ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY

DIPLOMOVÁ PRÁCE

Implementace D.A.L.I. protokolu pro STM8 mikrokontroler

Praha 2009 Bc. Martin Knotek

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 20.1.2009 Martin Kroset .....

podpis

Poděkování

Tímto bych rád poděkoval především firmě STMicroelectronics za poskytnutí prostředků a podpory při tvorbě této diplomové práce. Dále bych rád poděkoval vedoucímu diplomové práce Ing. Janu Fischerovi, CSc. za jeho ochotu a vstřícný přístup při řešení problémů a v neposlední řadě svému nejbližšímu okolí za podporu při studiu.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Martin Knotek**

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

Název tématu: **Implementace D.A.L.I. protokolu pro STM8 mikrokontroler**

Pokyny pro vypracování:

1. Prostudujte strukturu D.A.L.I. komunikačního protokolu i jeho nestandardních modifikaci a odhadněte minimální výpočetní prostředky a paměťové nároky pro programovou implementaci D.A.L.I protokolu.
2. Navrhněte a realizujte programovou knihovnu D.A.L.I. komunikačního protokolu pro „slave" zařízení řízené pomocí mikrořadiče STM8, dále knihovnu D.A.L.I. komunikačního protokolu pro „master" zařízení, obojí včetně fyzické vrstvy D.A.L.I protokolu emulovaného programem.
3. Vytvořte krátký uživatelský manuál v anglickém jazyce (5-10 stran) popisující tyto programové knihovny. Přizpůsobte PC demonstrační program (založený na existující ST7DALI vývojové desce) pro prácí s STM8.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Jan Fischer, CSc.

Platnost zadání: do konce zimního semestru 2009/2010

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 3. 9. 2008

ANOTACE:

Úkolem diplomové práce bylo vytvořit programovou knihovnu D.A.L.I. komunikačního protokolu pro 8-bitový mikrokontroler STM8 společnosti STMicroelectronics.

D.A.L.I. protokol je používán především k automatizaci a řízení osvětlení ve velkých kancelářských budovách a objektech. Jeho význam roste s každým rokem, a proto se jej mnoho výrobních společností snaží zahrnout do svého portfolia

K vývoji programové knihovny byl použit jazyk C, vývojové prostředí ST Visual Develop firmy STMicroelectronics a vývojová deska určená k testování aplikací pro STM8S2xx mikrokontroler.

První část knihovny je určena pro *master* zařízení, které má za úkol adresovat a ovládat *slave* zařízení příkazy definovanými v normě IEC Standard 60929, Annex E & G. Jeho další funkcí je komunikovat a interpretovat příkazy vysílané demonstračním programem na PC.

Druhá část knihovny je učena pro *slave* zařízení, které ovládá vlastní světelné zařízení v závislosti na přijatých příkazech z *master* zařízení.

Jako poslední část práce byl vytvořen demonstrační program pro PC, který komunikuje s *master* zařízením po USB sběrnici nebo sériovém portu a předává mu příkazy zadané uživatelem přes grafické rozhraní.

THE ANNOTATION:

The main goal of this master thesis was to create a software library for D.A.L.I. communication protocol for STMicroelectronics 8-bit microcontroller STM8.

The D.A.L.I protocol is mainly used for automation and light management in office buildings. Its significance is growing every year; therefore, many automation companies are willing to include support of this protocol into their portfolio.

Development of this application was accomplished in C programming language, with support of STMicroelectronics Visual Develop IDE and tested on STM8S2xx evaluation board.

First part of the software library concerns with master device software. The main task of the *master* unit is to address and control *slave* devices with aid of commands defined in the IEC Standard 60929, Annex E & G norm. Another task for the *master* device is to communicate and interpret commands sent by the demonstration program running on PC.

Second part of the software was created for the *slave* unit that controls the light device depending on commands received from the *master* unit.

The last part of my work was to create a demonstration program for the PC. This application communicates with the *master* device over the USB bus and passes the commands placed by the user in the graphical interface.

Obsah

1 ÚVOD 1

2 DALI (DIGITAL ADDRESSABLE LIGHTING INTERFACE) 3

2.1 HISTORIE 3
2.1.1 SDRUŽENÍ AG_DALI 3
2.1.2 DRUHY NOREM 3
2.2 PRINCIPY DALI SÍTÍ 4
2.2.1 MOŽNOSTI KOMUNIKACE MEZI ZAŘÍZENÍMI 4
2.2.1.1 Master komunikace 4
2.2.1.2 Slave komunikace 5
2.2.2 ZPŮSOBY ADRESACE SLAVE ZAŘÍZENÍ 5
2.2.3 NASTAVENÍ SCÉN 5
2.3 ELEKTROINSTALACE V DALI SÍTI 5
2.3.1 KLASICKÁ ELEKTROINSTALACE BEZ DALI 6
2.3.2 ELEKTROINSTALACE ZA POUŽITÍ DALI 7
2.4 VARIANTY ZAPOJENÍ DALI SÍTĚ 8
2.4.1 ZÁKLADNÍ DALI SMYČKA 8
2.4.2 ROZŠÍŘENÁ DALI SMYČKA 9
2.5 VÝHODY DALI INSTALACÍ 10
2.6 NEVÝHODY DALI INSTALACÍ 10
2.7 MINIMÁLNÍ VÝPOČETNÍ A PAMĚŤOVÉ NÁROKY 10

3 DALI PROTOKOL 12

3.1 MASTER/SLAVE TOPOLOGIE 12
3.1.1 DRUHY MASTER ZAŘÍZENÍ 12
3.2 FYZICKÁ VRSTVA 13
3.2.1 BI-FÁZOVÉ KÓDOVÁNÍ 13
3.2.2 RYCHLOST PŘENOSU 14
3.2.3 ELEKTRICKÁ SPECIFIKACE 14
3.3 KOMUNIKACE MEZI ZAŘÍZENÍMI 15
3.3.1 KOMUNIKAČNÍ RÁMCE 15
3.3.1.1 Master → Slave forward rámec 15
3.3.1.2 Master → Master forward rámec 15
3.3.1.3 Backward rámec 16
3.3.2 DOBY USTÁLENÍ 16
3.3.3 PŘÍSTUPOVÉ PRIORITY 16
3.3.4 TRANSAKCE 18
3.3.5 DETEKCE KOLIZÍ 18
3.3.5.1 Pravidla pro detekci kolizí 18
3.3.5.2 Body detekce kolizí 18
3.4 ADRESAČNÍ POSTUPY 19
3.4.1 NÁHODNÉ PŘIŘAZENÍ ADRES 19
3.4.2 PŘIŘAZENÍ FYZICKÝM ZVOLENÍM 20
3.4.3 PŘEDNASTAVENÍ 20
3.5 ZPŮSOB ADRESACE SLAVE PRVKŮ 20
3.5.1 NASTAVENÍ Y BITU 21
3.5.2 NASTAVENÍ S BITU 21

3.5.3 BROADCAST PŘÍKAZ 21
3.5.4 ADRESACE SPECIÁLNÍCH PŘÍKAZŮ 21
3.5.5 PŘÍKLADY 2 BYTOVÝCH FORWARD ZPRÁV 21
3.6 TYPY PŘÍKAZŮ VE FORWARD RÁMCI 22
3.6.1 PŘÍKAZY PŘÍMO OVLIVŇUJÍCÍ ÚROVEŇ OSVĚTLENÍ 22
3.6.2 PŘÍKAZY NEPŘÍMO OVLIVŇUJÍCÍ ÚROVEŇ OSVĚTLENÍ 23
3.6.3 KONFIGURAČNÍ PŘÍKAZY 23
3.6.4 DOTAZOVÉ (QUERY) PŘÍKAZY 23
3.6.5 SPECIÁLNÍ PŘÍKAZY 23

4 VÝVOJOVÉ PROSTŘEDKY 25

4.1 VÝVOJOVÉ PROSTŘEDÍ ST VISUAL DEVELOP 25
4.2 PROGRAMOVACÍ MODUL RAISONANCE RLINK 27
4.3 STM8 VÝVOJOVÁ DESKA 28
4.3.1 PROCESOR STM8S 28
4.4 PROUDOVÝ OMEZOVAČ 29

5 MASTER KOMUNIKAČNÍ MODUL 31

5.1 PRINCIP ČINNOSTI 31
5.2 POPIS FUNKCÍ A STAVOVÝCH PROMĚNNÝCH 32
5.2.1 ROZDĚLENÍ DO SOUBORŮ 32
5.2.1.1 Soubor main.c 32
5.2.1.2 Soubor it.c 32
5.2.1.3 Soubor dalimaster.c 33
5.2.2 SEZNAM VYTVOŘENÝCH FUNKCÍ 33
5.2.3 FUNKCE VOLANÉ UŽIVATELEM 33
5.2.3.1 init_DALI (dalimaster.c) 33
5.2.3.2 get_flag (dalimaster.c) 34
5.2.3.3 send_data (dalimaster.c) 34
5.2.4 VNITŘNĚ VOLANÉ FUNKCE 35
5.2.4.1 receive_data (dalimaster.c) 35
5.2.4.2 receive_tick (dalimaster.c) 35
5.2.4.3 send_tick (dalimaster.c) 36
5.2.4.4 CollisionDet (dalimaster.c) 36
5.2.4.5 set_timer_delay (it.c) 36
5.2.4.6 Repeat_after_CD (dalimaster.c) 36
5.2.4.7 enable_EXTI (dalimaster.c) 36
5.2.4.8 disable_EXTI (dalimaster.c) 37
5.2.4.9 set_DALIOUT (dalimaster.c) 37
5.2.4.10 get_DALIOUT (dalimaster.c) 37
5.2.4.11 get_DALIIN (dalimaster.c) 37
5.2.4.12 DataReceived (it.c) 37
5.3 PRINCIP PŘENOSU DAT V MODULU MASTER 37
5.3.1 ČASOVÁNÍ POMOCÍ TIMER3 37
5.3.2 ODESLÁNÍ DAT POMOCÍ SEND_DATA FUNKCE 39
5.3.3 PERIODICKÉ VOLÁNÍ SEND_TICK 40
5.3.4 PŘIJÍMÁNÍ DAT POMOCÍ RECEIVE_DATA FUNKCE 43
5.3.5 PERIODICKÉ VOLÁNÍ RECEIVE_TICK 44

6 SLAVE KOMUNIKAČNÍ MODUL 47

6.1 PRINCIP ČINNOSTI 47
6.2 POPIS FUNKCÍ A STAVOVÝCH PROMĚNNÝCH 47
6.2.1 ROZDĚLENÍ DO SOUBORŮ 47
6.2.2 SEZNAM VYTVOŘENÝCH FUNKCÍ 48
6.2.3 FUNKCE VOLANÉ UŽIVATELEM 48
6.2.3.1 init_DALI (dalislave.c) 48
6.2.3.2 get_flag (dalislave.c) 49
6.2.3.3 send_data (dalislave.c) 49
6.2.4 VNITŘNĚ VOLANÉ FUNKCE 49
6.2.4.1 receive_data (dalislave.c) 49
6.2.4.2 receive_tick (dalislave.c) 50
6.2.4.3 send_tick (dalislave.c) 50
6.2.4.4 DataReceived (stm8_it.c) 50
6.2.4.5 Ostatní procedury 50
6.3 PRINCIP PŘENOSU DAT V MODULU SLAVE 50
6.3.1 ČASOVÁNÍ POMOCÍ TIMER3 51
6.3.2 ODESLÁNÍ DAT POMOCÍ SEND_DATA FUNKCE 52
6.3.3 PERIODICKÉ VOLÁNÍ SEND_TICK 52
6.3.4 PŘIJÍMÁNÍ DAT POMOCÍ RECEIVE_DATA FUNKCE 53
6.3.5 PERIODICKÉ VOLÁNÍ RECEIVE_TICK 54

7 DEMONSTRAČNÍ PROGRAM 56

7.1 VÝVOJOVÉ PROSTŘEDÍ A POŽADAVKY 56
7.2 STAV PO SPUŠTĚNÍ 56
7.3 HLAVNÍ OBRAZOVKA 57
7.3.1 POSUVNÍK ÚROVNĚ OSVĚTLENÍ 57
7.3.2 FADE TIME / FADE RATE 57
7.3.3 GROUPS 58
7.3.4 SCENE 59
7.4 BALLASTS LIST OBRAZOVKA 59
7.5 CONNECTION SETUP OBRAZOVKA 61
7.6 LOG OBRAZOVKA 61
7.7 SEARCH BALLASTS OBRAZOVKA 63
7.8 MANUAL COMMANDS OBRAZOVKA 64

8 ZÁVĚR 66

LITERATURA 68

PŘÍLOHA A 71

PŘÍLOHA B 80

PŘÍLOHA C 82

PŘÍLOHA D 83

PŘÍLOHA E 84

Seznam obrázků

Obr. 1: Ovládání osvětlení pomocí PC programu 4
Obr. 2: Příklad elektroinstalace DALI 6
Obr. 3: Klasická elektroinstalace bez DALI 7
Obr. 4: Elektroinstalace za použití DALI 8
Obr. 5: Základní DALI smyčka 9
Obr. 6 Rozšířená DALI smyčka 9
Obr. 7: Změny logických úrovní 13
Obr. 9: Master → Slave datový rámec 15
Obr. 11: Slave → Master datový rámec 16
Obr. 13: Body detekce kolizí 19
Obr. 14: Logaritmická křivka tlumení 22
Obr. 15: Vývojové prostředí ST Visual Develop 26
Obr. 16: Simulátor prostředí ST Visual Develop 27
Obr. 17: Raisonance Rlink modul 27
Obr. 18: STM8 vývojová deska 28
Obr. 19: Schéma proudového omezovače 30
Obr. 20: Hlavní diagram Master modulu a diagram příjmu dat 31
Obr. 21: Diagram obsluhy přerušení timer3 (Master modul) 38
Obr. 22: Diagram funkce send_data Master modulu 39
Obr. 23: Průběh signálu DALI rámce 41
Obr. 24: Vývojový diagram metody send_tick 42
Obr. 25: Vývojový diagram funkce receive_data 43
Obr. 26: Vývojový diagram funkce receive_tick (Master modul) 44
Obr. 27: Diagram obsluhy přerušení timer3 (Slave modul) 51
Obr. 28: Diagram funkce send_data Slave modulu 52
Obr. 29: Vývojový diagram funkce send_tick (Master modul) 53
Obr. 30: Vývojový diagram funkce receive_tick (Master modul) 54
Obr. 31: Hlavní obrazovka demonstračního programu 57
Obr. 32: Výběr skupin slave zařízení 59
Obr. 33: Ballasts List obrazovka 60
Obr. 34: Obrazovka nastavení sériové komunikace 61
Obr. 35: Obrazovka log (ukládání záznamů) 62
Obr. 36: Search Ballasts obrazovka a Ballasts List 63
Obr. 37: Obrazovka Manual Commands 64

Seznam tabulek

Tabulka 1: Seznam funkcí modulu Master 33
Tabulka 2: Možné hodnoty proměnné flag (Master modul) 34
Tabulka 3: Seznam funkcí modulu Slave 48
Tabulka 4: Možné hodnoty proměnné flag (Slave modul) 49
Tabulka 5: Stupně Fade Time/ Fade rate 58

Slovník pojmů

- *master* – Komunikační zařízení DALI sítě, jehož úkolem je řídit slave zařízení a případně komunikovat s ostatními master zařízeními.

