České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Komunikační protokol a jednotky pro distribuovaný sběr dat a řízení

Praha, 2009 **Autor: Stanislav Hrbek**

Prohlášení

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

V Praze dne 20. 5. 2009

podpis

Poděkování

Děkuji především vedoucímu práce panu Ing. Pavlu Píšovi za vedení a pomoc při realizaci tohoto projektu. Dále děkuji svému nejbližšímu okolí za podporu při studiu.

Abstrakt

Tato diplomová práce se zabývá protokolem uLan, jeho implementací na různých platformách a aplikačním software, využívajícím jeho rozhraní.

V teoretické části je pojednáváno o vlastnostech protokolu, jeho možnostech a způsobech použití v praxi. Dále je zde detailně vysvětlena implementace protokolu na všech úrovních a to především pro použití bez operačního systému. Práce obsahuje také stručný popis objektového rozhraní protokolu a prohlížeče pro něj vytvořeného. Je zde též zahrnut popis aplikace převodníku USB-uLan - důležité pro praktickou část práce.

Praktická práce zahrnuje úpravu implementace protokolu, tak aby umožňovala realizaci logických jednotek. Dále zahrnuje popis využití těchto změn při realizaci dvou logických jednotek pracujících v rámci jednoho fyzického zařízení.

Abstract

This diploma thesis is focused to uLan communication protocol, its implementation on various platforms and to application software using the protocol for data acquisition and instruments control.

Theoretical part of this work discusses about protocol properties, its suitability for different kinds of field applications. There is also protocol implementation explained in detail, especially for small embedded devices (system-less version). It also contains simple description of object interface and browser designed for object values changes and readings. Description of USB-uLan aplication, which is important for practical part of work, is included too.

The extension of protocol driver implementation to allow multiple logical units to be implemented in single physical device utilizing one communication interface hardware has been realized in practical part of the work. The device firmware utilizing feature of multiple logical units in single physical device has been developed as well.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Stanislav Hrbek**

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

Název tématu: **Komunikační protokol a jednotky pro distribuovaný sběr dat a řízení**

Pokyny pro vypracování:

1. Otestujte jednotky pro distribuovaný sběr dat a řízení, které využívají fyzickou vrstvu RS-485 na katedře Řídicí techniky vyvíjeného multimaster protokolu uLan. Prověřte možnosti a zdokumentujte použití tohoto řešení pro sběr dat ze zdrojů přesných, nikoliv však příliš rychlých signálů.
2. Implementujte firmware, který umožní využití některé z těchto jednotek jako přesného AD převodníku a zároveň jako převodníku sběrnice uLan USB. Při práci spolupracujte s týmem z katedry, který pracuje na využití komunikace pro domovní automatizaci. Převodník by měl komunikovat i s programem CHROMuLAN vyvíjeným na Přírodovědecké fakultě Univerzity Karlovy.
3. Doplňte knihovny pro komunikaci s jednotkami o vrstvu umožňující snadný přístup ke slovníku proměnných přístupných přes sběrnici uLan.

Seznam odborné literatury:

lJames P. Lynch, ARM Cross Development with Eclipse, Olimex, 2005
lThe Insider's Guide To The Philips ARM7-Based Microcontrollers (LPC21xx)
http://www.hitex.co.uk/arm/lpc2000book/
lARM microcontroller Wiki - http://www.open-research.org.uk/ARMuC/
luLan communication project- http://ulan.sourceforge.net/

Vedoucí: Ing. Pavel Píša

Platnost zadání: do konce letního semestru 2009/10

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 27. 2. 2009

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní době, kdy je v informačních sítích typu internetu běžně dosahováno rychlostí 1Gb/s se zdá, že se sběrnice, založené na sériových standardech RS-232,RS-422 a RS-485, zařadí do historických archivů po boku starším paralelním rozhraním. Ale není tomu tak, tyto sběrnice jsou stále používány v odvětvích, kde datová prostupnost není nejdůležitějším parametrem.

Pro sběrnice používané v průmyslové automatizaci, dopravních systémech nebo v inteligentních budovách jsou důležitější parametry jako například: determinističnost (naopak od náhodného přístupu k médiu), zaručená doba odezvy, snadnost a levnost implementace v zabudovaných systémech, nízká energetická náročnost, šumová imunita ,možnost galvanického oddělení či topologie sítě.

Jednou z takovýchto sběrnic je sběrnice uLan vyvinutá spoječností PiKRON s.r.o. Protokol sběrnice uLan je používán v zařízeních pro kapalinovou chromatografii a systémech sběru dat. Protokol má potenciál použití v domovní automatizaci nebo průmyslových výrobách, kde nejsou kladeny příliš velké požadavky na datovou propustnost. Jedním z rozsáhlých projektů je systém CHROMuLAN.

Nevýhodou tohoto protokolu je chybějící schopnost zajistit provoz multifunkčních jednotek a to především převodníku USB-uLan a A/D převodníku v jedné jednotce. Tato práce má za cíl tento problém vyřešit.

Kapitola 2

uLan

2.1 Základní informace

uLan je zprávově orientovaný multi-master komunikační protokol primárně určený pro vestavěné aplikace. Používá devítibitový znakový formát pro adresaci stanic a jeho základem je spojová a fyzická vrstva standardu RS-485.

Znaky jsou po sběrnici přenášeny stejným asynchronním kanálem jako na RS-232 s odlišností v paritním bitu, který je použit na rozlišení znaků datových od znaků řídicích a adresace. Fyzická vrstva se skládá z jednoho páru kroucených vodičů a budičů standardu RS-485.

Použití devítibitového kódování znaků zjednodušuje přenos dat ve dvojkové soustavě a snižuje vytížení procesoru inteligentních jednotek, protože se procesor nemusí starat o znaky přenášené do dalších uzlů. Jeden z problémů devítibitových přenosů dat je chybějící standardizovaný protokol zpráv. Toto místo se snaží zaplnit protokol uLan.

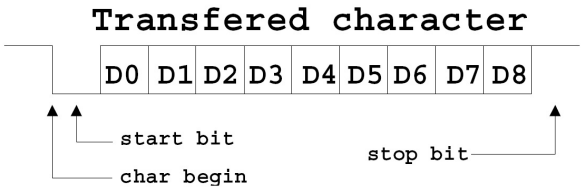
2.2 Historie

Protokol ulan byl navržen v roce 1992 během vývoje nového modulárního systému pro společnost Laboratory Instruments. Hlavními kritérii návrhu byly vlastnosti: multi master sběrnice, jednoduchá fyzická vrstva, deterministický přístup na sběrnici, levné připojení k PC a nízké energetické požadavky pro tehdy vyráběné mikrokontroléry a příslušenství.

Fyzická vrstva RS-485 byla vybrána pro svoji dobrou šumovou imunitu, sběrnicovou topologii, minimální počet vodičů (pouze 2 vodiče) a možnost galvanického oddělení. Toto řešení s výhodou používají i další průmyslové sběrnice, například Profibus.
Mnoho současných mikrokontrolérů implementuje devítibitové rozšíření v jejich UARTech (všechny Intel 8051 a 8096, mikrokontroléry řad Motorola 683xx, Hitachi H8 a další). Intel vyvinul multiprotokolární UART i82510, který je velmi dobře použitelný pro implementaci devítibitových komunikačních rozhraní na straně personálních počítačů.
Návrh protokolu se zdá být tak dobrým, že rámcový formát a šablona pro výměnu dat nebyla pozměněna více než deset let. Objektově orientovaná vrstva byla plánována od začátku. Implementována byla o trochu později v roce 1995, když protokol uLan našel uplatnění v řídicí jednotce spalovacího motoru na přírodní plyn. Byl použit pro získávání dat a nastavování parametrů.
Software na straně PC se časem měnil a vyvíjel. První implementace na straně PC byla určena pro prostředí MS DOS, vlastní obsluha komunikace byla napsána v assembleru x86. První aplikací, která tuto implementaci využívala, byl program pro testování a ladění připojených jednotek založený na prostředí Turbo Visions. Implementace protokolu byla přepsána jako platformově nezávislý ovladač. Původně byl vyvinut na operačních systémech s linuxovým jádrem, poté byl rozšířen o systémově nezávislou vrstvu pro podporu jádra Windows NT. Podpora Windows 2000/XP byla přidána později.

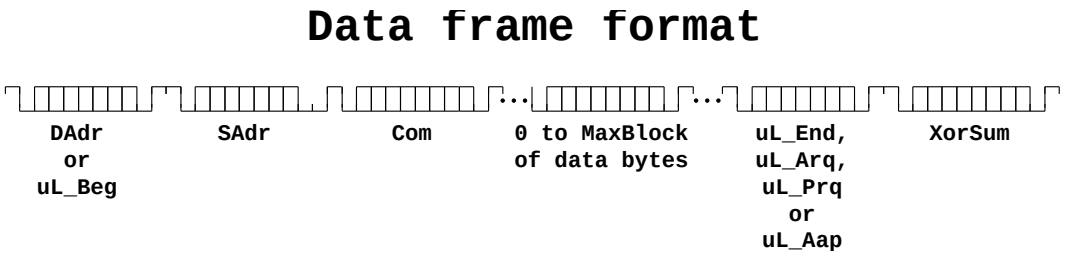
2.3 Popis protokolu

2.3.1 Datový rámec



Obrázek 2.1: Formát přenášeného znaku

Datový rámec je základní jednotkou komunikace uLan. Skládá se z posloupnosti devítibitových znaků. Protože se jedná o asynchronní komunikaci, je každý znak opatřen startbitem a stopbitem. Celková doba přenosu jednoho znaku je jedenáctkrát delší než doba přenosu bitu. Průběh přenosu jednoho znaku je na obrázku 2.1. Pro zvýraznění bude nejvýznamnější bit přenášeného devítibitového znaku D8 vykreslen tučně. Znaky s bitem D8=1 jsou speciální řídící znaky, které se mohou v datovém rámci vyskytovat jen na určitých pozicích a ohraničují přenášená data.



Obrázek 2.2: Formát datového rámce

Datový rámec začíná cílovou adresou (DAdr - destination address) nebo znakem neadresného počátku rámce (**uL_Beg**) s nastaveným bitem D8. Tento znak zachytí všechna zařízení a podle něj se rozhodnou jestli jsou pro ně určeny další znaky. Další znaky jsou již bez nastaveného bitu D8. Jedná se o adresu odesílatele rámce (SAdr - source address) a číslo služby, pro kterou je rámec určen (Com - command). Dále jsou vysílána data rámce

bez nastaveného bitu D8. S daty není vyslána délka. Ta se může pohybovat v rozsahu nula až maximální povolená délka dat MaxBlock. Maximální délku dat je nutné zvolit podle nejdelší přípustné doby odezvy mezi zařízeními, rychlosti komunikace a počtu aktivních zařízení pro konkrétní aplikaci. Předpokládaná hodnota je v rozsahu 256 B až 10 kB. Přenos až dvou byte délky dat by prodlužoval délku rámce a vyžadoval by od všech přístrojů počítání přijímaných byte, proto jsou data ukončena pouze znakem s nastaveným D8. Podle použitého ukončovacího znaku ze čtyř možných ukončovacích znaků (**uL_End**, **uL_Arq**, **uL_Prq**, **uL_Aap**) se rozhoduje o dalším průběhu komunikace po skončení rámce. Jako poslední znak rámce je vyslán kontrolní znak (XorSum - vypočítaný ze všech přenesených znaků včetně DAdr a ukončovacího znaku) bez nastaveného bitu D8.

Pro zajištění zotavení při výpadku zařízení a pro umožnění bezkolizního přepínání výstupních budičů sběrnice jsou definovány následující časové relace. Mezi vysíláním jednotlivých znaků rámce nesmí být větší prodleva než doba vyslání jednoho znaku. Pokud k ní dojde a je cílovým přístrojem rozpoznána, je zpráva prohlášena za poškozenou. Protože při vysílání rámců a odpovědí dochází k změně směru přenosu dat přes transceivery sběrnice, jsou definovány minimální a maximální časy jednotlivých úkonů. Tyto časy jsou definovány poměrem k době přenosu jednoho znaku. Při ukončení vysílání signálu na sběrnici musí být vypnut výstupní budič po skončení vysílání posledního znaku zprávy v době kratší, než je doba vysílání jednoho znaku. Při přechodu z příjmu na vysílání znaku je nutné po příjmu posledního znaku počkat před zapnutím výstupního budiče minimálně po dobu vysílání jednoho znaku. Tato doba by však neměla být delší než doba vysílání dvou znaků. Pokud je čekání na znak delší než trojnásobek vysílání znaku, je předchozí vysílaný rámec považován za nepotvrzený a spojení je ukončeno s chybou.

Popis jednotlivých možných ukončení datového rámce:

- **uL_End** (end) nejjednodušší ukončení rámce. Příjemce nesmí vyslat žádný znak na sběrnici do uvolnění sběrnice. Vysílající přístroj prohlásí okamžitě rámec za odvysílaný a nekontroluje, že byl přijat. Tyto rámce se využívají k přenosu informačních zpráv, u nichž nedoručení nevede ke kritické chybě. Dále je nutné, je použít, pokud je naadresováno více cílů zprávy. V tomto případě by při snaze potvrdit zprávu více přístroji vedl ke kolizi na sběrnici a i k chybě komunikace.

Pro kontrolu přenosu těchto rámců je možné použít vhodného průběhu komunikace mezi aplikacemi, kdy jedna po doručení zprávy reaguje vložením zprávy s odpovědí do výstupní fronty zpráv s adresou DAdr rovnou SAdr původní zprávy.

