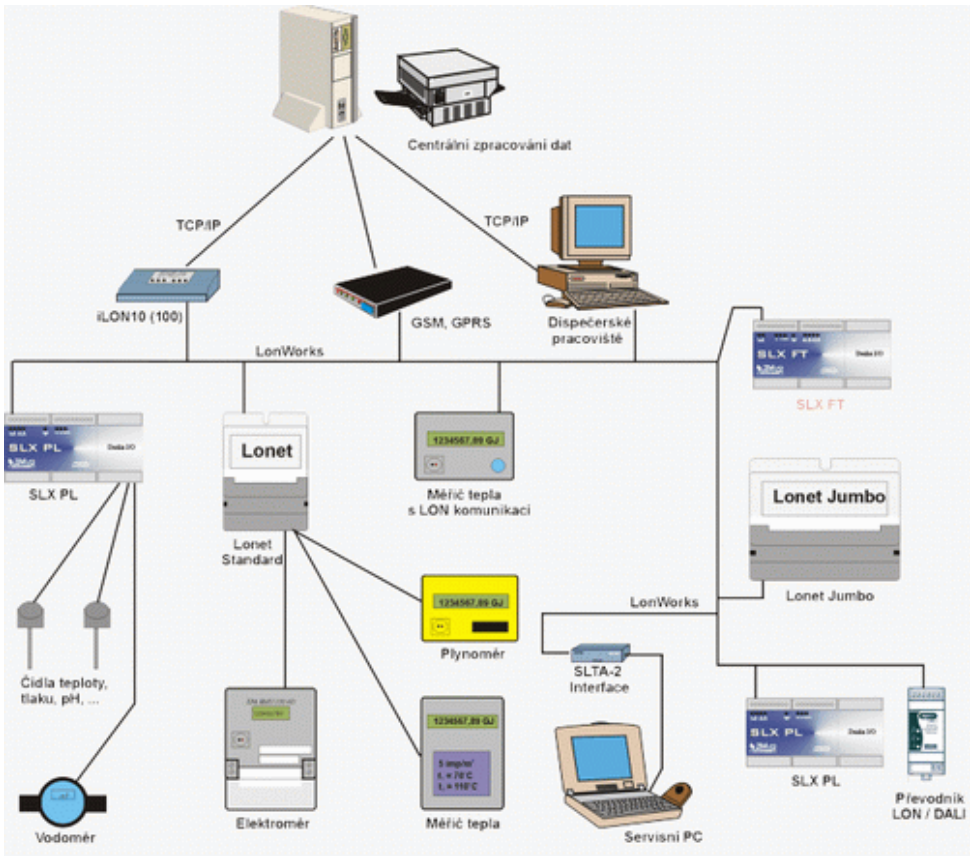
Je-li k síti LonWorks připojen počítač (PC), lze celou síť konfigurovat a řídit a v neposlední řadě také zobrazovat řízené procesy. Při použití vizualizačního programu (např. LonMaker) lze monitorovat jednotlivé subsystémy sítě (např. v budově: přístupový systém, vzduchotechnika, vytápění atd.) jedním programem na jediném PC. Současně je možné na tomto PC realizovat důležité vyhodnocovací algoritmy, jejichž složitost převyšuje možnosti zařízení zapojených v síti LonWorks. Počítač se připojuje k síti buď přes rozhraní RS-232 a sériový adaptér (Serial LONTolk Adapter – SLTA), nebo přes rozhraní USB, případně zvláštní kartou, která se zasune do volného slotu PC.

2.4. Spolupráce zařízení od různých výrobců v jedné síti

Uzly s vlastnostmi umožňujícími jejich vzájemnou koexistenci na jedné síti se označují jako interoperabilní. K zajištění interoperability musí výrobci uzlů respektovat množství doporučení, jež jsou rozdělena do několika úrovní. Aby byla doporučení řádně specifikována a byla možná kontrola vlastností nabízených produktů nezávislá na jejich výrobcích, byla zřízena nadnárodní asociace LonMark. Jejím úkolem je vydávat a aktualizovat již zmíněná doporučení a výrobkům (uzlům), které je splňují, přidělovat značku LonMark. Jestliže tedy daný uzel vlastní tuto značku, není důvod se obávat o jeho kompatibilitu se sítí LonWorks. Součástí činnosti asociace LonMark je rovněž standardizace typů nejčastěji používaných síťových proměnných. Proto byl vytvořen seznam standardních typů síťových proměnných (Standard Network Variable Type – SNVT [6]), jehož použití velmi usnadňuje spolupráci různých uzlů v jedné síti.

2.5. *Možnosti použití*

Koncept LON je v současné době podporován a využíván asi 4 000 výrobci a organizacemi. Z největších jsou to např. ABB Network Partner, Allen-Bradley, AT&T, British Petroleum, British Telecom, General Electric, Goldstar Industrial Systems, Hewlett-Packard, Hitachi, Johnson Controls, KABA Benzing, Lexel, Microsoft Corporation, Mitsubishi, Molex, Motorola, NASA, Olivetti, Philips Lighting, Sauter, Schneider Electric, Siemens, Toshiba, ZPA a mnoho dalších. Jejich počet stále roste, stejně jako počet aplikací. Na následujícím obrázku je příklad použití síťě LonWorks (Obrázek 2.2).



Obr. 2.2 Příklad použití sběrnice LonWorks a systému LONET od ZPA Trutnov

Teoreticky lze technologii LonWorks využít v každém zařízení. V praxi je však její použitelnost omezena na oblasti, kde není požadována výměna dat rychlostí větší než

2,5 Mb/s a kde postačuje doba odezvy sítě v rozmezí 3 až 10 ms. Technika LonWorks tak v současné době nachází uplatnění téměř všude, namátkou:

- v budovách při řízení technických zařízení budov (sledování a řízení spotřeby energií, klimatizace, požární a zabezpečovací signalizace, v elektronických zámcích, ve výtazích, při ovládání žaluzií atd.),
- v kancelářských přístrojích (kopírky, faxy apod.), v domácích spotřebičích, v zařízeních pro automatickou úpravu životního prostředí,
- v platebních systémech supermarketů, v nápojových a jídelních automatech, v identifikačních systémech,
- v průmyslu při řízení výrobních linek i v jednotlivých strojích, v inteligentních průmyslových snímačích (digitální elektroměry, snímače tlaku, hladiny, teploty atd.) a akčních členech (řízení ventilů a pohonů, ovládání spínačů), v testerech elektronických obvodů,
- v dopravních prostředcích (automobilech, tramvajích, na železnici) i v dopravní infrastruktuře (řízení křižovatek, řízení provozu a vybírání poplatků na dálnicích a silnicích),
- ve zdravotnictví v přístrojích pro monitorování stavu pacientů,
- v řízení venkovního osvětlení i osvětlení interiérů,
- v zavlažovacích systémech atd.

Například při použití sítí LonWorks ke sledování a řízení veřejného osvětlení v městských aglomeracích je každý sloup se svítidlem vybaven neuronovým chipem. Ten informuje centrálu o aktuálním stavu svítidla a současně umožňuje na dálku ho rozsvěcet a zhasínat podle požadavků centra nebo podle intenzity okolního osvětlení. Vzniká tak velmi efektivní řízení veřejného osvětlení přinášející značné úspory energie.

Velmi zajímavé možnosti nabízí technika LonWorks v kombinaci s internetem např. teplárenským společnostem a firmám provozujícím tzv. facility management. Zde je možné ji využít v podobě poměrně jednoduchého řešení založeného na síti LonWorks a jednom PC namísto nákladného udržování složitých dispečinků. Jak je známo, lze v současné době bezproblémově a především poměrně levně sledovat systémy na dálku při použití PC z internetové kavárny, přístrojů typu PDA nebo výkonnějších mobilních telefonů (prostřednictvím programů založených na funkci appletů Java, obsažených v každém operačním systému Microsoft Windows).

V nepříliš vzdálené budoucnosti bude s technikou LonWorks možné splnit i sny o plně automatizované domácnosti – od kávovaru, přes vařič, chladničku, vytápění, domácí vodárnu a čističku odpadních vod až po zabezpečení objektu. Existuje unikátní možnost vzájemného datového propojení všech uvedených i dalších přístrojů a zařízení. Až se jednou ráno vzbudíte zvukem z rádia, než se osprchujete, celý dům „sám„ ožije – zapne se topení, kávovar připraví oblíbený nápoj a PC automaticky vyzvedne zprávy z internetu. Po snídani zaklapnete dveře svého domu, který se v tu chvíli stane dalším zabezpečeným objektem ve městě, dojedete do zaměstnání, kde bude stačit protáhnout kartu čtečkou a vše kolem začne fungovat tak, jak má. Možná za tím vším, aniž byste to věděli, bude právě technika LonWorks.

2.6. Přednosti produktů kompatibilních s LonWorks

Síť LonWorks je prudce se rozvíjející řídicí síť, v celosvětovém měřítku podporovaná mnoha velkými firmami. Již nyní je síť LonWorks celosvětovým standardem. Proč nevyužít praxí ověřený komunikační protokol, který dokáže spojit výrobky různých výrobců. A nejenom to. Proč nevyužít celou podporu, kterou firma Echelon nabízí.

Jestliže jakýkoliv systém podporuje standard LonWorks, automaticky se stává součástí velké „skládačky„ a tím součástkou řešící konkrétní problematiku v rámci velkého projektu. V tu chvíli může na jednom fyzickém objektu koexistovat množství dalších subsystémů, původem i od jiných výrobců, které společně komplexně vyřeší požadavky zákazníka. Každý z těchto subsystémů se totiž velmi snadno stává součástí otevřeného systému. Společně s jednodušší instalací (stačí jedno vedení pro několik subsystémů) a vyplývajícími podstatně menšími instalačními náklady při zachování možnosti systém v budoucnu snadno modifikovat podle požadavků zákazníka (i jen programovým přepojením síťových proměnných) tak existuje další pádný argument, který by měl přesvědčit investora, že právě systém založený na technice LonWorks je ten pravý.

Další výhodou je, že síť LonWorks je sítí distribuované inteligence. Není tedy nutný výkonný – a drahý – centrální řídicí prvek. Je tomu tak proto, že síť může základní funkce vykonávat sama. Protože každý uzel sítě má vlastní inteligenci, lze tyto uzly

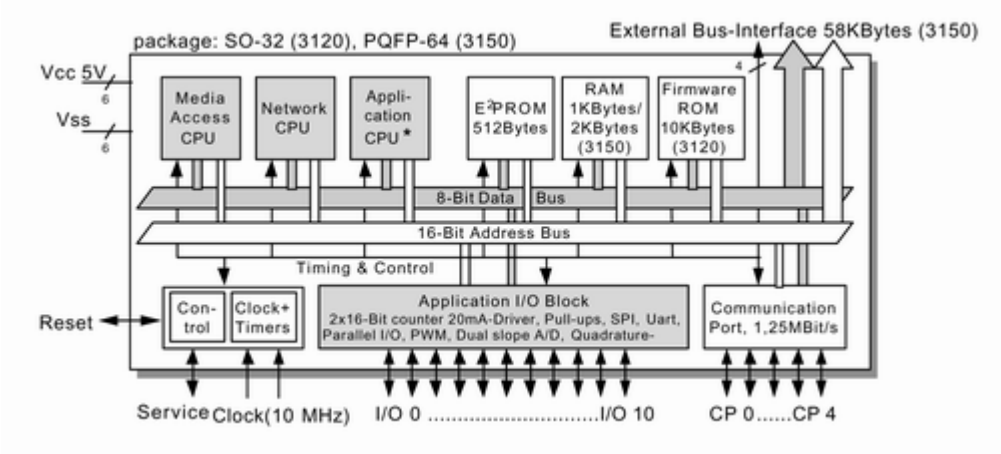
řídit. Toho se mnohdy velmi efektivně využívá (např. při měření a hlídání spotřeby energií).

Závěrem lze říci, že zařízení schopná pracovat v síti LonWorks přinesou uživateli:

- finanční úsporu, primární (investice do instalace) či sekundární (úspora energie, času, práce atd.),
- svobodu, tj. skutečnost, že za své peníze zákazník získává systém snadno modifikovatelný podle jeho požadavků,
- otevřený systém, který umožní později přidat další subsystémy s již minimálními náklady,
- velmi výkonný systém, díky distribuované inteligenci.

3. Neuron chip

Neuron chip je hlavní hardwarovou komponentou každého uzlu. Mezi hlavní, od začátku na výrobě neuron chipů podílející se společnosti, patří firmy Toshiba a Cypress. Na následujícím obrázku 3. je zhruba blokové schéma vnitřku takového Neuron chipu. Konkrétně by mělo odpovídat chipům Toshiba. Příklad některých vyráběných typů je na obrázku 3.1 a jejich stručný bližší popis v tabulce 3.1.



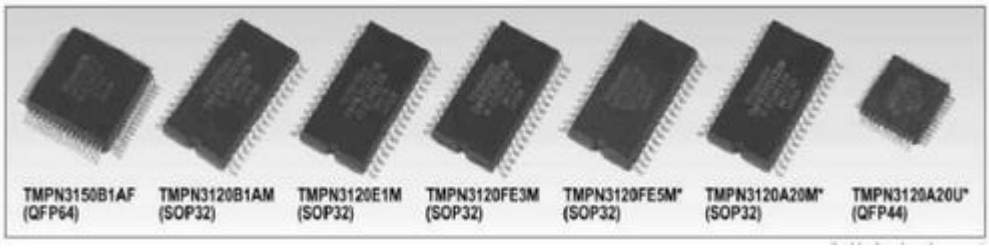
Obr. 3.1 Blokové schéma vnitřku Neuron chipu.

Základem každého aktivního chipu je CPU a paměť. V Neuron chipu se obvykle vyskytují tři nezávislé CPU vykonávající následující operace:

- ***Media access CPU*** **(CPU pro přístup na médium)** - to ovládá a řídí všechny sériové komunikační porty (Communication Port, 600 až 1.25MBit/s na obr. 3.) na úrovni Linkové vrstvy (Data Layer) OSI modelu komunikace. Model je popsán v kapitole Lon Talk. Na výstupním portu je již kompletně "zapouzdřený" signál do paketu protokolu LonTalk, který je přímo určený pro přenos do dalšího uzlu. CPU má tedy na starosti i buferování vstupů a výstupů, řízení vysílání paketů dle priority, detekce kolize na sběrnici, tvorbu rámců, opravný kód CRC apod.

- ***Network CPU*** **(síťový CPU)** - to zpracovává veškeré informace a požadavky na služby poskytované protokolem LonTalk na úrovni síťové vrstvy (Network layer) OSI modelu komunikace. řídí časovací služby využívané v různých stavech zpracování signálů, adresování uzlů a správné směrování paketů apod.
- ***Application CPU*** **(aplikační CPU)** - to provádí zpracování dané uživatelské aplikace napsané jazykem Neuron C. Přeložený a slinkovaný program se do neuron chipu může přenést po síti prostřednictvím komunikačního portu. Toto CPU však nemusí zastávat uvedenou funkci a může se použít libovolný externí CPU či MCU, které bude data, například ze senzorů, zpracovávat a poté vysílat přes I/O porty do neuron chipu na vyslání. Poté aplikační CPU vykonává jen funkci zprostředkovatele dat z I/O portů pro síťový CPU.

Uživatel / programátor má však možnost příkazy přímo ovládat jen aplikační CPU. Zbylá CPU již pracují samostatně automaticky dle vnitřního firmwaru a parametrů v programu pracovaných v aplikačním CPU. Pro uložení aplikačního programu a aktualizovatelné části firmwaru neuron chipu se využívá vnitřní Flash EEPROM paměti, případně externí paměti (dle konkrétního typu Neuron chipu). Vnitřní RAM slouží klasicky pro datové proměnné aplikačního programu a část také jako prostředek pro uchování dočasných dat komunikace (například pro fronty komunikačního portu). Vnitřní ROM obsahuje pevnou neměnnou část firmwaru neuron chipu. Externí softwarově ovládané I/O piny chipu mohou obvykle sloužit k libovolné komunikaci s okolím nebo pro monitorování firmwaru či čtení ID neuron chipu.



Obr. 3.2 Některé Neuron chipy vyráběné firmou Toshiba

Product No.	EEPROM (in bytes)	RAM (in bytes)	ROM (in bytes)	8-bit CPU	External memory I/F	16-bit timer/ counter	A/D CONVERTER	Maximum operation Frequency (MHz)	Package
TMPN3150B1AF	512	2K	No	3	Available	2 ch	-	10	QFP64-P-1414 -0.80A
TMPN3120B1AM	512	1K	10K	3	No	2 ch	-		SOP32-P-525 -1.27
TMPN3120E1M	1K	1K	10K	3	No	2 ch	-		SOP32-P525 -1.27
* TMPN3120A20U	1K	1K	16K	3	No	2 ch	3 ch	20	QFP44-P-1010 -0.80
* TMPN3120A20M									SOP32-P-525 -1.27
TMPN3120FE3M	2K	2K	16K	3	No	2 ch	-		SOP32-P-525 -1.27
* TMPN3150FE5M	3K	4K	16K	3	No	2 ch	3 ch		SOP32-P-525 -1.27

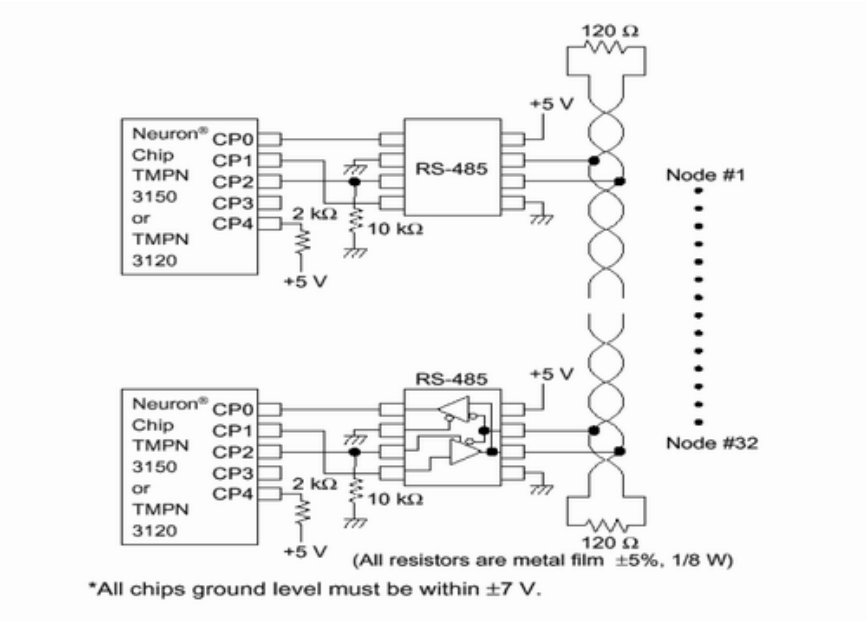
Tab.3.1 Stručné parametry Neuron chipů Toshiba

3.1. IO pro přístup na médium - transceivery

Pro úspěšnou komunikaci po daném typu sběrnice či sítě je však nutné na výstup komunikačního portu Neuron chipu připojit buď daný transceiver nebo někdy jen oddělovací transformátor, který bude zajišťovat ochranu neuron chipu, přizpůsobení nebo modulaci signálů pro daný typ fyzického média (koax, dvojlinka, optika, radiový přenos, přenos po síťovém vedení apod.). V tabulce 4.1 jsou uvedeny některé integrované obvody transceiverů, vyráběné nebo podporované firmou Echelon, pro médium typu vodič a pro různé uspořádání a konfiguraci sítě (sběrnice - Line, volná topologie - free apod.). Popis topologií podporovaných standardem LonWorks je v kapitole Lon Talk. Z hlediska napojení různých transceiverů, neuron chip obsahuje 5 vývodů komunikačního portu, které lze široce softwarově konfigurovat jak po stránce funkce (klasický sériový signál proti signálové zemi, nebo pro diferenční signál apod.), tak i po stránce nastavení komunikační rychlosti v malých krocích od 600 bit/s do 1,25 Mbit/s. Proto je možné použít různé tranceivery od různých výrobců.