- *slave* – Komunikační zařízení DALI sítě, jehož úkolem je řídit světelné zařízení podle příkazů přijatých od master prvku DALI sítě.

- *modul Master* – Programová knihovna vytvořená pro použití v master zařízeních, která byla vytvořena pro účely této diplomové práce.

- *modul Slave* - Programová knihovna vytvořená pro použití v slave zařízeních, která byla vytvořena pro účely této diplomové práce.

- *short adresa* – adresa DALI komunikačního zařízení, s rozsahem 0-63.

- *broadcast příkaz* – Příkaz adresovaný všem zařízením na DALI síti

- *forward rámec* – zpráva odeslaná master zařízením do sítě DALI

- *backward rámec* – zpráva odeslaná slave zařízením jako odpověď na dotaz od master zařízení.

1 Úvod

D.A.L.I. (Digital Addressable Lighting Interface, v dalším textu uváděn jako DALI) je komunikační protokol vyvinutý na základě požadavků výrobců osvětlovacích systémů na jednotný komunikační protokol pro propojení zařízení od různých výrobců.

Tento protokol byl vyvíjen od začátku devadesátých let 20. století, ale komerčního využití se dočkal až v roce 1998. Lze jej používat bez nutnosti platby licenčních poplatků, aby se tak zajistilo jeho co nejrychlejší rozšíření mezi výrobce osvětlovacích systémů a dalších zařízení pracujících se sítí DALI.

Poprvé byl přijat jako norma v Evropě v roce 2002 pod označením IEC Standard 60929, Annex E & G. V roce 2004 byl přijat jako norma v USA pod označením NEMA Standard 234-2004.

DALI využívá bi-fázové kódování s rychlostí 1200 bits/s a je schopno adresovat až 64 *master* a 64 *slave* zařízení zapojených do své sítě. Ke komunikaci a ovládání je zapotřebí pouze dvouvodičového vedení a řízená osvětlení je možno díky adresování ovládat jednotlivě nebo je sdružovat do skupin a ovládat tak celou skupinu najednou jakoby to byl jediný prvek.

STM8 je nový 8-bitový mikrokontroler pro obecné použití jednoho z největších výrobců mikrokontrolerů na světě, firmy ST Microelectronics. Jeho rozměry, nízké energetické nároky a vysoká výkonnost jej předurčují pro účely vysoce efektivních průmyslových zařízení.

Narozdíl od svého předchůdce, mikrokontroleru ST7, nemá STM8 zabudován hardwarový interface pro DALI protokol, je nutné tedy vytvořit programovou knihovnu, která bude tento interface nahrazovat.

V této diplomové práci, zadané a vykonávané ve společnosti STMicroelectronics se zabývám realizací programové knihovny pro komunikaci *master* a *slave* zařízení po DALI síti. Protože mikrokontroler STM8 obsažený v obou zařízeních nemá hardwarovou podporu protokolu DALI, vytvořil jsem pro ně fyzickou vrstvu protokolu programově, včetně kontroly přenosu, detekce kolizí, nastavení priorit zpráv, řízení opakování zpráv a dalších požadavků vyplývajících z normy pro DALI protokol. Ke sběrnici DALI jsem připojil *slave* prvky ovládající světelná zařízení podle příkazů přijatých z *master* zařízení připojeného k téže sběrnici.

Pro lepší prezentaci dosažených výsledků jsem navíc vytvořil demonstrační řídící program pro PC, kterým se pomocí jednoduchého a názorného grafického menu ovládají světelná zařízení připojená ke slave prvkům. Demonstrační program komunikuje s *master* zařízením přes USB sběrnici nebo sériový port. Programovou knihovnu pro *master* zařízení jsem proto navrhnul tak, aby sloužila jako brána mezi daty přijatými po sériové lince z demonstračního programu na PC a sběrnicí DALI.

Diplomovou práci jsem rozčlenil do dvou hlavních částí.

První část je tvořena druhou, třetí a čtvrtou kapitolou. V druhé kapitole je popsán základní princip DALI sítí a umožňuje čtenáři rychlé zorientování v probírané tématice. Třetí kapitola se pak zabývá podrobným popisem DALI normy. Čtvrtá kapitola popisuje vývojové prostředky, které byly použity při práci na vývoji programové knihovny a demonstračního programu.

Druhá část diplomové práce se již zabývá způsobem, jakým jsem řešil zadané úkoly. Pátý a šestý oddíl této diplomové práce popisuje vlastní programovou knihovnu pro komunikační moduly master a slave. Poslední, sedmý oddíl, se zabývá popisem demonstračního programu pro PC ve stylu uživatelského manuálu.

V příloze A je navíc uveden uživatelský manuál této programové knihovny v anglickém jazyce, určený pro mezinárodní uživatele tohoto řešení.

2 DALI (Digital Addressable Lighting Interface)

Tato kapitola slouží k pochopení základních principů DALI sítí, jejích výhod a nevýhod oproti současným osvětlovacím systémům. Konkrétní detaily týkající se DALI protokolu jsou poté uvedeny a rozebrány v kapitole 3.

2.1 Historie

Myšlenka centralizovaného ovládání osvětlovacích systémů budov a komunikace mezi jednotlivými zařízeními není zdaleka nová.

Ze začátku bylo ale nutné vyřešit problémy spojené s velkou různorodostí a množstvím typů světelných zařízení. Dalším problémem bylo, že tato zařízení musí pracovat co nejúsporněji a přitom v silně zarušeném prostředí. A jako vždy, standardizace a vývoj systémů schopných komunikovat a ovládat produkty od různých výrobců s sebou nese i značně vyšší náklady při vývoji.

2.1.1 Sdružení AG_DALI

Dnes jsou výrobci používající standard DALI sdruženi do obchodní společnosti AG-DALI se sídlem ve Frankfurtu nad Mohanem. Společnost se stará o co největší rozšíření tohoto protokolu, jeho aktualizaci a v neposlední řadě o přijímání nových členů z řad výrobců, kteří jsou ochotni tento protokol přijmout a podporovat. Jejími členy jsou všichni významní výrobci světelných zařízení a velké společnosti zabývající se automatizací.

2.1.2 Druhy norem

První digitálně ovládaný světelný prvek byl vyvinut již v roce 1991 společností Tridonic. První zařízení odpovídající normě DALI bylo vyvinuto v roce 1998.

V roce 2000 byl v Evropě DALI protokol postoupen ke standardizaci pod označením IEC Standard 60929, Annex E & G a v roce 2002 byl tento standard oficiálně schválen.

O dva roky později, v roce 2004, byl standard převzat americkým elektrotechnickým sdružením výrobců NEMA a vydán pod označením NEMA Standard Publication 243-2004. Narozdíl od evropské normy je tato volně dostupná na internetu.

V roce 2008 byla vydána zatím nejnovější aktualizace DALI normy pod označením IEC 62386, která upravuje a specifikuje především použití 3 bytové komunikace mezi jednotlivými *master* zařízeními.

Komunikační knihovna vytvořená v rámci této diplomové práce již plně odpovídá normě IEC 62386 z roku 2008.

2.2 Principy DALI sítí

Pro komunikaci v DALI síti jsou potřeba dva druhy zařízení. Prvním je *master*, který má možnost rozesílat příkazy zařízením ovládajícím samotné osvětlení, tzv. *slave* zařízení. *Master* má možnost komunikovat i s ostatními *master* zařízeními.

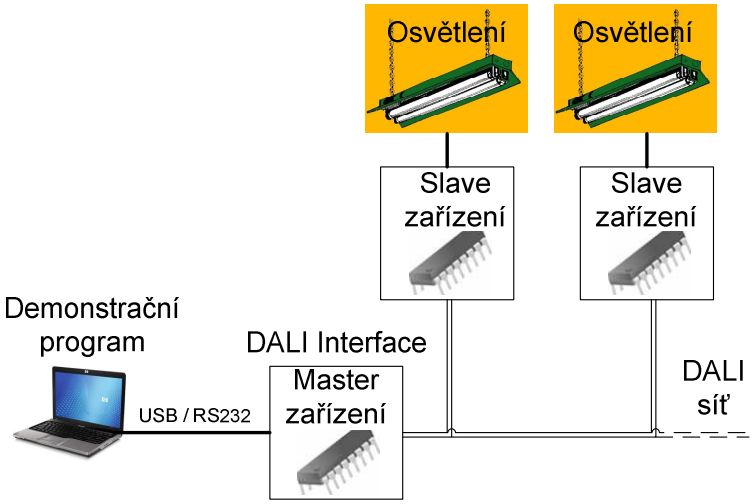
2.2.1 Možnosti komunikace mezi zařízeními

Komunikace mezi prvky DALI sítě je zajištěna pomocí 1, 2 a 3-bytových zpráv. Délka zprávy záleží na tom jestli se jedná o komunikaci *Master-Slave* nebo *Master-Master*.

2.2.1.1 Master komunikace

Hlavní funkce zařízení typu *master* je ale ovládat *slave* prvky sítě, kterým předává příkazy pomocí 2-bytových zpráv.

Master velmi často slouží jako interface mezi vyšším prvkem ovládání a DALI sítí. V našem případě je vyšším prvkem ovládání demonstrační program pro PC, vytvořený pro účely této diplomové práce.



Obr. 1: Ovládání osvětlení pomocí PC programu

Maximální počet zařízení typu *master* v síti DALI je 64, vzájemně spolu mohou komunikovat pomocí 3-bytových zpráv.

2.2.1.2 Slave komunikace

Maximální počet zařízení typu *slave* v sítích DALI je 64 a mohou komunikovat pouze s zařízeními typu *master*, které je plně ovládají.

Od *master* přijímají příkazy v podobě 2-bytových zpráv a na příkazy typu „Query“ odpovídají v podobě 1-bytové zprávy.

2.2.2 Způsoby adresace slave zařízení

Existují tři způsoby, kterými lze adresovat a tím i ovládat *slave* zařízení.

- *Broadcast* příkazy
- Individuální adresování
- Skupinové adresování

Broadcast příkazy jsou takové, které přijmou všechna *slave* zařízení a budou na ně reagovat. Z toho důvodu není vhodné takto adresovat zprávy na které čeká *master* odpověď, protože všechna *slave* zařízení začnou odpovídat současně.

Individuální příkaz je určen pouze pro jediné *slave* zařízení, které na něj musí zareagovat. Ostatní *slave* zařízení příkaz sice přijmou, ale po vyhodnocení adresy příjemce zprávu zahodí.

Každé *slave* zařízení může patřit až do šestnácti skupin a na skupinovou zprávu reagují jen zařízení do této skupiny patřící, ostatní zprávu zahazují. Opět není vhodné skupinově adresovat zprávy na které se očekává odpověď ze strany *slave* zařízení.

2.2.3 Nastavení scén

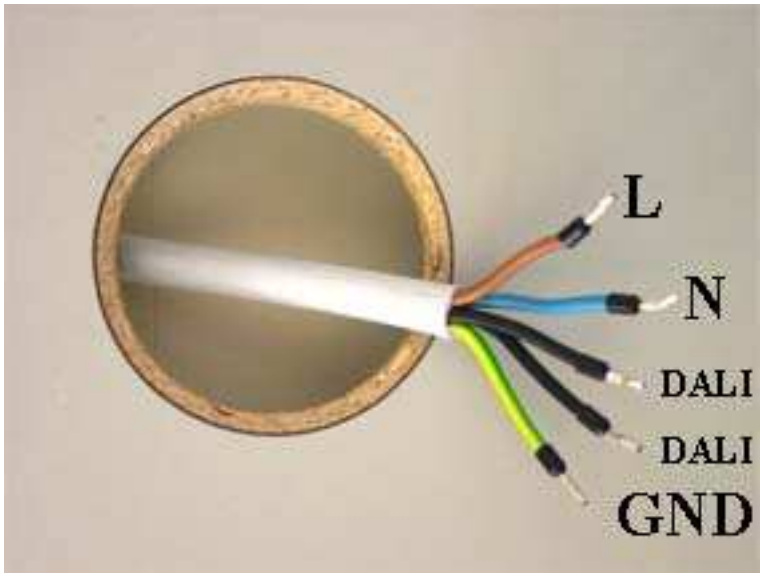
Každé *slave* zařízení může mít až šestnáct předdefinovaných úrovní osvětlení (scén). Scény je nutné definovat ve vyšším ovládacím prvku, v našem případě demonstračním programu na PC.

2.3 Elektroinstalace v DALI síti

Inteligentní světelné zařízení pro práci v DALI síti se sestává ze dvou částí. Prvním je předřadník. Předřadník mimo činností spojených s DALI komunikací ovládá i činnost

samotné světelné lampy podle příkazů přijatých od ovládacího zařízení (*master*). Světelná lampa je druhou částí světelného zařízení a dohromady tvoří zařízení typu *slave*.

K přenosu signálu jsou potřeba dva vodiče, není nutné aby byl tento pár vodičů kroucený ani stíněný. V zájmu co nejnižších nákladů na elektroinstalaci v budovách jsou DALI vodiče velmi často již obsaženy v kabelu, spolu s napájecími vodiči pro samotné světelné zařízení.



Obr. 2: Příklad elektroinstalace DALI [17]

V normě je také definovaná nutnost galvanického oddělení DALI sběrnice od vstupu mikroprocesoru, který řídí *slave* zařízení, vhodnými optočleny. Tato ochrana je zde předepsána především z důvodu možnosti špatného zapojení vodičů a tím připojení napájecích vodičů na místo DALI komunikace.