- uL_Arq (acknowledge request) umožňuje okamžitou kontrolu doručení rámce. Příjemce rámce musí reagovat na toto ukončení jednoznakovou odpovědí bez nastaveného D8. Odpověď **uL_ACK** potvrzuje příjem rámce. Odpověď uL_NAK nebo odmlka po dobu delší než doba vysílání tří znaků je považována za chybu příjmu. Ta může být způsobena zaneprázdněností zařízení, přeplněním vstupní fronty nebo chybou přenosu rámce. Zařízení, které je schopné zjistit, že k chybě došlo v důsledku jeho dočasného zaneprázdnění nebo dočasného přeplnění vstupní fronty, může odpovědět **uL_WAK**. Tato odpověď může být chápána vysílajícím zařízením pro zjednodušení stejně jako **uL_NAK** nebo, je-li to možné, výhodněji. Když dojde k nepotvrzení rámce, snaží se většinou vysílající zařízení při příštím přidělení sběrnice o opakování vyslání rámce. Počet pokusů o opakování je předdefinován. Tento postup může zbytečně zatěžovat sběrnici v případě, že přijímač má přeplněnu vstupní frontu a nebo je celkově přetížen. Výhodnější je, aby při příštím přidělení sběrnice vysílači tento přístroj vyřizoval zprávy pro ostatní přístroje a k zprávě pro přetížený přístroj se vrátil až po vyřízení ostatních zpráv nebo po určité době.

- uL_Prq (proceed request) označuje rámec za určený k okamžitému zpracování. Tento rámec by měl být příjemcem okamžitě zpracován. Způsob zpracování závisí na čísle služby Com a je závislý na aplikaci. Obecně platí, že služby vyžadující okamžité zpracování zprávy mají číslo služby Com 80h na rozdíl od služeb pro příjem do vstupní fronty s Com $\geq$ 80h. Pro některé služby se požaduje okamžitá odpověď příjemce vysílači rámcem s odpovědí. Jiné služby předpokládají vyslání dalšího rámce původním vysílačem. Tyto zprávy musí být zpracovány okamžitě pod přerušením a proto musí být zpracovány rychle a zcela asynchronně s ostatními aplikacemi. To je výhodné pro trasování a inspekci běžících programů v cílovém zařízení, pro vyčítání krátké stavové informace ze zařízení a pro čtení typu zařízení. Příkladem jsou zprávy přechodu do krokovacího režimu, provedení jednoho kroku a pokračování v normálním běhu aplikace, které se skládají pouze z jednoho rámce. Další příklady jsou čtení typu zařízení, stavové informace a paměti

zařízení. V těchto případech se po odvysílání rámce vyžadujícího danou informaci očekává přijetí rámce s odpovědí. Zápis paměti, naopak po vyslání rámce s informacemi kam se bude zapisovat, vyžaduje odvysílání rámce se zapisovanými daty. Nemůže-li příjemce požadovanou činnost provést, rámec ignoruje. Předpokládá-li vysílač odpověď, pak po neúspěšném čekání prohlásí rámec za nepotvrzený.

- **uL_Aap** (acknowledge and proceed request) vyžaduje shodnou činnost jako **uL_Prq**. Před vlastní zpracováním rámce příjemcem vyžaduje potvrzení přijetí rámce a potvrzení schopnosti okamžitě předat rámec službě Com vysláním **uL_ACK**. Pokud rámec nemůže být zpracován okamžitě, například požadovanou činnost nelze provést pod přerušením, je vysláno **uL_NAK**

2.3.2 Řídící znaky

Tyto znaky mají nastaven bit D8 a jsou přijímány všemi zařízeními, ohraničují rámce a rozhodují o stavu obsazení sběrnice. Sběrnice je obsazena po fázi arbitrace přístupu vysláním cílové adresy s nastaveným bitem D8 a nulovým D7. Sběrnice se uvolňuje vysláním vlastní adresy s nastavenými bity D8 a D7 nebo při chybě vysláním **uL_Err**.

- **DAdr** 100h všeobecná adresa
- **DAdr** 101h až 164h adresa cílového zařízení
- **uL_Beg** 175h neadresný počátek rámce, většinou odpovědi
- **uL_END** 17Ch konec rámce
- **uL_ARQ** 17Ah konec rámce a žádost o potvrzení
- **uL_PRQ** 179h konec s žádostí o okamžité zpracování
- **uL_AAP** 176h konec s žádostí o potvrzení a okamžité zpracování
- **uL_ERR** 17Fh chyba bez uvolnění sběrnice
- **uL_ERR** 1FFh chyba s uvolněním sběrnice

Kódy těchto znaků kromě cílové adresy jsou voleny tak, aby jejich minimální Hammingova vzdálenost byla rovna 2. V případě chyby při přenosu znaku pouze v jednom bitu je tato chyba rozpoznána a přenos je považován za chybný.

Při přenosu potvrzení rámce se využívá definovaných kódů s Hammingovou vzdáleností rovnou 4, proto jsou uvedeny také v tomto odstavci přesto, že mají nulový D8.

- **uL_ACK** 019h potvrzení rámce
- **uL_NAK** 07Fh negativní potvrzení rámce
- **uL_WAK** 025h nepotvrzeno a bude potřeba čekat

2.3.3 Přenos zprávy

Přenos zprávy v síti uLan může vyžadovat i více než přenos pouze jednoho datového rámce. Může vyžadovat potvrzení a nebo okamžité provedení služby, které může vyžadovat přenos dalších rámců. Specifikace dovoluje i velmi složitý průběh zpracování zpráv přesto, že většinou nejsou tyto vlastnosti používány ani implementovány. Například je možná konstrukce, kdy zařízení A upozorní zařízení B na přijetí příštího datového rámce speciálním způsobem k tomuto účelu vytvořenou službou. Potom jinou službou předá zařízení C příkaz k vyslání určitých dat do zařízení B. Zařízení C předá data zařízení B. B potvrdí příjem dat, poté převezme sběrnici přístroj A, který uvolní sběrnici. Tento postup umožní zařízení A přenést data z C do zařízení B, aniž by musel celý datový blok načíst a poté znovu vyslat. V stávajících aplikacích však nebylo nutné takto složitá schémata komunikace implementovat. Jestliže se nejedná o přenosy kritické na čas a není nutno z důvodů synchronizace dopravit zprávu více zařízením naráz, je možné docílit podobných výsledků přenosem několika samostatných zpráv.

Aplikace nebo zpracování předcházejících zpráv vyžaduje přenos zprávy. Tuto informaci uloží do výstupní fronty.

Zařízení, které potřebuje vyslat zprávu, čeká na uvolnění sběrnice. Pak provede připojení na sběrnici. Pokud je tato fáze úspěšná, vyšle rámec. Pokud je to vyžadováno, provede

se kontrola potvrzení rámce a přenos dalších datových rámců. Pak je uvolněna sběrnice. Není-li detekována chyba, je zpráva označena jako provedená. V opačném případu je zvýšeno počitadlo nezdařených pokusů. Při dosažení předdefinované hodnoty je zpráva označena za nedoručenou a pokus o vyslání se neopakuje.

Aplikace může být upozorněna o provedení zprávy, pokud to požaduje. Aplikace je vždy informována o zjištěné chybě v přenosu.

Každé zařízení, které má být schopno přijímat zprávy, musí mít přidělenu vlastní adresu a musí kontrolovat všechny znaky s nastaveným D8. Přijme-li znak s nastaveným D8 a nulovým D7 rovný svojí adrese nebo znak rovný všeobecné adrese, přejde se do stavu naadresovaný a přijme datový rámec. Není-li rámec úspěšně přijat (shodný XorSum a dostatek paměti), je ignorován a v případě **uL_ARQ** nebo **uL_AAP** je vysláno **uL_NAK**. V opačném případě je rámec uložen do paměti. Je-li přijato **uL_ARQ**, je rámec potvrzen **uL_NAK**. Není-li požadováno provedení služby, přejde se do stavu čekání na rámec s nastaveným příznakem naadresovaný. V případě **uL_AAP** je rámec potvrzen až po rozpoznání služby. Po přijetí **uL_ARQ** nebo **uL_AAP** je rámec okamžitě předán příslušné službě. Ta provede požadovanou akci a může čekat na další jí určený datový rámec nebo vyslat rámec. Pro přijetí dalšího rámce ve stavu naadresovaný jsou rozšířeny rozpoznávané adresy o adresu **uL_Beg**. Do počátečního stavu čekání na rámec bez příznaku naadresovaný se zařízení vrátí po příjmu znaku s nastavenými bity D8 a D7. Pro některé služby může být počátek rámce s adresou **uL_Beg** povinný.

2.3.4 Arbitrace přístupu k médiu

Toto je jedna z nejsložitějších částí každého komunikačního protokolu, který připouští připojení více rovnocenných zařízení a nemá pevně určeného mastera.

Nejjednodušší na počet přenesených znaků je náhodný přístup s příposlechem nosné a detekcí kolizí. Tato metoda vyžaduje schopnost rozpoznat kolizi. Zároveň při vysokém zatížení sítě vede k snížení kapacity.

Proto jsou výhodnější postupy, kdy je dopředu rozhodnuto, které zařízení má právo vysílat. Toho lze docílit předáváním oprávnění vysílat mezi zařízeními. K tomu je nutné vysílat speciální rámce s předáním řízení. Při ztrátě tohoto rámce v důsledku chyby přenosu nebo výpadku zařízení, je nutné vygenerovat nový. Zároveň je pro každé zařízení nutné znát svého následníka v logickém kruhu předávání řízení.

Další možností je definování přípravné fáze před získáním oprávnění vysílat (arbitrace), která zaručí, že toto oprávnění získá pouze jedno zařízení. Tato fáze spotřebuje sice určitou část kapacity sběrnice, ale nevyžaduje rozpoznávání kolizí ani předávání oprávnění vysílat. Tato možnost byla vybrána pro protokol uLan. Zároveň byla doplněna o částečné rotování priority mezi zařízeními. Toho je docíleno určením minimálního času, po který se nesmí zařízení začít připojovat, v závislosti na adrese posledního předcházejícího zařízení s oprávněním vysílat.

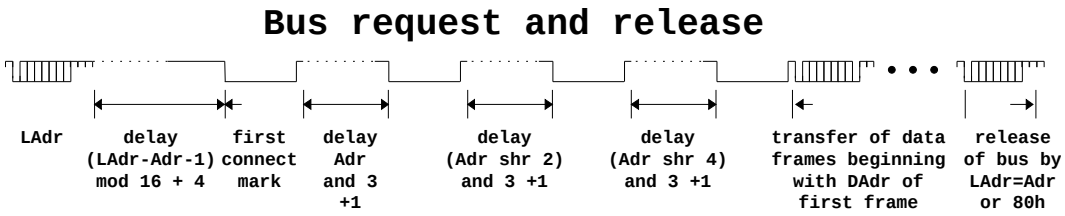
Každé zařízení, které má být schopno samostatného vysílání zprávy, musí trvale sledovat stav obsazenosti sběrnice. Zařízení musí přijímat všechny znaky s nastaveným D8. Je-li nastaven bit D7, je sběrnice považována za uvolněnou, v opačném případě za obsazenou. Podle toho je nastaven příznak neobsazenosti sběrnice **uLF_NB**. Protože arbitrační fáze je založena na vysílání znaků 000h bez nastaveného bitu D8, je nutné, aby zařízení schopné vysílání zprávy, měla po dobu nastavení příznaku uLF_NB povolen i příjem znaků s nulovým D8. Po příjmu jakéhokoliv znaku se příznak **uLF_NB** znuluje a není již nutné přijímat znaky s nulovým D8. Protože může dojít k výpadku zařízení v době, kdy obsadilo sběrnici, je nutné definovat způsob, jak obnovit práci sběrnice. Všechna zařízení mohou periodicky testovat aktivitu sběrnice funkcí **uL_STROKE**. Tato funkce nesmí být znovu volána dříve, než po době odpovídající vyslání 40 znaků. Maximální doba mezi voláními není definovaná a pouze ovlivňuje maximální dobu, po kterou nemusí být dané zařízení schopné obsadit sběrnici. Funkce **uL_STROKE** sleduje, jestli byl mezi jejími voláními přijat alespoň jeden znak. V případě neaktivity sběrnice nastaví příznak **uLF_NB** a při dalším volání zaktivuje vysílací systém zařízení. Protože fáze arbitrace obsahuje čekání po dobu přenosu minimálně 4 znaků s povoleným příjmem všech znaků, nedojde ani v případě nastavení **uLF_NB** v důsledku nepřijímání znaků s nulovým D8 během komunikace mezi jinými zařízeními ke kolizi. Během komunikace není dovolena

prodleva delší, než přenos 3 znaků (viz popis datového rámce) a tím je zaručeno opětné vynulování **uLF_NB**. Tento postup má výhodu, že komunikace nevyžaduje alokaci čítače prodlevy. **uL_STROKE** může být volána z libovolného časového přerušení nebo z libovolné periodicky volané funkce.

Vlastní proces připojení začíná v okamžiku detekce uvolnění sběrnice (změna **uLF_NB** z 0 do 1) nebo po vložení zprávy do výstupní fronty v době, kdy je nastaven **uLF_NB**. Nejdříve je z adresy posledního sběrnici uvolňujícího zařízení LAdr a vlastní adresy Adr vypočítána doba, po kterou testuje klid na sběrnici, podle vzorce 2.1.

$$(LAdr - Adr - 1) mod 16 + 4 \tag{2.1}$$

Doba se měří v době přenosu jednoho znaku. Není-li známa LAdr nebo při chybě na sběrnici musí tato doba odpovídat minimálně přenosu 20 znaků. Je-li přijat libovolný znak nebo detekována nízká úroveň na sběrnici, je sběrnice prohlášena za obsazenou a zařízení musí čekat na uvolnění sběrnice.



Obrázek 2.3: Obsazení a uvolnění sběrnice

Tento postup zajišťuje rotaci priority mezi maximálně 16 zařízeními. K následné kolizi by mohlo dojít mezi zařízeními s rozdílem adresy dělitelným 16 nebo při současném zapnutí dvou zařízení. Konečné jednoznačné rozhodnutí o právu vysílat je provedeno vysíláním kombinace znaků 000h a prázdných intervalů. Tato kombinace je pro každé zařízení dána jeho vlastní adresou. Prázdný znak odpovídá odpojení výstupního budiče. V této době je udržováno napětí sběrnice ve stavu odpovídajícímu logické jedničce zakončovacími odpory, které mohou být v každém zařízení. Pokud je v době vysílání prázdného znaku zařízením detekována nízká úroveň, znamená to, že došlo ke kolizi snahy více

zařízení o připojení a zařízení, které vysílalo prázdný znak nesmí v připojování pokračovat a musí sběrnici považovat za obsazenou. Na začátku vysílaní kombinace znaků je vyslán nulový znak. Pak je vyslán 1 až 4 prázdné znaky podle nejnižších dvou bitů vlastní adresy. Poté je vyslán nulový znak. Vysílání prodlevy a nulového znaku se opakuje ještě dvakrát pro další bity vlastní adresy. Tento postup zaručuje jednoznačné přidělení sběrnice jednomu ze 64 zařízení. Vysílání nulového znaku odděleného prodlevami umožňuje vzájemnou synchronizaci mezi zařízeními, která mají delší počáteční část arbitrační kombinace shodnou i při nenulové době odezvy na přerušení.