3.1.1. Transceiver RS - 485 / RS-422

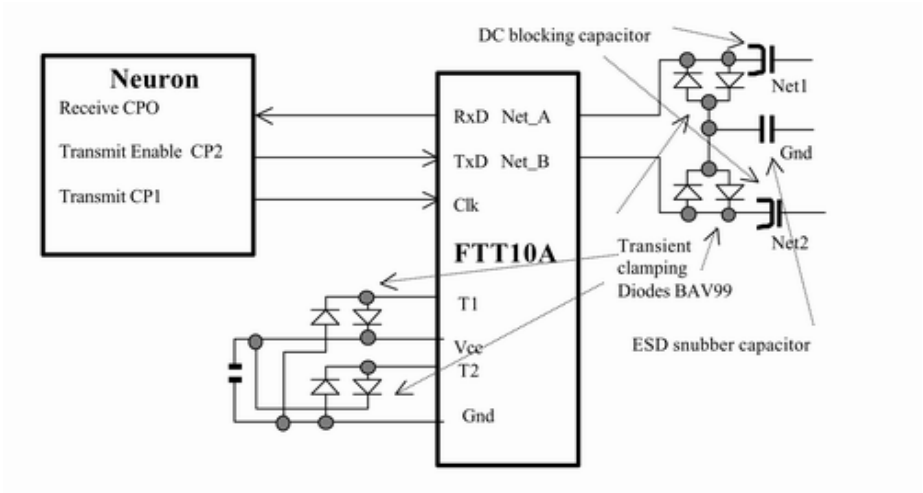
Jako pro přístup na sběrnici RS-485 lze využít libovolné transceivery podporující standard EIA-485, tedy i například známé transceivery firmy Maxim. Příklad propojeni neuron chipu s transceiverem RS-485 je na následujícím obrázku 3.3.



Obr. 3.3 Příklad připojení Neuron chipu na sběrnici RS-485 prostřednictvím RS-485 transceiveru

3.1.2. Free topology transceiver - transceiver pro libovolnou topologii sítě)

V nabídce firmy Echelon je několik integrovaných obvodů transceiverů vhodné pro LonWorks sítě libovolné topologie (tvaru). Na obrázku 3.4 je příklad propojení neuron chipu s transceiverem FTT10A. činnost obvodu lze přiblížit jeho označením jako integrovaný oddělovací transformátor, který chrání neuron chip před zničím prostřednictvím zkratu, nějakého výboje nebo přepětí v síti. Tento tranceiver je určen pro fixní přenosovou rychlost 78 kbit/s.



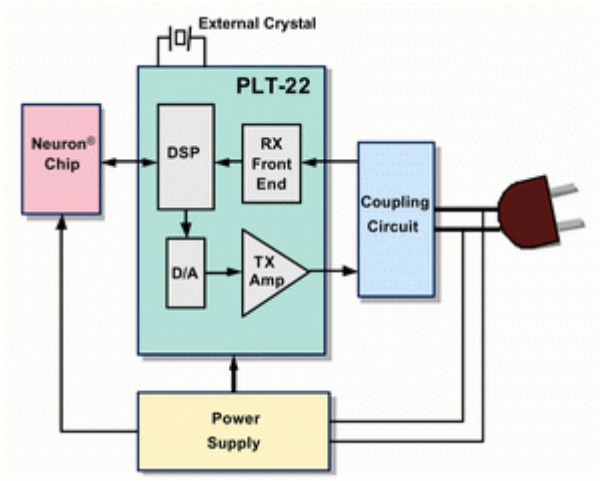
Obr. 3.4 Příklad propojení Neuron chipu a transceiveru pro libovolnou topologii sítě

3.1.3. Link Power Transceiver - transceiver pro přenos dat po napájecím vedení

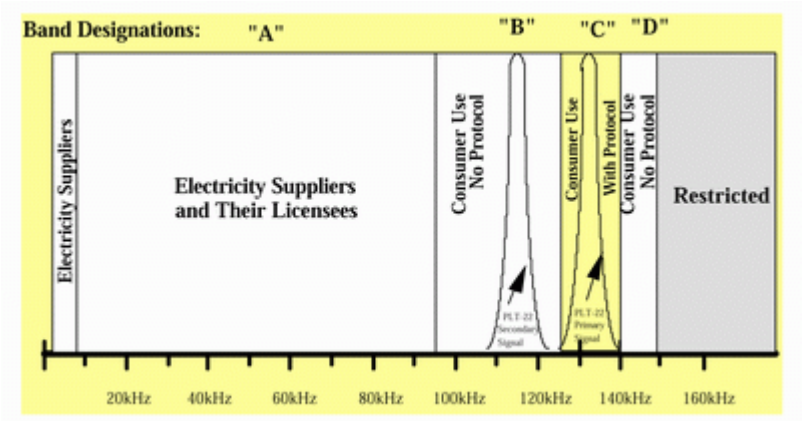
V nabídce firmy Echelon lze například najít transceiver LPT-10, který umožňuje přenášet data a po jedné zkroucené dvojlince (twisted pair) společně s napájením uzlu +5 V DC & 100 mA. Vzdáleně to lze přirovnat k telefonní lince, kde společně se signálem se přenáší i napájecí napětí telefonního přístroje. LPT-10 rovněž umožňuje kombinovat libovolné topologie v síti LonWorks a v datové komunikaci je přímo kompatibilní s předchozím transceiverem FTT10A.

3.1.4. Powerline Transceiver (EIA 709.2-A-2000) - transceiver pro přenos dat po síťovém napájecím vedením 230V

Transceiver PLT-22 umožňuje provádět komunikaci po klasickém síťovém měděném vedení 230V splňující standard EIA 709.2-A-2000. Blokové zapojení uzlu LonWorks sítě pro tento transceiver je na obrázku 3.5 Komunikace splňuje regule FCC (USA) a European CENELEC EN50065-1 pro signalizaci a přenos dat pro síťovém napájecím vedení v pásmu 125 až 140 kHz - viz. obrázek 3.6 Taková to komunikace má však pomalý přenos dat pouze přenosovou rychlostí 4,8 kbit/s, což je značně nevýhodné. Naopak lze takto vytvořit síť pro řízení budov a provozů bez nutnosti natahování nové kabeláže, což představuje finančně výrazně nižší náklady.



Obr. 3.5 Blokové schéma uzlu pro komunikaci po síťovém napájecím vedení



Obr.3.6 Rozložení frekvenčních pásem pro komunikaci po síťovém nap. Vedení

Technologie LonWorks nabízí univerzální komunikaci vhodnou nejen pro řízení spotřebičů a automatizaci budov (klimatizace, topení, světlo apod.), ale i regulaci v průmyslu. Jako síťový protokol je použit LonTalk, který je jako firmware součástí každého uzlu a umožňuje přenos dat po libovolném médiu a topologii sítě.

4. LonTalk

4.1. OSI model

Tento protokol, který byl navržen v roce 1989 firmou Echelon a standardizován jako EIA 709.1 Standard, definuje přístup na sběrnici a řízení přenosu paketu (zpráv - messages) po existující síti. Síťový protokol LonTalk byl navržen dle ISO OSI referenčního modelu (viz. obrázek 4.1). To umožňuje programům běžícím na aplikačním CPU komunikovat s aplikací běžící na jiném uzlu tvořeného Neuronovým chipem kdekoliv ve stejné síti. Služby protokolu, které jsou vyvolávány programy a objekty pracující na aplikační hladině OSI modelu.

OSI vrstva	**funkce**	**služby**
aplikační	aplikační program	interpretace síťových proměnných, služby sítě
prezentační	interpretace dat	příjem a odeslání síťových proměnných, explicitních zpráv a cizích rámců, apracování konfiguračních zpráv
relační	řídí spojení	autentizace, služba dotaz/odpověď
transportní	doručení paketů	služba bez potvrzování, s potvrzováním a bez potvrzování s opakováním
síťová	způsob adresování	unicast, multicast, broadcast
linková	přístup ke sběrnici	provádí konverzi datových typů a struktur mezi komunikační a aplikační vrstvou
fyzická	elektrické propojení	zajišťuje přenos signálu komunikačním kanálem

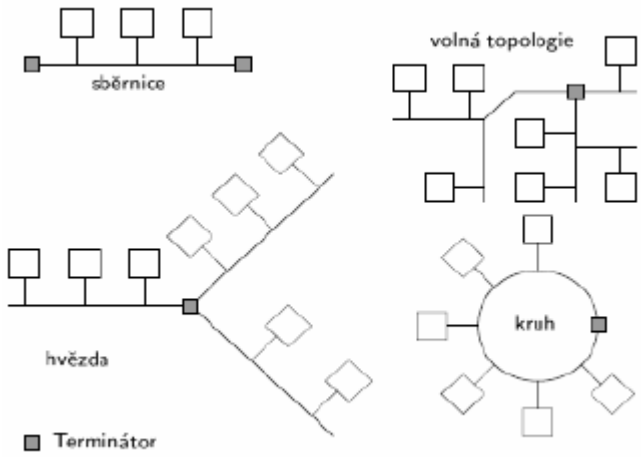
Obr. 4.1 OSI model pro LonTalk protokol

4.2. Fyzická vrstva OSI modelu (Physical OSI layer)

Fyzická vrstva definuje propojení po fyzickém komunikačním médiu. Již zmíněnou velkou výhodou protokolu LonTalk je možnost přenosu po libovolném médiu, pro který existuje transceiver. Ten je přímo napojen na k tomu určeným pinům neuron chipu.

Dohromady tedy tvoří uzel sítě. V současné době se využívají tyto média, resp. existují pro ně transceivery vhodné pro napojení na neuron chip (viz. obrázek 4.2 a tabulka 4.1):

- Manchesterem kódovaný signál po zkroucený pár vodičů (twisted wires pair) izolovaný transformátorem a sběrnicovou topologií s rychlostí přenosu 78kbps až 1.25Mbps
- Zkroucený pár vodičů libovolnou topologií (kruh, hvězda, sběrnice) a kombinací kabelů s průřezem od 0.65 až 1.3mm
- Napájený zkroucený pár vodičů s libovolnou topologií se společným přenosem dat a napájení libovolného zařízení (ostatní viz bod výše)
- Výkonový síťové vedení, kde se společně přenášejí data i střídavé, příp. stejnosměrné, napájení (například síťový rozvod 230V). Tato komunikace se v Evropě řídí standardem CENELEC.
- Optické kabely v podobě dvou vláknového přenosu (dopřenný a zpětný přenos po samostatném vláknu) nebo jednovláknový přenos (oba směry se šíří po jednom vláknu)
- Radiový (RF) přenos s využitím frekvencí 49MHz, 400 - 450MHz, 900MHz s rozprostřeným spektrem, 1.2GHz s rozprostřeným spektrem nebo 2.4GHz s rozprostřeným spektrem.
- Infračervený (IR) přenos obvyklý a vhodný pro přenosové aplikace
- Koaxiální kabel vhodný pro vysokorychlostní přenos nebo ve spojení s přenosem dat, obrazu i zvuku



Obr. 4.2 Příklady typů topologií LonWorks sítě

Samozřejmostí je využití a navázání více druhu médií, kdy například od vzdálené řídící stanice s využije síťové vedení a pak pro místní přístroje zkroucený pár. Zároveň je podporovány tzv. vícenásobné komunikační kanály (multiple communications channels), kde pojem kanál (channel) se uvažuje fyzické transportní médium pro datové pakety (telegramy) s možností připojit až 32385 uzlů. Každá síť může být složena z jednoho nebo více kanálů tvořených i různými fyzickými médii. Pro přenos z jednoho kanálu do druhého se využívají routery.

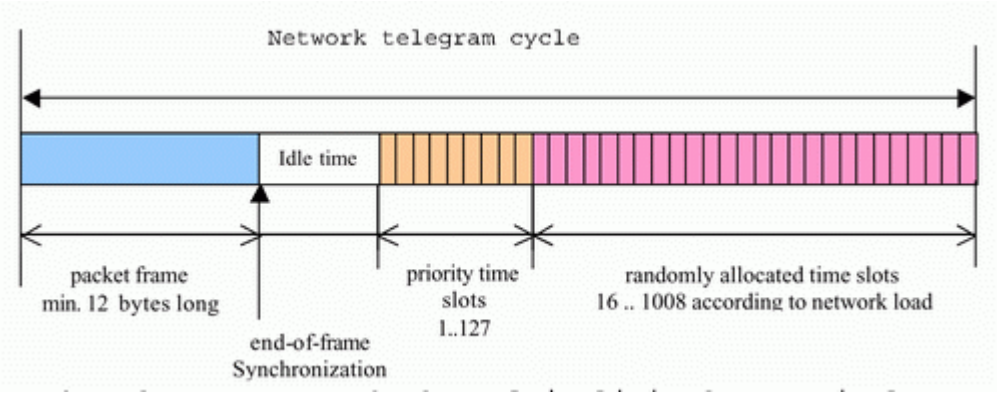
název transceiveru	rychlost sítě	topologie sítě	nódů na kanál	vzdálenost mezi nódama	typ izolace
RS485	<=1Mb/s	sběrnice	32	1400m	volitelná
TP/XF1250	1,25Mb/s	sběrnice	64	130m	transformátor
TP/XX78	78kb/s	sběrnice	64	1400m	transformátor
FTT10A	78kb/s	sběrnice	64	2700m	transformátor
FTT10A	78kb/s	volná	64	500m	transformátor
LPT10	78kb/s	sběrnice	128	2200m	transformátor
PLT22	4,8kb/s	libovolná	-	5000 m	volitelná

Tab. 4.1 Některé transceivery pro síť LonWorks z nabídky firmy Echelon

4.3. Linková vrstva OSI modelu (Data Link OSI layer)

Linková vrstva ovládá a řídí přístup na médium (viz. obrázek 4.3) a provádí kódování dat pro případnou opravu chyby vzniklé přenosem. Zde se využívá cyklický kód CRC (viz. obrázek 4.4).

4.3.1. Přístup na médium



Obr. 4.3 Schéma přístupu daného uzlu na sběrnici - metoda CSMA/CA

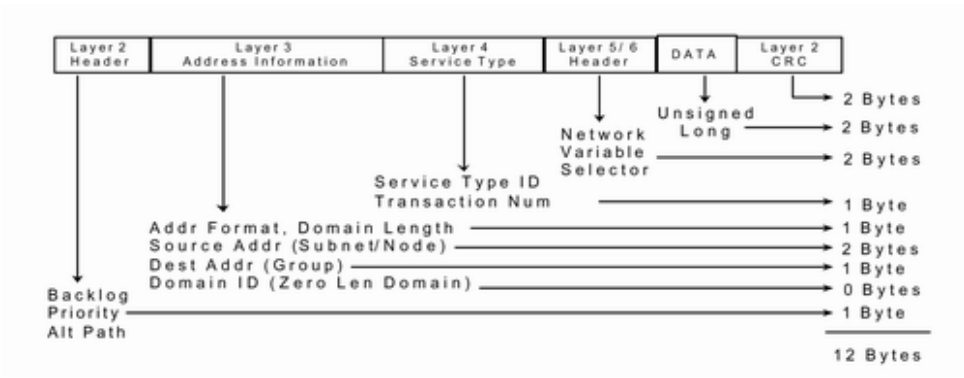
Pro přístup se zde využívá známá metoda CSMA/CA pro přenášení paketů dle obrázku 4.3. Všechny uzlu, resp. jejich neuron chipy sledují přenos po síti a pro jejich přístup čekají na stav nečinnosti (Idle state), kdy nikdo nevysílá. Vysílání předchozího uzlu je ukončeno synchronizačním bitem, tzv. End-of-frame Synchronization, ukončující přenášený rámec. Pak každý uzel odpočítává tzv. Priority time slots, kdy mohou určité uzly nebo zprávy míst vyšší prioritu než další a tímto způsobem se na sběrnici dostanou přednostně dříve, protože je jim odpočítáván kratší čas. Každý neuron chip uzlu má proto několik front s rozdílnou prioritou pro rozdělení priorit jednotlivých paketů. Pak následuje čekání dle náhodně vygenerované doby, tzv. randomly allocated time slots, a pokud se do té doby neobjeví na sběrnici komunikace, vyšle uzel svůj paket. Počet čekacích slotů se automaticky zvyšuje s rostoucí vytížeností sítě. Je možné využít i detekce kolize, ale tu musejí pak podporovat všechny transceivery zapojené v síti.

Rychlost přenosu [kBd]	Propustnost sítě[pakety/s]
9,8	100
15,9	192
39,1	337
78,1	410
156,3	508
312,5	615
625	696
1250	1021

Tab. 4.2 Datová rychlost přenosu v porovnání s propustností sítě (počet paketů / s)

4.4. Síťová vrstva OSI modelu (Network OSI layer)

Síťová vrstva je zodpovědná za správné doručení paketu cílovému uzlu nebo více uzlům.



Obr. 4.4 Rámec LonTalk protokolu

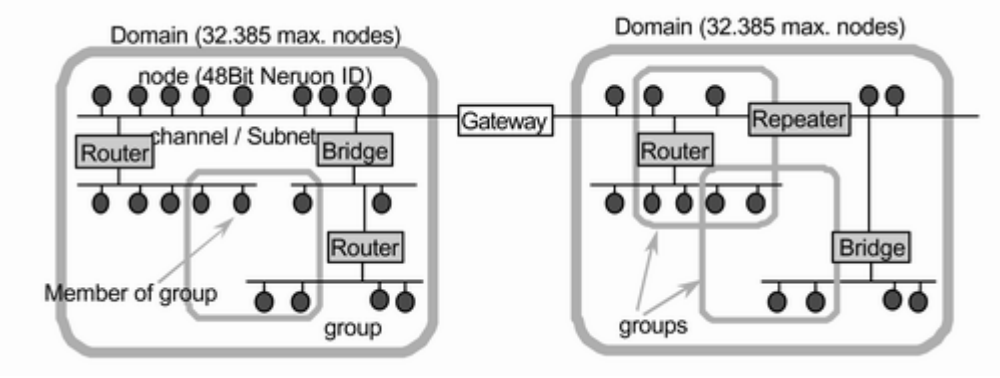
4.4.1. Adresování sítě

Zde se využívá 3-úrovňové adresace k identifikování daného uzlu (viz. obrázek 4.5). První úroveň hierarchie je doména (domain) jejíž identifikátor je délkou volitelný mezi hodnotami 0, 1, 3 nebo 6 bajtů. Každý uzel může být členem maximálně dvou domén. K propojení domén slouží brány (Gateways).

Druhá úroveň adresování je podsíť (subnet). V každé doméně muže být až 255 podsítí. Podsíť je tvořená logickou skupinou uzlů z různých kanálů (adresování samotných uzlů je kanálově nezávislé). K propojení podsítí slouží routery.

Třetí úroveň (nejnižší) tvoří samotný uzel (node), který je adresován 48-bitovým identifikačním číslem (Neuron ID). V každé podsíti může být až 127 uzlů, tzn. až 32385 uzlů v jedné doméně. Protože každý uzel může být zároveň členem dvou domén, může sloužit jako mezidoménová brána a posílat například data z jednoho senzoru do dvou domén najednou.

K adresování více zařízení najednou slouží skupina (group). Každé zařízení smí být členem maximálně 15 skupin. Zařízení nemusí být v jedné podsíti, ale musí být v jedné doméně. Adresování pomocí skupin je velmi užitečné pro koncept síťových proměnných. V rámci jedné domény může být definováno až 255 skupin.



Obr. 4.5 Adresování uzlů v síti

Obrázek 4.5 ukazuje příklad adresování v síti. Můžeme zde vidět, že routry nemohou být členy podsítě. Bridges obvykle slouží ke spojení dvou různých kanálů. Gateway se používá ke spojení domén.