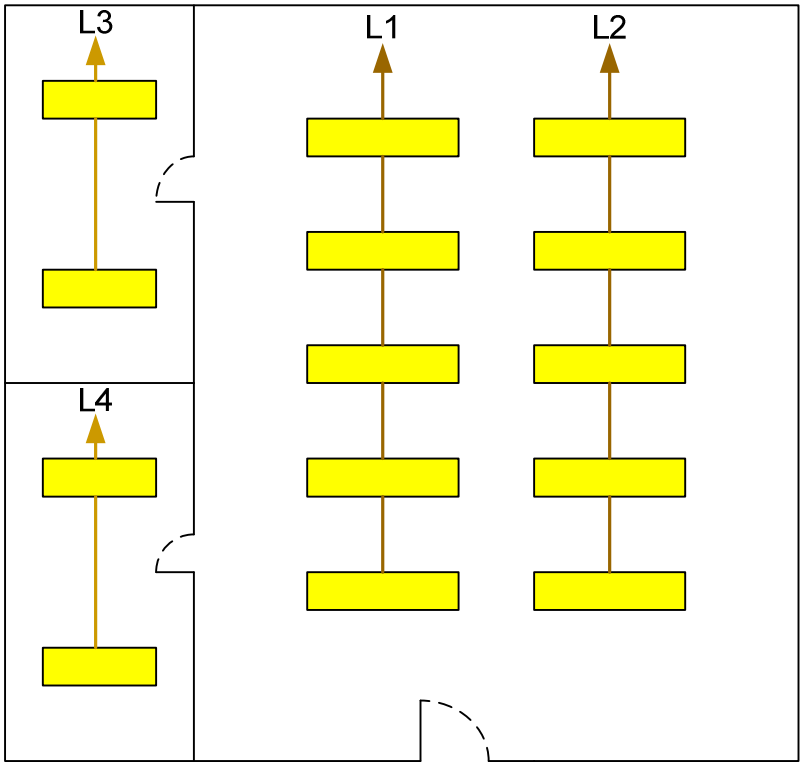
2.3.1 Klasická elektroinstalace bez DALI

V klasickém světelném obvodu je nutné vyčlenit jednu samostatnou fázi pro každý světelný obvod.

Pokud budeme chtít navíc ovládat všechna osvětlení z jednoho místa, musíme na toto místo přivést všechny fáze použité ve světelném obvodu. Toto samozřejmě radikálně zvyšuje náklady na délku rozvodů a tedy cenu elektroinstalace.

Ovládací prvky světelného obvodu, jako například vypínače, stmívače nebo pohybové senzory musí být zapojeny na začátek světelného obvodu, podle fyzického umístění těchto prvků se tedy opět značně mění náklady na rozvody.

Výhodou je, že samotné světelné zařízení je velmi levné a jeho instalace jednoduchá.



Obr. 3: Klasická elektroinstalace bez DALI

2.3.2 Elektroinstalace za použití DALI

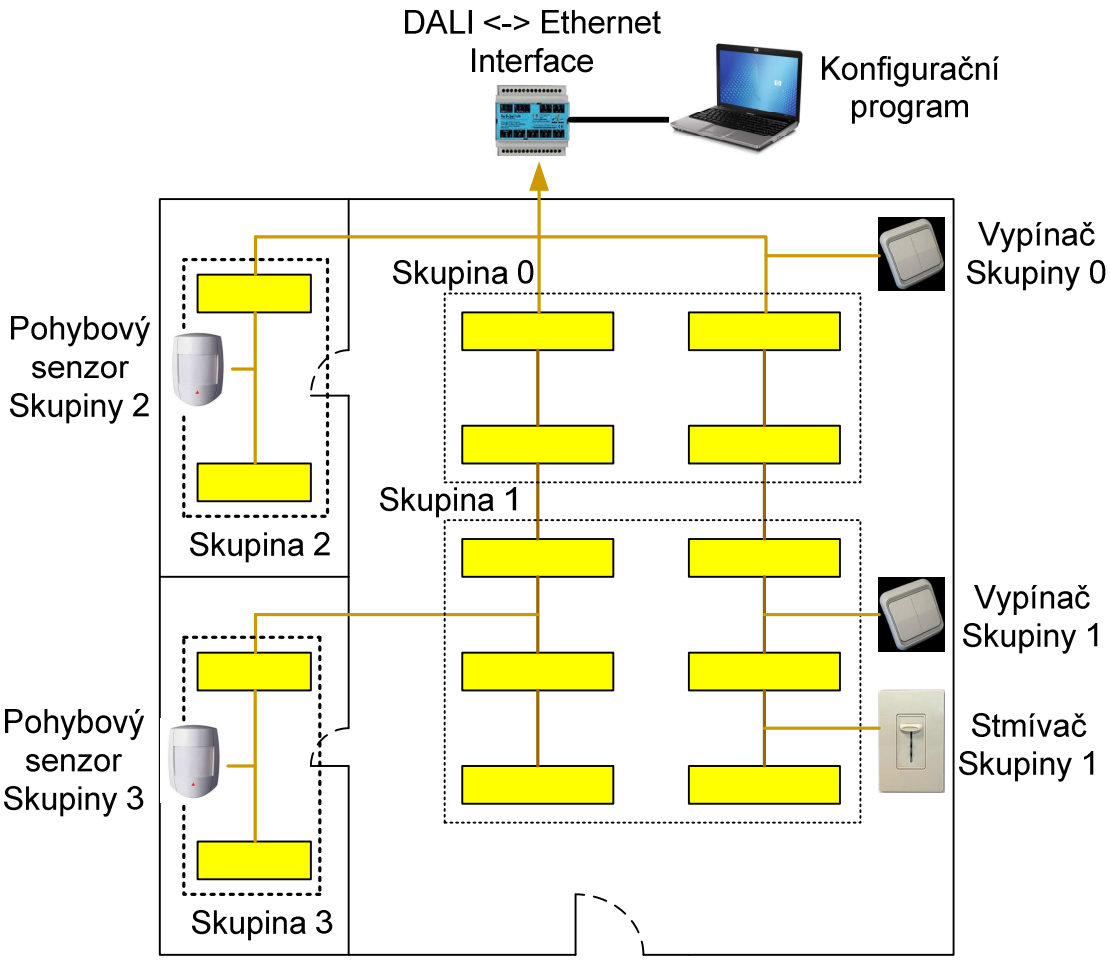
Použití DALI sítí při elektroinstalaci je výhodné především z pohledu nákladů na rozvody. Pro celý světelný obvod nám bude stačit jediná fáze (příslušně proudově dimenzovaná.

Při připojování ovládacích prvků k DALI sběrnici nezáleží ve kterém místě sběrnice bude prvek připojen. Rozvody se tedy velmi zjednoduší a zlevní.

Rozdělení *slave* prvků do skupin a následné přiřazení těchto skupin k ovládacím prvkům je nutno udělat přes konfigurační rozhraní, které může být s DALI sítí spojeno přes Ethernet nebo jiné průmyslové sítě za použití k tomu určených převodníků.

Instalovaná DALI zařízení jsou také schopna předávat informace o svém stavu, umožnit rychlejší odhalení případné poruchy a snížit tak náklady na údržbu.

Nevýhodou jsou samozřejmě vyšší náklady na pořízení světelných, ovládacích a převodních zařízení. Tyto náklady se rychle zhodnotí především v případě nutnosti změny složení skupiny světelných zařízení, které ovládá určitý *master* prvek, například při přestavbě příček v univerzálním otevřeném kancelářském prostoru. V DALI síti je toto přeskupení provedeno softwarově, v klasických případech bez DALI nákladnou změnou rozvodů.



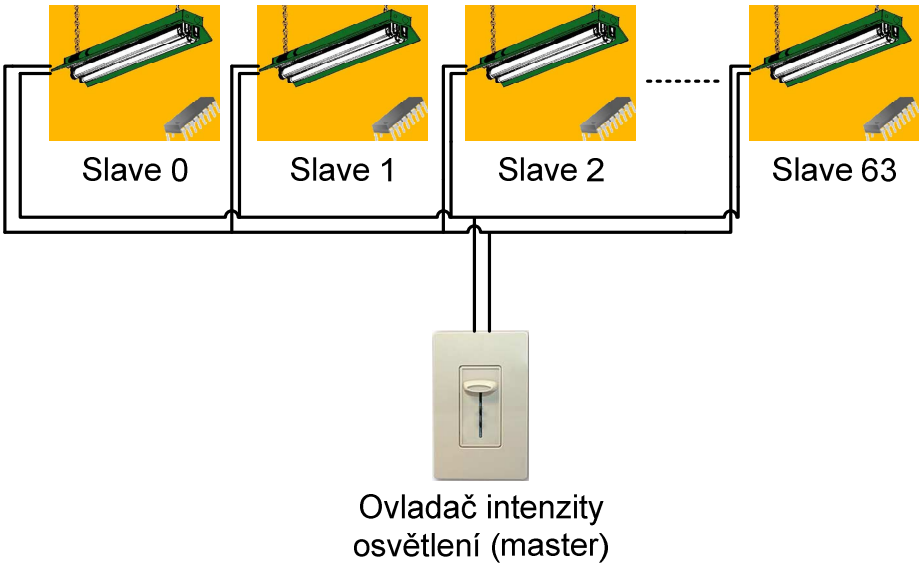
Obr. 4: Elektroinstalace za použití DALI

2.4 Varianty zapojení DALI sítě

Díky sběrnicové struktuře DALI sítě je možné prvky fyzicky připojovat v jakémkoliv místě. Důraz by měl být však kladen na přehlednost a logické uspořádání prvků v síti.

2.4.1 Základní DALI smyčka

Základní DALI smyčka na Obr. 5 obsahuje pouze jedno *master* zařízení a až 64 *slave* zařízení. V tomto případě *master* představuje jednoduchý modul určený pro stmívání/ rozsvěcení všech světel najednou, bez možnosti ovládání jednotlivých světel individuálně nebo po skupinách.

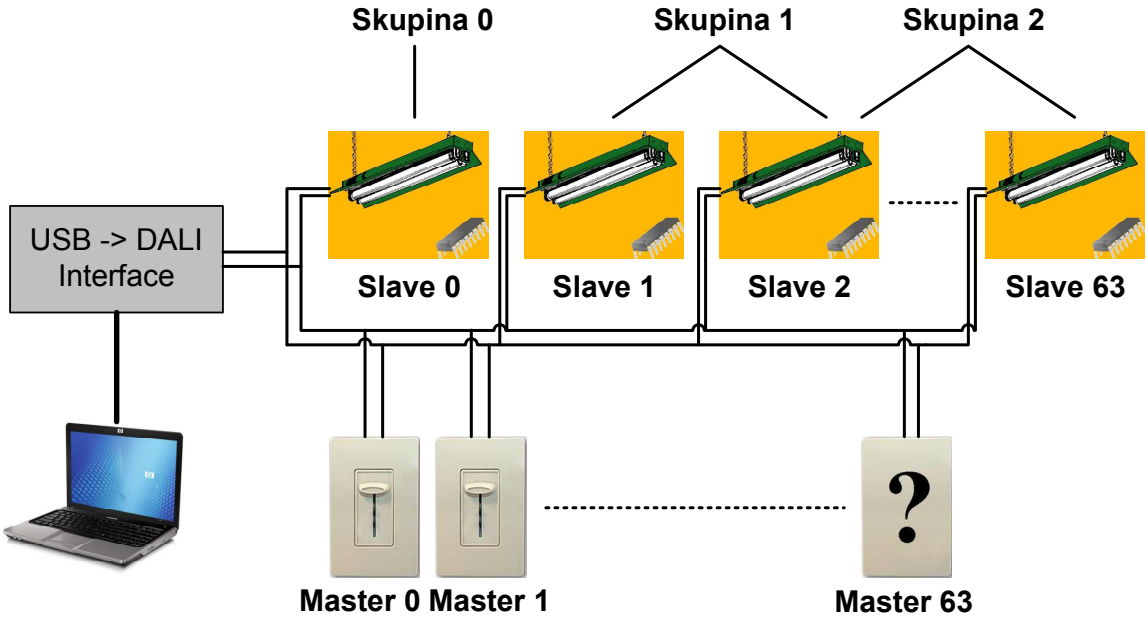


Obr. 5: Základní DALI smyčka

2.4.2 Rozšířená DALI smyčka

V rozšířené smyčce se může vyskytovat až 128 adresovatelných zařízení, 64 *slave* a 64 *master* zařízení. *Master* zařízení mohou být různých druhů, které jsou podrobně popsány v kapitole 3.1.1.

Slave zařízení mohou být v tomto případě rozdělena až do šestnácti skupin nebo ovládána individuálně. Poté musí být příslušný *master* modul nakonfigurován aby komunikoval s tou určitou skupinou nebo samostatným *slave* zařízením. K tomu je potřeba použít programovací nástroj, v našem případě demonstrační program pro PC.



Obr. 6 Rozšířená DALI smyčka

2.5 Výhody DALI instalací

V dlouhodobém pohledu jsou náklady na vytvoření a údržbu DALI sítě menší než náklady na systém s ovládáním osvětlení u klasických instalací, přesto je zatím neochota zavádět tyto systémy značná. Zde jsou uvedeny některé z výhod DALI systémů:

- Jednoduchá instalace, jednoduché rozvody
- Energetické úspory
- Menší náklady na údržbu
- Rychlá doba návratnosti
- Vysoká flexibilita (softwarové přeskupení světelných zařízení)
- Větší kontrolovatelnost (každé světelné zařízení může být ovládáno samostatně)
- Možnost měření spotřeby jednotlivých světelných zařízení, skupin nebo celých místností a tedy možnost samostatného účtování pro případné nájemce prostor.

2.6 Nevýhody DALI instalací

Nevýhody omezující masové rozšíření zařízení pracujících v DALI sítích jsou následující.

- Vyšší počáteční náklady na nákup zařízení
- Potřeba inicializace a konfigurace přes počítač
- DALI určeno jen pro účely ovládání osvětlení
- Malé portfolio produktů podporujících DALI

2.7 Minimální výpočetní a paměťové nároky

Výpočetní a paměťové nároky *modulů Master* a *Slave* na programovou implementaci DALI protokolu nejsou příliš vysoké a měl by jim odpovídat i méně výkonný mikroprocesor.