2.3.5 Systém rozpoznávání jednotek a dynamického přidělení adres

Aby bylo možné zjistit, která zařízení jsou připojeny k síti uLan, je zavedena služba zjištění typu zařízení **uLCo_SID** (send identification). Po přijetí rámce s hodnotou Com rovnou **uLCo_SID** a zakončením **uL_PRQ** nebo **uL_AAP** zařízení vyšle rámec s textovým řetězcem zakončeným znakem nula.

".mt ulad21 v 0.71 .uP 51x .oi .dy"

Například AD převodník popisuje výše uvedený řetězec. V řetězci jsou jednotlivé části uvozeny znakem "." a písmenným kódem. Poté může následovat slovní popis. Funkci zařízení popisuje ".mt" (sekce module type), v tomto případě "ulad21". Typ procesoru ".uP" je v tomto případě "51x" (rodina I8051 s vnější pamětí RAM). Příznak ".dy" možnost dynamického přidělení adresy a příznak ".oi" ukazuje, že zařízení vlastní objektové rozhraní.

Systém dynamického přidělení adresy umožňuje jednoznačnou identifikaci každého připojeného zařízení podle jeho 32 bitového výrobního čísla. Zařízení může být kdykoliv připojeno do sítě a nemusí mít nastavenu vlastní adresu. O přidělení adres se stará server dynamických adres, který zároveň monitoruje činnost připojených zařízení. S využitím

servisních protokolů je možné do jedné sítě zapojit i více těchto serverů. Pouze jeden z nich je však zvolen za hlavní a pouze ten smí publikovat do sítě svoji adresu a přidělovat adresy zařízením. Pouze při výpadku může začít s přidělováním adres jeho náhradník.

Činnost sítě s jedním serverem je založena na cyklickém zjišťování stavu všech zařízení. V první fázi je vyslána zpráva s jedním rámcem s Com rovným **uLCo_GST** (get status) se všeobecnou adresou a podslužbou 0 zakončeným **uL_PRQ**. Poté jsou vysílány zprávy jednotlivým zařízením. První rámec zprávy s Com rovným **uLCo_GST** obsahující určení podslužby 10h až 1Fh (pro rozlišení typu zjišťovaných údajů - základní údaje jsou 10h) a může obsahovat předpokládané výrobní číslo zařízení. Rámec je zakončen **uL_PRQ** nebo **uL_AAP**. Zařízení po přijetí této zprávy zkontroluje, jestli odpovídá jeho výrobní číslo číslu ve zprávě. V případě rozdílu předpokládá, že se jeho adresa kryje s adresou jiného zařízení, a proto vynuluje svoji adresu a přejde do stavu hledání nové volné adresy. Je-li výrobní číslo v pořádku vyšle rámec s požadovanými údaji. Tím je server informován o funkčnosti zařízení a i o jeho základním stavu. Zařízení přejde do stavu hledání nové adresy také tehdy, nepřijme-li po dobu více než tří cyklů dotazů rámec s dotazem na jeho stav. To že není dotazován a na síti je aktivní server dynamického přidělování adres pozná podle toho, že přijme třikrát rámec **uLCo_GST** s podslužbou 0 a ani jednou za tuto dobu není dotazován na svůj stav.

Zařízení ve stavu hledání nové adresy periodicky po určité době vysílá na server, od kterého přijme zprávu **uLCo_GST** s podslužbou 0, požadavek na zařazení do jeho tabulky adres. Tento požadavek je uložen v rámci s Com = **uLCo_NCS** (network controll services), s podslužbou **uLNCS_ARQ** (address request), svým výrobním číslem a zakončením **uL_END**. Server zjistí, jestli již dané zařízení nemá v tabulce připojených zařízení. Je-li to potřeba nalezne pro zařízení novou volnou adresu a zařadí ho do tabulky připojených zařízení. Nakonec oznámí zařízení zprávou s rámcem s Com = **uLCo_NCS**, podslužbou **uLNCS_ASE** (address set) a výrobním číslem zařízení jeho novou adresu. Pokud bylo nutno vytvořit novou položku v tabulce připojených zařízení, je standardním serverem vygenerován požadavek **uLCo_SID** pro zjištění typu zařízení. Po zjištění typu zařízení jsou informace o nově připojeném zařízení dodány systému modelů zařízení. Tento systém se snaží spojit položku z tabulky připojených zařízení s již existujícím

modelem zařízení. Není-li model nalezen, je podle typu zařízení ".mt" vytvořen model nový.

Periodicky zjišťovaný stav je též standardně předáván do systému modelů zařízení. Dojde-li víckrát po sobě k chybě při čtení stavu, považuje se zařízení za vypnuté a je z tabulky připojených zařízení vyřazeno. O této skutečnosti je informován i model zařízení. Ten vyhlásí chybu komunikace s přístrojem.

Existují-li záložní servery dynamického přidělení adres, mohou využívat pro tvorbu tabulky připojených zařízení zpráv **uLCo_NCS** s podslužbou **uLNCS_ASE**, které jsou vysílány s všeobecnou adresou.

Kapitola 3

Implementace protokolu

Ovladač je implementován z relativně nezávislých úrovní a subsystémů. Jednotlivé součásti ovladače jsou:

- Obecná podpora alokace, iteraci, plnění, čtení a manipulace se zprávami skládajícími se z jednoho nebo i více rámců. Tato část zahrnuje i seznam volných bloků ***free_blk***, frontu odchozích zpráv připravených ke zpracování ***prep_bll***, aktuálně zpracovávané zprávy *work_bll*, frontu zpracovaných zpráv připravených pro distribuci k operátorům ***proc_bll*** a seznam zpráv čekajících na provedení zpracování ze strany operátorů ***opan_bll***.

- Obecná podpora vytváření a rušení instancí driveru a volby podpory chip driverů podle přiřazeného hardware.

- Interface driveru pro přístup z uživatelského prostoru systémů s ochranou paměti s implementacemi pro Linux, Windows NT a Windows WDM driver model, případně pro integraci do firmware malých vestavěných zařízení bez operačního systému.

- Implementace obecného stavového automatu (UFSM) pro ty případy, kdy je pro realizaci protokolu využité rozhraní UART, které zpracovává data na úrovni jednotlivých znaků, ať již s frontou FIFO nebo bez ní. Tato vrstva není využita, pokud je možné celé rámce nebo zprávy přenést do vlastního hardware vykonávajícího protokolem předepsané vysílání a příjem dat (do této kategorie spadá připojení převodníku uLan-USB k počítači PC).

- Vlastní chipové drivery implementují funkce ***ul_call_fnc*** ***fnc_recch*** (funkce příjmu znaku), ***fnc_sndch*** (funkce vyslání znaku), ***fnc_wait*** (čekání na příjem nebo po určitý čas), ***fnc_connect*** (připojení na médium), ***fnc_finishtx*** (čekání na konec vysílání, uvolnění sběrnice), ***fnc_pool*** (test na příznak přerušení) v případě využití centrálního UFSM . Pro případ kdy jsou do cílového HW přenášeny celé zprávy nebo je použit servisní thread se tyto funkce nepoužívají a o požadavku na zpracování dat z výstupní fronty je chipový driver notifikován funkcí ***fnc_stroke***.

Vlastní průběh komunikace:

Klient/operátor otevře vlastní driver voláním open. Pro ukládání dat se používá předem alokované paměti rozdělené na bloky konstantní délky. Tyto bloky jsou na počátku vloženy na jednosměrně zřetězený seznam ***free_blk***. Během přípravy zprávy založené voláním UL_NEWMSG a založení hlavičky prvního rámce jsou zapisovaná data ukládaná do alokovaných bloků zřetězených za touto hlavičkou. K hlavičce rámce lze do zprávy přidat další rámec voláním UL_TAILMSG. Zpráva je po uvolnění UL_FREEMSG přesunuta na frontu ke zpracování ***prep_bll*** a pokud není aktivní, je aktivován chipový driver startem UFSM nebo funkcí ***fnc_stroke***. Pokud přijde na řadu zpracování dané zprávy, ještě před přijmutím dat bude zpráva přesunuta na frontu ***work_bll***. Po dokončení celé přijímací sekvence, bude zpráva zařazena do fronty ***proc_bll***. Před provedením filtrace a distribuce ze strany klientů/operátorů je zpráva přesunuta do fronty ***opan_bll***. Operátor testuje svojí privátní přijímací frontu UL_INEPOLL, přijme první rámec zprávy UL_ACCEPTMSG, případně její další rámce UL_ACTAILMSG, vyčte data a nakonec uvolní zprávu voláním UL_FREEMSG. Registrace typů zpráv o které má operátor zájem se provádí voláním UL_ADDFILT.

3.1 Stavový automat

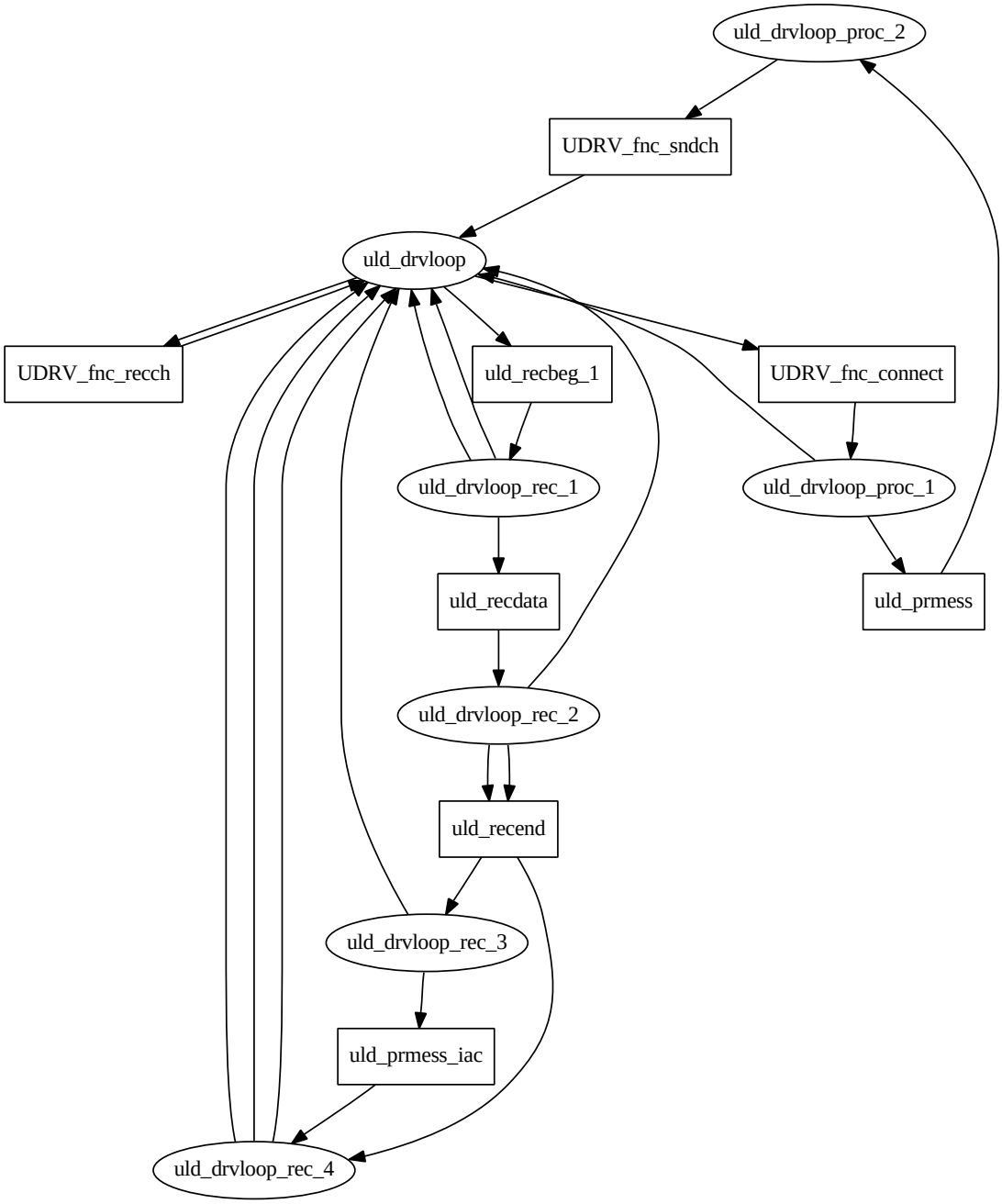
Spojovací protokol je naprogramován jako konečný stavový automat. Funkce tohoto automatu (UFSM) bude podrobně popsána v následujících odstavcích. Pro podporu schopnosti porozumět jednotlivým přechodům a voláním nižších úrovní automatu budou stavy, přechody a volání pod-úrovní vyobrazeny pomocí schémat stavových automatů. V těchto schématech představují ovály a osmihrany stavy automatu (v kódu reprezentované funkcemi), které budou podrobněji poslány a obdélníky znázorňují volání nižších úrovní automatu. Šedě vybarvené ovály a osmiúhelníky znázorňují vstupní body nižších úrovní automatu. Osmihrany obsahují možnost návratu do vyšší úrovně automatu. Šipkami jsou znázorněny přechody mezi jednotlivými stavy automatu a volání podprogramů.