Každá zpráva zaslaná protokolem LonTalk (Network Protokol Data Unit –NPDU) obsahuje adresu odesílatele a adresu příjemce v jednom z pěti možných formátů (tabulka 4.3). Typ 0-2b využívají logické adresy, pouze typ 3 využívá fyzickou adresu (48 bit Neuron-ID)

Typ	Formát adresy	Velikost adresy (bytes)	Příjemce
0	doména (podsíť =0)	3	Všechy uzly v doméně
0	Doména – podsíť	3	Všechny uzly v podsíti
1	Doména – skupina	3	Všechny uzly ve skupině
2a	Doména – podsíť – uzel	4	Jeden uzel v podsíti
2b	Doména –podsíť – uzel-skupina	6	Jeden uzel ze skupiny vyžaduje potvrzení
3	Doména – Neuron-ID	9	Jeden uzel

Tab. 4.3 Typy adresování [2]

4.5. Transportní vrstva OSI modelu (Transport OSI layer)

Transportní vrstva zajišťuje spolehlivost doručení paketů, tj.provádí kontrolu správného přenosu paketů sítí od vysílajícího uzlu k cílovému, zajišťuje potvrzování přijetí paketu, ničí duplikátně vyslané pakety a další služby. Blíže bych se zastavil u čtyřech základních:

4.5.1. Služba potvrzování došlého paketu či zprávy (End-to-End Acknowledged service)

Po zprávě, resp. paketu, vyslaném uzlem sítě dalšímu uzlu nebo skupině uzlů se vždy očekává zpětné potvrzení o úspěšném doručení (acknowledgement) od každého uzlu. Jestliže vysílající uzel nedostane potvrzení od všech příjemců, vyčká určitou nastavenou dobu a provede nové odeslání zprávy. Doba i maximální počet pokusů o znovu vyslání je nastavitelný. Potvrzení o přijetí zprávy, resp. vyslání potvrzení, se provádí automaticky neuron chipem daného uzlu. K zamezení duplikátního příjmu tytéž zprávy se využívá číslování zpráv a potvrzení tzn. Transaction ID číslem.

4.5.2. Služba Dotaz/Odpověď(Request/Response)

Ta je využívána k vyslání zprávy jednomu či více uzlům, od kterých se očekává zaslání nějaké konkrétní odpovědi. Ta může obsahovat i přenášená data, což se využívá při volání vzdálených procedur nebo v client/server aplikacích. Tato služba se využíva zejména při konfiguraci zařízení. Příchozí zpráva je uzlem nebo externí aplikací uzlu zpracována a výsledek vyslán jako odpověď s určitým nastavitelným časovým zpožděním.

4.5.3. Služba nepotvrzeného zasílání zpráv s opakováním (Unacknowledged repeated service)

Toho se využívá při jednom či opakovaném hromadném zasílání zpráv velkému počtu uzlů, kdy by hromadné odpovědi od každého zahltilo síť. Opakováním se zvyšuje spolehlivost doručení zprávy. Nepotvrzované zasílaní se používá pro dosažení velké přenosové rychlosti.

4.5.4. Služba nepotvrzeného zasílání zpráv (Unacknowledged service)

Tato služba je podobná předchozí, tzn. od vyslané zprávy se neočekává ani odpověď ani potvrzení. Je možné ji zasílat i pouze jednomu uzlu sítě. Tak to komunikující aplikace však musí být odolné proti ztrátě paketů či zpráv. Tato služba se využívá zejména pro přenos konfiguračních parametrů do zařízení.

4.5.5. Autentizace

Autentizace dovoluje příjemci ověřit identitu odesilatele. Pouze zprávy zasílané službou s potvrzováním nebo službou dotaz/odpověď mohu být autentizovány. Pro autentizaci se používá 48-bitový klíč a náhodně vygenerované číslo pro danou transakci. Klíč musí být uložen na serveru i u klienta a lze jej změnit pomocí konfiguračního nástroje, toto musí být ale prováděno velmi opatrně, protože nový klíč je zasílán nešifrovaný. Podrobný popis mechanismu lze nalézt v [8], nebo v [2], kde je také uveden kódovací algoritmus.

4.6. Relační vrstva OSI modelu (Session OSI layer)

Pakety mohou použít třídu služeb známé jako žádost-odpověď (request-response) požadující akci od nějakého vzdáleného uzlu. LonTalk relační vrstva definuje standardní kódy zpráv pro síťový management (network management messages) a diagnostiku (network diagnostic messages). Network management messages usnadňují instalaci a řízení sítě, kde příkazy umožňuje měnit nastavení a konfiguraci neuron chipů, resp. obsah jejich EEPROM. Network diagnostic messages zajišťují diagnostiku sítě a případně opravy.

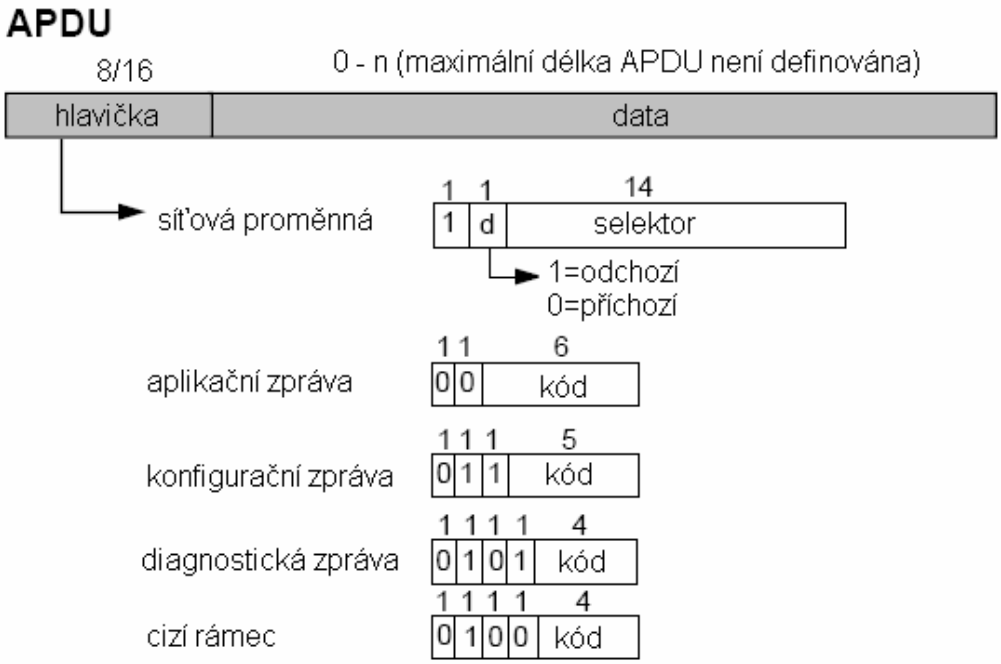
Tato vrstva také definuje ověřovací protokol pro ověřování zpráv (authenticated messages), který umožňuje příjemci zprávy zjistit, zda ten co zprávu vyslal, je k tomu oprávněný. Tento způsob zabraňuje neoprávněnému přístupu na uzel a do aplikace. Totiž každý uzel má 48-bitový ověřovací klíč (Authentication Key). Příjemce zprávy si tak může ověřit, zda ten vysílající má ten samý klíč.

Dále tato vrstva provádí rozhraní mezi 6. a 7. vrstvou protokolu běžícím v hostitelské aplikaci a nižšími vrstvami běžící jako firmware na neuron chipech jednotlivých uzlů.

5. Prezentační (Presentation OSI layer) a Aplikační vrstva (Application OSI layer) OSI modelu

V této kapitole je uveden podrobnější popis prezentační a aplikační vrstvy protokolu LonTalk. Důraz je kladen na ty oblasti, které bylo třeba implementovat v zařízení vyvíjeném v rámci této práce tj. v zařízení, které je založeno na MIP API[1], který je popsán v kapitole 6. Jsou zde popsány služby, které tyto vrstvy poskytují nadřazeným vrstvám a jaké funkce sami.

5.1. Aplikační rámec (APDU)



Obr. 5.1 Aplikační rámec

Aplikační rámec - část zprávy, která je zpracovávána aplikační vrstvou, Application Protocol Data Unit (APDU), obsahuje hlavičku a data. Začlenění aplikačního rámce v rámci celé zprávy protokolu LonTalk je na obrázku v příloze A. Obrázek 5.1 znázorňuje strukturu aplikačního rámce a způsob, jakým se rozlišují jednotlivé typy rámce podle toho, jakou zprávu rámec obsahuje. Jednotlivé typy rámců korespondují se službami,

[1] Microprocessor Interface Program Application interface

které aplikační vrstva nabízí (kromě aliasingu síťových proměnných). Pokud rámec obsahuje data síťové proměnné, je v jeho hlavičce selektor (o něm viz více v 5.3) a velikost hlavičky je v tomto případě 2 bajty. Ostatní typy aplikačního rámce obsahují hlavičku o velikosti 1 bajt. V tomto případě část hlavičky obsahuje kód zprávy, dle kterého jsou rozlišována další data v rámci, jak ukazuje tabulka 5.1.

rozsah	typ zprávy	použití
0…62	aplikační zpráva	
63		služba dotaz/odpověď, když je příjemce online
64…78	cizí rámec	pro funkce gateway
79		pro odpověď, když je příjemce offline
80…95	diagnostická zpráva	
96…127	konfigurační zpráva	
128…255	síťová proměnná	nejvyšší bit =1 indikuje update síťové proměnné, druhý bit indikuje směr transakce (vstup, výstup)

Tab. 5.1Rozsah kódů v hlavičce aplikačního rámce

5.2. Služby prezentační a aplikační vrstvy

Obě vrstvy společně nabízejí 6 služeb:

- **Odesílání a příjem síťových proměnných**

je služba, pomocí které může aplikace odesílat aktualizované hodnoty svých výstupních síťových proměnných. Služba zajišťuje odeslání odpovídající zprávy všem příjemcům dle aktuální konfigurace, která je dána propojením síťových proměnných s ostatními zařízeními.

- **Služba pro odesílání aplikačních zpráv**

slouží aplikaci, pokud potřebuje komunikovat pomocí zpráv, které jsou specifické pouze pro danou aplikaci a které nejsou součástí protokolu LonTalk.

- **Aliasing síťových proměnných**

je služba, která umožňuje použití vícenásobných vstupních i výstupních síťových proměnných.

- **Služba přenos cizích rámců**

Používá se pro přenos rámců jiného protokolu.

- **Konfigurační zprávy**

slouží pro aktualizaci a vyčítání konfiguračních dat. Toto je obvykle prováděno nástrojem pro správu sítě LonWorks (aplikace LonMaker). Obsluha konfiguračních zpráv je jádrem činnosti aplikační a prezentační vrstvy.

- **Diagnostické zprávy**

slouží pro zjištění stavu zařízení. Nadřazená vrstva, tj. samotná řídicí aplikace, používá pouze službu pro odesílání a příjem síťových proměnných a případně ještě službu pro odesílání aplikačních zpráv. Ostatní služby pracují uvnitř aplikační vrstvy, můžeme říci, že pracují na pozadí, aplikace o jejich činnosti není nijak informována. Jak již bylo řečeno výše, aplikace pouze odesílá a přijímá data proměnných a způsob jakým jsou odesílány (pomocí jaké služby, jakým zařízením, zda bude použita autentizace atd.) je záležitostí protokolu LonTalk a je to určeno konfigurací, která je nastavována zpravidla konfiguračním nástrojem. Správa SP a těchto konfiguračních dat je hlavním úkolem aplikační vrstvy.

5.3. Selektor síťové proměnné

Selektor slouží jako identifikátor jednoho propojení proměnné, tedy výstupní proměnná jednoho zařízení má stejný selektor jako vstupní proměnná přijímajících zařízení. Každé propojení má v rámci celé sítě unikátní selektor. Selektor slouží pro identifikaci síťové proměnné doručené do zařízení. Proměnné v jednom zařízení tedy musí mít různý selektor. Selektor je součástí hlavičky aplikačního rámce (obrázek 5.1) a jeho hodnota je automaticky přiřazována konfiguračním nástrojem sítě.

5.4. Síťové proměnné (NV)

Aplikační data se obvykle vyměňují prostřednictvím síťových proměnných, které tvoří třídu zpráv, kde jsou data označena jako Neuron C proměnná a tak je s nimi i zacházeno. Tyto proměnné zjednodušují vývoj a instalování systému, definují a přiřadí data do určité skupiny dle jejich fyzikálního významu, včetně jednotek. Takto přenášená data mají pevně definováno, co prezentují za hodnoty a jak se s nimi má ve vzdálené aplikaci zacházet. Proměnné představují u daného zařízení datové vstupy a výstupy. Maximální velikost jedné je 31 bytů. Propojení výstupních proměnných jednoho zařízení se vstupními jiných zařízení se provádí při konfiguraci sítě. Toto logické

propojování je označováno jako binding a provádí se zpravidla pomocí nějakého konfiguračního nástroje (např. LonMaker). Aplikace běžící v zařízení pouze aktualizuje výstupní proměnné a dále se již nestará o doručení hodnot proměnných k ostatním zařízením, toto už je úkolem protokolu LonTalk a záleží na aktuální konfiguraci, jakým zařízením a jak budou data doručena. Obdobně u vstupních proměnných se aplikace nemusí starat o příchozí data, je-li přijata nová hodnota proměnné, je o tom aplikace informována. Např. v Neuron Chipu je spuštěna odpovídající obslužná funkce pro každou proměnnou. U vstupních proměnných lze ale také využít mechanismu dotazování se na hodnoty vstupních proměnných (polling). Tento mechanismu lze využít jen tehdy, pokud dotazující se zařízení zná adresu dotazovaného zařízení. V obvyklé implementaci protokolu LonTalk se ale tyto adresy neukládají do konfiguračních struktur. Mechanismus proto používají zejména konfigurační nástroje, protože ty mají tyto adresy uloženy. Také jej mohou používat aplikace, které mají k datům konfiguračního nástroje přístup.

5.4.1. Standardní síťové typy proměnných (SNVT)

LonTalk protokol definuje několik typů standardních proměnných, tzv. Standard Network Variable Types (SNVT). SNVT určuje velikost proměnné a její konkrétní fyzikální význam. Sdružení LonMark spravuje a vydává jejich seznam nazývaný SNVT Master List. Tento seznam je dispozici nejen jako textový dokument [6] nebo http://types.lonmark.org, ale také v podobě binárních souborů, které jsou volně ke stažení [6] a ke kterým sdružení LonMark také zveřejňuje API pro čtení a zápis do souborů (podrobněji o tomto API je pojednáno v 6 kapitole). Specifikace SNVT určuje uložení dat v proměnné (např. zda jde o celočíselné číslo, kolik bytů zabírá), fyzikální rozměr, jednotku, rozlišení a název. Každý typ má přidělen číselný index, je označován jako SNVT Index. Příklad definice jedné proměnné SNVT je ve výpisu 5.1, definice také specifikuje další parametry jednotlivých částí proměnné tabulka 5.2.

```
typedef enum {
      SNVT_switch = 95,
} SNVT_t;

typedef struct {
      unsigned char      value;
      char               state;
} tSNVT_switch;
```

Výpis 5.1Definice proměnné SNVT_switch

	Parametr	hodnota
value	typ dat	unsigned short
	Minimum	0
	Maximum	200
	škálovací konstanty (A,B,C)	5, -1, 0
	výpočet hodnoty	$A.10^{B}$.(data + C)
	Rozlišení	0,5
state	typ dat	signed short
	Minimum	-1
	Maximum	1
	škálovací konstanty (A,B,C)	1, 0, 0
	výpočet hodnoty	$A.10^{B}$.(data + C)
	Rozlišení	1

Tab. 5.2 Síťová proměnná SNVT_switch a její reprezentace

5.5. Konfigurační parametry (CP)

Konfigurační parametry (Configuration Properties) slouží k nastavení běhu zařízení. Konfigurační parametry se používají pro nastavení časové periody, pro nastavení offsetů, konstant PID regulátoru apod. Hlavní výhodou je, že kopie jejich hodnot je uložena v databázi konfiguračního nástroje a při výměně zařízení je lze opět jednoduše a rychle nahrát do nového zařízení, takže zařízení je nakonfigurováno zcela stejně, jako bylo zařízení měněné. Konfigurační parametry mají své standardní typy (SCPT - Standard Configuration Properties Type), které jsou spravovány rovněž sdružením LonMark.

5.6. Interoperabilita

Interoperabilita je vlastnost systémů, která jim umožňuje pracovat jako jeden z prvků distribuované řídicí aplikace. Kde každý takovýto prvek může být nahrazen prvkem jiného výrobce bez nutnosti provést změny v řídicí aplikaci. Význam interoperability vzrůstá především v distribuovaných řídicích aplikacích, které obsahují mnoho spolupracujících zařízení, protože se v takovýchto instalacích obvykle využívají zařízení mnoha výrobců. K zajištění interoperability je třeba zajistit nejen spolupráci mezi zařízeními na všech úrovních dle ISO/OSI modelu. Je také třeba zajistit shodný mechanismus přístupu ke sdíleným datům (identifikaci dat, autentizaci přístupu).

Shodnou definici dat představující typ dat, velikost, uspořádání (zda jsou data big-endian nebo little-endian, kódování textů). Shodný význam dat, což znamená stejnou logickou či fyzikální interpretaci dat. Shodnost chování aplikace, aplikace se zvenku musí projevovat stejně, vnitřní konkrétní implementace algoritmů může být libovolná. Shodnost komunikačního rozhraní zařízení, datové vstupy a výstupy zařízení musí být stejné.

V technologii LonWorks je mechanismus přístupu k datům zajištěn autentizací (4.5.5) a identifikace dat je zajištěna selektorem (5.3). Definici dat a význam dat zajišťují standardní typy síťových proměnných (SNVT) popsané v 5.4.1. Zbývá tedy zajistit definice rozhraní zařízení a shodnost aplikace, k tomuto slouží funkční bloky, funkční profily a profily zařízení.

5.6.1. Funkční bloky (Functional object)

Funkční bloky sdružují vstupní a výstupní síťové proměnné, konfigurační parametry, které souvisejí s jednou aplikací zařízení (v jednom zařízení může být implementováno několik aplikací, např. senzor teploty vzduchu a senzor vlhkosti vzduchu). Funkční blok je častí profilu celého zařízení.

5.6.2. Funkční profily (SFPT)

Funkční profily jsou šablony pro funkční bloky. Funkční profily specifikují pomocí SNVT povinné a volitelné vstupní a výstupní proměnné a povinné a volitelné konfigurační parametry. Jestliže nějaký funkční blok implementuje nějaký funkční profil, musí obsahovat všechny položky funkčního profilu, které jsou povinné. Sdružení LonMark specifikuje standardní funkční profily Standard Functional Profile Templates (SFPT) pro různá zařízení, seznam všech SFPT lze nalézt na webové stránce sdružení LonMark, každý standardní profil má přiděleno unikátní číslo (ID) functional profile number, nebo functional profile key. Funkční profil je vždy definován pro jednu specifickou aplikaci, např. pro senzor přítomnosti osob v prostoru je specifikován funkční profil Occupancy controller.

5.6.3. Profily zařízení (Device Interface)

Profil zařízení (Device Interface) popisuje zařízení jako celek.
definuje:
• identifikátor profilu zařízení Standard Program ID (SPID)
• jaké funkční profily zařízení implementuje
• jaké další síťové proměnné a konfigurační parametry implementuje
• implicitní hodnoty konfiguračních parametrů
• vlastnosti implementace protokolu LonTalk (např. maximální možný použitelný počet proměnných, zda jsou podporovány dynamické síťové proměnné)
• parametry fyzické vrstvy
Každé zařízení, které splňuje specifikace sdružení LonMark musí ve svém profilu obsahovat speciální funkční profil Node Object, který obsahuje síťové proměnné, které používají konfigurační nástroje pro správu všech ostatních funkčních bloků v zařízení a pro zjišťování stavu zařízení.

5.6.4. XIF soubor

XIF soubor obsahuje popis profilu zařízení, formát souboru je specifikován sdružením LonMark. XIF soubor by měl být vždy dodáván výrobcem spolu se zařízením, protože jej používají konfigurační nástroje pro načtení informací o zařízení do své databáze. XIF soubor je textový soubor, lze jej tedy vytvořit i ručně, ale častěji je vytvářen přímo vývojovými nástroji při programování aplikace pro procesor Neuron Chip, takovým nástrojem je např. Node Builder od firmy Echelon.

5.7. Konfigurační data protokolu LonTalk

Je-li zařízení tvořeno pouze procesorem Neuron Chip (popsáno v kapitole 6), jsou všechna konfigurační data uložena přímo v něm. Dále budu popisovat pouze variantu, kdy je zařízení založeno na architektuře, ve které je Neuron Chip pouze v roli komunikačního koprocesoru, na architektuře host-based nodes nebo MIP (kapitola 6). Tuto architekturu jsem použil pro svou aplikaci. V tomto případě jsou některá data uložena v procesoru Neuron Chip a některá v implementaci aplikační vrstvy v aplikačním.