<u>Hardwarové nároky</u>

Součásti, které by měl mít mikroprocesor možnost používat vycházejí z principu činnosti komunikace popsaných dále v kapitolách 5 a 6.

Pro používání *modulů Master* a *Slave* je nutné mít k dispozici timer s možností přerušení v případě přetečení čítače. Rychlost časování procesoru pro samotnou obsluhu komunikace stačí v řádu jednotek MHz, ale pokud má procesor navíc vedle obsluhy

komunikace stíhat i jiné operace spojené se zpracováním přijatých nebo odeslaných dat, je nutné volit výkonnější mikroprocesor.

Poslední nutnou součástí mikroprocesoru jsou 2 piny pro obecné využití s možností externího přerušení, které se v inicializační proceduře přiřadí DALI komunikaci.

Vhodná je také, v případě *Slave modulu,* možnost uvedení zařízení do klidového režimu s nízkou spotřebou v době, kdy se čeká na příchod dalších příkazů.

Paměťové nároky

Nároky mé programové knihovny na paměť kódu jsou 4kB pro *Slave modul* a 8,5kB pro *Master modul*, což je srovnatelné s ostatními programy, které mají za úkol emulovat komunikaci. Procesor STM8S2, na kterém byla programová knihovna testována má přitom velikost programové paměti 128kB.

Pokud by chtěl uživatel rozšířit mnou vytvořenou programovou knihovnu a ke kódu pro komunikaci *slave* zařízení přidat i moduly, které přímo příkazy zpracovávají a podle příslušné normy ovládají samotné světelné zařízení, přidávají se k výše jmenovaným i o něco vyšší nároky na datovou paměť.

3 DALI protokol

Tato kapitola podrobně popisuje DALI komunikační protokol tak jak je definován v normě IEC Standard 60929, Annex E & G z roku 2002 a posléze rozšířen normou IEC 62386 z roku 2008.

3.1 Master/slave topologie

Zařízení v DALI síti pracují buďto se statutem *master* nebo *slave*.

Slave jsou zařízení, která ovládají světelné prvky, odpovídají na dotazy řídících prvků (*master*), ale sama komunikaci nezahajují.

Master prvky sítě jsou všechna řídicí zařízení, která komunikují a zadávají příkazy *slave* zařízením a jsou také schopna komunikovat mezi sebou.

3.1.1 Druhy Master zařízení

Existuje několik základních druhů *master* zařízení, jejichž úkolem je posílat DALI příkazy nejen zařízením ovládajícím osvětlovací prvky ale i komunikovat s ostatními *master* prvky. Jsou to tyto:

- Vypínač (zapnout/vypnout) a ovladač intenzity osvětlení (včetně rychlosti zhasínání / rozsvěcení)
- Ovladač scény (vyvolá přednastavený režim osvětlení)
- Senzory pohybu
- Regulátor denního světla v otevřené smyčce (řídí osvětlení v závislosti na intenzitě denního světla)
- Regulátor denního světla v uzavřené smyčce (řídí osvětlení v závislosti na celkové intenzitě světla, tj. umělého a denního)
- Plánovač (světelné efekty, spouštění přednastaveného režimu v daný okamžik)
- Brána (propojení DALI sítě s ostatními systémy – TCP/IP, LonWorks...)
- Sequencer (spuštění sekvence přednastavených příkazů)
- Zdroj energie pro DALI smyčku
- Nouzové osvětlení – nouzové nebo alespoň záložní světlo
- Obecné zařízení s analogovým vstupem
- Zařízení pro uchování dat

- Neznámé zařízení - takové zařízení na které se nehodí žádný z výše uvedených popisů

Další typy zařízení mohou vzniknout kombinací základních druhů uvedených výše. Maximální počet druhů *master* zařízení je 64 a zatím jsou normou specifikovány pouze výše uvedené druhy a ostatní jsou rezervovány pro budoucí využití.

3.2 Fyzická vrstva

DALI protokol je bi-fázový (Manchesterský) asynchronní sériový protokol. Bi-fázově jsou kódovány všechny bity v datovém rámci, s výjimkou dvou stop bitů (žádná změna úrovně).

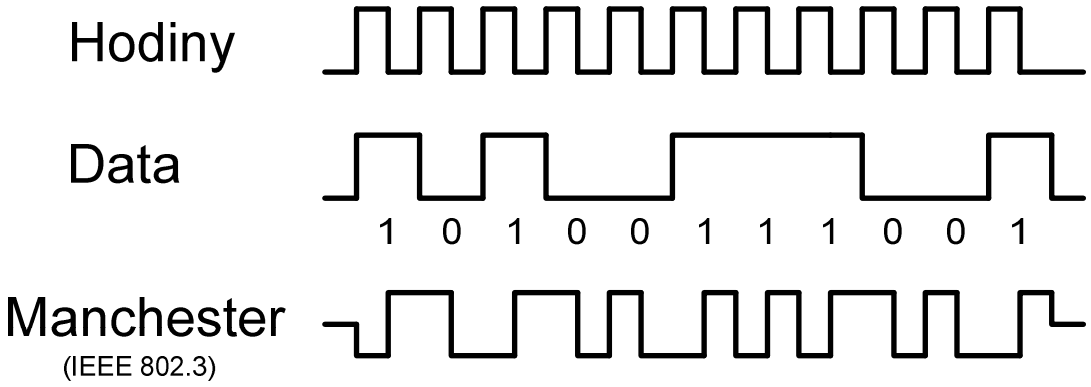
3.2.1 Bi-fázové kódování

Při použití bi-fázového kódování (Manchesterské) je logická 1 vyjádřena jako přechod úrovně z nízké do vysoké hladiny, pro logickou 0 platí opak a jde o hranu přechodu z vysoké do nízké úrovně.

Obr. 7: Změny logických úrovní

Bi-fázové kódování má hranu nesoucí datovou informaci uprostřed periody bitu. Pokud má předchozí bit stejnou logickou hodnotu jako aktuální, je nutná přítomnost hrany i na začátku periody bitu. Tato hrana ovšem nenese informaci a je použita jen pro nastavení signálové úrovně na správnou hodnotu.

Tento princip odpovídá normě IEEE 802.3 (použití také v Ethernetových sítích) a ilustruje jej následující obrázek:



Obr. 8: Princip bi-fázového kódování [25]

3.2.2 Rychlost přenosu

Rychlost přenosu je normou specifikována na 1200bits/s ±10% pro všechna zařízení v DALI síti.

3.2.3 Elektrická specifikace

Elektrická specifikace definuje napěťové úrovně sběrnice, klidovou úroveň sběrnice a délky náběžných a spádových hran.

Napěťové úrovně sběrnice

Pro vysokou úroveň na sběrnici je definována hodnota napětí mezi +9,5V až +22,5V, pro nízkou úroveň na sběrnici to je -6,5V až +6,5V.

Pásmo napětí mezi +6,5V až +9,5V se považuje za nedefinovanou úroveň.

Klidová úroveň sběrnice

V případě že na sběrnici právě nedochází ke komunikaci, musí být napětí na sběrnici nastaveno na vysokou úroveň.

Délka náběžné a spádové hrany signálu

Trvání náběžné i spádové hrany signálu by neměla být kratší jak 10µs a ne delší než 100µs.

3.3 Komunikace mezi zařízeními

Prvky zapojené do DALI sítě spolu komunikují předepsanými způsoby závislými na typu zařízení. Důležitá je také kontrola kolizí na DALI síti, bez které by nešlo, při větším počtu zařízení na DALI síti, efektivně komunikovat.

3.3.1 Komunikační rámce

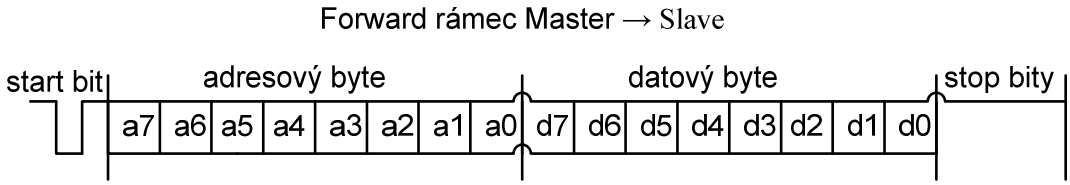
Při komunikaci *master → slave* se používají rámce s 1 adresovým bytem a 1 datovým bytem, pro *master → master* komunikaci je použit rámec s 1 adresovým bytem a 2 datovými byty. Oba tyto rámce se označují jako *forward rámce*.

Pro odpověď na zprávu od *master* zařízení se používají *backward rámce*, které jsou pro všechna zařízení stejná a obsahují pouze jeden datový byte.

3.3.1.1 Master → Slave forward rámec

Master → Slave forward rámec se skládá z 19 bitů s rozdělením:

- 1 start bit
- 1 adresový byte
- 1 datový byte
- 2 stop bity
- Délka trvání 15,83ms

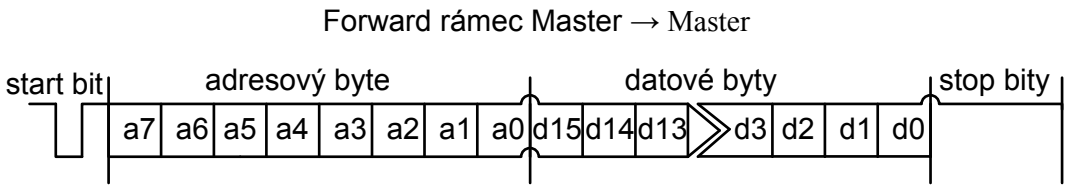


Obr. 9: *Master → Slave* datový rámec

3.3.1.2 Master → Master forward rámec

Master → Master forward rámec se skládá z 27 bitů s rozdělením:

- 1 start bit
- 1 adresový byte
- 2 datové byty
- 2 stop bity
- Délka trvání 22,5ms

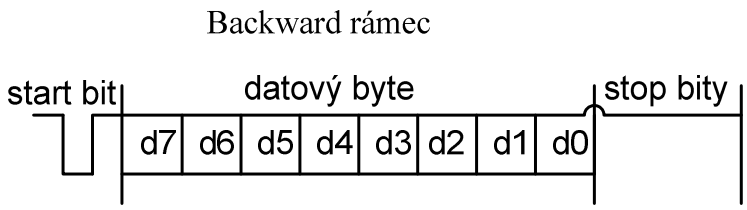


Obr. 10: *Master* → *Master* datový rámec

3.3.1.3 Backward rámec

Backward rámec se skládá z 11 bitů s rozdělením:

- 1 start bit
- 1 datový byte
- 2 stop bity
- Délka trvání 9,17ms



Obr. 11: *Slave* → *Master* datový rámec

3.3.2 Doby ustálení

Doby ustálení se liší mezi jednotlivými rámci.

<u>Mezi forward - backward</u>:

Minimálně 2,92ms, maximálně 9,17ms, jakákoliv delší prodleva je považována za odpověď NE nebo žádnou odpověď, podle druhu dotazu.

<u>Mezi backward – forward a forward – forward:</u>

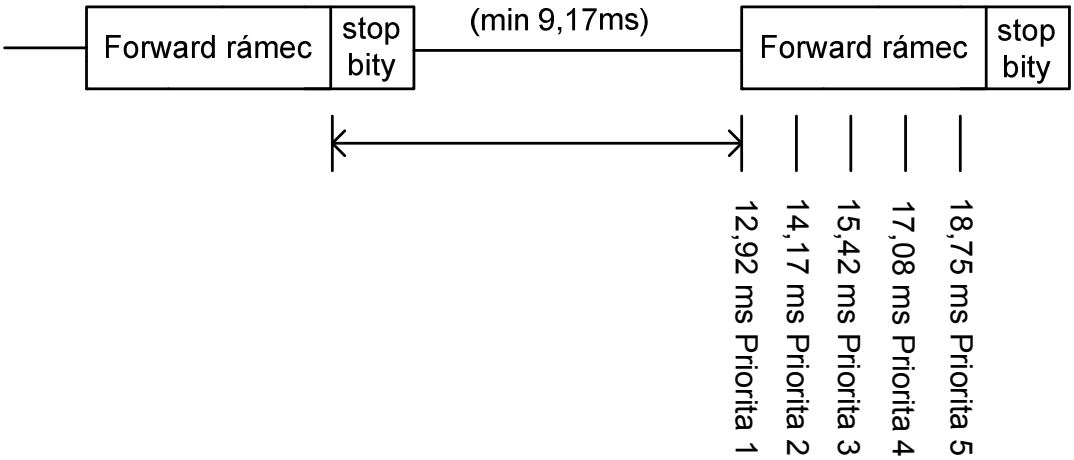
Minimální délka 9,17ms, dále se doba ustálení řídí podle priorit přiřazených *forward rámci*.

3.3.3 Přístupové priority

Dva po sobě jdoucí *forward rámce* jsou odesílány s časovou prodlevou, která odpovídá prioritě odesílané zprávy.

K zajištění přístupu na sběrnici ve správném pořadí a minimalizování šance kolizí slouží pět stupňů priorit zpráv, kde 1 je nejvyšší priorita, 5 nejnižší.

Výjimkou je *forward rámec* s prioritou 1 a jiného typu než typ „Query“. Zde je časová prodleva snížena na 10,83ms respektive 11,25ms v případě že poslední bit má logickou hodnotu 0 respektive 1.



Obr. 12: Přístupové priority a časová prodleva [13]

Priorita 1

Priorita 1 je určena pro všechny příkazy v transakci s výjimkou prvního a pro zprávy, které musí být zopakovány.

Priorita 2

Tato priorita je určena pro vynucené řídicí příkazy a příkazy mající vliv na nastavení intenzity osvětlení a dobu pohasínání. Příkladem je příkaz „GO TO SCENE“.

Priorita 3

Prioritu 3 využijí obecné konfigurační zprávy jako například příkaz „ADD TO GROUP“, která přiřadí adresované *slave* zařízení do určené skupiny.

Priorita 4

Priorita 4 je určena pro automatické řídicí příkazy. Takovým příkazem je například „STEP UP“ od *master* zařízení automaticky nastavující hladinu osvětlení podle změřené úrovně denního světla.

Priorita 5

Nejnižší priorita 5 je učena ověřovacím a zjišťovacím zprávám. Jako příklad lze uvést příkaz „QUERY BALLAST“, zjišťující přítomnost adresovaného *slave* zařízení.