Nejvyšší úroveň stavového automatu ovladače je schematicky znázorněna na obrázku 3.1. Ze schématu je patrné, že centrální uzel tvoří funkce ***uld_drvloop***. Tato funkce slouží jako jakýsi rozcestník. Rozděluje možné směrování operací do tří směrů a to:

- Příjem zpráv
- Odesílání zpráv
- Odposlech sběrnice

3.1.1 Odposlech sběrnice

Není-li požadována jiná akce (odeslání/přijímání zpráv), volá se *funkční odkaz* ***UDRV_fnc_recch***. Tento odkaz vyvolá funkci danou architekturou chipu, který je obsluhován danou instancí driveru (do driveru lze zakompilovat a používat podpory všech čipů naráz, WDM i Linux driver je kompilovaný takto). Ve všech případech je úkolem této funkce přijmout znak ze sběrnice a zapsat jej do znakového bufferu ovladače. Funkce pro různé architektury budou popsány později. Po zachycení znaku se přechází zpět na centrální uzel (***uld_drvloop***).



Obrázek 3.1: Schéma nejvyšší úrovně automatu

3.1.2 Příjem zpráv

Pokud byl odposlechnut znak (nachází se ve znakovém bufferu ovladače), který je roven adrese zařízení a má nastavený bit **D8**, popřípadě se jedná o broadcastovou adresu (0)

nebo je-li zapnut promiskuitní režim ovladače, volá se podprogram příjmu začátku zprávy (funkce ***uld_recbeg_1***, popis se nachází níže) a poté se přechází do stavu ***uld_drvloop_rec_1***.

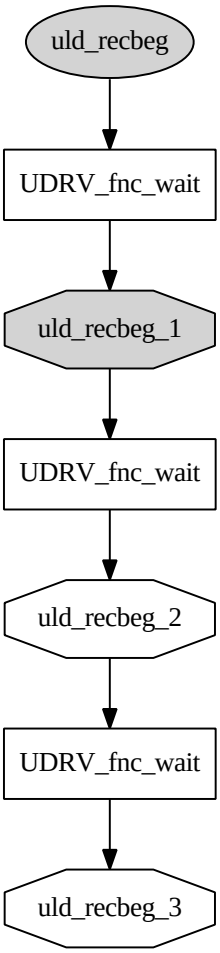
Ve funkci ***uld_drvloop_rec_1*** se vytvoří paměťový blok typu *ul_mem_blk* (zpráva) a naplní se hlavička tohoto bloku. Vytvořená zpráva se přenese do pracovního úložiště ovladače a jako reakce na vypršení doby nečinnosti (timeout) se nastaví ***uld_drvloop_rec_error***. Dále ze zavolá podprocedura přijímající data obsažená ve zprávě ***uld_recdata*** (popis lze nalézt níže) a pokračuje se stavem ***uld_drvloop_rec_2***.

Ve funkci ***uld_drvloop_rec_2*** se nachází rozcestník, který v případě, že jsou povoleny procedury okamžité reakce přechází do stavu ***uld_drvloop_rec_3***, v opačném případě do stavu ***uld_drvloop_rec_4***. V každém případě se však před skokem do výše zmíněných stavů volá podprogram příjmu konce rámce ***uld_recend*** (zakončovací znaky atd.).

Funkce ***uld_drvloop_rec_3*** pouze kontroluje zda nedošlo během přijímání k chybě a volá podprogram okamžitých reakcí ***uld_prmess_iac***. Poté přechází do stavu ***uld_drvloop_rec_4***. V této funkci je nastavena timeouová funkce ovladače zpět na prázdný ukazatel a vygeneruje se známka zprávy, která slouží jako jedinečný identifikátor při pozdějším rozřazování zpráv na vyšších úrovních protokolu. Nakonec se přijatá zpráva přesune do vstupní fronty ovladače a naplánuje se rozřazení přijatých zpráv. Pokračuje se přechodem do centrálního stavu (***uld_drvloop***).

3.1.2.1 Příjem začátku rámce

Příjem začátku rámce je ilustrován schématem 3.2.



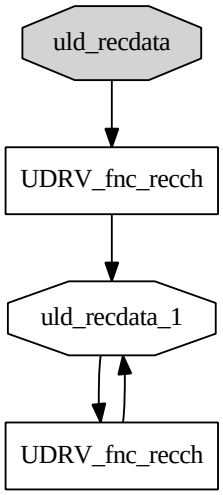
Obrázek 3.2: Schéma příjmu začátku rámce

V tomto podprogramu se přijímají znaky voláním funkce ***UDRV_fnc_wait***. Tato funkce se liší od ***UDRV_fnc_recch*** nutností přijmutí znaku do určitého časového okamžiku, pokud se takto nestane, generuje se timeout a přechází se na příslušnou chybovou proceduru. Je-li vstupním bodem řetězce ***uld_recbeg*** zavolá se funkce ***fnc_wait*** a přechází se do stavu ***uld_recbeg_1***. Zde se opět zkontroluje, zda má být toto zařízení skutečně příjemcem zprávy podle vyslané adresy příjemce. Tato adresa se uloží v příslušném lokálním registru a v příznakovém slově ovladače se nastaví devátý bit na hodnotu log. 1. Pokud je zapnut promiskuitní mód ovladače nastaví se i šestnáctý bit v tomto slově. Opět se volá ***fnc_wait*** a přechází se do stavu ***uld_recbeg_2***.

V této funkci se uloží **sadr** přijímané zprávy a čeká se na další znak. V další funkci (***uld_recbeg_3***) se uloží **cmd** zprávy. Následuje návrat o úroveň výš.

3.1.2.2 Příjem dat

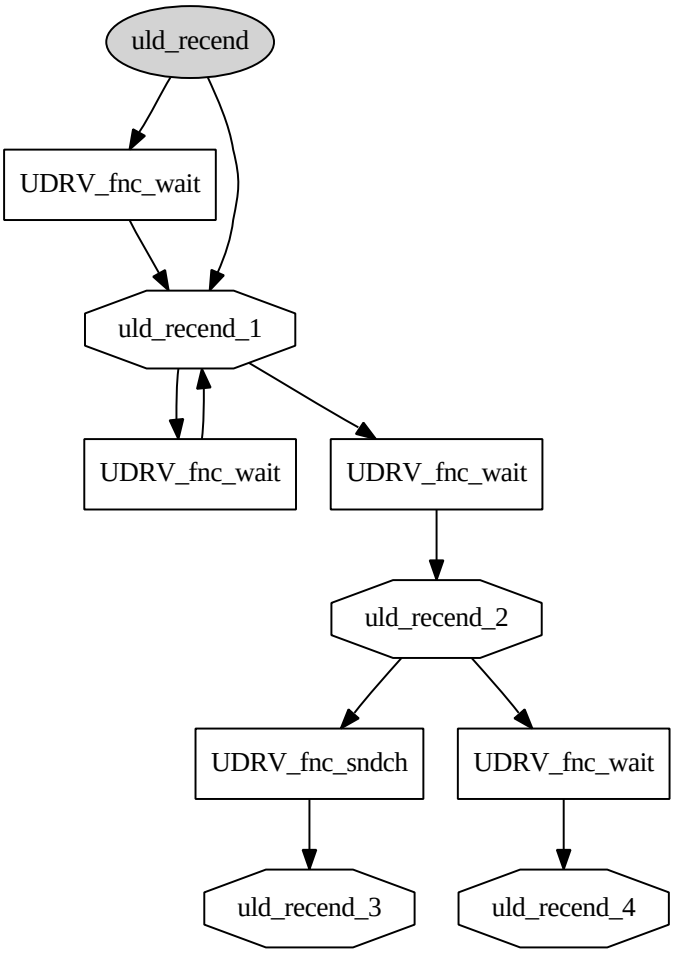
V této proceduře se sekvenčně načítají data a pomocí funkcí dataiterátorů zapisují do předem vytvořené struktury typu ***ul_mem_blk***. Stavové schéma 3.3 tento proces vyobrazuje.



Obrázek 3.3: Schéma příjmu dat

3.1.2.3 Příjem konce rámce

Graf na obrázku 3.4 znázorňuje přijímání znaků ve stavech ***uld_recend*** a ***uld_recend_1***. Dokud se ve znakovém bufferu ovladače nenalézá znak s nastaveným devátým bitem (ukončovací znak), neprovádí se žádná akce. Podle hodnoty tohoto znaku (**UL_ARQ**, **UL_PRQ** atd.) se nastaví příznak spojení a také příznak v přijímaném rámci a přechází se do dalšího stavu (***uld_recend_2***). Je-li vyžadováno potvrzení zprávy a není aktivován promiskuitní mód odešle se potvrzovací znak a přechází se do zakončovacího stavu vysílání (***uld_recend_3***). Je-li aktivován promiskuitní mód, zkontroluje se, zda byla zpráva potvrzena (někým jiným). Pokud se tak nestalo bude přijatá zpráva označena jako chybná (příznakem ***UL_BFL_FAIL***).



Obrázek 3.4: Schéma příjmu konce rámce

3.1.3 Odesílání zpráv

Je-li v centrálním uzlu zjištěna přítomnost zprávy ve výstupní frontě ovladače (***prep_bll***), provede se připojení na sběrnici pomocí funkce ***UDRV_fnc_connect***(odkazu na funkci - veškeré funkce začínající UDRV jsou pouze odkazy na speciální funkce příslušné k danému hardwaru, budou popsány později) a přejde se do stavu ***uld_drvloop_proc_1***.

Funkce ***uld_drvloop_proc_1*** především přesune první zprávu z výstupní fronty do vnitřního úložiště ovladače pro aktuálně zpracovávanou zprávu ať už vysílanou nebo přijímanou. Dále se zde definuje timeout funkce, kterou v tomto případě bude ***uld_drvloop_proc_error***. Timeout funkce bude volána dojde-li při pokusu o vyslání

zprávy k přetečení přednastaveného časovače. Následně se zavolá vysílací podřetězec se vstupním bodem ***uld_prmess***, který se postará o samotné odeslání zprávy a jeho funkce bude vysvětlena v další podsekci.

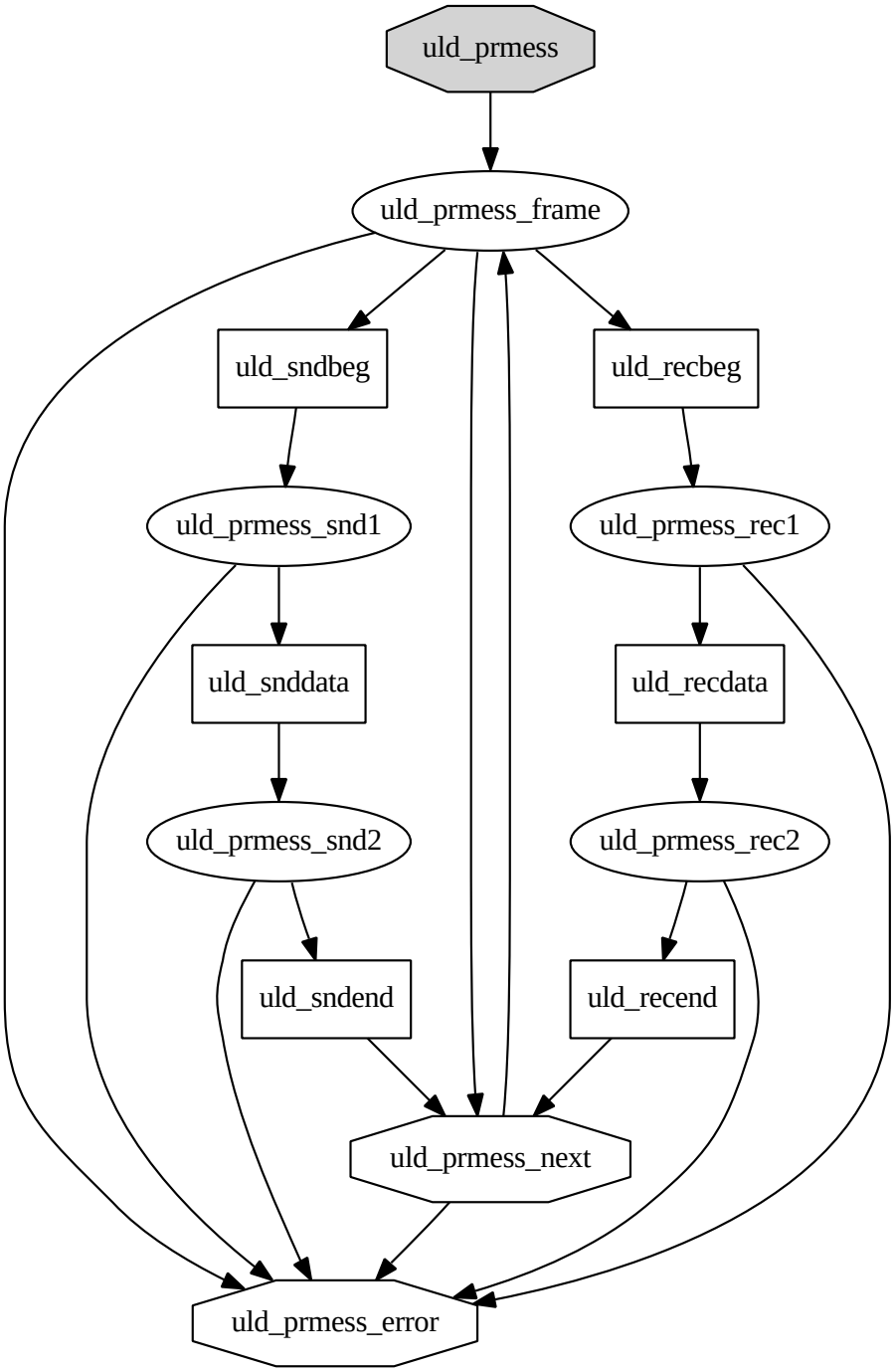
Dojde-li během vysílání k timeoutu nebo nějaké jiné chybě, provede funkce ***uld_drvloop_proc_error*** opětovný pokus o vyslání zprávy, pokud není u zprávy nastaven příznak **UL_BFL_NORE** nebo nebyl překročen maximální počet vysílacích pokusů (***UL_RETRY_CNT***). V opačném případě nastaví u zprávy příznak chyby (**UL_BFL_FAIL**) a zařadí ji do vstupní fronty ovladače, čímž informuje vyšší vrstvy protokolu o neschopnosti odeslání zprávy příjemci.