5.7.1. Komunikační koprocesor

V komunikačním koprocesoru jsou uloženy:

• Informace týkající se konfigurovatelných parametrů (popis umístění zařízení, komunikační rychlost), podrobnosti o těchto datech lze nalézt v dokumentaci výrobce procesoru Neuron chip. Tyto parametry jsou uchovávány v EEPROM paměti.

• Struktura základních parametrů obsahuje např. Neuron ID, SPID, počet položek v tabulce adres, počet síťových proměnných. Obsah této struktury nelze měnit konfiguračním nástrojem, proto je také někdy označována jako read only structure.

• Tabulka domén 1 obsahuje dvě položky, protože zařízení může být členem maximálně dvou domén. Položka obsahuje identifikátor domény, základní adresu a klíč pro autentizaci.

• Tabulka adres 1, adresy z této tabulky se používají při příjmu zpráv. Přesněji řečeno, do této tabulky nahlíží síťová vrstva, když u přijaté zprávy rozhoduje, zda je tomuto zařízení určena. Tabulka může mít maximálně 15 adres a jednotlivé adresy (položky) mohou být různého typu.

5.7.2. Aplikační procesor

V aplikačním procesoru jsou převážně informace týkající se síťových proměnných:

• Položky tabulky adres 2 jsou používány při odesílání hodnot výstupních síťových proměnných. Tato tabulka se používá pouze u zařízení, která používají ECS[2] zprávy a může obsahovat až 65535 adres. Položky této tabulky jsou aktualizovány ECS zprávami.

• Tabulka domén 2 se v aplikačním procesoru používá pouze, jsou-li využívány ECS zprávy. Tabulka může mít teoreticky až 65535 záznamů. Je třeba ale brát ohled na to, že tato tabulka se využívá pouze pro adresy při odesílání zpráv a případně při dotazování se na hodnotu síťových proměnných.

• Ke každé proměnné se váže konfigurace proměnné, která obsahuje informace týkající se doručování síťových proměnných (jakou službou se data odesílají, jakou mají prioritu.

[2] Extended Network Management Command Set – rozšířená množina zpráv [2]

• U každé síťové proměnné se také ukládá informace, která je dána jejím typem. Je zde také uložen SNVT Index[3].

• Hodnoty síťových proměnných lze považovat za konfigurační data, pouze pokud to jsou konfigurační parametry implementované pomocí síťových proměnných. Pak je třeba je také ukládat, jinak jsou hodnoty síťových proměnných udržovány pouze v paměti RAM[4].

• Popis zařízení (Self-Identification Structures) obsahuje informace uložené v profilu zařízení. Maximální počet síťových proměnných, aktuální počet proměnných, počet funkčních profilů a jejich ID. Tyto informace uložené v zařízení musí být v souladu s informacemi, které jsou uloženy v XIF souboru tohoto zařízení.

5.8. Odesílání a příjem síťových proměnných

Je-li z rozhraní MIP nebo OpenLDV přijata zpráva, která je aktualizací některé proměnné, je úkolem aplikační vrstvy dle selektoru aktualizovat hodnotu proměnné a případně informovat nadřazené vrstvy (řídicí aplikaci) o změně hodnoty síťové proměnné.

Při aktualizaci hodnoty výstupní proměnných aplikace aktualizuje hodnoty dané proměnné a aplikační vrstva protokolu LonTalk se stará o odeslání hodnoty proměnné všem zařízením, ke kterým je proměnná připojena. K odeslání zprávy může aplikační procesor využít adresy uložené v komunikačním koprocesoru, kterému ve zprávě k odeslání předává pouze index do tabulky adres 1 v komunikačním koprocesoru. Tento způsob je označován jako implicitní adresování. Druhý způsob je označován jako explicitní adresování a je používán spolu s ECS zprávami, aplikační procesor předává komunikačnímu procesoru zprávu včetně cílové adresy.

5.9. Konfigurační zprávy(Network Management Commands)

Konfigurační zprávy (Network Management Commands) slouží k aktualizaci konfiguračních dat, většinou jsou zasílány pomocí služby dotaz/odpověď. Zprávy jsou většinou požadavkem na inicializaci, čtení nebo aktualizaci nějaké položky v některé z konfiguračních tabulek. Konkrétní požadavek a struktura zasílaných dat se rozlišuje

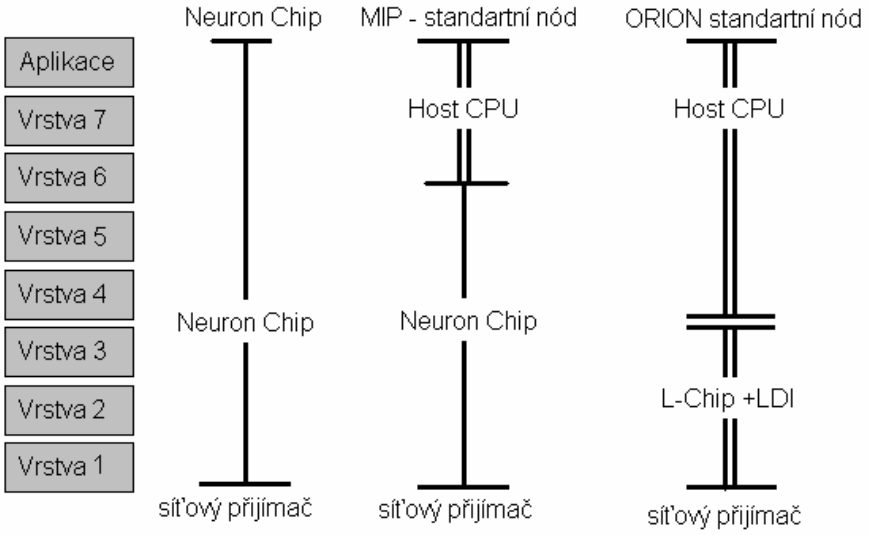
[3] Každá síťová proměnná má přidělené číslo [6]
[4] Random Access Memory

dle kódu zprávy, který je součástí hlavičky aplikačního rámce a případně ještě dle velikosti přijatých dat. Přesné datové struktury lze nalézt v příloze A. Konfigurační zprávy se používají zejména při instalaci zařízení do sítě a při propojování síťových proměnných.

6. Implementace síťě LonWorks

6.1. Přístupy ke sběrnici LonWorks

Úkolem mé práce bylo zprovoznit komunikaci LonWorks pro PC104 kartu s využitím dodávaných driverů pod operačním systémem Windows. Následně provést implementaci nejnižšího driveru pro kartu pod operační systém reálného času On-time. Musel jsem se seznámit se způsoby, jakými lze v současné době realizovat přístup ke sběrnici LonWorks. V následujících odstavcích je uveden rozbor jednotlivých možností, které se v současné době nabízejí a jejich nejdůležitější vlastnosti. Na obrázku 6.1 jsou znázorněny tři možnosti implementace sítě.



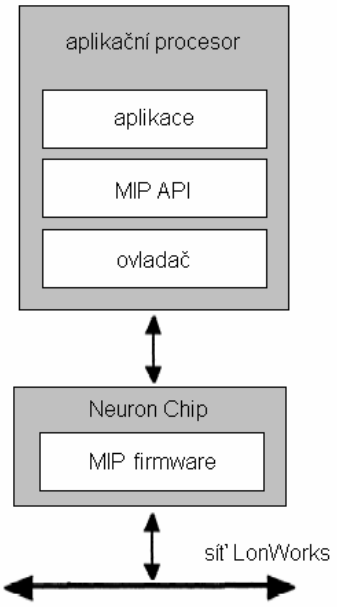
Obr. 6.1 Implementace sítě

6.1.1. Neuron Chip (hosted nodes)

Většina zařízení pro LonWorks je založena na mikroprocesoru Neuron Chip (podrobněji je popsán v kapitole 3). Neuron Chip obsahuje implementaci celého protokolu LonTalk. Uživatelská aplikace běží přímo v procesoru Neuron Chip a umožňuje pracovat maximálně s 62 síťovými proměnnými. K naprogramování procesoru Neuron Chip je potřeba vývojové prostředí NodeBuilder.

6.1.2. Standartní nód MIP (Host-based nodes)

Tento způsob přístupu využívá Neuron Chip pouze jako komunikační koprocesor, je potřeba do něj nahrát speciální Microprocessor Interface Program (MIP) firmware, samotná uživatelská aplikace běží na jiném libovolném aplikačním procesoru (host procesor). Tento přístup je vhodný pro aplikace, které není možné realizovat pomocí samotného procesoru Neuron Chip. Z důvodu náročnosti uživatelské aplikace na výpočetní výkon nebo na velikost paměti nebo na počet potřebných síťových proměnných (v této konfiguraci je možné použít až 4096 síťových proměnných). Neuron Chip v tomto módu zajišťuje pouze 2. – 5. vrstvu protokolu LonTalk, vrstvy 6 a 7 je potřeba implementovat v aplikačním procesoru. Komunikace mezi procesorem Neuron Chip a aplikačním procesorem je specifikována zvláštním protokolem Network Interface Protocol [11], který musí být implementován v podobě ovladače (Network Interface Driver). Na úrovni fyzické vrstvy může komunikace probíhat po lince RS232 (označováno SLTA), pomocí paralelního rozhraní (MIP/P20, MIP/50) nebo pomocí dvoubránové paměti (MIP/DPS). Z hlediska softwaru je nutné v aplikačním procesoru implementovat ovladač (Network Interface Driver) a 6.–7. vrstvu protokolu LonTalk. Architektura postavena na MIP je rozebrána v kapitole 6.3.



Obr. 6.2 Architektura postavená na MIP

6.1.3. ORION Stack

K rozhraním firmy Loytec je dodávána knihovna orion.dll, která obsahuje implementaci ORION API tj. API pro přístup k ORION Stacku, připomeňme, že ORION Stack pro mikroprocesory není volně k dispozici a je nutné jej zakoupit. Použití ORION Stacku usnadňuje velké množství funkcí, které ORION API poskytuje. Hlavní rozdíl mezi MIP aplikacemi a aplikací založené na ORION Stacku spočívá v tom, že ORION Stack obsahuje implementaci 6. a 7. vrstvy protokolu LonTalk, zatímco OpenLDV toto neobsahuje. Shrnuto, ORION Stack nabízí na PC nejlepší přístup k síťovému rozhraní na nízké úrovni. Ještě poznamenejme, že ORION Stack je dodáván k rozhraním firmy Loytec, ale lze k těmto rozhraním přistupovat také stejně jako k MIP pomocí čtyř základních funkcí ldv_open, ldv_close, ldv_read, ldv_write.

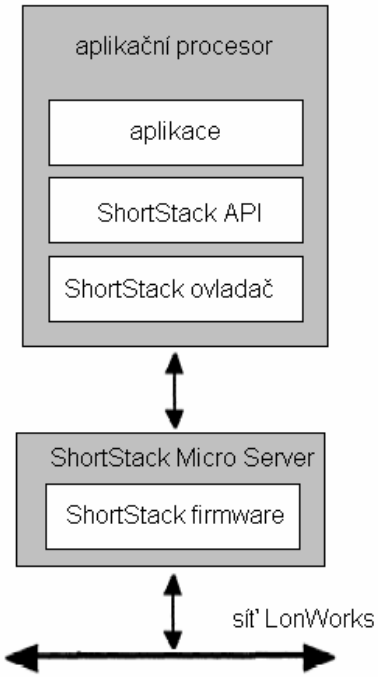
6.1.4. OpenLDV API

OpenLDV je API pro aplikace, které chtějí k síťovým rozhraním firmy Echelon přistupovat na nízké úrovni, z hlediska ISO/OSI modelu je to přístup k 5. vrstvě. Pomocí tohoto API lze přijímat a odesílat většinu zpráv LonTalk protokolu, některé zprávy obslouží přímo síťové rozhraní a dále je nepropouští, je to většina konfiguračních zpráv. Síťové rozhraní zde vystupuje ve stejné roli jako komunikační koprocesor v MIP architektuře. Rozhraní OpenLDV API je téměř shodné s MIP API. Lze tedy říci, že implementaci 6. a 7. vrstvy protokolu LonTalk pro OpenLDV API lze použít také pro aplikaci založené na MIP API.

6.1.5. Short Stack Development Kit

Jedná se o další variantu, ve které Neuron Chip vystupuje v roli komunikačního koprocesoru a samotná aplikace běží na aplikačním procesoru. Short Stack Development Kit je zdarma poskytován firmou Echelon za podmínky použití procesoru Neuron Chip nebo Smart Transceiver ve vyvíjeném zařízení, kit obsahuje také speciální firmware pro Neuron Chip (Short Stack Micro Server). V tomto případě je možné použít maximálně 62 síťových proměnných. Komunikace mezi procesorem Neuron Chip a aplikačním procesorem je realizována pomocí sériových rozhraní SCI nebo SPI. Z hlediska softwaru je nutné v aplikačním procesoru pouze naprogramovat ovladač Serial Driver, což je rozhraní, které v aplikačním procesoru poskytuje přístup k

periferiím SCI nebo SPI pro vrstvu ShortStack API (ta je dodávána ve zdrojovém kódu jako součást Short Stack Development Kit). Architektura zařízení při použití Short Stack Development Kit je na obrázku 6.3.



Obr. 6.3 Architektura postavená na ShortStack

6.1.6. Přístup z počítače

Především pro konfiguraci a správu sítě LonWorks (pomocí nástroje LonMaker) je vhodné mít k síti přístup z počítače (PC). Hardwarově je toto většinou řešeno síťovými rozhraními pro PC, která jsou vyráběna několika výrobci Echelon, Loytec, Gesytec) obvykle ve formě karty do PCI či PC104 slotu nebo jako převodník na LPT, USB port. Z hlediska funkčního se tato rozhraní téměř neliší ani v závislosti na výrobci ani v závislosti na provedení (PCI, LPT, USB). Odlišují se především softwarem, který lze spolu s daným rozhraním použít, proto jsou následující odstavce členěny dle softwarového řešení. LNS Server pochází z dílny firmy Echelon, ale lze jej použít i s rozhraními firmy Loytec Gesytec. LNS Server je v podstatě server pro model server-klient. Aplikace jako LonMaker vystupují v roli klienta. Díky tomuto řešení klient-server je možné mít LNS Server spuštěn na jednom serverovém počítači a klienti (aplikace) mohou být spouštěny z počítačů připojených přes lokální ethernet či internet. LNS Server poskytuje klientům (aplikacím) objektový přístup k síti LonWorks pomocí technologií COM a ActiveX (je tedy závislý na operačním systému Windows). Nabízí

objekty, pomocí kterých lze efektivně instalovat, spravovat a monitorovat celou síť. LNS Server využívá pro ukládání všech dat o aktuální konfiguraci sítě vlastní databázi (LNS global database), která obsahuje v podstatě kopii fyzické sítě (kopii konfigurací uložených ve všech zařízeních sítě).
Pro vývoj klientských aplikací je třeba The LNS Application Developer's Kit. Lze ovšem také klientskou aplikaci vyvíjet na platformě Java, k dispozici je balík LNS HMI Developer's Kit for the Java Platform.

6.2. Instalace sítě LonWorks

Existuje několik různých způsobů jak instalovat síť LonWorks. Správný způsob instalace záleží na mnoho faktorech, jako je zkušenost tvůrce sítě, požadované flexibility nebo požadavky koncového uživatele či vhodností pro různé aplikace dle velikosti instalované sítě. Dále se budu zabývat pouze způsoby instalace, které se budou při instalaci PC104 karty používat. Názvy instalačních scénářů jsem nepřekládal a jsou stejné jako v literatuře [8], kde jsou popsány I další scénaře.

6.2.1. Samoinstalace nódu (Self-installed nodes)

Zařízení je z výroby vybaven možností pro konfigurování sítě a má za tímto účelem uživatelské rozhranní, které umožňuje nastavovat některé konfigurační parametry. Jako uživatelské rozhranní slouží většinou několik jumperů pro nastavení parametrů. Používá se převážně při vytváření kopii zařízení, které jsou v omezené míře konfigurovatelná. Tento způsob instalace není vhodný pro flexibilní instalace a má své uplatnění v sériové výrobě. V aplikačním programu zařízení musí být funkce provádějící čtení uživatelského rozhranní a modifikaci konfiguračních parametrů.

6.2.2. Instalace s navržením celé struktury (Engineering system installation scenario)

Tato instalace se provádí ve dvou krocích. V první fázi návrhu je navržena celá struktura aplikace, tj. jsou zvolena jednotlivá zařízení a jsou vybrány způsoby fyzického propojení zařízení. Provádí se propojování výstupních proměnných se vstupními proměnnými jiných zařízení (binding). Lze také nastavit hodnoty konfiguračních

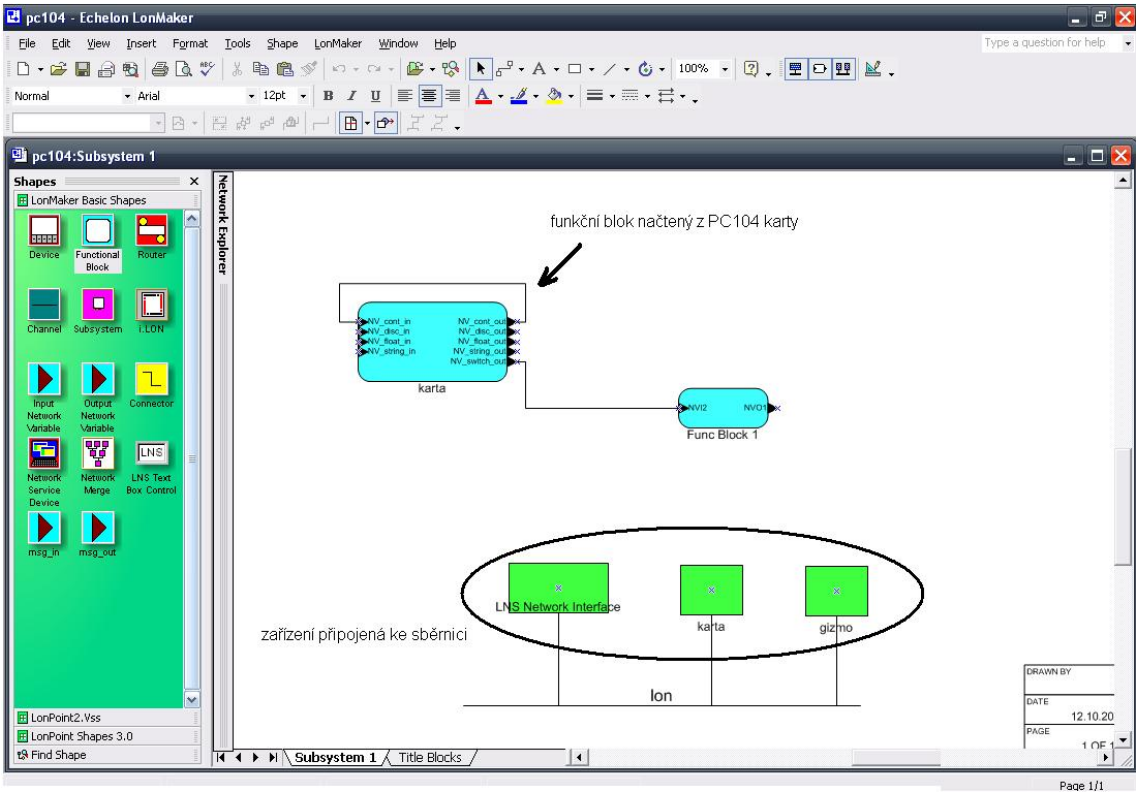
parametrů pro jednotlivá zařízení. Všechny tyto konfigurační informace jsou ukládány do databáze konfiguračního nástroje, celý návrh lze tedy provádět bez fyzicky existující (instalované) sítě LonWorks, je zapotřebí pouze instalační nástroj a profily zařízení (XIF soubory). V druhé realizační fázi je konfigurační nástroj již připojen k fyzické síti LonWorks a konfigurace celé sítě je z databáze nahrávána do jednotlivých zařízení. Proces instalace zařízení do sítě LonWorks je označován jako instalace zařízení do sítě (commision). Algoritmus instalace zařízení do sítě je podrobně popsán [8]. Při instalaci zařízení se také nahrávají do zařízení hodnoty konfiguračních parametrů a také lze nahrát aplikační program do zařízení. Instalace se nejčastěji používá v návrhu sítě pro automatizované budovy, kde se nejprve navrhne systém a na základě návrhu dojde k realizaci sítě.