3.3.4 Transakce

Transakce je sekvence jednoho nebo více *forward rámců*, odeslaných z jediného *master* zařízení, na které není očekávána odpověď. Transakce se může skládat až z 10ti *forward rámců*, v konfiguračním módu je limit zvýšen až na 19 *forward rámců*.

Pokud se jedná o transakci příkazů stejného typu, priorita první zprávy odpovídá těm uvedeným v kapitolách 0 až 0, následující zprávy jsou již odesílány s prioritou 1.

3.3.5 Detekce kolizí

V případě existence více *master* zařízení v jedné DALI síti, je nutné implementovat detekci kolizí, která ošetří takové případy, kdy se více *master* zařízení snaží odesílat data ve stejnou chvíli.

3.3.5.1 Pravidla pro detekci kolizí

Master zařízení musí okamžitě zastavit přenos *forward rámce* a celé transakce v případě že detekuje kolizi na DALI sběrnici.

Zrušený přenos rámce nebo celé transakce smí být opakován až po uplynutí časové prodlevy, která odpovídá prioritě se kterou byla zpráva nebo transakce přenášena. Transakce musí být zopakována celá od začátku.

Nutnost detekce kolizí se týká pouze *master* zařízení, respektive *forward rámců*, *slave* zařízení nemusí mít detekci kolizí implementovanou.

3.3.5.2 Body detekce kolizí

Detekce kolizí musí být prováděna ve třech bodech:

a) Přechod sběrnice z vysoké do nízké úrovně (E):

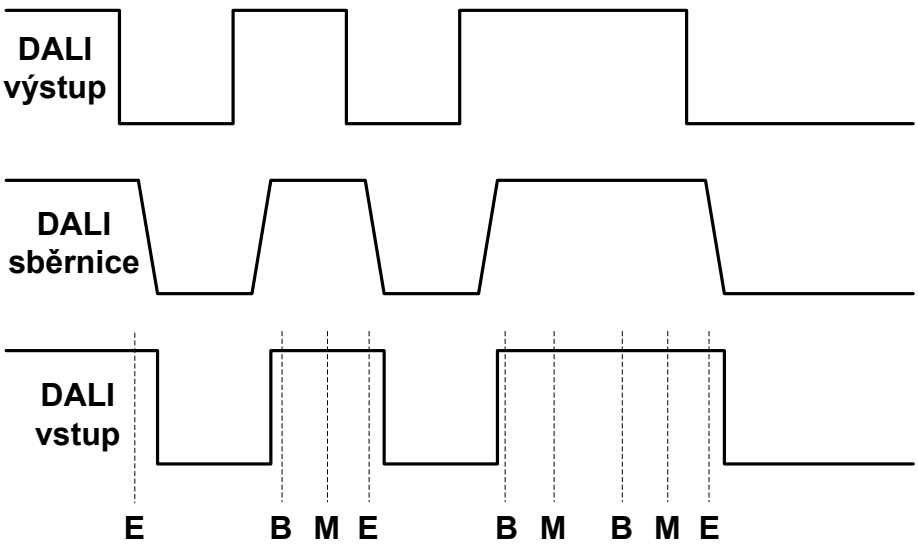
Během maximálně 26µs před přechodem sběrnice z vysoké do nízké úrovně je nutné zkontrolovat stav sběrnice. Pokud se sběrnice již nachází v nízké úrovni, je indikována kolize.

b) Uprostřed půl-bitu vysoké úrovně (M):

V polovině každého půl-bitu vysoké úrovně se provede kontrola stavu sběrnice. Pokud je na sběrnici indikována nízká úroveň, je toto vyhodnoceno jako kolize.

c) Po přechodu sběrnice do vysoké úrovně (B):

Během maximálně 26μs po přechodu sběrnice z nízké do vysoké úrovně je nutná kontrola zda se nachází sběrnice ve vysokém stavu . Nízká úroveň na sběrnici indikuje kolizi.



Obr. 13: Body detekce kolizí [2]

3.4 Adresační postupy

Další funkcí *master* zařízení je přiřazovat adresy *slave* zařízením, pokud ještě žádnou nemají, nebo jim je z hlediska přehlednosti vhodné přiřadit novou. To jakým způsobem je adresace zajištěna určují tři základní postupy, které jsou popsány níže.

3.4.1 Náhodné přiřazení adres

Algoritmus je založen na vyhledávání *slave* zařízení s nejmenším náhodně vygenerovaným číslem.

- Po přijetí příkazu „INITIALISE“ se *slave* zařízení na 15minut přepne do adresačního módu.

- Následuje příkaz „RANDOMISE“, po kterém *slave* zařízení vygeneruje 24-bitové náhodné číslo.
- *Master* zařízení poté vhodným algoritmem hledá *slave* zařízení s nejnižším náhodným číslem.
- *Slave* s nejnižším náhodným číslem je přiřazena první *short adresa* a je vyřazeno z procesu dalšího vyhledávání příkazem „WITHDRAW“.
- Algoritmus se cyklicky opakuje dokud není všem *slave* zařízením přiřazena *short adresa* (adresa 0 – 63).
- Adresační mód je ukončen zasláním příkazu „TERMINATE“.

3.4.2 Přiřazení fyzickým zvolením

Podmínkou pro tuto adresaci je existence zařízení, kterým uživatel upozorní *master* zařízení, kterému *slave* prvku má přiřadit danou *short adresu*. Toto zařízení může být například tlačítko.

Výhodou tohoto způsobu je, že si uživatel může jednoduše vybrat pořadí *slave* prvků v jakém budou očíslovány, aby se dosáhlo co největší přehlednosti.

3.4.3 Přednastavení

Short adresa slave prvku je v tomto případě již nastavena například pomocí adresových přepínačů. Uživatel musí však dbát na to, aby se adresy zvolené přepínači nepřekrývaly s adresami zvolenými automaticky.

Slave prvek s pevně nastavenou *short adresou* se také musí vyjmout z procesu automatické adresace, která byla popsána v předchozích dvou bodech.

3.5 Způsob adresace slave prvků

Master prvek může *slave* zařízení adresovat příslušným nastavením adresového byte. Jsou možné tři typy adresací, *broadcast*, skupinově nebo jednotlivě.

<u>Obecný tvar adresového byte:</u>

YAAAAAAS

3.5.1 Nastavení Y bitu

Nastavením Y bitu určujeme jestli adresujeme jeden určitý *slave* prvek (Y = 0) nebo skupinu prvků (Y = 1).

Při volání jediného *slave* zařízení představují bity A *short adresu* volaného *slave* prvku (0-63).

Pro volání skupiny *slave* prvků vypadá adresový byte následovně: 100AAAAS, kde bity A tentokrát značí číslo skupiny, kterou chceme ovládat (0-15).

3.5.2 Nastavení S bitu

S bit určuje zda byte následující po adresovém byte bude přímo ovládat hladinu osvětlení (S = 0) nebo bude obsahovat jiný příkaz *slave* prvku (S = 1).

3.5.3 Broadcast příkaz

Broadcast příkaz, to jest příkaz na který musí reagovat všechna *slave* zařízení, má prvních 7 bitů tvořených ze samých logických 1. Poslední bit má označení S a jeho funkce byla vysvětlena výše.

3.5.4 Adresace speciálních příkazů

Protože adresový prostor příkazového byte již nestačil počtu příkazů, používá se adresový byte pro skupinu speciálních příkazů. Adresový byte poté začíná prefixem 101 nebo 110. Celé adresové byty vypadají následovně: 101CCCC1 nebo 110CCCC1, kde bity C jsou určeny pro identifikaci speciálního příkazu.

3.5.5 Příklady 2 bytových forward zpráv

- 11111110 10010001 (254, 145)

  Příkaz pro všechny *slave* prvky (*broadcast*) pro přechod do úrovně osvětlení 145.

- 00000010 10010001 (2, 145)

  Příkaz pro *slave* prvek se *short adresou* 2 pro přechod do úrovně osvětlení 145.

- 10000010 10010001 (130, 145)

  Příkaz pro 2. skupinu *slave* prvků pro přechod do úrovně osvětlení 145.

- 00000011 10010001 (3, 145)

  Příkaz „QUERY BALLAST“ pro *slave* prvek se *short adresou* 2, pomocí kterého *master* zařízení zjišťuje přítomnost adresovaného *slave* v DALI síti.

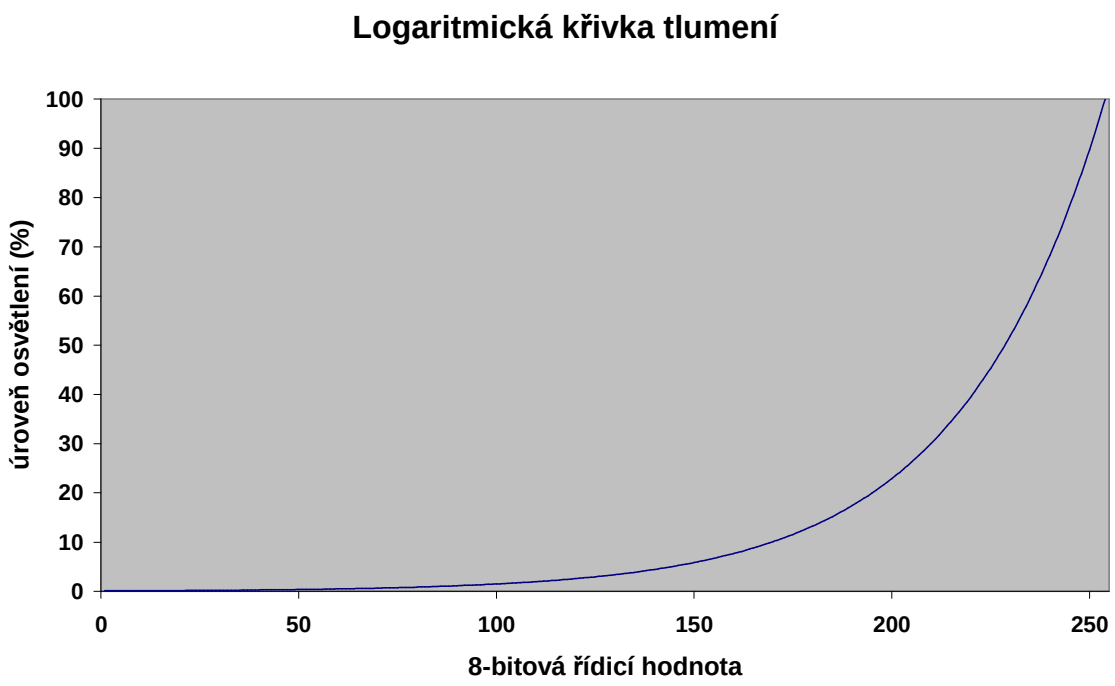
3.6 Typy příkazů ve forward rámci

Existuje několik druhů příkazů, které může příkazový byte obsahovat. Příkazy se liší svou charakteristikou a reakcí *slave* zařízení na ně. Pro úplný seznam všech příkazů, viz Příloha B této diplomové práce.

3.6.1 Příkazy přímo ovlivňující úroveň osvětlení

Je to takový příkaz, jehož hodnota S bitu v adresovém bytu je 0 a v příkazovém bytu je požadovaná úroveň hladiny osvětlení.

Rozsah možných digitálních hodnot požadované hladiny osvětlení je od 1 do 254, přičemž výkon který musí světelné zařízení vyzářit je dán logaritmickou křivkou vyjádřenou vzorcem $X(n) = 10^{\frac{n-1}{253/3}-1}$ s následujícím průběhem:



Obr. 14: Logaritmická křivka tlumení

V příloze C se nachází tabulka s vypočtenými hodnotami výkonu světelného zařízení pro každou digitální hodnotu požadované hladiny osvětlení.

3.6.2 Příkazy nepřímo ovlivňující úroveň osvětlení

Jedná se o příkazy 0 až 31, které nějakým způsobem ovlivňují úroveň osvětlení zvolených *slave* prvků. Příkladem takového příkazu je „UP“ respektive „DOWN“ příkaz pro zvýšení respektive snížení úrovně osvětlení o jeden stupeň.

3.6.3 Konfigurační příkazy

Příkazy s číslem 32 až 143 nastavují chování *slave* prvků, jejich zařazení do skupin a ukládání dat. Příkladem může být příkaz „ADD TO GROUP“, kterým se adresované *slave* zařízení zařazuje do určené skupiny.

Aby se zvýšila pravděpodobnost přijetí konfiguračního příkazu na maximum, musí se během 100ms po prvním přenosu odeslání zopakovat. Během těchto 100ms by neměly být odeslány žádné jiné zprávy adresující stejné *slave* zařízení. Pokud se tak stane, budou tyto zprávy ignorovány.

3.6.4 Dotazové (Query) příkazy

Na příkazy s číslem 144 až 255 očekává *master* zařízení odpověď. Odpověď může být tří druhů:

- ‘ANO’ = 11111111 (*backward rámec* obsahuje samé 1)
- Data – XXXXXXXX (8-bitová informace)
- ‘NE’ (*slave* prvek nereaguje)

Jako příklad použiji opět příkaz “QUERY BALLAST”, na který bude odpověď 11111111 v případě přítomnosti *slave* prvku s odpovídající *short adresou* v DALI síti.

3.6.5 Speciální příkazy

Příkazy 256 až 272 využívají pro svou identifikaci 4 bitů v adresovém bytu a jsou určeny především pro adresační algoritmy popsané v kapitole 3.4. Jako příkladem můžou být příkazy “INITIALISE”, “RANDOMISE” nebo “WITHDRAW” zmiňované také v kapitole 3.4.

Pro příkazy “INITIALISE”, “RANDOMISE” platí stejné pravidlo jako pro všechny konfigurační příkazy a musí být během 100ms po prvním přenosu odeslány znovu.