Pokud proběhne přenos zprávy v pořádku, přechází se do bodu ***uld_drvloop_proc_2***. V této funkci dojde k nastavení timoutové funkce zpět na hodnotu prázdného ukazatele a do znakového bufferu ovladače se přemístí hodnota adresy vysílajícího zařízení s nastavenými bity **D8** a **D7**, po jejímž odeslání funkcí ***UDRV_fnc_sndch*** se sběrnice uvolní. Nakonec bude zpráva testována na příznak **UL_BFL_M2IN**, je-li výsledek testu kladný přesune se zpráva do vstupní fronty ovladače. Tímto způsobem jsou vyšší vrstvy protokolu informovány o doručení odeslané zprávy je-li to vyžadováno.

3.1.3.1 Vysílací podřetězec

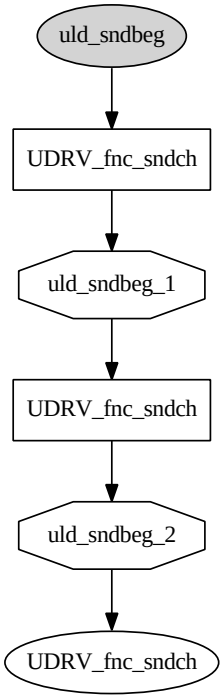
Tento podřetězec je schematicky znázorněn na obrázku 3.5.

V tomto podprogramu se nalézá důležitý stav ***uld_prmess_frame***, ve kterém je obsažen rozhodovací mechanismus zajišťující přechod do dalšího stavu podle příznaku (**flag**) v hlavičce zprávy.



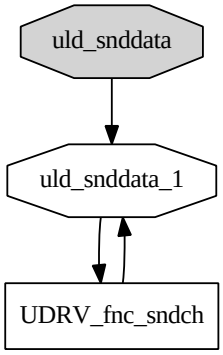
Obrázek 3.5: Schéma vysílacího podřetězce

Jednou možností je vyslání zprávy na sběrnici má-li nastaven příznak **UL_BFL_SND**. V takovém případě je naplněna hlavička rámce a odeslána voláním podprogramu ***uld_sndbeg*** schéma tohoto podprogramu nalezneme na obrázku 3.6.



Obrázek 3.6: Schéma podprogramu vysílání hlavičky rámce

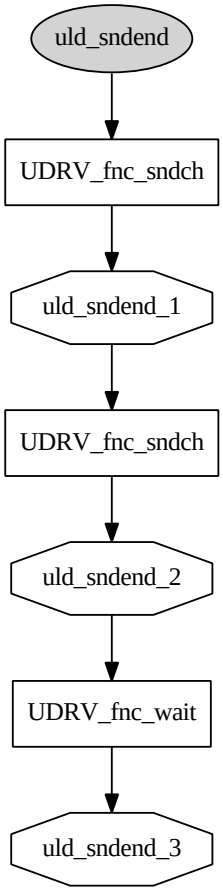
Následně se přechází do stavu ***uld_prmess_snd1***. Tato funkce volá podprogram (***uld_snddata***) vysílání dat obsažených ve zprávě. Jednoduché schéma 3.7 samo názorně vysvětluje funkci podprogramu.



Obrázek 3.7: Schéma podprogramu vysílání dat

Poté se přechází do stavu ***uld_prmess_snd2***, kde je volána procedura vyslání zakončovacích znaků ***uld_sndend***. Tuto proceduru lze popsat stavovým schématem 3.8. V

prvním kroku této procedury je vyslán zakončovací znak (nastavený devátý bit). Jaký ze zakončovacích znaků bude vyslán, je dáno příznakovým slovem spojení. Dále se vysílá kontrolní znak exkluzivního součtu. Následně je kontrolováno, zda má být zpráva potvrzena, pokud tak být nemá, procedura končí. V opačném případě se vyčká na příchod potvrzovacího znaku.



Obrázek 3.8: Schéma vysílání zakončovacích znaků

Bylo-li vysílání zdárně ukončeno, přechází se do stavu ***uld_prmess_next***, kde je rámec testován na příznak **UL_BFL_TAIL**, který určuje zda má zpráva více rámců. Pokud ano přejde se do stavu ***uld_prmess_frame*** a je odeslán následný rámec. Pokud zpráva není vícerámcová, podprogram končí. Do stavu ***uld_prmess_next*** se je možné dostat i ze stavu ***uld_prmess_frame***, pokud vysílaná zpráva nemá nastavený žádný ze sledovaných příznaků.

Druhá možnost přechodu ze stavu ***uld_prmess_frame*** vede do stavů ***uld_prmess_rec1*** a ***uld_prmess_rec1*** přes podprogramy přijímání hlavičky, dat a konce rámce, pokud je u rámce zprávy nastaven příznak **UL_BFL_REC**. K tomuto ději dochází, pokud má zpráva více rámců a následné rámce jsou určeny k okamžitému příjmu odpovědi. Jako příklad je možné uvést identifikaci v síti uLan **uLCo_SID**, kdy jsou u prvního rámce nastaveny příznaky **UL_BFL_TAIL** a **UL_BFL_SND** a je tedy vyslán na sběrnici. Následný rámec ve zprávě **uLCo_SID**, který má však nastaven příznak **UL_BFL_REC** a provede se jeho nahrazení (doplnění) rámcem přijatým jako odpověď od identifikovaného zařízení, stále zůstává součástí zprávy. Zpráva bude tedy ve vyšších vrstvách zpracována jako vyslaná, pokud se vrátí do vstupní fronty (musí mít nastaven **UL_BFL_M2IN**). Tento způsob komunikace je využíván kvůli rychlosti odpovědi, vynechá se zdlouhavá arbitrace přístupu na sběrnici. Sběrnice je přidělena po celou dobu jednomu masteru, který vyzývá příjemce k odpovědi, aniž by předtím musel získat pomocí arbitrace oprávnění vysílat. Na zprávy tohoto typu (**UL_PRQ**) je na přijímací straně reagováno voláním procedur okamžité reakce.

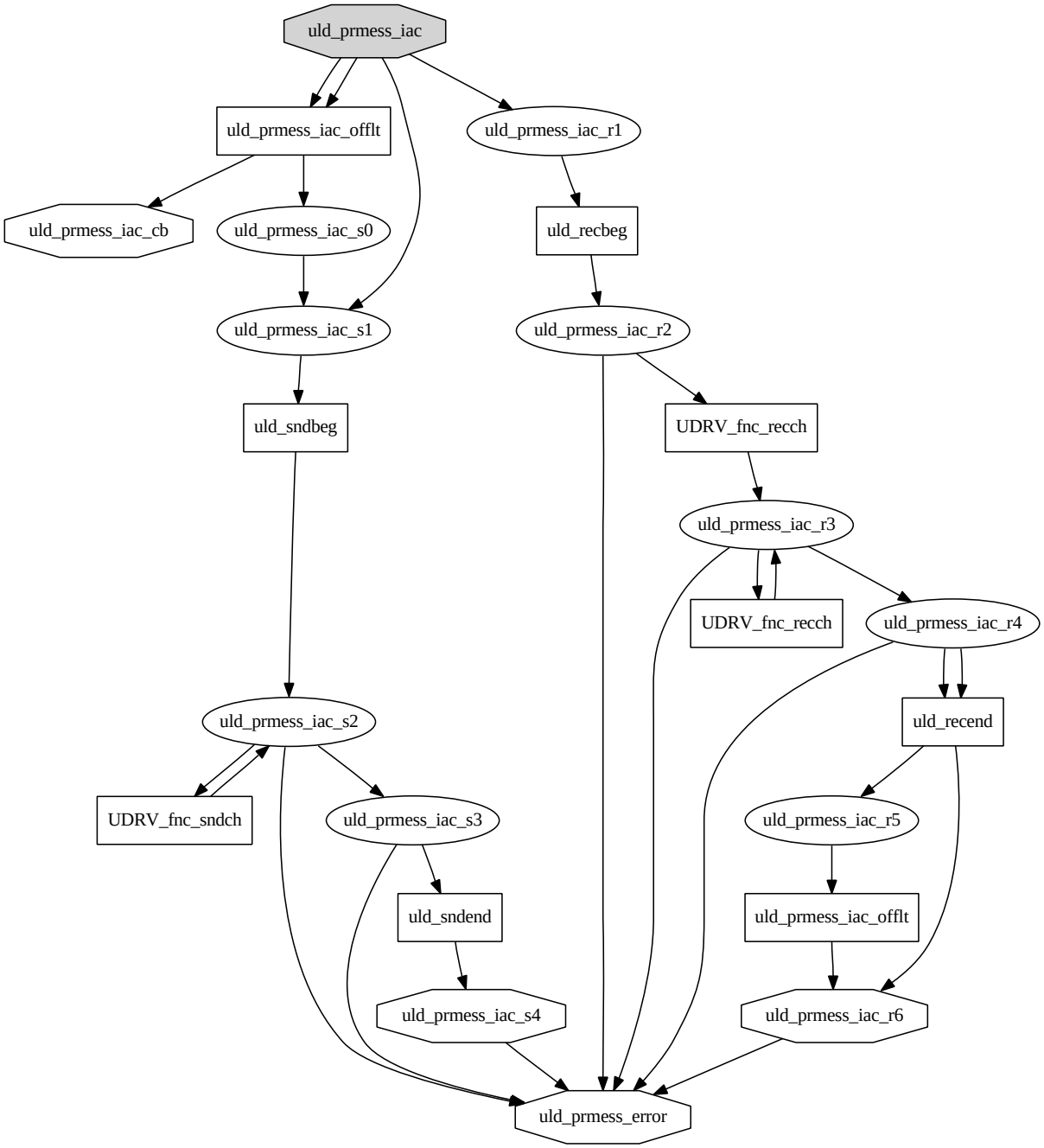
Vyskytne-li se v jakémkoliv stavu tohoto řetězce chyba volá se chybová funkce popsaná výše.

3.1.4 Procedury okamžité reakce

Tento podprogram je volán v přijímací části automatu, má-li zpráva nastaven příznak **UL_PRQ**. Jak je možné sledovat na schématu 3.9, vstupním bodem tohoto subautomatu je funkce ***uld_prmess_iac***. Tato funkce obsahuje přepínač akcí k okamžitému vykonání. Jako klíč k rozřazení je použit **com** (command) přijatého rámce. Podle tohoto klíče bude vybrána ze struktur okamžitých reakcí zaregistrovaných u ovladače struktura obsahující data potřebná pro vykonání správné reakce. Tato struktura taktéž obsahuje registr, ve kterém je hodnota určující k jaké akci přísluší. Možné akce jsou následující:

- volání callback funkce
- vyslání přednaplněného bufferu na sběrnici
- vyslání dat generovaných uživatelem definovanou funkcí

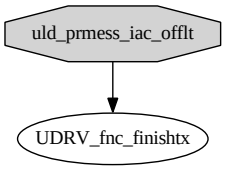
- příjem dalšího rámce



Obrázek 3.9: Schéma procedur okamžité reakce

3.1.4.1 Callback funkce

Pokud je v příznakovém slově struktury okamžitých reakcí nastaven příznak **UL_IAC_BFL_CB_OFFLT**, bude volána funkce ***uld_prmess_iac_offlt***, která zajistí odpojení od sběrnice voláním procedury chipového ovladače ***UDRV_fnc_finishtx*** (obrázek 3.10). Celý proce je zakončen voláním funkce ***uld_prmess_iac_cb***.



Obrázek 3.10: Callback funkce

3.1.4.2 Vysílání na sběrnici

Jsou dvě varianty vysílání odpovědi na sběrnici. V prvním případě se podle příznaku ve struktuře okamžitých reakcí začne s vysíláním již naplněného bufferu (**UL_IAC_OP_SNDBUFF**) voláním funkce ***uld_prmess_iac_s1***. Pokud však byl příznak roven **UL_IAC_OP_SND**, dojde k odpojení od sběrnice a volání funkce ***uld_prmess_iac_s0***, ve které je umožněno buffer nejprve naplnit. V obou případech dojde k přechodu do stavu ***uld_prmess_iac_s1***. Tento stav obsahuje naplnění hlavičky rámce daty a to takovými, že příkaz bude mít hodnotu 0x7f a místo **dadr** příjemce bude vyslán znak **UL_BEG**. Následuje vyslání hlavičky na sběrnici a přejde se do stavu ***uld_prmess_iac_s2***. V této funkci budou vyslána na sběrnici všechna data z bufferu. Nakonec je vyslán konec rámce s zakončovacím znakem **UL_END**.

3.1.4.3 Přijímání rámců

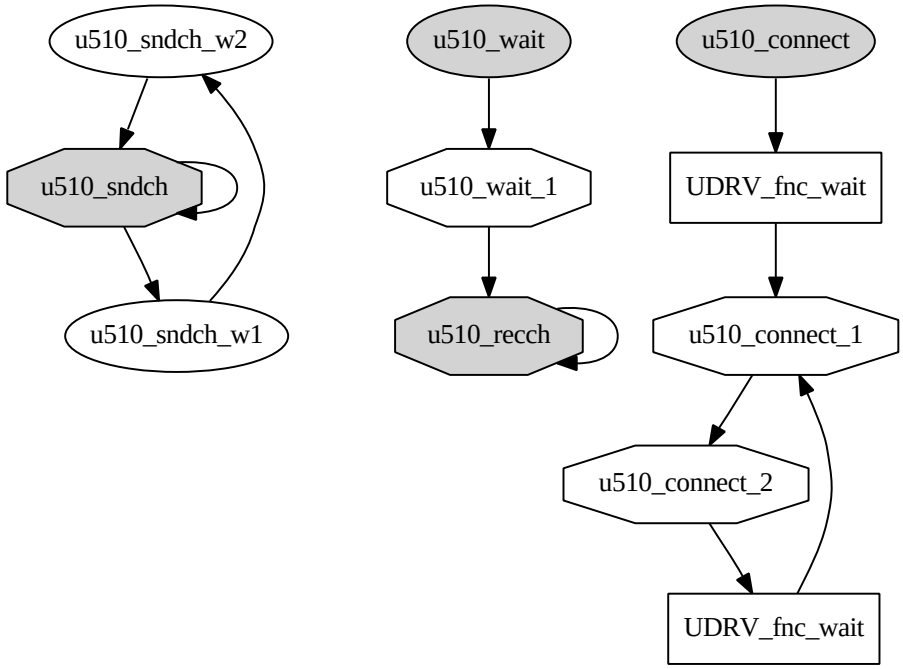
Tato procedura je velmi podobná podprogramu přijímání klasických rámců s jedním rozdílem a to, že ve stavu ***uld_prmess_iac_r3*** nejsou data přijímána do struktury rámce, ale do aktuálně vybrané struktury okamžitých reakcí. Pokud je z různých důvodů aktivován přechod do stavu ***uld_prmess_error***, je volána příslušná chybová funkce.