6.2.3. Instalace s postupným přidáváním zařízení (Conner-as-you-go installation scenario)

Tato instalace se taky někdy nazývá ad-hoc systém installation[5]. Vzhledem k tomu že jsem vytvářel zařízení, tak je tento způsob instalace sítě pro mé účely nejlepší. Využíval jsem ho při seznamování se sítí LonWorks tak i při následném testování PC104 karty pod operačním systémem Windows i On-time. Konfigurační nástroj je připojen k fyzické síti LonWorks, návrh nelze provádět bez fyzicky existující sítě, a při konfigurace celé sítě se postupně instalují zařízení do sítě (commision) a postupně se provádí propojování výstupních proměnných se vstupními proměnnými mezi zařízeními (binding). Konfigurace každého nódu se při jakýchkoliv změnách konfigurace ukládá do databáze konfiguračního nástroje. Do zařízení se také nahrávají hodnoty konfiguračních parametrů a také lze nahrát aplikační program. Tato instalace se používá pro malé a jednoduché systémy nebo pro systémy u nichž předem neznáme celou strukturu sítě nebo neznáme pořadí instalace zařízení.

Obrázek 6.4 zobrazuje příklad sítě vytvořené programem Lonmaker za účelem testování sítě. Tato síť byla použita při testování ovladače, který byl napsán pro On-time (kapitola 7). V síti jsou zapojené dva nódy (PC104 karta a deska z Mini Evaluatio kitu). Třetí nód je karta zprostředkovávající rozhraní mezi PC (konfiguračním nástrojem) a sítí Lonworks.

[5] Instalace za provozu sítě



Obr. 6.4 Síť Lonwork vytvořená nástrojem Lonmaker

6.3. Microprocessor Interface Program (MIP) a Host Application (HA)

Tato kapitola pojednává o rozboru microprocessor interface program (MIP) [11], protože ho obsahuje PC104 kata (kapitola 8.2.) použitá v této práci. Záměrem této kapitoly bylo rozebrat MIP z pohledu aplikace (HA[6]) v aplikačním procesoru. HA [12] je věnována samostatná podkapitola.

Ukážeme si zde, jak je implementován ovladač pro operační systém Windows (6.3.2.), jaké funkce nabízí rozhraní ovladače aplikaci a jaké procesy jsou v MIP obsaženy. Implementaci ovladače pro tento systém jsem neprováděl, protože je ovladač dodáván, ve formě instalace, již od výrobce. Všech těchto znalostí bylo ovšem použito při psaní vlastního ovladače pro On-time. Který popisuji v kapitole 7, kde se odkazuji na tuto implementaci.

[6] Host Application – aplikace běžící v aplikačním procesoru

6.3.1. Microprocessor Interface Program (MIP)

Síťové rozhraní je zařízení, které zprostředkovává komunikační rozhranní mezi aplikačním procesorem (host processor) a sítí Lonworks. Jeden ze způsobu realizace rozhraní je neuron chip ve kterém běží MIP. MIP je firmware, který transformuje neuron chip v komunikační koprocesor pro připojení aplikačního procesoru. MIP umožňuje přesunutí obsluhy vyšších vrstev Lontalk protokolu z neuron chipu do aplikačního procesoru (více v kapitole 6.1.2.). Z pohledu rozhraní aplikačního procesoru rozlišujeme tři druhy MIP.

- **MIP/P20**

Využívá 11 bitové paralelní rozhranní v neuron chipu mezi aplikačním a síťovým rozhraním. Procesor přistupuje k neuron chipu jako k 8 bitový I/O[7] portům nebo jako k 8 bitovým paměťovým oblastem. MIP/P20 běží na neuron 3120 chipu nebo neuron 3150 chipu, ale obvykle se používá neuron 3120 chip.

- **MIP/P50**

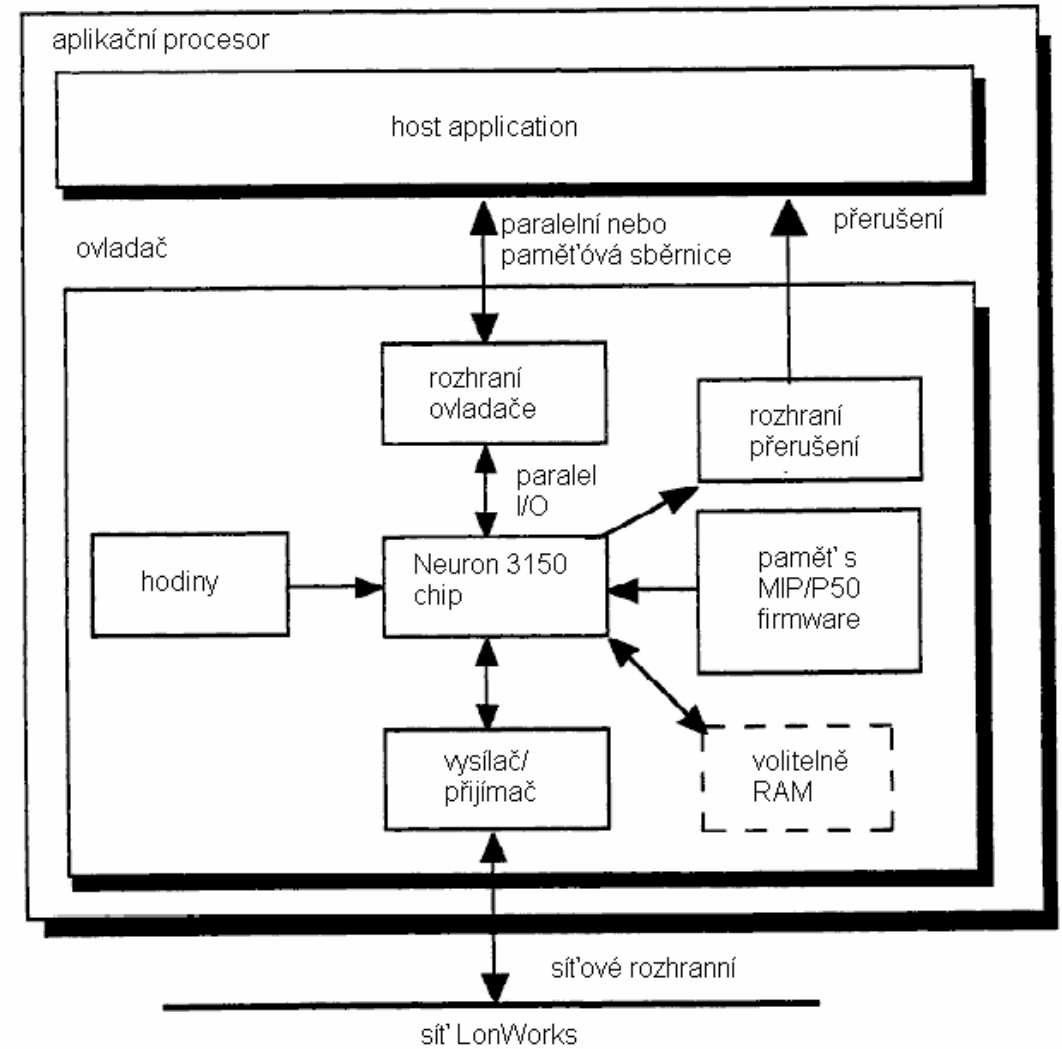
Využívá 11 bitové paralelní rozhranní v neuron chipu mezi aplikačním a síťovým rozhraním s přidáním přerušení od paměti, které je generováno zápisem z neuron chipu. Stejně jako u MIP/P20 tak u MIP/P50, procesor přistupuje k neuron chipu jako k 8 bitový I/O[8] portům nebo jako k 8 bitovým paměťovým oblastem. MIP/P50 umožňuje větší datový tok než MIP/P20 a to i když není využíváno přerušení. MIP/P50 vyžaduje pro svůj běh neuron 3150 chip. Firmware MIP/P50 obsahuje PC104 karta použitá v této práci. Na obrázku 6.5 je proto ukázána architektura síťového rozhraní MIP/P50.

- **MIP/DPS**

Využívá dvoubránové paměti v neuron chipem ke zprostředkování rozhraní mezi aplikačním a síťovým rozhraním. Z důvodu dvoubránové paměti musí být v MIP/DPS hardwarové semafóry, které kontrolují přístup do sdílené paměti od aplikačního a síťového rozhraní. MIP/DPS vyžaduje neuron 3150 chip, rychlou dvoubránovou paměť a obvykle se používá v kombinaci s 32 bitovým mikroprocesorem vyšší třídy.

---

[7] I-input (vstupní), O-output (výstupní)
[8] I-input (vstupní), O-output (výstupní)



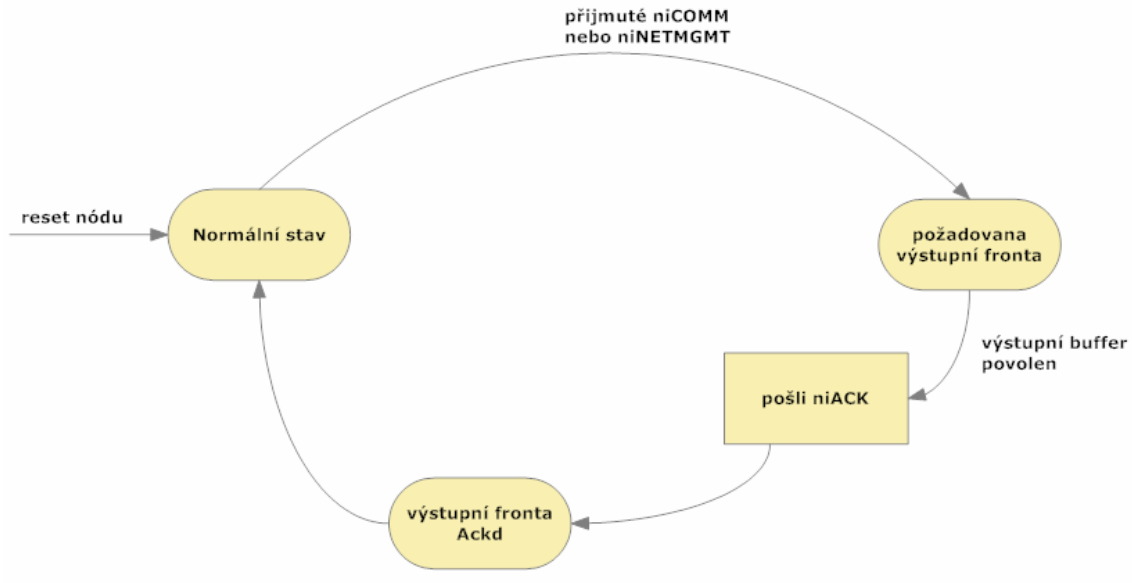
Obr. 6.5 Architektura síťového rozhraní MIP/P50

6.3.2. Implementace síťového ovladače z pohledu MIP

Jak již bylo zmíněno v úvodu (kapitoly 6.3.), je zde rozebrána implementace z pohledu operačního systému Windows . Všechny přenosy mezi ovladačem a síťovým rozhraním jsou pomocí přenosu nulového (NULL) tokenu, přenosu dat, přenosu ACK_RESYNC (popsáno v *Parallel I/O Interface to the Neuron Chip*). Ovladač přijme aplikační frontu (obrázek 6.12) z aplikace, překládá ho do frontu pro síťovou vrstvu a tuto přeloženou frontu posílá ovladači rozhraní prostřednictvím paralelního I/O protokolu. Když je v aplikační frontě nastaven příkaz `niCOMM`[9] nebo `niNETMGMT`, musí ovladač čekat, dokud není připravena výstupní fronta k poslání. Obrázek 6.6 ukazuje stavový

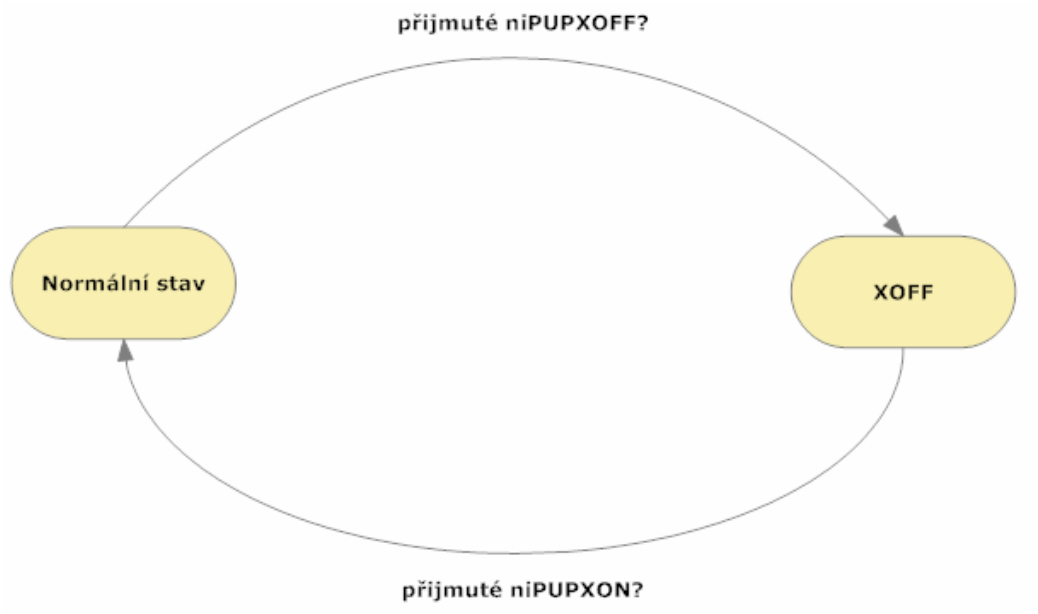
[9] Proměnná, která je použita v HA. Všechny proměnné použité v textu budou psány tímto fontem.

přechodový diagram síťového rozhraní při downlinku[10]. Přesný popis co se v kterém stavu děje je v uživatelské příručce k MIP [11].



Obr. 6.5 Stavový přechodový diagram síťového rozhraní při downlinku

Při uplinku[11] ovladač překládá frontu ze síťové vrstvy do aplikačních fronty (obrázek 6.12) a čeká dokud HA není připravená ke čtení fronty. Obrázek 6.7 ukazuje stavový přechodový diagram síťového rozhraní při uplinku.

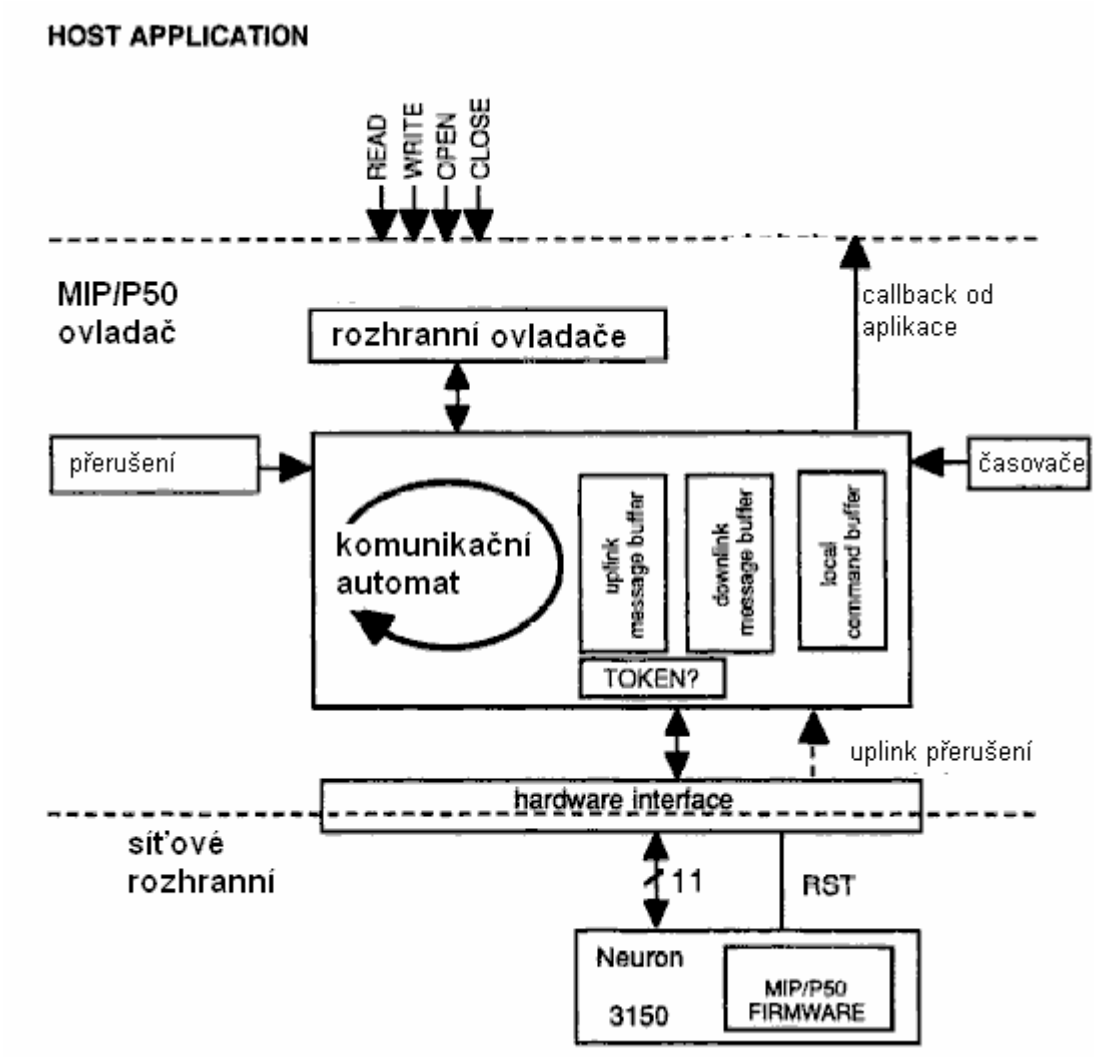


Obr. 6.6 Stavový přechodový diagram síťového rozhraní při uplinku

[10] Poslání fronty z aplikace do ovladače
[11] Vyslání fronty z ovladače do aplikace