4 Vývojové prostředky

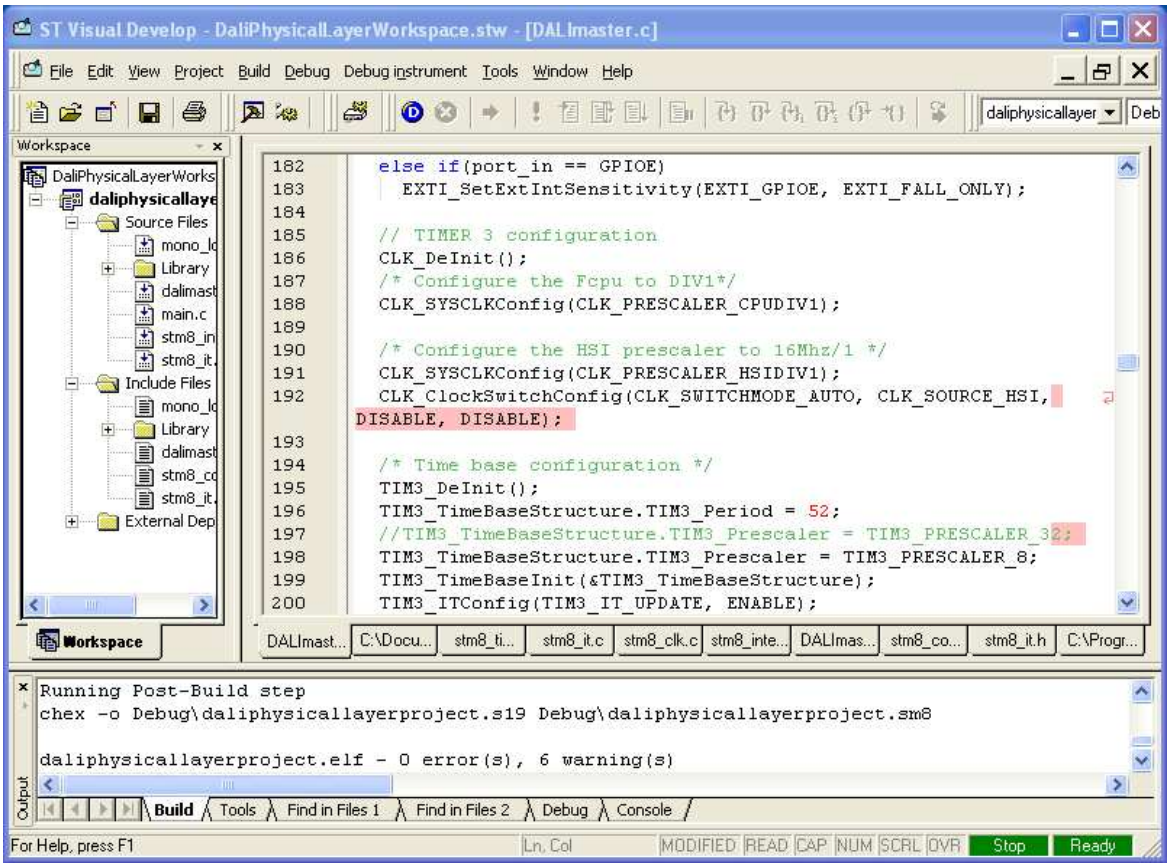
Při vývoji kódu programové knihovny této diplomové práce jsem použil ST Visual Develop vývojové prostředí, provozované na PC s operačním systémem Windows. Pro účely vývoje *Master modulu* byla použita vývojová deska STM8/128 EVAL s 8-bitovým procesorem typu STM8S. Pro spojení vývojové desky a ST Visual Develop byl použit programovací modul Raisonance Rlink. Vývojová deska byla přes proudový omezovač připojena ke dvěma *slave* zařízením, obsahujícím řídicí prvky pro ovládání připojených světelných zařízení, na kterých se ověřovala funkčnost programu. Sestava vývojových prostředků prostředků odpovídá obrázku Obr. **1: Ovládání osvětlení pomocí PC programu** v kapitole 2.

4.1 Vývojové prostředí ST Visual Develop

Prostředí ST Visual Develop obsahuje všechny standardní prostředky pro vývoj profesionálních aplikací pro 8-bitové mikrokontrolery společnosti ST Microelectronics. Pro překlad programu jsou použity compilery Cosmic a Raisonance.

Odlaďování programu

S pomocí programovacího modulu Raisonance Rlink je možné vyvíjený program nahrát do procesoru na vývojové desce a dále jej v ST Visual Develop krokovat, používat zarážky (breakpoints), sledovat stavy proměnných, číst a dokonce i zapisovat do paměti za běhu programu.



Obr. 15: Vývojové prostředí ST Visual Develop

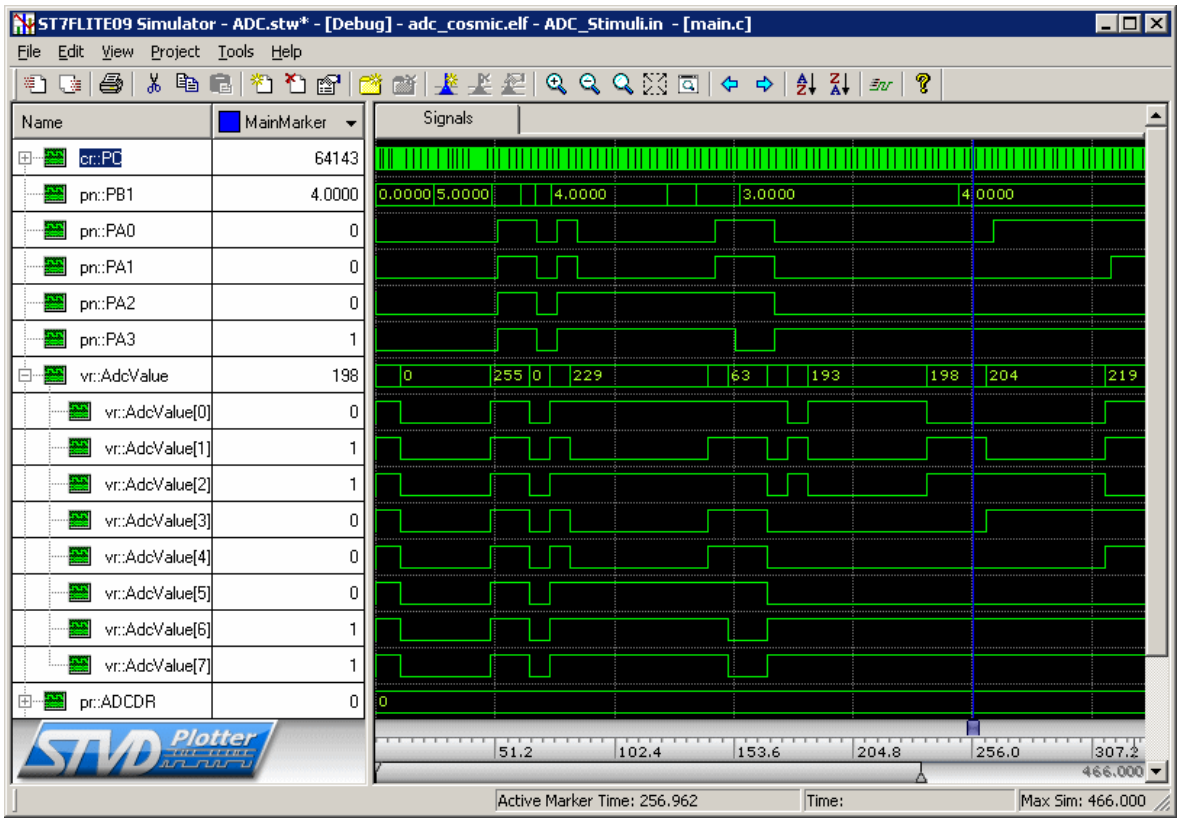
Simulátor

Další zajímavou funkcí je Simulátor, kde můžeme simulovat analogové nebo digitální signály přicházející na vstupní pin simulovaného mikrokontroleru a sledovat odezvu našeho programu na výstupních pinech. Vstupní piny lze budit buďto ručně nebo lze zadat budící funkci ve formě skriptu.

V grafickém okně si zvolíme, které výstupní piny chceme sledovat a jejich odezva na vstupní signál se bude poté vykreslovat v závislosti na časové ose, která je dána počtem cyklů procesoru.

Pro správnou funkci simulátoru není potřeba žádné další hardwarové vybavení.

Při vývoji zadaného úkolu jsem tuto funkci simulátoru nevyužíval, protože jsem již měl k dispozici potřebné hardwarové vybavení, tj. vývojovou desku s mikrokontrolerem STM8.



Obr. 16: Simulátor prostředí ST Visual Develop

4.2 Programovací modul Raisonance Rlink

Modul Rlink je prostředek pro odlaďování a programování ST mikrokontrolerů bez nutnosti vyjmutí procesoru z vývojové desky. Rlink podporuje JTAG, ICC a SWIM protokoly, přičemž pro STM8 vývojovou desku byl použit posledně jmenovaný, SWIM protokol.

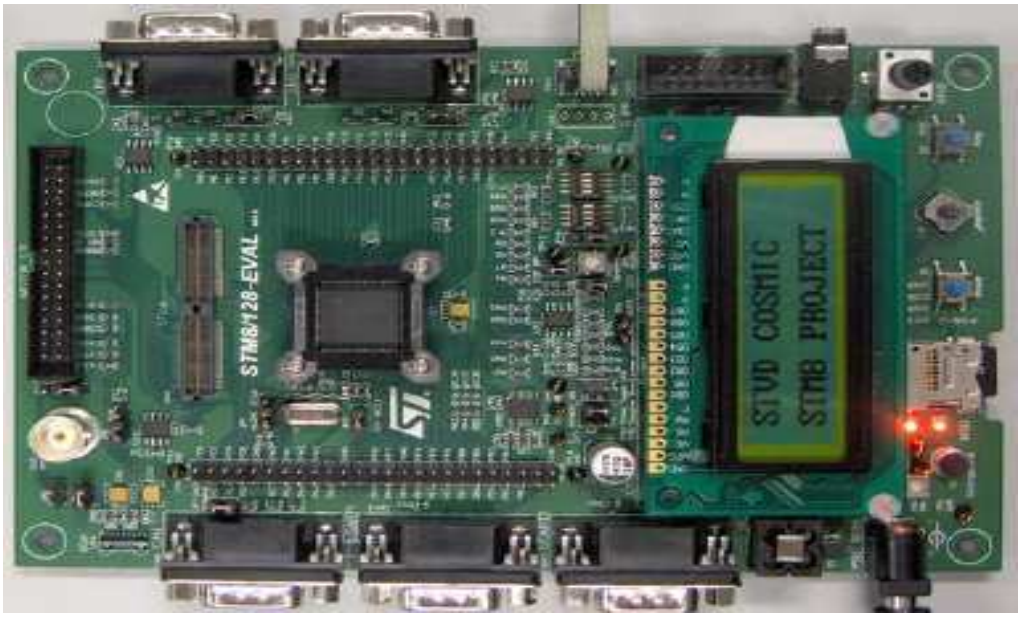
Obr. 17: Raisonance Rlink modul [10]

4.3 STM8 vývojová deska

Použitá vývojová deska STM8/128 EVAL obsahuje 8-bitový procesor STM8S2xx v 80-pinovém pouzdře a dále obvody a konektory pro testování periferií procesoru.

Na desce je možno testovat například LCD displej, I^2C sběrnici, EEPROM paměť, USART rozhraní, IrDA port, LIN a CAN2.0 sběrnice, Audio obvody MicroSD karty a další.

Všechny piny procesoru jsou vyvedeny na lištu pinů pro usnadnění přístupu k jednotlivým pinům a možnost připojení rozšiřovacích obvodů.



Obr. 18: STM8 vývojová deska [10]

4.3.1 Procesor STM8S

Samotný procesor typu STM8S byl uveden na trh v polovině roku 2008. Byl vytvořen jako nástupce velmi oblíbeného a rozšířeného mikrokontroleru ST7, oproti kterému má až 10x větší MIPS (miliony instrukcí za sekundu) výkon a velmi nízkou spotřebu.

Byl vyroben 130nm technologií, je určen především pro automobilový a spotřební průmysl a je velmi vhodný pro bateriově napájené aplikace díky své nízké spotřebě.

<u>Hlavní parametry:</u>

- 130nm technologie
- Harvardská architektura (fyzicky oddělená paměť dat a programu)

- maximální výkon 20MIPS při taktovací frekvenci procesoru 24MHz
- průměrně 1,8 cyklu procesoru na instrukci
- Interní 16Mhz RC oscilátor (HSI s 1,5% přesností)
- Interní 128kHz RC oscilátor (LSI) pro módy s nízkou spotřebou (např. HALT mód)
- 3x16 bitové + 1x 8-bitový timer
- Periferie pro komunikaci (CAN2.0, USART, LIN-UART, SPI, I2C)
- 68 pinů pro obecné použití
- 38 pinů s možností externího přerušení
- 8-128kb Flash paměti
- 32 – 80 pinové pouzdro
- velká odolnost proti šumu

4.4 Proudový omezovač

Normou je dána povinnost galvanicky oddělit DALI sběrnici od pinů mikroprocesoru. K tomu jsou v proudovém omezovači použity optočleny U_4, U_5. Důvodem je nebezpečí mylného připojení napájecích vodičů pro světelné zařízení (síťové napětí), na svorky datové DALI sběrnice a tím zničení mikroprocesoru.