3.1.5 Chipové ovladače

Aby byla umožněna práce ovladače na různých architekturách chipů, používá stavový automat abstraktní funkce pro připojení na sběrnici (včetně arbitrace přístupu na medium), čtení a vysílání znaku. Tyto funkce jsou nastaveny rutinou pro inicializaci podpory konkrétního hardware. Zatím jsou připraveny ovladače pro UARTY (univerzální asynchronní přijímače/vysílače) typu 82510, 16450 a OX16950. V následující podsekci bude popsána implementace zmíněných funkcí pro UART 82510.

3.1.5.1 Ovladač pro UART 82510

Funkce tohoto ovladače je schématicky znázorněna na obrázku 3.11.



Obrázek 3.11: Schéma ovladače pro UART 82510

Procedura příjmu znaku je pro tuto variantu obvodu implementována funkcí ***u510_recch***. V této funkci se na začátku zjistí, zda bylo dokončeno odesílání předchozího znaku, pokud odeslán ještě nebyl, čeká se na další přerušení od obvodu, které je kontrolováno funkcí ***u510_pool***. Bylo-li vysílání dokončeno, přepne se budič sběrnice do stavu příjmu. Bude-li přijat znak, uloží se do znakového bufferu ovladače, v opačném případě se čeká na další

přerušení.

Příjem znaku má ve své kompetenci také procedura ***u510_wait***, která čeká určitou přednastavenou dobu a jakmile tato doba odezní nebo je aktivováno přerušení od příjmu znaku, přejde se do stavu ***u510_wait_1***. V tomto stavu je zkontrolováno zda přišel nový znak, pokud ano, volá se funkce ***u510_recch***, v opačném případě dojde k návratu z procedury z příznakem timeoutu.

Vyslání znaku na sběrnici je zajištěno podprogramem se vstupním bodem ***u510_sndch***. Není-li budič sběrnice v režimu vysílání, přejde se přes smyčku tvořenou ***u510_sndch_w1*** a ***u510_sndch_w2*** zpět do stavu ***u510_sndch***, přičemž mezi w1 a w2 je vyčkáno na příchod přerušení a po jeho příchodu, je budič přepnut do režimu vysílání. Byl-li budič v režimu vysílání ale nebyl dokončen předchozí transfer, vyčká se na další přerušení od obvodu. Není-li vstupní registr prázdný, čeká se tak dlouho, dokud nebude vyprázdněn. Nakonec je znak odeslán na sběrnici.

Podprogram zajišťující připojení na sběrnici (arbitrace o médium) má vstupní bod ***u510_connect***. Zde dojde k výpočtu pevného času čekání daného vysílací adresou zařízení a po jeho uplynutí se přejde do stavu ***u510_connect_1***. Pokud během čekání přišel nějaký znak, znamená to, že již někdo vysílá a ovladač nic na sběrnici posílat nebude. Pokud nic nepřišlo přepne se budič do režimu vysílání a pokusí o vyslání znaku s nastaveným dominantním stavem (logická úroveň 0), po přerušení se přejde do stavu ***u510_connect_2***. Zde se nachází kontrola správnosti připojení, nebyla-li již připojovací sekvence dokončena, vyčká se po dobu odpovídající času vysílání 1 až 4 znaků podle nejnižších dvou bitů vlastní adresy a dochází k dalšímu cyklu. Tento postup se provádí třikrát, tedy pro nižších 6 bitů vlastní adresy.

3.2 Vyšší vrstvy protokolu

Mezi stavovým automatem a aplikacemi se nachází vrstva, která zajišťuje větší univerzálnost protokolu. Jednou z výhod je možnost běhu více aplikací nad jediným ovladačem,

další výhodou je aplikovatelnost ovladače na více operačních systémech popřípadě na zařízeních bez systému. Hlavním představitelem této vrstvy je klient a služby s ním spojené.

3.2.1 Klienti

Aplikace využívají služeb protokolu prostřednictvím objektů, které nazýváme klienty driveru. Klient je abstraktní strukturou mimo jiné umožňující funkci driveru na více operačních systémech (windows, linux, bez systému) s jednotným rozhraním. Klient je při použití bez systému struktura typu **ul_opdata**. Nejdůležitější součásti klienta jsou následující:

- **udrv** - ukazatel na strukturu ovladače ke kterému klient přísluší
- **opnext** a **opprew** - reprezentují odkaz na předcházející a následující instanci klienta v listu klientů ovladače
- **message** - první rámec připravené zprávy
- **data** - ukazatel na data v prvním rámci zprávy
- **recchain** - seznam členů obsahujících přejaté zprávy a pomocné informace (především známku zprávy)
- **filtchain** - seznam členů obsahujících informace pro filtraci příchozích zpráv

3.2.2 Služby klientů

Aplikace volají funkce z univerzální knihovny s parametrem struktury klienta, kterážto obsahuje veškeré potřebné informace pro provedení operace. Nejdůležitější služby jsou:

3.2.2.1 Vytvoření klienta

Klient se vytváří voláním funkce ***ul_open***, jejímž důležitým vstupním parametrem je jméno zařízení. Podle jména zařízení bude rozhodnuto, k jakému ovladači bude klient

zaregistrován. Pod operačními systémy může pracovat i více instancí ovladače zároveň (běžně omezeno na čtyři) jako jiná vstupně/výstupní rozhraní počítače. Použitelná jména zařízení jsou uvedena v tabulce 3.1.

Jméno zařízení	platforma
COMx	Windows
/dev/ulan$x$	Linux
null	bez systému
subDevIdx$x$	bez systému s více zařízeními

Tabulka 3.1: Tabulka použitelných jmen zařízení

V tabulce 3.1 je za jménem zařízení uvedeno x, za které je nutné dosadit kladnou celou číslici. Tato číslice u windows a linuxu označuje pořadí komunikačního rozhraní . Pokud je jako jméno zařízení uveden prázdný ukazatel, bude zařízení nakonfigurováno pro práci v režimu bez systému s jednou instancí ovladače.

Nakonec se vytvořený klient zaregistruje do listu klientů příslušících k ovladači.

3.2.2.2 Zrušení klienta

Tuto operaci zajišťuje funkce ***ul_close***. Nejprve je odstraněna zpráva zpracovávaná klientem, existuje-li, dále odstraní svůj odkaz z listu klientů ovladače. Nakonec klient odstraní svou vlastní instanci z paměti.

3.2.2.3 Načítání a zápis dat

Funkce ***ul_read*** překopíruje data z aktuálně zpracovávaného rámce klinta do bufferu, který již bezprostředně využijí aplikace. Funkce ***ul_write*** pracuje na stejném principu s tím rozdílem, že směr toku dat je opačný.

3.2.2.4 Vytvoření zprávy

Funkce ***ul_newmsg*** vytvoří rámec zprávy podle vstupních parametrů, nastaví zprávě příznak **UL_BFL_SND** a nastaví tento rámec jako **message** klienta. Je-li třeba vytvořit vícerámcovou zprávu volá se po vytvoření prvního rámce funkce ***ulan_tailmsg***. Tato

funkce vytvoří další rámec podobným způsobem jako ***ul_newmsg*** ale zároveň nastaví rámec **message** jako svého předchůdce.

3.2.2.5 Uvolnění zprávy od klienta

Funkce ***ul_freemsg*** nejprve zjistí, zda má aktuálně zpracovávaná zpráva klienta již přidělenou známku (**stamp**). Známkou jsou označeny odeslané zprávy, jako jedinečným identifikátorem. Pokud zpráva má přidělenu známku, je pouze odstraněna z paměti. V opačném případě je známka vygenerována a přiřazena zpracovávané zprávě. Dále je-li u zprávy nastaven příznak ***UL_BFL_M2IN*** vytvoří se člen filtračního seznamu klienta (filtchainu) a tento člen bude mít nastaven stav **UL_OPST_ONCE**. Význam tohoto stavu bude vysvětlen později v sekci věnované klientům. Nakonec je zpráva předána do výstupní fronty ovladače a bude odeslána.

3.2.2.6 Vyzvednutí zpráv aplikací

Potřebuje-li aplikace nová data, zavolá nad svým klientem funkci ***ulan_acceptmsg***, která vyzvedne z **recchainu** klienta první člen a rámec jenž tento člen obsahuje, nastaví jako **message**, tedy první rámec zpracovávané zprávy klienta. Je-li zpráva více rámcová, tak pomocí funkce ***ulan_actailmsg*** získá aplikace atributy hlavičky a ukazatel na data dalšího rámce aktuálně zpracovávané zprávy.

3.2.2.7 Přidání filtračního členu do filtchainu klient

Aplikace určují, u které z přijímaných zpráv mají zájem pomocí nastavení permanentních členů ve filtchanech svých klientů. Přidávání filtračních členů realizuje funkce ***ul_addfilt***. Nejprve vytvoří člen podle vstupních parametrů, které vyznačují jaká může být hlavička přijímané zprávy (filtruje se podle **dard**, **sadr**, **com** a **stamp**). Následně je vyplněn stav tohoto členu, který určuje, jaký způsob filtrace bude použit a nakonec je člen přidán do filtchainu klienta.

3.2.2.8 Nastavování atributů ovladače

Knihovna obsahuje též funkce, které přímo nastavují proměnné ovladače. Jednou takovou funkcí je ***ul_setmyadr*** nastavující adresu, kterou bude zařízení identifikováno v síti uLan.

Dalším příkladem může být funkce ***ul_setpromode***, která zajišťuje aktivaci a deaktivaci promiskuitního módu ovladače, při kterém ovladač přijímá veškerou komunikaci na sběrnici. Funkce ***ul_setidstr*** vytvoří strukturu okamžitých reakcí pro identifikaci zařízení na sběrnici a nastaví její identifikační řetězec.

3.2.3 Předávání zpráv

Po zařazení zprávy do vstupní fronty ovladače nebo volání funkce ***ulan_inepoll*** je spuštěna procedura, jejíž hlavním úkolem je rozřadit zprávy klientům podle prvků v jejich filtrovacích seznamech (**filtchainech**), v případě, že není driver kompilován pro operační systém windows realizuje tuto proceduru funkce ***ulan_do_bh*** . Tato funkce spustí pro každou zprávu ve vstupní frontě ovladače podprogram rozřazování zpráv pro klienty realizovaný funkcí *ulan_proc_arrived*. Tato funkce obsahuje filtry, podle kterých bude zajištěno správné předání zprávy do přijímacích seznamů klientů. Filtr pro každého klienta zaregistrovaného k ovladači vyhledává ve filtračním seznamu člen splňující požadavek na shodnost **dadr**, **sadr**, **com** a **stamp**. Pokud má člen filtračního seznamu nějaký z těchto porovnávaných atributů nulový, bude podmínka shodnosti tohoto atributu přeskočena. Byl-li nalezen filtrační člen splňující výše zmíněné podmínky, bude další akce odvozena od hodnoty stavu tohoto členu.

Má-li člen stav **UL_OPST_ONCE**, přijatá zpráva projde filtrem pouze jednou a zařadí se do **recchainu** klienta, to zajistí následné odstranění tohoto členu z filtračního seznamu. Bude-li stav členu roven **UL_OPST_FILT**, bude zpráva s danými atributy vždy přidána do **recchainu** klienta obsahujícího tento člen ve **filtchainu**.

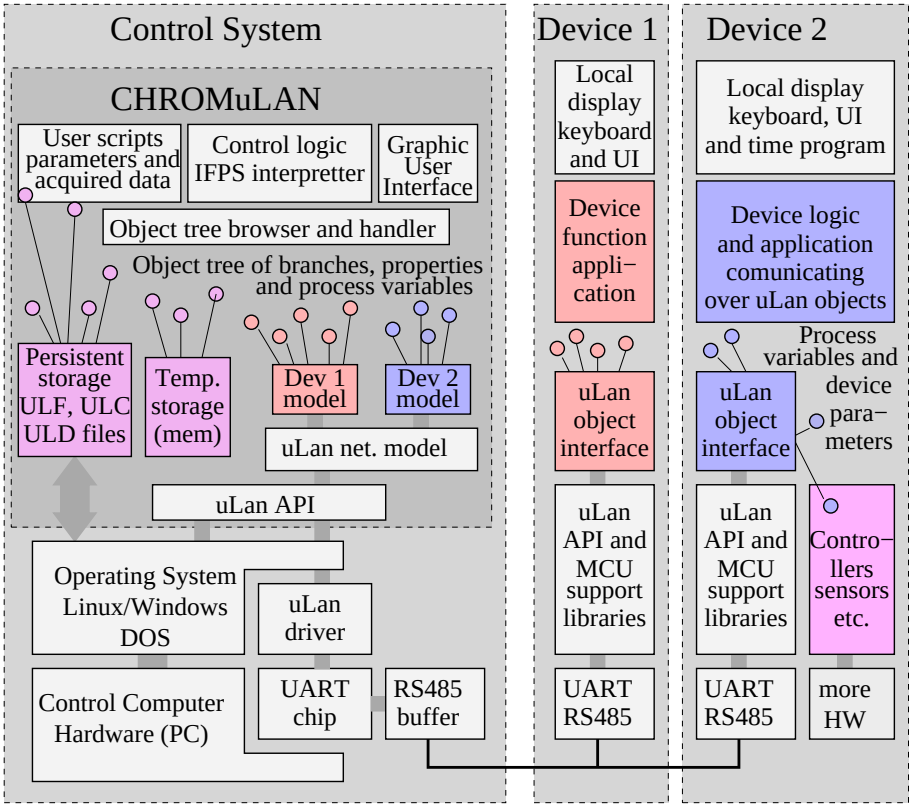
Kapitola 4

Objektové rozhraní a prohlížeč proměnných

Objektové rozhraní protokolu uLan bylo navrženo tak, aby bylo umožněno flexibilní nastavování/dotazování proměnných/objektů daného přístroje a pro zasílání příkazů. Zprávový formát je navržen tak, aby byl co nejkratší, ale ne na úkor obecnosti. Toto vyústilo v komplexnější profil přístroje. Objektové rozhraní protokolu uLan umožňuje uložit popis objektů přímo v přístroji a dává možnost automaticky vytvořit profil přístroje z popisu jeho objektů.