6.3.2.1. Stavový diagram síťového rozhraní MIP/P50

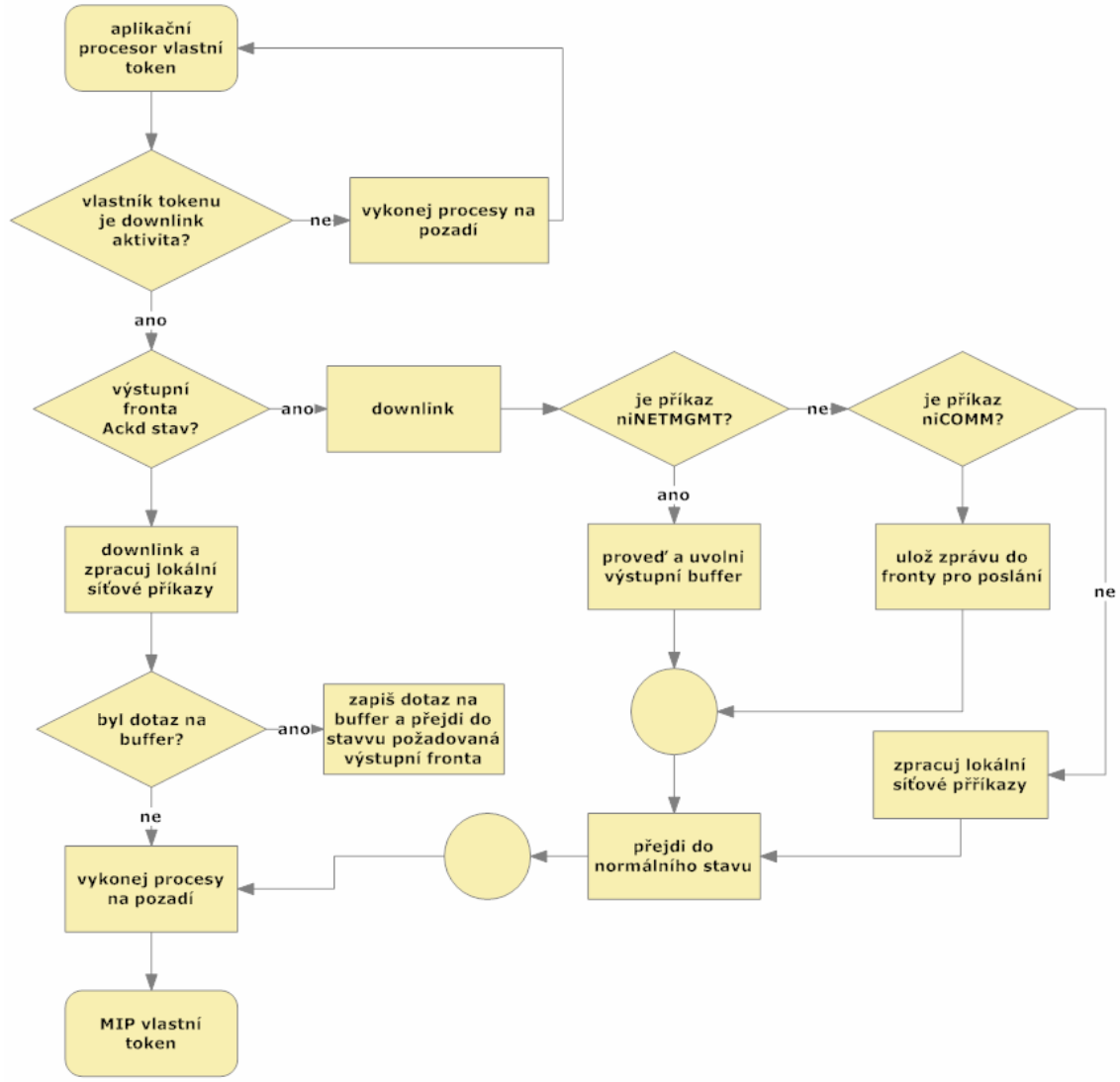
Obrázek 6.7 ukazuje blokový diagram síťového rozhraní MIP/P50 na kterém je vidět co implementuje ovladač a tedy co je potřeba implementovat v ovladači pro On-time (kapitola 7.3.). Je zde naznačeno, že ovladač je stavový automat, který operuje nad frontami zpráv, když má NULL token. Procesy probíhající v ovladači jsou popsány v kapitole 6.3.2.2. Na obrázku je vidět jaké funkce zprostředkovávají rozhraní mezi ovladačem a HA. Přístup funkcím read, write, open, close zprostředkovává pod Windows dynamická knihovna wldv32.dll (dodávaná výrobce), pro On-time je tedy opět nutná vlastní implementace těchto funkcí (kapitola 7.4)



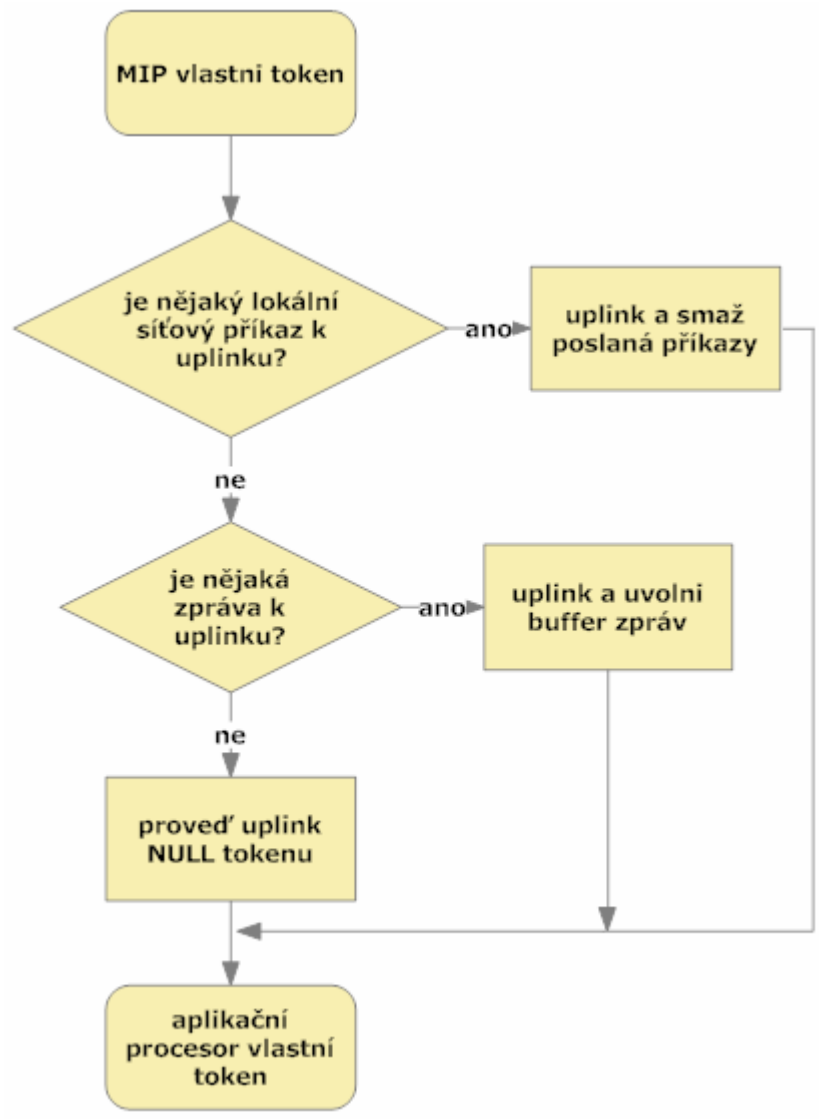
Obr. 6.6 Stavový diagram síťového rozhraní MIP/P50

6.3.2.2. Procesy v MIP/P50

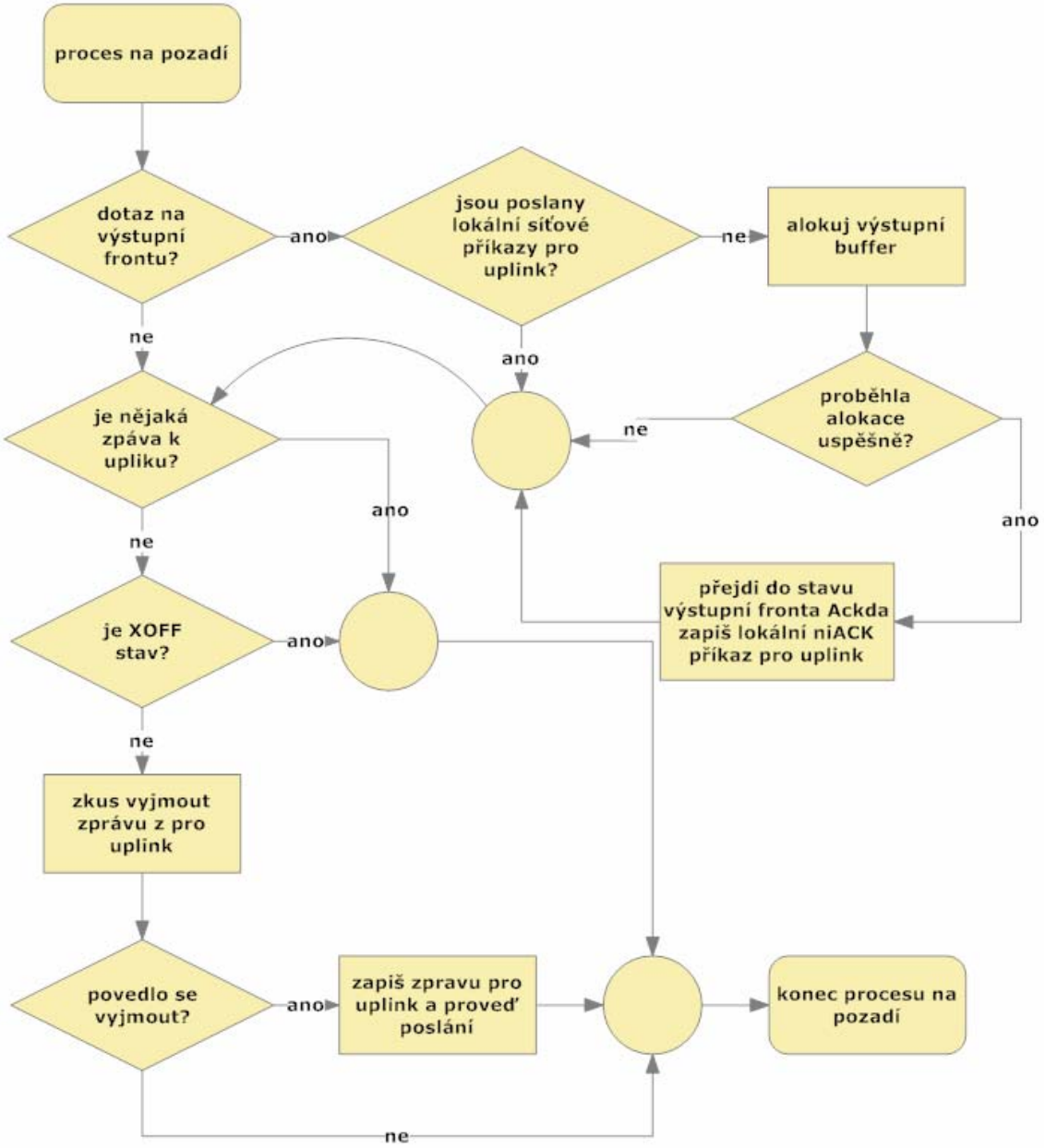
Na následujících obrázcích jsou zachyceny procesy, které popisují předávání NULL tokenu a ukazují proces ovladače. Obrázky 6.7 a 6.8 popisují procesy na kterých je ukázána implementaci ovladače MIP/P50 (obrázek 6.6). Obrázek 6.7 ukazuje co ses děje za procesy, když NULL token vlastní aplikační procesor a jak se předá token ovladači. Obrázek 6.8 popisuje, jak se dostane NULL token od ovladače aplikačnímu procesoru. Obrázek 6.9 ukazuje proces, který běží na pozadí a slouží k obsluze hardwarového rozhraní (obrázek 6.6) MIP.



Obr. 6.7 Proces předání tokenu MIP



Obr. 6.8 Proces předání tokenu aplikačnímu procesoru



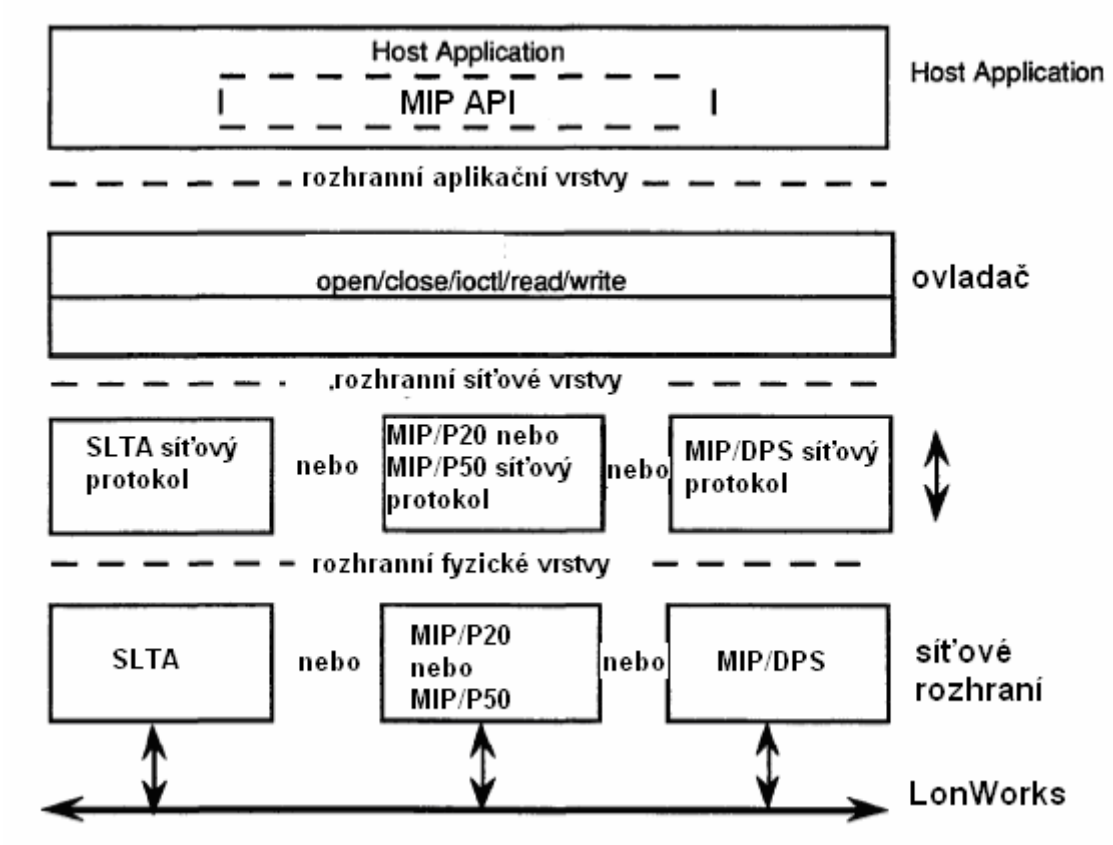
Obr. 6.9 Proces běžící na pozadí MIP

6.3.3. Host Application (HA)

V rámci této práce byl vytvořen i program, tzv. Host Appplication (HA) [11], která běží v aplikačním procesoru. V tomto případě je aplikační procesor AMD GEODE LX800 (kapitola 8.3.), který je postaven z důvodu implementace programu pro On-time (kapitola 7) na architektuře x86. Pro vývoj aplikace bylo použito Microsoft Visual Studio 2008 s modulem pro On-time, všechny zdrojové kódy jsou psány v C++. Byla vyvinuta jedna HA, která může být pomocí podmíněného překladu přeložena pro OS Windows nebo On-time.

6.3.3.1. Architektura HA

Architektura aplikace je definován protokolem pro komunikaci mezi aplikačním a síťovém rozhraním. Architektura HA je znázorněna na obrázku 6.10.



Obr. 6.10 Architektura Host Application

Rozhraní aplikace využívá k přístupu do sítě Lonworks rozhraní aplikační vrstvy. Rozhraní jsem vyvíjel pouze pro MIP API, protože PC104 karta (kapitola 8.2) je

dodávána s ovladačem (pro OS Windows) pro toto API. HA lze samozřejmě použít s libovolnou kartou, která disponuje rozhraním MIP API. Popis MIP API je uveden v kapitole 6.3.3.4.

6.3.3.2. Network Interface Selection a Host Selection mód

V jakém módu bude karta pracovat určuje firmware nahraný v neuron chipu. Pro Network Interface Selection mód je firmware NSI a pro Host Selection mód je firmware MIP/P50 (implementovná karta). Hlavní rozdíl mezi nódy je počet maximální proměnných, které nód může nabídnout. Další rozdíl je v tom, kdo se stará o ukládání konfigurační tabulky síťových proměnných. Kdo zjišťuje příjem zpráv pro nód. Všechny rozdíli jsou shrnuty v tabulce 6.1.

	Network Interface Selection	Host Selection
Dekódování cílové adresy	neuron chip	neuron chip
Výběr síťové proměnné	neuron chip	HA
Konfigurační tabulka síťových proměnných	neuron chip	HA
Maximální počet síťových proměnných v konfigurační tabulce	62	4096
Formát aplikačního fronty síťové proměnné	`ProcessedNV` (`msg`_type=1)	`UnprocessedNV` (`msg`_type=0)

Tab. 6.1 Srovnání Network interface selection a host selection

Protože je karta v host selection módu, tak se musí HA kontrolovat zda příchozí zpráva je určena pro tento nód. Kontrola se provádí pomocí porovnání adresy v příchozí zprávě s adresami uloženými v konfigurační tabulce proměnných. HA se také stará u odchozích zpráv o vyplnění aplikačního rámce (obrázek 5.1), který je v HA implementován funkcí `ni_msg_hdr_init` (Výpis 6.1). Za povšimnutí zde také stojí, že `msg_type` je zde nastavena na nulu (host selection mód viz. Tabulka 6.1).

HA se také stará o konfigurační tabulka síťových proměnných, kterou při spuštění aplikace nahraje z disku pomocí funkce `load_NV_config` (Výpis 6.2) a funkce `exit_func`, která tabulku nahraje na disk při ukončení HA. Tabulka se totiž může při běhu HA, změnit. Data konfigurační tabulce mění Lonmaker při konfiguraci sítě. Datová struktura konfigurační tabulky je posána v kapitole 6.3.3.3.

```
void ni_msg_hdr_init( int msg_size, ServiceType service, boolean
priority,boolean local, boolean auth, boolean implicit, byte msg_tag )
{

    const static NI_Queue queue[ 2 ][ 2 ] =         /* define MIP
queue */
        { { niTQ, niTQ_P }, { niNTQ, niNTQ_P } };

    msg_out.ni_hdr.q.queue  = queue[ service == UNACKD ][ priority ];
                /* Transaction queue            */

    msg_out.ni_hdr.q.q_cmd    = local ? niNETMGMT : niCOMM;
             /* Data transfer for network interface or network
    msg_out.ni_hdr.q.length   = sizeof( ExpMsgHdr ) + sizeof(
ExplicitAddr )
                                + msg_size; /* Header size

    msg_out.msg_hdr.exp.tag       = msg_tag;
    msg_out.msg_hdr.exp.auth      = auth;
    msg_out.msg_hdr.exp.st        = service;
    msg_out.msg_hdr.exp.msg_type  = 0;      /* MIP doesn't process NVs
    msg_out.msg_hdr.exp.response  = 0;      /* Not a response message
    msg_out.msg_hdr.exp.pool      = 0;      /* Must be zero
    msg_out.msg_hdr.exp.alt_path  = 0;      /* Use default path
    msg_out.msg_hdr.exp.addr_mode = ! implicit;   /* Addressing?
    msg_out.msg_hdr.exp.cmpl_code = MSG_NOT_COMPL; /* Zero
    msg_out.msg_hdr.exp.path      = 0;      /* Use primary path
    msg_out.msg_hdr.exp.priority  = priority;
    msg_out.msg_hdr.exp.length    = msg_size;
                                            /* Message size
}
```

Výpis. 6.1 Inicializace odchozího aplikační fronty

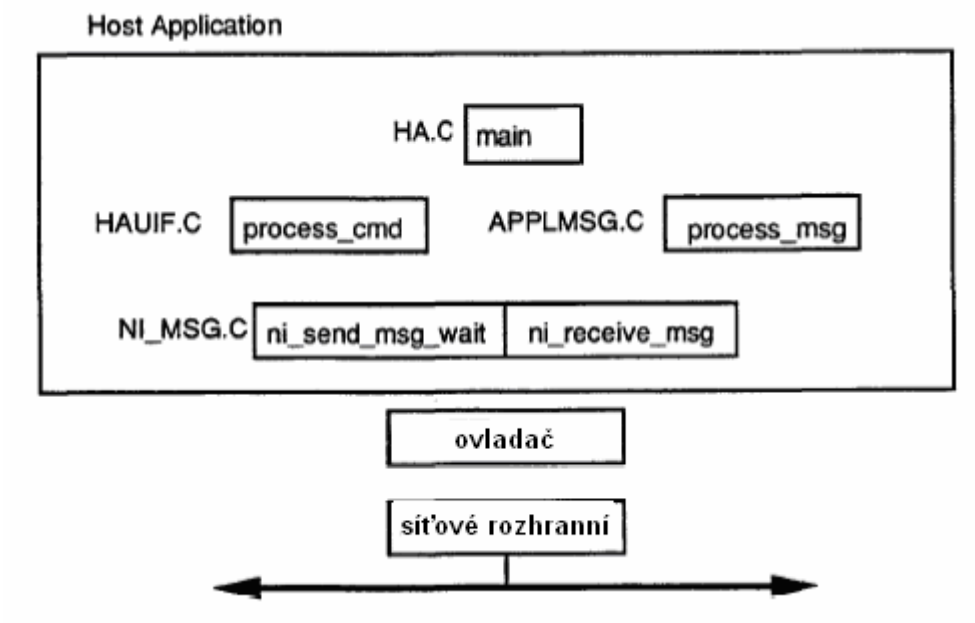
```
void load_NV_config( void ) {       // called from main() startup code
    int fd;
    unsigned byte_count;

    fd = open( "NVCONFIG.TBL", O_RDONLY | O_BINARY );
    if( fd < 0 ) return;        // No file yet
    byte_count = read( fd, nv_config_table, sizeof( nv_config_table )
);
    if( byte_count == sizeof( nv_config_table ) ) {     // success
        printf( "Network variable config table loaded\n" );
        close( fd );
        return;
    }
    printf( "Unable to load NV config table from file\n" );
    close( fd );
}
```

Výpis. 6.2 Funkce pro nahrávání tabulky síťových proměnných

6.3.3.3. Datová struktura HA

Na datové struktuře HA (obrázek 6.11) je naznačena hierarchie jednotlivých zdrojových kódů a o co se daný kód stará. V případě varianty HA pro On-time by HA ještě navíc zahrnoval ovladač (OntimeDriver.c), protože je v této implementaci její součástí.