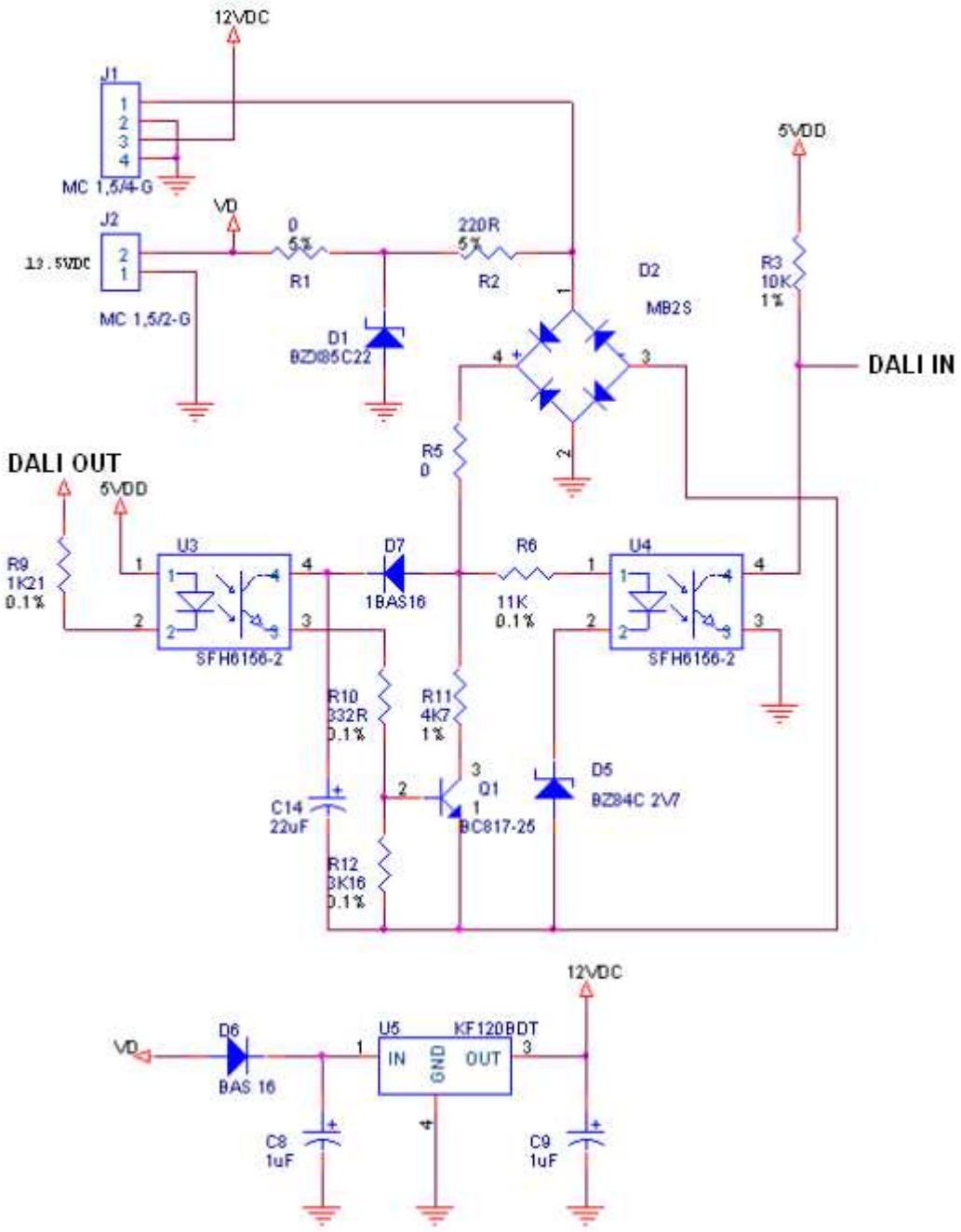
Další podmínkou normy je, že každé DALI zařízení musí snést proud o hodnotě až 250mA, který obvodem protéká když je sběrnice ve stavu low. V případě realizace obvodu galvanického oddělení za použití SMD součástek by byly optočleny, schopné snést tak vysoký proud, příliš drahé. Proto se používá proudový omezovač, podle schématu na obrázku Obr. 19. Optočleny zde fungují jen jako spínače a vlastní proud svede k zemi tranzistor Q1.

Pro udržení sběrnice v úrovni low během přenosu se používá kondenzátor C7, který zachovává napětí na bázi tranzistoru Q1 potřebné k jeho otevření. Proti vybíjení kondenzátoru slouží dioda D6.

Zenerova dioda D5 zajišťuje, že bude dodržena poměrně široká napěťová specifikace DALI normy, to jest úroveň high: 9,5V až 22,5V a low úroveň -6,5V až +6,5V.

Diodový můstek slouží jako ochrana proti přepólování vodičů sběrnice, nedokáže již ale chránit proti přepětí. K tomuto účelu by se musel použít přepěťový omezovač.

Seznam součástek pro stavbu proudového omezovače je uveden v příloze D.



Obr. 19: Schéma proudového omezovače [12]

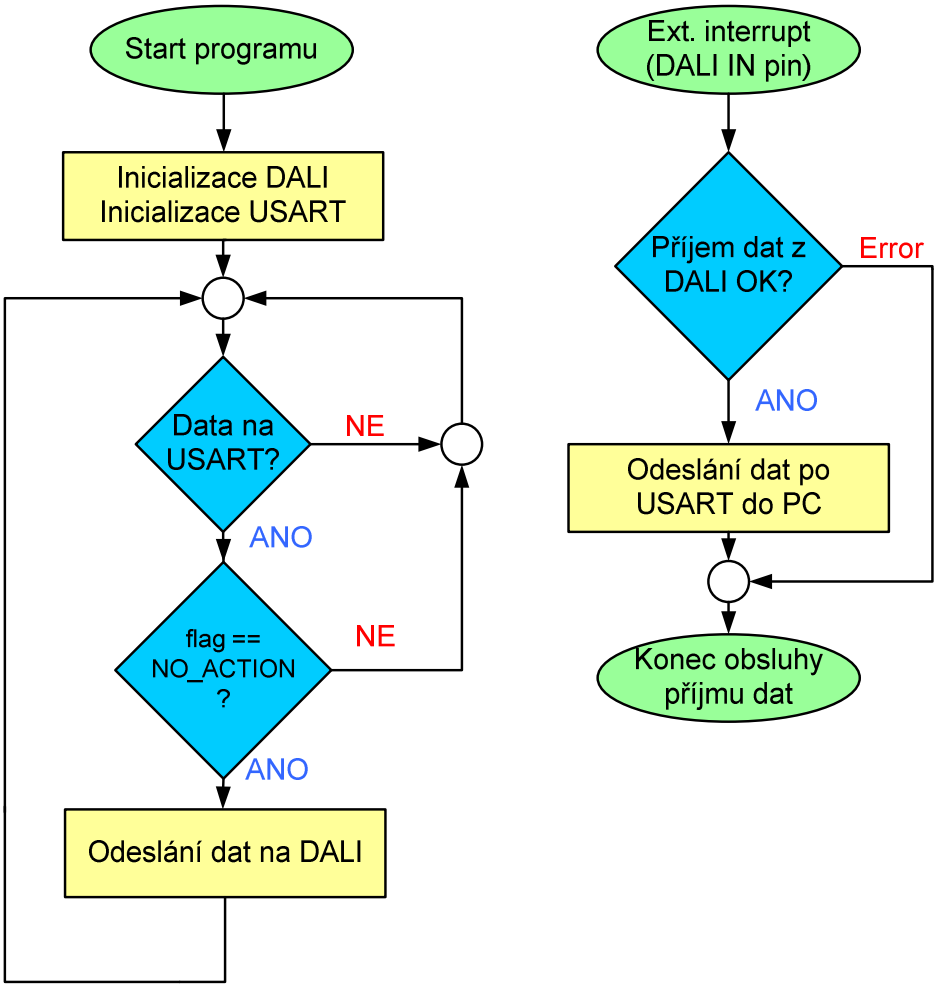
5 Master komunikační modul

Tato kapitola se zabývá popisem programu určeného pro implementaci do *master* zařízení, který jsem vytvořil pro účely této diplomové práce. V tomto případě bude *master* fungovat jako interface mezi příkazy odeslanými z demonstračního programu spuštěného v počítači PC a DALI sběrnicí.

5.1 Princip činnosti

Prvním krokem který program provede po svém startu je inicializace komunikace se sítí DALI a USART komunikace s demonstračním programem na PC. Poté se přepne do klidového stavu a čeká na další příkazy.

Přes sériovou linku jsou z demonstračního programu přijímány 2 byty, kde prvním je adresový byte a druhým je datový byte. Modul tedy nemusí provádět žádnou další konverzi a může přijatá data postoupit funkci pro odesílání dat na DALI sběrnici.



Obr. 20: Hlavní diagram *Master* modulu a diagram příjmu dat

Pouze pokud právě neprobíhá odesílání nebo přijímání dat po DALI sběrnici, je započato odesílání dat, což je indikováno stavem proměnné „flag“. Po odeslání dat *master* povolí možnost externích přerušení a tím se připraví na možnost příjmu případné odpovědi.

V případě indikace spádové hrany na vstupním DALI pinu je zahájeno přijímání dat. Modul je schopen kontrolovat několik parametrů přijímaného signálu, například předepsané doby trvání jednotlivých bitů a zajistit tak správnost přijatých dat. V případě konfliktu ohlásí error a ukončí přijímání.

Poté co jsou data z DALI sběrnice přijata, začíná odesílání odpovědi do PC, kde ji příslušně zpracovává demonstrační program.

5.2 Popis funkcí a stavových proměnných

V této kapitole jsou popisovány funkce a stavové proměnné programové knihovny, které jsem naprogramoval pro účely této diplomové práce. Použité funkce je nutné stručně popsat, včetně významu jejich parametrů, aby byl výklad principu a fungování celého modulu v následujících kapitolách srozumitelný.

5.2.1 Rozdělení do souborů

Pro přehlednost a správnou funkci jsou výkonné části celého programu rozděleny do třech souborů. Jsou to main.c, it.c a dalimaster.c

5.2.1.1 Soubor main.c

V tomto souboru jsou napsány především inicializační procedury a probíhá zde hlavní cyklus mikroprocesoru. Tento cyklus je ovšem prázdný, protože všechny výkonné operace jako například příjem dat z USART nebo spuštění příjmu dat z DALI sběrnice jsou řešeny pomocí přerušení.

5.2.1.2 Soubor it.c

V tomto souboru jsou obsluhována veškerá přerušení. V programu je nejčastěji použito přerušení přetečení čítače timer3, pomocí kterého se časuje průběh DALI datového signálu (emulace fyzické vrstvy). Další dvě používaná přerušení jsou pro příjem a odesílání dat přes sériovou linku USART.

5.2.1.3 Soubor dalimaster.c

Soubor dalimaster.c byl vytvořen jako místo pro uložení všech procedur týkajících se operací na DALI sběrnici. Hlavní funkcí této programové knihovny je možnost odesílání a přijímání dat po sběrnici DALI. Ostatní funkce slouží jako podpora dosažení tohoto cíle.

5.2.2 Seznam vytvořených funkcí

Tabulka uvádí funkce programové knihovny vytvořené pro modul *Master*

Definice funkcí	Umístění
void **init_DALI**(GPIO_TypeDef* port_out,GPIO_Pin_TypeDef pin_out, GPIO_TypeDef* port_in, GPIO_Pin_TypeDef pin_in);	dalimaster.c
void **send_data**(u8 addrByte, u8 cmdByte, u8 dataByte, u16 priority, u8 BytesToSend);	dalimaster.c
void **send_tick**(void);	dalimaster.c
void **receive_data**(void);	dalimaster.c
void **receive_tick**(void);	dalimaster.c
u8 **get_flag**(void);	dalimaster.c
BoolT **CollisionDet**(void);	dalimaster.c
void **Repeat_after_CD**(void);	dalimaster.c
void **enable_EXTI**(void);	dalimaster.c
void **disable_EXTI**(void);	dalimaster.c
BoolT **set_DALIOUT**(BoolT pin_value);	dalimaster.c
BoolT **get_DALIOUT**(void);	dalimaster.c
BoolT **get_DALIIN**(void);	dalimaster.c
void **DataReceived**(u8 answer,u16 answerExt, u8 bit_count);	stm8_it.c
void **set_timer_delay**(u16 settling_time);	stm8_it.c

Tabulka 1: Seznam funkcí modulu *Master*

5.2.3 Funkce volané uživatelem

Tyto funkce musí volat a používat sám uživatel této programové knihovny.

5.2.3.1 init_DALI (dalimaster.c)

Funkce se nachází v souboru dalimaster.c a bez jejího předchozího volání nebude možné používat DALI komunikaci. Parametry této funkce jsou vstupní port a pin a výstupní port a pin, které uživatel použije pro DALI komunikaci.

Pin pro vstup dat musí navíc patřit do rozsahu portů A až E, protože ostatní porty neumožňují využít externího přerušení.

Pro časování je použit timer3. Timer3 se tedy v případě využití této programové knihovny nesmí používat k jiným účelům, ani přenastavovat jeho parametry.

5.2.3.2 get_flag (dalimaster.c)

Procedura get_flag vrací 8-bitovou hodnotu stavové proměnné flag, která vyjadřuje aktuální stav *Master modulu*. Uživateli se hodí především pro čekání na stav „NO_ACTION“, kdy bude moci bezpečně odeslat další zprávu na DALI sběrnici.

Proměnná flag může nabývat následujících hodnot:

Název	Číselná hodnota	Význam
NO_ACTION	0	Právě není prováděna žádná operace
SENDING_DATA	1	Odesílání dat na DALI sběrnici
RECEIVING_DATA	2	Přijímání dat z DALI sběrnice
REPEAT_SEND	3	Opakované odesílání dat na DALI sběrnici
ERR	4	Detekován error
CD_ERR	5	Detekována kolize

Tabulka 2: Možné hodnoty proměnné flag (*Master modul*)

5.2.3.3 send_data (dalimaster.c)

Tato procedura má na starosti veškeré operace spojené s odesláním DALI rámce na sběrnici. Měla by být volána jen v případech, že hodnota proměnné flag se rovná „NO_ACTION“, to znamená že se právě neprovádí žádná jiná operace, kterou by toto volání narušilo.

<u>2-bytový nebo 3-bytový rámec:</u>

Z prototypu funkce uvedené v Tabulce Tabulka **1: Seznam funkcí modulu *Master*** je zřejmé, že stejná funkce se používá jak pro odeslání 2 bytové zprávy, tak pro odeslání rozšířeného (Extended) rámce, složeného ze 3 bytů. Při hodnotě parametru 1 se odesílá 1 bytová zpráva, kterou *master* odpovídá na dotaz typu „Query“ od jiného *master* zařízení. Pro 2 bytový rámec je hodnota parametru 2, pro rozšířený rámec je BytesToSend rovno 3.

V případě odesílání 2 bytového rámce, může být hodnota parametru dataByte libovolná. V případě odesílání 1 bytové zprávy, může být hodnota parametrů cmdByte a dataByte libovolná.

Priorita rámce:

Dalším parametrem, který musí uživatel zadat je „priority“, kterým se určuje doba prodlevy mezi odeslanými rámci. Délky prodlev příslušné jednotlivým prioritám jsou popsány v kapitole 3.3.3. Parametr priority může tedy nabývat hodnot „PRIORITY1“ až „PRIORITY5“.

Dvojnásobné odeslání:

Ještě před startem přenosu se zjišťuje jestli není nutné zadaný příkaz odeslat dvakrát. Dvojnásobné odeslání zprávy je nutné u konfiguračních příkazů (příkazový byte 32 – 143) a u příkazů “INITIALISE”a “RANDOMISE”.