Každá aplikace implementující objektové rozhraní má definován slovník proměnných, které je možné pomocí dotazů číst nebo do nich zapisovat. Sériové číslo a adresa jednotky jsou příkladem proměnných, jež vlastní každá jednotka s objektovým rozhraním a umožňují dynamickou adresaci v síti uLan.

Objektové rozhraní bylo využito i při realizaci projektu CHROMuLAN, z dokumentace k němu vytvořené byl vyňat obrázek 4.1. Na tomto obrázku je možné pozorovat objekty (kruhy ocáskem) vystupující z bloků objektového rozhraní směrem k aplikacím, které objekty používají.

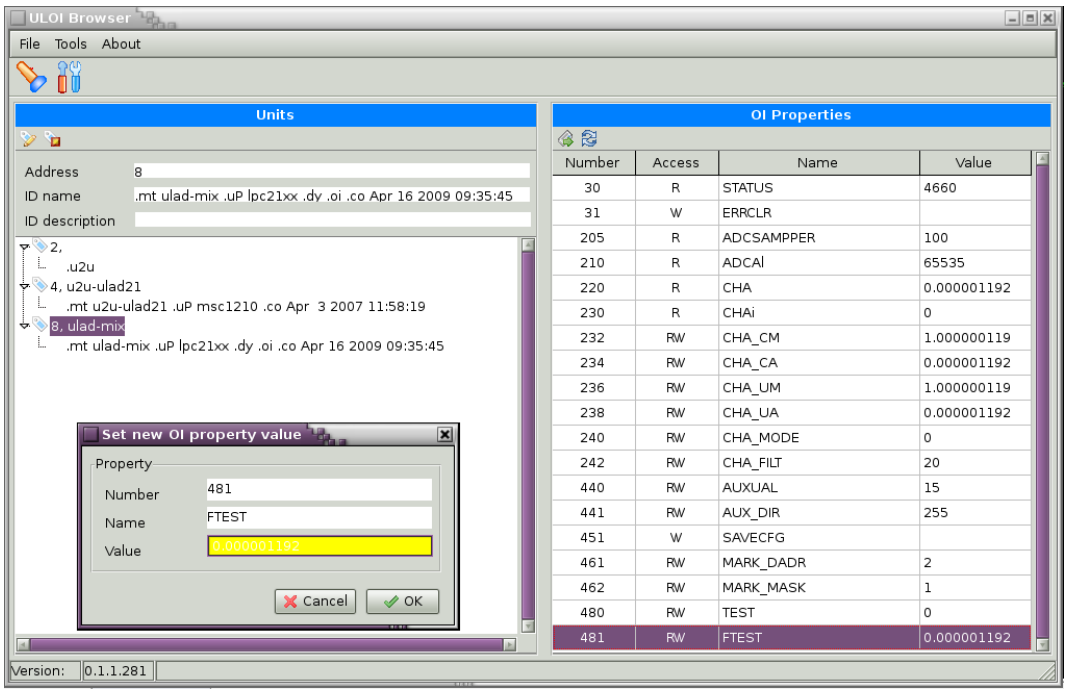


Obrázek 4.1: Blokové schéma z projektu CHROMuLAN s objektovým rozhraním

Objektové rozhraní implementují dvě knihovny. Jedna obsahuje funkce potřebné pro vysílání dotazů a zpracování odpovědí na straně mastera. Druhá obsahuje funkce umožňující na tyto dotazy odpovědět patřičným způsobem.

4.1 Prohlížeč proměnných

Pro prohlížení proměnných objektového rozhraní je možné na straně PC využít například utilitu *ul_dysn* nebo ***ul_lcscan***. Tyto utility pracují v textovém režimu a nejsou tedy příliš vhodné k vizualizaci proměnných daného přístroje, z tohoto důvodu byl vyvinut grafický prohlížeč proměnných (*uloi_browser*).



Obrázek 4.2: Screenshot prohlížeče proměnných

Prohlížeč proměnných (obrázek 4.2) automaticky po startu a kdykoliv je stisknut *refresh* identifikuje zařízení připojená ke sběrnici uLan a pokud tato zařízení implementují objektové rozhraní, načte slovníky jejich proměnných. Prohlížeč poskytuje možnost jednoduchého zadávání a vyčítání hodnot proměnných a možnost readresace jednotlivých uzlů v síti.

Kapitola 5

Převodník USB-uLan

Převodník uLan USB přijímá všechny zprávy pro něj určené a přeposílá je do počítače a naopak.

5.1 Universal serial bus - USB

Univerzální sériová sběrnice (USB - *Universal Serial Bus*) je od druhé poloviny 90.let minulého století novým standardem pro připojování periferních zařízení k počítači a přenos dat mezi nimi. Sběrnice se na trhu prosadila díky vlastnostem, které znamenaly pro uživatele jednoduché použití a pro výrobce levnou a rychlou implementaci do nových i již hotových zařízení. Mezi tyto vlastnosti patří především možnost připojovat zařízení za chodu počítače, napájení zařízení s menším odběrem přímo ze sběrnice, rychlost přenosu až do 12Mb/s, univerzálnost použití a možnost připojení více zařízení k jedné sběrnici.

Další rozšíření USB na trhu umožnila nová verze specifikace, která dovoluje přenos až do 480MB/s a tím i připojení datově náročnějších zařízení. Navíc s rozvojem mobilní digitální techniky a uvedením dodatku *On-The-Go* je možné propojovat některá zařízení přímo, bez potřeby počítače jako hostitele. Vzhledem k těmto vlastnostem rozhraní USB vytlačuje dosud používané starší paralelní a sériové rozhraní počítačů.

Výhodou USB je také standardizace základních tříd zařízení se stejným účelem, pro které jsou definovány přenosové protokoly. Podpora těchto tříd v operačních systémech pak představuje pro uživatele zjednodušení instalace nových zařízení, které patří do

podporované třídy a které tak nepotřebují zvláštní programové vybavení.

Rozšíření sběrnice USB a použití i v nekomerčních projektech je umožněno širokou součástkovou základnou vyráběných elektronických obvodů s implementovaným rozhraním USB.

Sběrnice USB je složena z několika typů prvků. Jsou to hostitel (*HOST*), původně vždy jako součást stolního počítače, rozbočovače (*HUB*) a koncová zařízení (*DEVICE*). Dále sběrnici tvoří propojení vždy mezi dvěma prvky typu *point-to-point*. Každý prvek se skládá z několika logických vrstev. Komunikace mezi hostitelem a koncovým zařízením probíhá skrz logické komunikační kanály.

Topologie sběrnice má stromovou strukturu. Hostitel tvoří centrální jednotku každé sběrnice a zároveň je spolu s kořenovým rozbočovačem první vrstvou sběrnice. V každé sběrnici USB může být pouze jeden hostitel, který řídí přidělování média metodou výzvy v logickém kruhu.

Stejně jako kořenový, tak i ostatní rozbočovače rozšiřují sběrnici o další přípojné body a zvyšují počet vrstev sběrnice. Většina rozbočovačů je součástí zařízení, které má v jednom fyzickém pouzdře kromě rozbočovače i koncové zařízení.

Koncové zařízení obsahuje několik logických bran (*endpointy*) s vlastní FIFO pamětí, přes které zařízení komunikuje hostitelem. Jednotlivé brány jsou sdruženy do rozhraní.

Pro některé negativní vlastnosti, mezi něž patří například topologie sběrnice, omezená maximální vzdálenost zařízení od počítače, nemožnost galvanického oddělení, se tato sběrnice nerozšířila do průmyslu, kde se i nadále používají vhodnější sběrnice založené na RS-485. Pro připojení takové sběrnice k počítači je zapotřebí převodník mezi těmito dvěma rozhraními. Takový převodník byl vyvinut i pro síť uLan.

5.2 Implementace převodníku

V této sekci bude podrobněji popsána funkce převodníku podle jeho softwarové implementace ve verzi 2. Na začátku programu se vytvoří dva klienti, jeden pro zápis a druhý pro čtení, se jménem zařízení rovným null, takže oba pracují nad jednou instancí driveru. Funkcí ***ul_setidstr*** zapíše do bufferu struktury okamžitých reakcí, příslušející k identifikačním zprávám, svůj identifikační řetězec a funkcí ***ul_setmyadr*** nastaví svou im-

plicitní adresu. Dále je přidán do filtračního seznamu přijímacího klienta člen povolující příjem všech zpráv určených pro převodník. Pokračuje se nastavením parametrů pro USB zařízení, vytvořením dvou endpointů a připojením na sběrnici USB.

Po provedení těchto inicializačních operací přechází program do nekonečné smyčky, ve které jsou vždy zkontrolovány události na USB a sběrnici uLan. Jsou-li na sběrnici USB připravena data, budou přečtena a vyslána na sběrnici uLan funkcí ***ul_dcnv_send***. V této funkci je vytvořena zpráva z dat poslaných přes USB, přičemž je nutné překódovat příznakové slovo zprávy a nastavit u něj příznak ***UL_BFL_M2IN***, díky jemuž bude počítač informován o odeslání zprávy do sítě uLan. Zpráva může být buď jednorámcová nebo vícerámcová, podle toho bude zpráva vytvořena funkcí ***ul_newmsg*** nebo ***ul_tailmsg***. Nakonec je zpráva odeslána voláním funkce ***ul_write***.

Je-li zařízení připraveno vysílat na sběrnici USB, zkontroluje se, zda nejsou připravena data pro příjem voláním ***funkce ul_dcnv_rec***, ve které jsou data přečtena funkcí ***ul_read***. Bylo-li přečteno nenulové množství dat, budou odeslána přes sběrnici USB do počítače. V opačném případě je zahájen příjem zpráv realizovaný funkcí ***ul_dcnv_rec_start***, kde je nejprve pomocí funkce ***ul_inepoll*** zkontrolováno, zda není seznam přijatých rámců přijímacího klienta prázdný a je tedy co přeposílat. Pokračuje se přijímáním prvního rámce zprávy ze seznamu přijatých rámců funkcí ***ul_acceptmsg***. Má-li zpráva více rámců bude volána v dalším cyklu smyčky funkce ***ul_actailmsg*** zajišťující připojení dalších rámců ze seznamu přijatých rámců k prvnímu rámci zprávy. U každého rámce je rozhodnuto, zda jde o zprávu přijatou nebo o zprávu odeslanou testováním na příznak **UL_BFL_REC** a zda nejde o zprávu chybovou. Nakonec jsou rámce přeposláno do počítače přes sběrnici USB.

Kapitola 6

Úprava protokolu - logické jednotky

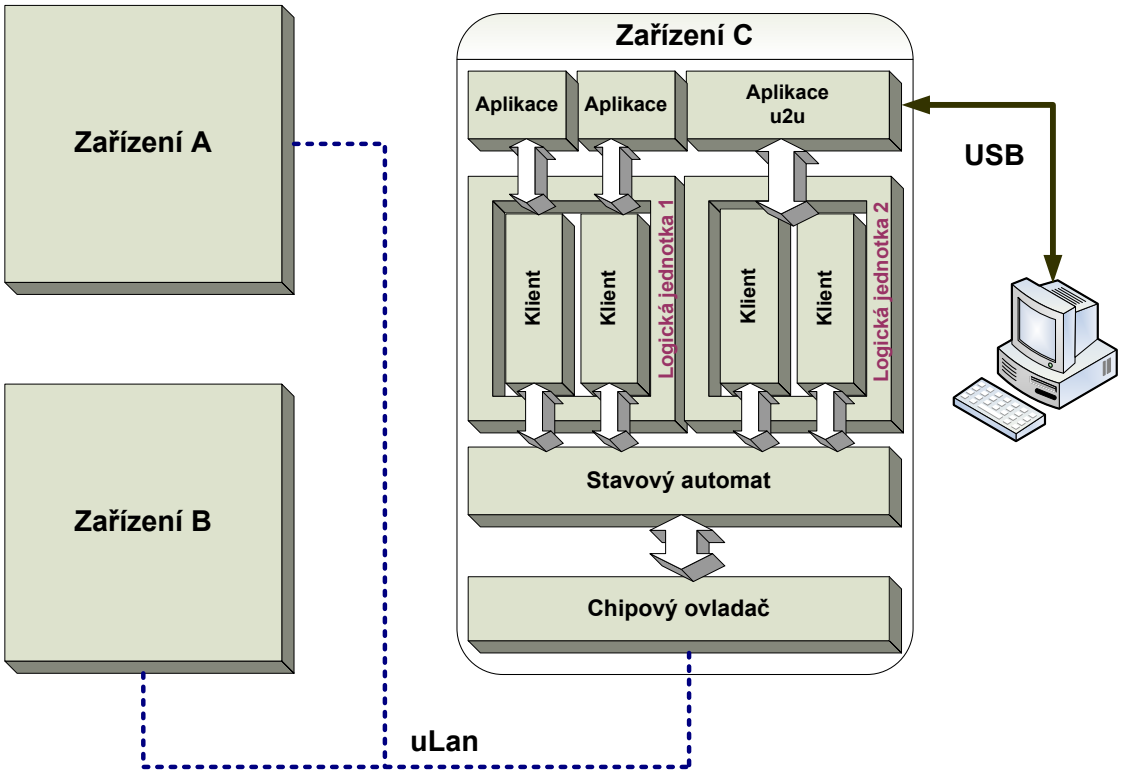
6.1 Motivace

Jelikož většinu větších aplikací není možné realizovat v zabudovaných zařízeních používajících sběrnici uLan, je nutné předávat data do výpočetně výkonnějších stanic (většinou osobní počítače). Takováto komunikace může být realizována v aplikačním softwaru používajícím služby protokolu uLan. V takovém případě však musí být každá aplikace vyžadující spolupráci s PC upravena a zařízení musí být fyzicky spojeno s počítačem (většinou sběrnicí USB). Další možností je použití v předchozí kapitole popsaného převodníku USB-uLan. Bude-li použit program převodníku, je jedno fyzické zařízení na sběrnici vyhrazeno čistě pro potřeby přenosu dat mezi sítí USB a uLan a to v případě využití univerzálních modulů, které obsahují i **A/D**, **D/A** převodníky nebo jiné součásti senzorových/akčních členů, není zcela optimální. Jedním z úkolů této práce je vytvoření programového vybavení umožňující práci zařízení, které bude zároveň realizovat převodník USB-uLan a jednotku sběru dat s **A/D** převodníkem.