Obr. 6.11 Datová struktura HA

Konfigurační tabulka je rozdělená na dvě části. Konfigurační tabulku, která se ukládá na disk a tabulku s obsahem hodnot proměnných. Konfigurační tabulka je pole typů `nv_struct` (záznam jedné proměnné). Tento typ je ukázán ve výpisu 6.3

```
typedef struct {
bits    selector_hi : 6;
bits    direction   : 1;
bits    priority    : 1;
bits    selector_lo : 8;
bits    addr_index  : 4;
bits    auth        : 1;
bits    service     : 2;
bits    turnaround  : 1;
} nv_struct;
```

Výpis. 6.3 Typ nv_struct

Tabulka s obsahem hodnot proměnných `nv_value_table` obsahuje velikost potřebnou pro alokaci dat proměnné, tato velikost je definována v SNVT (kapitola 5.4.1), jestli se jedná a vstup nebo výstup, název proměnné, čtecí (`read_**`) a zapisovací (`print_**`) funkci proměnné. Každý typ síťové proměnné má svoji vlastní

čtecí/zapisovací funkci, které na základě typu (SNVT) ví jaká data má očekávat. Příklad konfigurační tabulky hodnot pro čtyři proměnné je ve výpisu 6.4.

```
network_variable nv_value_table[ NUM_NVS ] =  { // RAM storage for NVs
  { 31,  NV_OUT, SNVT_info.string_out_name, print_asc,   read_asc   },
  {  1,  NV_OUT, SNVT_info.disc_out_name,   print_disc,  read_disc  },
  {  1,  NV_OUT, SNVT_info.cont_out_name,   print_cont,  read_cont  },
  {  4,  NV_OUT, SNVT_info.float_out_name,  print_float, read_float },
  { 31,  NV_IN,  SNVT_info.string_in_name,  print_asc,   read_asc   },
```

Výpis. 6.3 Příklad tabulky hodnot proměnných

Rozhraní mezi aplikací a uživatelem (HAUIF.C viz obrázek 6.11) zprostředkovává uživateli prostřednictvím klávesnice následující funkce

E – ukončení HA (`exit_func`)

N – zobrazení konfigurační tabulky (`NV_table`)

P – dotaz na znovunačtení hodnoty ivstupní proměnné (`NV_poll`)

U – zmenu hodnoty síťové proměnné (`NV_update`)

6.3.3.4. Popis MIP API

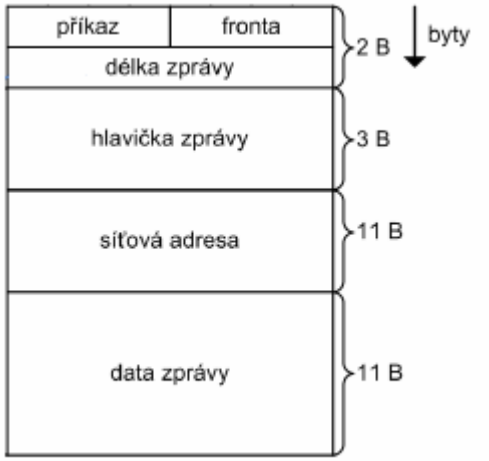
Obecný popis možností a architektury MIP byl již uveden v kapitole 6.1.2., zde se zaměřím na stručný popis MIP API samotného, detailní informace lze nalézt v [11]. MIP API je dostupné v podobě dll knihovny wldv32.dll, která je dodávána spolu s ovladači k PC104 kartě. Je také distribuována k Sample Host Application firmy Echelon. Dále je také dostupná implementace pro Linux a Max OS X. Pro On-time není tato implementace dostupná, proto v její implementaci v rámci této práce vidím jako přínos. Implementaci pro On-time se věnuje kapitola 7.4.

MIP API tvoří následující funkce (napojení na MIP je vidět na obrázku 6.6):

- `ldv_open` - slouží pro otevření spojení se síťovým rozhraním, kde funkce vrací identifikátor na otevřené zařízení, HA umožňuje pracovat s více zařízeními, proto se tento identifikátor používá jako parametr v ostatních funkcích
- `ldv_close` - slouží k uzavření spojení se síťovým rozhraním, tato funkce uvolňuje alokované místo

- `ldv_read` - slouží k vyčtení příchozí zprávy ze síťového rozhraní, funkce předává frontu z aplikace do ovladače
- `ldv_write` - slouží k odeslání zprávy do síťového rozhraní, funkce předává frontu z ovladače do aplikace

Prostřednictvím MIP API se posílají zprávy s formátem zobrazeným na obrázku 6.12., který ve zdrojovém kódu implementuje typ `ExpAppBuffer` (Výpis 6.5). Deklarace typů použitých v `ExpAppBuffer` jsou uvedeny v Příloze B, kde je uveden u jednotlivých bitů jejich význam.



Obr. 6.12 Formát aplikační fronty zprávy MIP API

```
typedef struct {
    NI_Hdr       ni_hdr;            /* Network interface header     */
    MsgHdr       msg_hdr;           /* Message header               */
    ExplicitAddr addr;              /* Network address              */
    MsgData      data;              /* Message data                 */
} ExpAppBuffer;
```

Výpis. 6.5 Deklarace typu ExpAppBuffer

7. Operační systém pro řízení v reálném čase (RTOS)

Oblast použití operačních systémů pro řízení v reálném čase RTOS (Real Time Operating System) neustále narůstá. Kromě tradiční oblasti řízení technologických procesů, kde se jedná často o kritické úlohy reálného času např. antiblokovací brzdový systém ABS u vozidel, a informačních systémů pro práci v reálném čase, kde se jedná o nekritický systém reálného času. Rychle narůstá použití RTOS zejména v oblasti vestavěných systémů (Embedded Systems). Aplikační oblasti použití vestavěných řídicích systémů s mikropočítači se neustále rozšiřuje a v současné době je lze nalézt ve všech oblastech lidské činnosti. Mezi hlavní oblasti jejich nasazení lze právě zařadit řízení v reálném čase. Tyto vestavěné systémy nás obklopují v každodenním životě od automobilů, domácích spotřebičů a mobilních telefonů v osobním životě, přes ordinace lékařů, průmyslovou automatizaci ve výrobních podnicích, až po kosmické lety. Vestavěné systémy jsou charakterizovány omezenou velikostí paměti, omezeným výkonem, nestandardními interfacy s okolním prostředím. Vestavěné systémy jsou spojeny s okolním světem pomocí senzorů, akčních členů a speciálních komunikačních spojení. Vestavěné systémy pracující v reálném čase v omezeném prostředí, omezení je dáno velikostí paměti a výkonem mikroprocesoru. Často je požadováno, aby vestavěné systémy poskytovaly své služby v přísně stanoveném čase vzhledem k uživateli a okolnímu světu. Je to právě paměť, rychlost a časová omezení, které určují použití RT operačního systému ve software vestavěného systému. RT systémy a vestavěné systémy mají požadavky na reálný čas, zavádí striktní časové limity pro dodání výsledku zpracování. Operační systémy, které se používají u klasických a vestavěných systému, se často liší pouze svým rozsahem nabízených služeb.

7.1. Srovnání RTOS s obecným operačním systémem

Mnoho NRT (non real-time) operačních systémů rovněž poskytuje obdobné služby jádra systému [14]. Základním rozdílem mezi obecným operačním systémem a RTOS je požadavek na „deterministické“ chování v čase u RTOS. Formálně „deterministické“ chování v čase znamená, že služby operačního systému spotřebují pouze známé a

požadované množství času. Teoreticky lze čas těchto služeb vyjádřit pomocí matematických vztahů. Tyto vztahy musí být přísně algebraické a nesmí obsahovat žádné náhodné složky. Náhodné prvky v čase služeb mohou způsobit náhodná zpoždění v aplikačním software a mohou způsobit, že aplikace náhodně nestihne stanovené RT časové termíny – a tento scénář je jasně nepřijatelný pro RT vestavěné systémy. Obecné NRT operační systémy jsou často zcela nedeterministické. Jejich služby mohou zavést náhodné zpoždění do aplikačního software a takto způsobit pomalou schopnost reagovat na podněty z aplikace a navíc v neočekávaných časových okamžicích. Jestliže je požadován po vývojáři NRTOS algebraický vztah popisující chování v čase některé ze služeb operačního systému (například posílání zpráv mezi procesy), nikdy nedostanete požadovaný algebraický vztah. Kromě toho vývojář NRTOS (jako je Windows, Unix nebo Linux) se na vás podívá rozpačitým nebo nechápajícím pohledem. Deterministické chování v čase jednoduše není cílem těchto obecných operačních systémů. Na druhé straně operační systémy reálného času často jdou až za základní požadavek determinismu. Většina služeb jádra těchto operačních systémů nabízí konstantní časování, které je nezávislé na zavedení. Jinak řečeno, algebraický vztah je tak jednoduchý jako vztah: T(message_send) = konstanta, bez ohledu na délku zaslané zprávy nebo na jiných faktorech jako je počet procesů, front a zpráv, které řídí RTOS.

7.2. Stručný popis On-time RTOS-32

On-time RTOS-32 je operační systém pro řízení v reálném čase, který je orientován pouze na skupinu procesorů založených na platformě Intel pro PC. Procesory jsou ale použity pro vestavěné PC systémy, jako je např. systém popsaný v kapitole 8.3. On-time je kompatibilní s Win32 API[12] a dynamickými knihovnami (DLL) Windows. Používá se jako RTOS pro vestavěné systémy s 32/64 - bitovými x86. Je to RT vestavěný OS pro kritické úlohy s podmnožinou jádra Windows použitelný od 16k paměti. Podporuje okolo 290-ti Win32 API funkcí.

[12] Rozhraní použité v operačních systémech Windows

7.2.1. Hlavní vlastnosti

plně integrovatelný do Microsoft Visual Studio, Borland Delphy a dalších

není potřeba licence pro každé zařízení (pouze licenci pro vývojové prostředí On-time)

dostupné všechny zdrojové kódy

zdarma testovací verze

7.2.2. Komponenty On Time RTOS-32

On-time je jednoprocesní multithreadový systém, který má modulární uspořádání jehož základem je modul RTTarget-32. Další moduly jsou volitelné (rozšiřují možnosti modulu RTTarget) a jsou zachyceny na obrázku 7.1.

Obr. 7.1 Uspořádání modulů On-time

RTTarget-32 – jádro operačního systému a vývojové nástroje

RTTarget-32 obsahuje všechny vývojové nástroje, které jsou potřeba ke spuštění 32 bitové aplikace pro vestavěný systém. Umožňuje spouštění aplikací kompilovaných pro Win32 na vestavěném systému. Program se vyvíjí pod Microsoft Windows s použitím standardního 32 bitového kompileru (Borland C/C++, Microsoft Visual C/C++, Borland Delphi). Provádí bootování a inicializaci hardwaru, run-time knihovny kde jsou implementovány funkce jako printf, mallock, fopen, spouští lokátor, umožňuje vzdálený debugging přes sériový a paralelní port, nebo ethernet. Podporuje běh až na 32 procesorech a podporu Windows DLL knihoven. Obsahuje také ovladače pro klávesnici, textový režim obrazovky, sériového portu, myši a další. Podrobný popis naleznete na http://www.on-time.com/rttarget-32.htm

RTKernel-32 – RT plánovač (Scheduler)

RTKernel-32 je preemptivní RT multivláknový plánovač pro On-time. Modul je malí (16k kódu, 6k dat), rychlí a nabízí RT časové odezvy. Zprostředkovává Plánovač (nabízí Priority scheduling, Round-Robin scheduling, Priority Ordedred Queues, Deterministi scheduling), přerušení, podporu pro více procesorů, debugging, emulaci

Win32 a další moduly (klávesnici, monitorování CPU, přerušení od timeru a další). Podrobný popis naleznete na http://www.on-time.com/rtkernel-32.htm

RTFiles-32 – implementace souborového systému

Umožňuje aplikaci přístup k souborům na disketě, hard disku, flash disku, USB disku, DVD a dalších. Podporuje souborové systémy ISO 9660, UTF-8, FAT-12, FAT-16, FAT-32. Podrobný popis naleznete na http://www.on-time.com/rtfiles-32.htm

RTIP-32 – implementace TCP/IP stacku

Modul implementuje jádro TCP/IP protokolu pro komunikaci prostřednictvím ethernetu a sériové linky skrze Berkley societ API. Zdrojový kód je kompatibilní s Windows WinSock 1.1 API. Podporuje protokoly UDP, TCP, BOOTP, NAT, NATP a další. Podporuje i protokoly vyšší vrstvy FTP, TFTP, Web server, POP3, SMTP, TELNET, DHCP, SNTP, SMA a další. Podrobný popis naleznete na http://www.on-time.com/rtip-32.htm

RTPEG-32 – implementace GUI (grafického rozhranní)

Modul je objektově orientovaná C++ GUI knihovna. Obsahuje ovladače pro VGA a SVGA/VESA grafický hardware. Podporuje vzhled podobný Windows, klávesnici, myš, dotykovou obrazovku, nástroj pro vývoj GUI. Podrobný popis naleznete na http://www.on-time.com/rtpeg-32.htm

RTUSB-32 – RT USB stack

Modul implementuje Universal Serial Bus (USB) host stack pro vestavěné systémy. Obsahuje protokol pro stack jádra, ovladače zařízení a nejnižší vrstvu komunikace API a třídu horní vrstvy ovladače. Jsou podporovány verze USB 1.1 a 2.0, pro tři typy řadičů (UHCI, OHCI, EHCI). Podrobný popis naleznete na http://www.on-time.com/rtusb-32.htm

7.3. Ovladač PC104 karty pro On-time

Jedním z cílů této práce bylo vytvoření ovladače pro PC104 kartu pro operační systém On-time. Výrobce karty dodává spolu s kartou ovladače pro Windows ve formě instalačního souboru a zdrojové kódy pro příklad implementace karty pro operační systém Linux. Výrobce nedodává detailní popis hardwaru, proto ovladače vycházejí ze zdrojových kódů implementace pro Linux a implementace rozhranní MIP/P50 (kapitola 6.3.2.1.). Chování rozhraní bylo testováno pomocí ovladače ve Windows a získaných informací bylo využito při psaní ovladače pro On-time. Protože zdrojové kódy pro Linux obsahují implementaci nejen PC104 karty použité v práci, ale i ostatních karet pro ISA a PCI rozhraní firmy Gesytec, bylo nutné se nejdříve v kódu zorientovat. Ovladač se skládá z několika částí (obrázek 6.6). Automatu pro obsluhu karty, který zprostředkovává uplink fronty (zápis dat do karty), samotné poslání dat na cílovou adresu v síti Lonwork zajistí neuron chip, který dekóduje cílovou adresu v síti. Downlink fronty (čtení dat z karty), čili data přijatá neuron chipem a uložena do fronty příchozích zpráv. Tyto procesy jsou popsány v kapitole 6.3.2.2. V případě příchozích dat karta vyvoláním přerušení (IRQ) a z obslužné rutiny se zavolá spouštění automatu pro obsluhu karty a tím dojde k downlink/uplink fronty dat. Pro odesílání dat slouží samostatný thread, který běží na pozadí a periodicky pouští automat pro obsluhu karty, čímž se zajistí uplink/downlink fronty dat. Protože je celá aplikace více threadová, takže může nastat situace, že ovladač i aplikace přistupují ke stejným datům, jsou v kódu použity semafory. Semafory povolují při změně společných přístup pouze jednomu threadu, tak že nastaví u těchto dat příznak, tím se znemožní operování z více thredu před ukončením zápisu. Po ukončení zápisu do sdílených dat, semafor smaže příznak a tím data uvolní jinému threadu. Mohlo by se zdát, že obsluha IRQ je nadbytečná, když se periodicky volá obsluha karty z threadu. Ale v případě, že před příchodem dat by thread starající se o obsluhu karty uspal, potom by muselo čekat, až se probudí a mohlo by nastat to, že vyprší čas pro příjem a data nebudou přijata. Aby toto nenastalo, využívá se obsluha IRQ, která zaručí příjem dat, protože ta volá automat pro obsluhu karty. Tím je zajištěno, že ne vyprší čas pro příjem dat.

Testování a ladění správné funkčnosti ovladače bylo možné až po implementaci rozhranní mezi aplikací a vlastním ovladačem, které je popsáno v následující kapitole.

7.4. Rozhraní pro přístup aplikace k ovladači karty

Rozhraní pro přístup aplikace k ovladači karty se v operačním systému Windows využívá DLL knihovna wldv32.dll (dodávaná výrobcem), která implementuje funkce `ldv_open`, `ldv_close`, `ldv_read`, `ldv_write`, napojení těchto funkcí na MIP je vidět na obrázku 6.6. Dalším úkolem proto bylo vlastní implementace funkcí `ldv_open`, `ldv_close`, `ldv_read`, `ldv_write` pro napsaný ovladač. Čímž proběhne náhrada knihovny wldv32.dll, která není pro On-time použitelná. Tyto čtyři funkce rozhrání mezi hardwarem (ovladačem) a vlastní aplikací. Tyto funkce nám umožňují tvořit výslednou aplikaci, která je téměř stejná pro operační systém Windows i On-time.

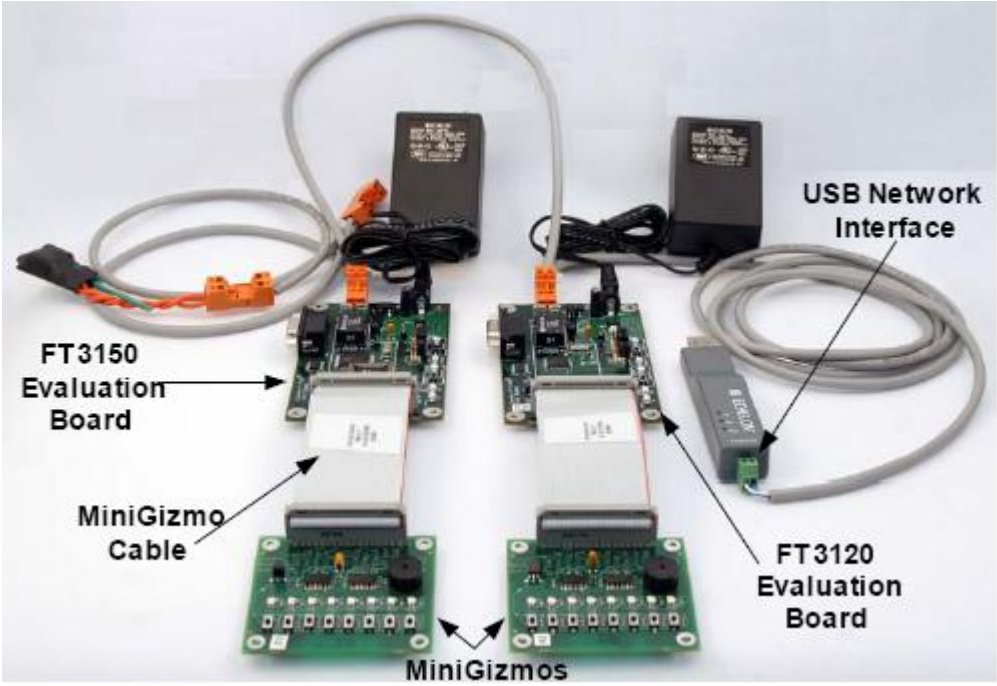
Funkce `ldv_open` zajišťuje inicializaci karty, otestuje komunikaci, proběhne inicializace semaforů, spustí se thread automatu pro obsluhu karty a povolí se přerušení od karty. Funkce `ldv_read` slouží pro alokaci paměti dat z příchozích zpráv a přidání do aplikační fronty (obrázek 6.12). Funkce `ldv_write` je opakem funkce read a slouží pro předávání dat z aplikace. Funkce `ldv_close` se používá pro ukončení práce s kartou. Ukončí se thread automatu pro obsluhu karty, zruší se přerušení od karty a uvolní se paměť nealokovaná aplikací.

8. Použitý hardware

V této kapitole je stručně popsaný hardware, který byl při práci použit a na kterém byla následně i celá práce testována. Za povšimnutí stojí, že zařízení spolu komunikují prostřednictvím Lonworks sítě a jsou od různých výrobců. Testovací síť byla složena ze dvou nódů. První nód byla vývojová deska z Kitu s chipem TF3150 (kapitola 8.1.). Druhý nód byl operátorský panel touch 11 (kapitola 8.3.) s PC104 kartou (kapitola 8.2.) v režimu Host Selection. Pro konfiguraci a sledování sítě, sloužilo PC s převodníkem USB/Lonworks z kitu (kapitola 8.1.).

8.1. Mini Evaluation Kit

Mini Evaluation Kit byl zakoupen pro účele testovaní sítě Lonworks. Kit obsahuje dvě vývojové desky, které obsahují neuron chip. Jedna obsahuje chip FT3150 a druhá FT3120. Tyto desky jsou propojeny pomocí kabelu s deskami vstupů/výstupů. Aplikace běží přímo v neuron chipu (kapitola 6.1.1). Protože nebylo mým cílem vyvíjet aplikaci pro neuron chip, použil jsem pro testování jednu z ukázkových aplikací firmy Echelon a to MGSwitch. Tato aplikace má jednu vstupní a jednu výstupní proměnou. Obě tyto proměnné jsou typu SNVT_switch (výpis 5.1.). Dále byl z tohoto kitu použit převodník USB na Lonworks. Tento prěvodník byl připojen do PC a sloužil rozhranní mezi PC a sítí Lonworks. PC s převodníkem sloužilo pro konfiguraci a monitorování sítě. Podrobný popis kitu je v pdf Mini EVK User's Guilde, který je uložen na přiloženém CD v adresáři Datasheets.



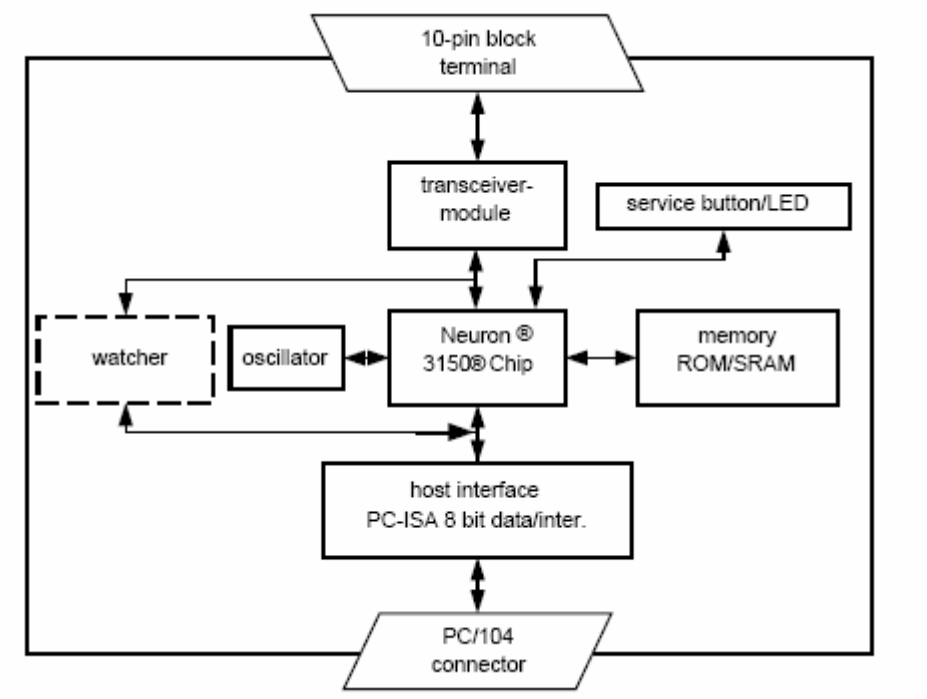
Obr. 8.1 Mini Evaluation Kit od firmy Echelon

8.2. PC104 karta od firmy Gesytec

Na implementaci této karty je postavena celá práce. Tato karta je v nabídce firmy Gesytec a vyrábí se ve dvou provedeních a to karta s firmwarem network interface (pro network selection mód) a s firmwarem MIP (pro host selection mód).

Obr. 8.2 PC104 karta od firmy Gesytec

Tato karta má sběrnici ISA a mapuje se do I/O prostoru. Adresa karty se spolu s hodnotou IRQ[13]se nastavuje pomocí DIP přepínače na kartě (z výroby adresa I/O prostoru 340-344 a IRQ 5). Karta má servis tlačítko, při jehož stisknutí je vysláno Neuron ID. Karta obsahuje LED, která signalizuje v jakém stavu se karta nachází. Na obrázku 8.3. je zobrazeno blokové schéma rozhraní PC104 karty. Podrobný popis karty, jako je popis pinů výstupního konektoru, tabulku DIP přepínače, stavy LED, naleznete v pdf LPC_Manual-E, který je uložen na přiloženém CD v adresáři Datasheets.



Obr. 8.2 Blokové schéma rozhraní PC104 karty

[13] Přerušení vyvolané kartou

8.3. Operátorský panel Touch 11

TOUCH11/PC je malý operátorský panel, určený jako grafické rozhraní člověk-stroj. Základem operátorského panelu je procesorová deska s integrovanými periferiemi (VGA, RS232, USB, ETHERNET, LPT, PS2, IRDA 115 kb/s, audio a slotem pro rozhraní ISA), s procesorem AMD GEODE LX800, pamětí RAM 256MB a CompactFlash kartou. Panel má LCD displej s úhlopříčkou 5,7“, rozlišení 640x480 bodů a podsvícení je bílými LED diodami. Displej je doplněn dotykovým panelem s odporovou technologii.

Procesorové desky, která je součástí panelu, bylo využito pro běh Host application, které byla nahrána na CompactFlash kartě. Aby tento panel mohl být zapojen do sítě Lonwork, byla do něho přidána PC104 karta (kapitola 8.2.)

Obr. 8.3 Operátorský panel od firmy SofCon

9. Závěr

V této práci jsem se seznámil s technologií LonWorks a zabýval jsem se realizací embedded zařízení, které komunikuje s ostatními zařízeními v síti LonWorks pomocí protokolu LonTalk. Provedl jsem rozbor přístupu k síti LonWorks a na základě tohoto rozboru jsem navrhl architekturu pro realizaci zařízení, které slouží jako nód v síti LonWorks. Toto zařízení je realizováno jako aplikace běžící v embedded zařízení s operačním systémem Windows, nebo systémem reálného času On-time. Pro přístup k síti LonWorks používá síťové rozhraní s ovladačem, který poskytuje MIP API.

Karta obsahující MIP firmware zde byla využita pouze jako komunikační koprocesor, proto bylo potřeba implementovat prezentační a aplikační vrstvu protokolu LonTalk. Implementace prezentační a aplikační vrstvy protokolu LonTalk spolu s implementací interpretace síťových proměnných dle sdružení LonMark byla proto podstatnou částí této práce. Z tohoto důvodu jsem také teoretickému rozboru prezentační a aplikační vrstvy protokolu LonTalk věnoval kapitolu 5. Popisu mé implementace je věnována kapitola 6. Mnou provedená implementace vychází ze specifikace protokolu LonTalk ANSI/EIA 709.1-B. Rozhraní síťových proměnných také obsahuje implementaci interpretace obsahu síťových proměnných dle standardů sdružení LonMark. Aplikace využívající rozhraní síťových proměnných je tedy kompatibilní s ostatními zařízeními jiných výrobců, která disponují rozhraním dle standardů sdružení LonMark. V rámci implementace rozhraní síťových proměnných jsem vytvořil aplikaci (Host Application) popsanou v kapitole 6.3.3., kterou lze použít pro čtení hodnot síťových proměnných ze sítě LonWorks a pro zasílání hodnot síťových proměnných do zařízení v síti LonWorks.

Práce se dá rozdělit na část implementace pro operační systém Windows. Kde jsem k přístupu ke kartě využil dodávaný ovladač a dll knihovnu, která vytváří rozhranní mezi ovladačem a aplikací. Zde jsem se zabýval implementací aplikace. Další část je implementace pro operační systém reálného času On-time (kapitola 7). Kde jsem využil aplikaci vytvořenou pro Windows a upravil ji tak, aby ji bylo možné na podmíněný překlad přeložit i pro On-time. Za největší přínos této práce považuji vytvoření ovladače PC104 karty pro On-time a vlastní implementaci rozhraní mezi ovladačem a aplikací. Protože tato implementace neexistovala. Ovladač vychází z implementace z pohledu

MIP (kapitola 6.3.2.) a byl napsán pro kartu v režimu Host selection (kapitola 6.3.3.2.). Rozhranní mezi ovladačem a aplikací je univerzální a lze jel využít i u jiných zařízení splňující specifikaci protokolu LonTalk.

Veškerý software vytvořený v rámci této práce je napsán v jazyce C++ v Microsoft Visual studiu 2008 s modulem pro On-time. Vyvinutý software by bylo možné ještě zdokonalit, definice síťových proměnných by mohla probíhat dynamicky při první spuštění aplikace, namísto statické definice v aplikaci. Také by bylo možné u aplikace vylepšit rozhraní, aby mohla sloužit jako samostatný modul v rozsáhlejším sytému. Věřím, že software a ovladač pc104 karty pro On-time, které jsem vytvořil v rámci této práce budou přínosné. Zejména pro toho, kdo se bude zabývat implementací zařízení komunikující prostřednictvím sítě LonWorks.

Literatura

[1] LonWorks Technology Device Data, Rev. 4, Motorola, 1997

[2] LonTalk Protocol Specification, V 3.0, Echelon Corporation, 078-0125-01A, 1994

[3] LonTalk Protokol, Echelon Corporation, 005-0017-01 Rev. C, 1993

[4] LonMark Layer 1 – 6 Interoperability Guidelines, V 3.3, 078-0014-01F, 2002

[5] LonMark Application Layer Interoperability Guidelines, V 3.3, 078-0120-01F, 2002

[6] LonMark SNVT Master List, Version 12, 2003

[7] LonMark SCPT Master List, Version 12, 2003

[8] Tiersch, LonWorks Technology An Introduction, Lontech, 3-932875-23-0, 2005

[9] Láska, Komunikace se sítí LonWorks, Diplomová práce, ČVUT FEL, Praha, 2007

[10] Neuron C Programmer's Guide, Echelon Corporation, 078-0002-02G

[11] LonWorks Microprocessor Interface Program User's Guide, Rev.3, Echelon Corporation, 078-0017-01C

[12] LonWorks Host Application Programmer's Guide, Rev. 2, Echelon Corporation, 078-0016-01B

[13]LonMaker User's Guide, Echelon Corporation, 078-0173-01B

[14] Hall, Modern operating system, Tanenbaum a.s., 1992

[15] LonWorks Installation Overview, LonWorks Engeneering Bulletin, 005-0006-01C

[16] Introduction to the LonWorks system, Echelon Corporation, 078-0183-01A

[17] Herout, U4ebnice jazyka C IV. Přepracované vydání, koop, 80-7232-220-6, České Budějovice, 2004

[18] Microsoft, MSDN [online], http://msdn.microsoft.com

[19] Mikula, Inteligentní sběnice LonWorks [online], http://automa.hjc.cz

[20] On-time, [online], http://www.on-time.com/rtos-32.htm

[21] Vojáček, Sběrnice LonWorks [online], http://automatizace.hw.cz

[22] Data sheets

Gesytec, Aachen

Loytec electronics, Wien

TAC GmbH, Oberhausen

SofCon s.r.o, Praha

NEWRON System, Columbiers

Seznam příloh

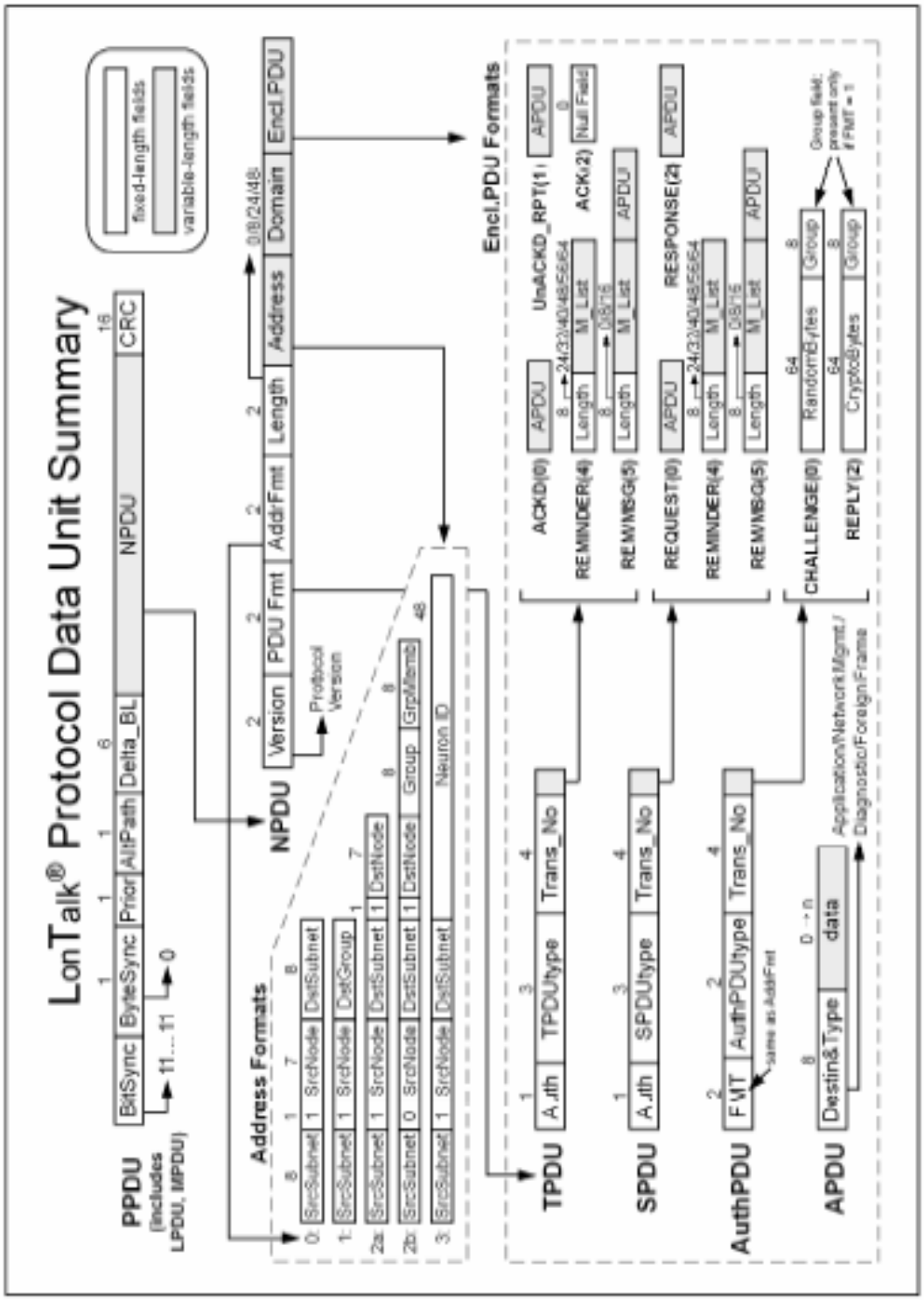
A . Začlenění aplikačního rámce v rámci celé zprávy protokolu LonTalk

B**.** Typy deklarovaná v typu ExpAppBuffer

C. CD s elektronickou podobou tohoto dokumentu, programem Host Application, ovladači pc104 kartu. Datasheety použitého hardwaru.

Příloha A

Začlenění aplikačního rámce v rámci celé zprávy protokolu LonTalk



Příloha B

Typy deklarovaná v typu ExpAppBuffer

```
typedef union {
    struct {
        bits queue  :4;         /* Network interface message queue
                                /* Use value of type 'NI_Queue'
        bits q_cmd  :4;         /* Network interface command with
queue */
                                /* Use value of type 'NI_QueueCmd'
        bits length :8;         /* Length of the buffer to follow
    } q;                        /* Queue option
    struct {
        byte     cmd;           /* Network interface command w/o queue
                                /* Use value of type 'NI_NoQueueCmd'
        byte     length;        /* Length of the buffer to follow
    } noq;                      /* No queue option
} NI_Hdr;
```

```
typedef struct {

    bits   tag       :4;        /* Message tag for implicit addressing
                                /*  cookie for explicit addressing */
    bits   auth      :1;        /* 1 => Authenticated
    bits   st        :2;        /* Service Type - see 'ServiceType'
    bits   msg_type  :1;        /* 0 => explicit message
                                /*      or unprocessed NV
/*-----------------------------------------------------------------*/
    bits   response  :1;        /* 1 => Response, 0 => Other        */
    bits   pool      :1;        /* 0 => Outgoing                    */
    bits   alt_path  :1;        /* 1 => Use path specified in 'path'*/
                                /* 0 => Use default path            */
    bits   addr_mode :1;        /* 1 => Explicit addressing,        */
                                /* 0 => Implicit                    */
                                /* Outgoing buffers only            */
    bits   cmpl_code :2;        /* Completion Code - see 'ComplType'
    bits   path      :1;        /* 1 => Use alternate path,         */
                                /* 0 => Use primary path             /
                                /*      (if 'alt_path' is set)
    bits   priority  :1;        /* 1 => Priority message
*/
/*-----------------------------------------------------------------*/
    byte   length;              /* Length of msg or NV to follow
                                /* not including any explicit address
                                /* field, includes code byte or
                                /* selector bytes
} ExpMsgHdr;
```

```
typedef union {
    RcvAddrDtl  rcv;
    SendAddrDtl snd;
    RespAddrDtl rsp;
} ExplicitAddr;
```

```
typedef union {
    ExpMsgHdr  exp;
    NetVarHdr  pnv;
} MsgHdr;
```

```
typedef struct {
    bits   tag       :4;        /* Magic cookie for correlating
                                /* responses and completion events
    bits   rsvd0     :2;
    bits   poll      :1;        /* 1 => Poll, 0 => Other
    bits   msg_type  :1;        /* 1 => Processed network variable
*/
/*-----------------------------------------------------------------*/
    bits   response  :1;        /* 1 => Poll response, 0 => Other
    bits   pool      :1;        /* 0 => Outgoing
    bits   trnarnd   :1;        /* 1 => Turnaround Poll, 0 => Other
    bits   addr_mode :1;        /* 1 => Explicit addressing,
                                /* 0 => Implicit addressing
    bits   cmpl_code :2;        /* Completion Code - see above
    bits   path      :1;        /* 1 => Used alternate path
                                /* 0 => Used primary path
                                /*      (incoming only)
    bits   priority  :1;        /* 1 => Priority msg (incoming only)
*/
/*-----------------------------------------------------------------*/
    byte   length;              /* Length  network variable to follow
                                /* not including any explicit address
                                /* not including index and rsvd0 byte
} NetVarHdr;
```