6.2 Nástin řešení

Jako systémové řešení zadaného úkolu se jeví úprava implementace komunikačního protokolu uLan, tak aby umožňoval nad jediným ovladačem (stavovým automatem) funkci více logických jednotek majících svou vlastní identifikaci (adresu) na sběrnici. Tyto jed-

notky musí pracovat na sobě zcela nezávisle a veškerá komunikace mezi těmito zařízeními musí mít stejné vlastnosti jako kdyby komunikovala po síti uLan. Zároveň logické jednotky musejí být z pohledu ostatních zařízení operujících na sběrnici uLan identifikovatelná a použitelná jako plnohodnotná zařízení ležící na síti uLan. Celou situaci dobře ilustruje obrázek 6.1.

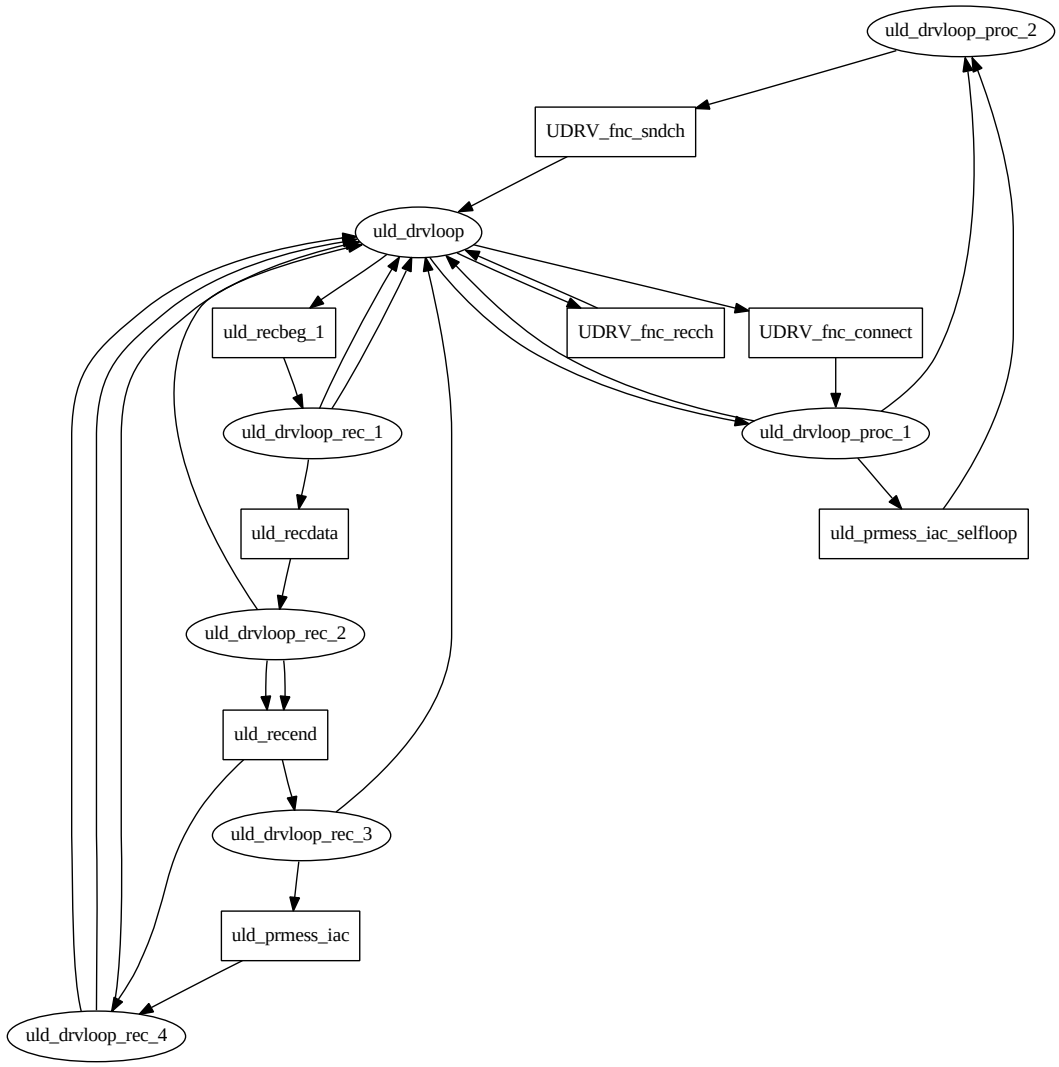


Obrázek 6.1: Blokové schéma zařízení se dvěma podzařízeními

6.3 Implementace změn ve stavovém automatu

První a nejdůležitější změnou ve struktuře ovladače (*ul_drv*) je záměna jednoduché skalární adresy zařízení za pole adres příslušných k logickým jednotkám. Počet možných jednotek je shora omezen hodnotou **UL_SUBDEVNUM_MAX**. Toto má za následek, že ovladač přijímá rámce pro každou jednotku (každou adresu z pole) a zařazuje je do vstupní fronty.

Dále bylo nutné přeskočit pokus o vysílání na sběrnici je-li odesílaná zpráva určena pro některou logickou jednotku příslušnou k témuž zařízení, které se chystá zprávu odeslat, z důvodu např. nemožného potvrzení zprávy. Nové schéma nejvyšší vrstvy stavového automatu je na obrázku 6.2. Přechody mezi stavy ***uld_drvloop***, ***uld_drvloop_proc_1*** a ***uld_drvloop_proc_2*** jsou podmíněny právě shodností **dadr** zprávy s jednou z adres z adresního pole ovladače. Pokud bylo dosaženo stavu ***uld_drvloop_proc_2*** právě tímto způsobem, je zpráva automaticky předána zpět do vstupní fronty ovladače.



Obrázek 6.2: Nové schéma nejvyšší úrovně automatu

Vyžaduje-li taková zpráva okamžitou reakci (příznak **UL_PRQ**), bude během přeskakování vysílacích procedur volána funkce ***uld_prmess_iac_selfloop***, která simuluje okamžitou

odpověď dané logické jednotky. Pokud je v této funkci vyžadována akce zaslání identifikačního řetězce, budou do rámce, který následuje po úvodním rámci zprávy s požadavkem, zapsána data přímo ze struktury okamžitých reakcí a aktivuje se přechod do stavu ***uld_drvloop_proc_2***. Další akce popsané v sekci Implementace protokolu nebyly zatím pro komunikaci mezi logickými jednotkami implementovány. Aby bylo možné rozlišit jaká struktura okamžitých reakcí patří k dané jednotce, obsahuje každá struktura okamžitých reakcí index jednotky (*subdevidx*), který též odpovídá indexu adresy v adresním poli ovladače. Nově je tedy správná struktura okamžitých reakcí vybrána nejen podle příkazu **com**, ale také podle indexu jednotky v adresním poli ovladače.

6.4 Implementace změn v při práci s klienty

V této sekci budou osvětleny mechanizmy příjmu, vysílání a předávání zpráv logickými jednotkami a jejich programová realizace. Logické jednotky jsou realizovány jedním nebo skupinou klientů s totožným indexem jednotky (*subdevidx*). Tento index určuje pod kterou adresou bude klient vysílat a které zprávy bude přejímat od ovladače. Index (*subdevidx*) je možné u klienta nastavit již při jeho vytváření voláním funkce ***ul_open*** s parametrem jména zařízení odpovídajícím řetězci *subDevIdx* a celočíselnou číslicí, která reprezentuje právě index jednotky (zápis je uveden v tabulce 3.1).

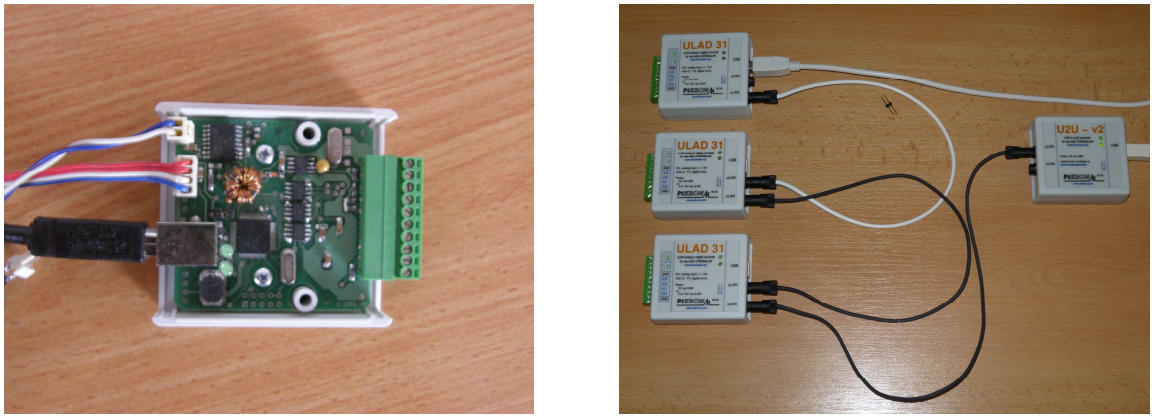
Chce-li klient vyslat zprávu, musí nejprve podle svého *subdevidx* získat adresu z adresního pole ovladače a tu použije jako **sadr** v hlavičce zprávy.

Procedura příjmu zpráv je komplikována možností přijímat zprávy od logických jednotek operujících na stejném fyzickém zařízení. V proceduře rozřazování zpráv klientům (***ulan_proc_arrived***) je nyní nutné kontrolovat, byla-li zpráva určena právě této jednotce. Pokud by se takto nestalo, byla by zpráva zpracovávána jako odeslaná (s příznakem proc=1) i u logické jednotky, která ji neodeslala. Je-li však zjištěno, že je zpráva určena pro tuto jednotku, nastaví se u členu přidaného do recchainu klienta stav rovný **UL_OPST_MESLOOP** a při volání funkce ***ul_acceptmsg*** získá zpráva příznak proc=0.

Kapitola 7

Zařízení obsahující převodník USB-uLan a přesný A/D převodník

Máme-li již upravený komunikační protokol, je nyní možné sloučit dvě existující aplikace jako dvě logické jednotky v rámci jediného zařízení. Sloučení bylo dosaženo na hardwaru (obrázek 7.1) s mikrokontrolérem LPC2148. Jako druhá součást zařízení, kromě převodníku UBS-uLan, byla zvolena aplikace ULAD31. Tato aplikace obsahuje funkce umožňující vyčítání hodnot z přesného A/D převodníku a na úrovni protokolu uLan umožňuje přístup k těmto hodnotám pomocí objektového rozhraní sběrnice. Aby byla umožněna správná funkce jednotky, jsou aplikace přepínány pomocí časového multiplexu s koeficientem 1:1.



Obrázek 7.1: Fotografie hardwaru použitého k realizaci projektu

Kapitola 8

Závěr

Hlavní náplní této práce byl návrh a implementace rozšíření protokolu uLan o schopnost provozu více logických jednotek v rámci jednoho fyzického zařízení.

Do úvodní části práce bylo připraveno obecnější pojednání o tomto zprávovém protokolu včetně drobného pohledu do jeho historie. Pro následující část práce byly získány informace o struktuře packetů, rámců a zpráv, které tento protokol využívá a letmý popis služeb, jež protokol poskytuje.

Dále byl pro potřeby této práce připraven detailní popis implementace protokolu především pro využití na platformách pro vestavěná zařízení bez operačního systému. Největší důraz byl v této části kladen na popis koncepce a implementace stavového automatu. Pro doplnění důležitých funkcí protokolu byl připojen základní popis objektového rozhraní, umožňující snadnou práci s proměnnými přístrojů, a prohlížeče objektů.

Hlavní motivací pro vznik rozšíření protokolu byla nutnost vyřazení jednoho universálního zařízení pro činnost převodníku USB-uLan, jehož popis funkce tato práce také obsahuje.

Největší podíl praktické části této práce tedy zaujímá úprava existujícího universálního ovladače, tak aby obsahoval schopnost realizace více jednotek v rámci jednoho zařízení uLan. Dále byl pro potřeby ladění a demonstrace zmíněné schopnosti ovladače vyvinut aplikační software sdružující aplikaci převodníku USB-uLan a A/D převodníku ULAD31 jako dvě logické jednotky v jediném zařízení uLan.

Výsledek této práce je nyní možné využít například v případě potřebujeme-li připojit nějaké jedno existující zařízení uLan k počítači pomocí sběrnice USB, aniž by bylo nutné výrazně měnit aplikační kód tohoto zařízení nebo přidávat další fyzické zařízení schopné komunikace po sběrnici uLan.

Literatura

[1] *Dokumentace na serveru.*
URL <http://ulan.sourceforge.net>

[2] *ULAD 10 - User Guide PiKRON s.r.o.* 2004.
URL <http://www.pikron.com/pages/products/support/manuals.html>

[3] Bartosiński, R.: *Implementace USB interface pro počítačové periferie.* Diplomová práce, České vysoké učení technické, Fakulta elektrotechnická, 2003.

[4] Píša, P.: Diplomová práce, České vysoké učení technické, Fakulta elektrotechnická, 1996.

[5] Píša, P.; Smolík, P.: *uLan Communication Protocol for Laboratory Instruments, Home Automation and Field Applications.* 15th International Conference on Process Control 05 [CD-ROM], Bratislava: Slovak University of Technology, 2005, ISBN 80-227-2235-9, 2005.

[6] Vysoký, O.; Píša, P.: Microprocessor Control System for Gas Engine Ignitions. Technická zpráva, CTU FEL, K335, 1995.