Start přenosu:

Po nastavení délky rámce podle hodnoty parametru BytesToSend se po dobu odesílání rámce vypne možnost externího přerušení a zapne se timer3. V obsluze přerušení přetečení čítače timer3 se bude periodicky volat procedura send_tick, která se stará o samotné odeslání dat ve správném formátu.

5.2.4 Vnitřně volané funkce

Vnitřně volané funkce jsou takové, které přímo nevolá sám uživatel, ale jsou zavolány nadřazeným programem. Nadřazený program spustil buďto uživatel nebo byl vyvolán přerušením.

5.2.4.1 receive_data (dalimaster.c)

Tato metoda je volána z obsluhy externího přerušení vstupního pinu určeného pro DALI komunikaci.

Její hlavní funkcí je inicializace proměnných pro příjem dat a zapnutí timer3, který se bude starat o periodické volání procedury receive_tick pro příjem DALI signálu.

5.2.4.2 receive_tick (dalimaster.c)

Tato metoda se volá při každém přetečení čítače timer3, pokud je hodnota stavové proměnné flag nastavena na „RECEIVING_DATA“. Perioda timer3 musí být dostatečně rychlá, aby se dosáhlo správného vzorkování, ale zároveň nesmí přijímání DALI dat příliš zatěžovat procesor, aby bylo možné provádět ještě další operace.

Master modul je schopen přijímat jak 1 bytové zprávy, které slouží k příjmu odpovědi na příkazy typu „Query“ tak 3 bytové zprávy odeslané jinými *master* zařízeními.

V případě mojí implementace se po přijetí zprávy zavolá funkce DataReceived, která odešle přijatá data do demonstračního programu po sériové lince USART.

5.2.4.3 send_tick (dalimaster.c)

Procedura send_tick se také volá pomocí přetečení čítače timeru3, ale hodnota stavové proměnné flag musí mít hodnotu „SENDING_DATA“ nebo „REPEAT_SEND“.

Samotná funkce send_tick již vytváří odpovídající bi-fázový formát protokolu. Signál je poté přes výstupní DALI pin mikroprocesoru odesílán do proudového omezovače a tak na sběrnici.

Dalším úkolem této metody je zajistit detekci kolizí. V případě zjištění kolize se přenos ukončí. Poté se čeká po dobu, která je definována délkou prodlevy pro příslušnou prioritu příkazu. Po uplynutí doby prodlevy se celý přenos zopakuje.

5.2.4.4 CollisionDet (dalimaster.c)

Volá se před plánovanou změnou stavu sběrnice z vysoké do nízké úrovně. Funkce vrátí hodnotu 0, pokud se sběrnice již nachází v nízké úrovni a došlo tak tedy ke kolizi. Pokud se vrácená hodnota rovná 1, je sběrnice stále ve vysokém stavu a ke kolizi nedošlo.

5.2.4.5 set_timer_delay (it.c)

Nastaví jak dlouhá bude doba prodlevy po zjištění kolize a tedy před možností přenos zopakovat. Doba prodlevy se určuje podle priority příkazu, který byl odesílán než došlo ke kolizi.

5.2.4.6 Repeat_after_CD (dalimaster.c)

Zavolá send_data se stejnými parametry, s jakými byla tato procedura spuštěna než došlo ke kolizi a tím umožní opakování přenosu od začátku.

5.2.4.7 enable_EXTI (dalimaster.c)

Povolí možnost externích přerušení na portu, který byl vybrán pro vstup DALI signálu.

5.2.4.8 disable_EXTI (dalimaster.c)

Zakáže možnost externích přerušení na portu, který byl vybrán pro vstup DALI signálu.

5.2.4.9 set_DALIOUT (dalimaster.c)

Nastaví výstupní pin pro DALI signál a tím i úroveň DALI sběrnice do stavu daného parametrem funkce set_DALIOUT. Parametr může nabývat hodnot 0 nebo 1.

5.2.4.10 get_DALIOUT (dalimaster.c)

Vrátí logickou hodnotu pinu určeného pro DALI výstupní signál.

5.2.4.11 get_DALIIN (dalimaster.c)

Vrátí logickou hodnotu pinu určeného pro DALI vstupní signál.

5.2.4.12 DataReceived (it.c)

Tato funkce se zavolá po úspěšném přijetí DALI zprávy a povolí odeslání přijatých dat demonstračnímu programu na PC. Samotné odeslání dat do PC se provede při přerušení vyvolaném USART odesílacím modulem.

Parametry funkce jsou přijatá data a počet přijatých bitů, aby se odlišilo kdy došlo k přijetí 1bytové a kdy 3bytové zprávy.

5.3 Princip přenosu dat v modulu Master

Tato kapitola je zaměřena na popis činností, které musí software vykonávat při odesílání nebo přijímání dat.

5.3.1 Časování pomocí timer3

Jak odesílání tak přijímání dat je v *Master modulu* časováno pomocí timer3. Hodnota předděličky a periody timer3 je nastavena tak, aby docházelo k přerušení přetečením čítače každých 13µs. Tato hodnota se vypočte podle následujícího vzorce:

$$T_{\text{int}\,errupt} = \frac{Přředděliča * \text{Perioda}}{f_{CLK}}$$

f_{CLK} je ovlivněno dalším parametrem, kterým je předdělička HSI hodinového signálu. Ta musí být rovna jedné a její nastavení se provádí v proceduře init_DALI.

Hodnotu HSI předděličky nesmí uživatel změnit, jinak nebude přenos DALI dat správně fungovat.

Pro nás je hodnota f_{CLK} rovna frekvenci HSI oscilátoru, tj. f_{CLK} =16MHz.

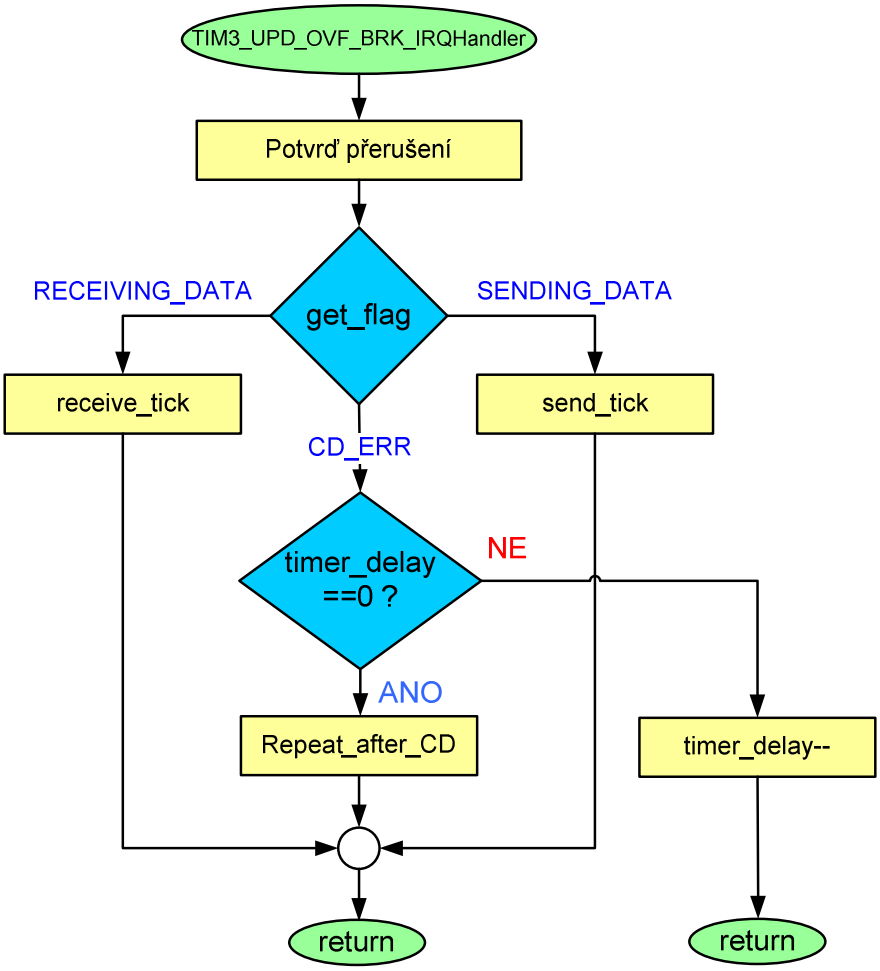
Hodnotu předděličky jsem zvolil 4 a periody 52. Potom výsledná perioda:

$$T_{\text{int}\,errupt} = \frac{4*52}{16\cdot 10^{6}\,Hz} = 13\cdot 10^{-6}\,s = 13\mu s$$

Pokaždé když dojde k přerušení, zjistí se hodnota stavové proměnné flag a podle toho jestli se právě data přijímají nebo vysílají se spustí procedura receive_tick nebo send_tick.

Hodnota CD_ERR je proměnné flag přiřazena po zjištění kolize na DALI sběrnici. Algoritmus nejdříve čeká zadanou dobu, která je úměrná prioritě odesílaného rámce před kolizí. Po uplynutí této doby, která je dána postupnou dekrementací proměnné timer_delay do nuly, je znovu odeslána původní zpráva.

Činnosti při přerušení timer3 shrnuje následující diagram.



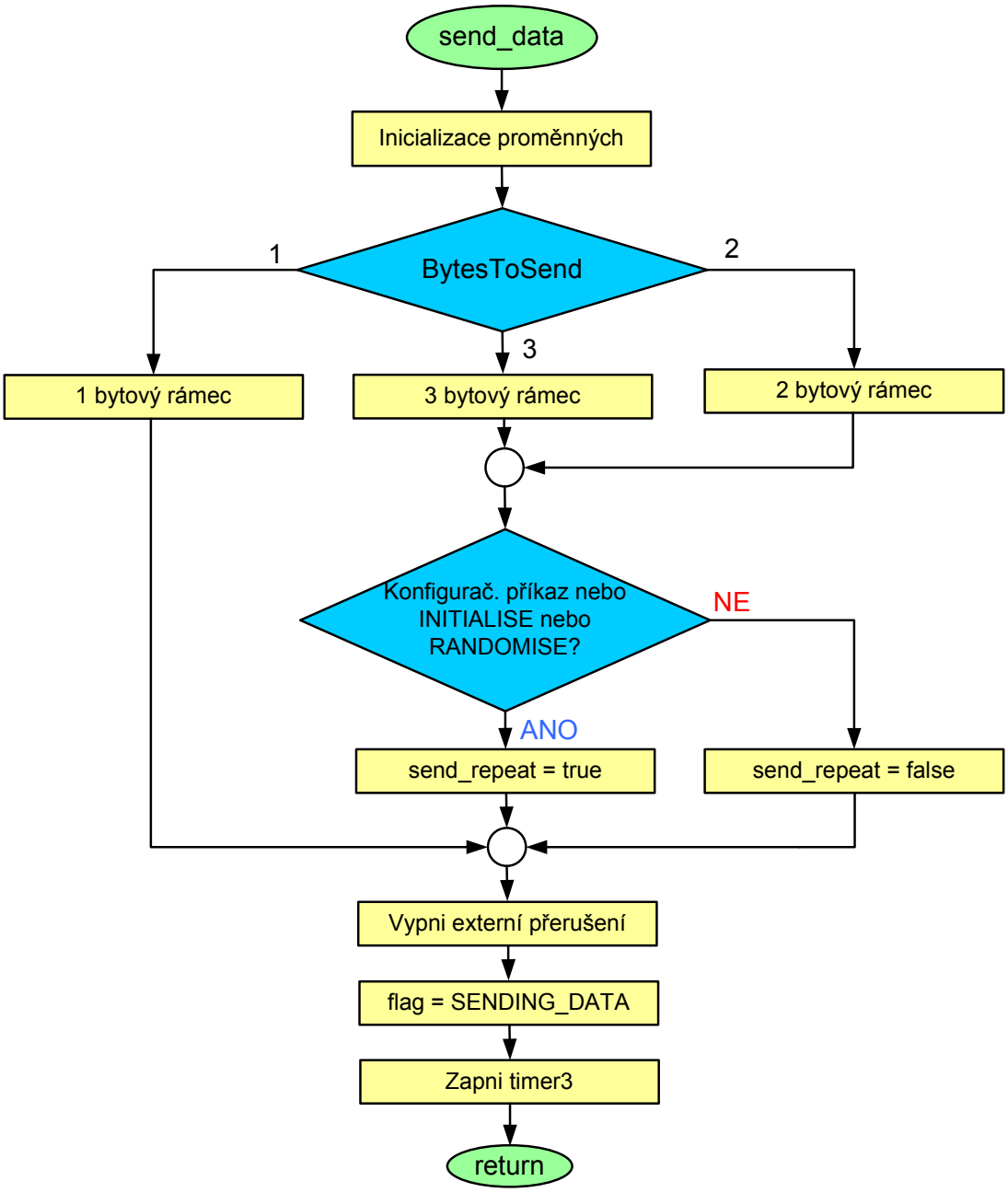
Obr. 21: Diagram obsluhy přerušení timer3 (*Master modul*)

5.3.2 Odeslání dat pomocí send_data funkce

Metoda send_data je volána uživatelem programové knihovny s parametry, které byly pospány v kapitole 5.2.3.3

Po inicializaci proměnných se algoritmus podle parametru BytesToSend rozhodne jestli půjde o odeslání 1 nebo 2 nebo 3 bytové zprávy. Podle toho nastaví hodnotu proměnné frame_length, která udává délku odesílaného rámce, bez stop bitů a doby ustálení, v ticích. Jeden tik je doba přetečení čítače timer3, v našem případě 13µs.

Další akcí je zjištění jestli rámec nebude nutné odeslat dvakrát. To je nutné pro konfigurační příkazy a příkazy „INITIALISE“ a „RANDOMISE“.



Obr. 22: Diagram funkce send_data *Master modulu*

Posledními kroky je vypnutí možnosti externího přerušení (nelze přijímat data v průběhu odesílání dat), nastavení stavové proměnné flag na „SENDING_DATA“ a konečně zapnutí timer3, který bude periodicky volat funkci send_tick a vytvářet tak příslušný bi-fázový signál DALI protokolu.

5.3.3 Periodické volání send_tick

Metoda send_tick se volá periodicky s každým přetečením (přerušením) čítače timer3. Počet těchto volání je ukládán v proměnné tick_count, která se s každým voláním inkrementuje a podle které se dále řídí časování průběhu signálu.

Dovnitř hlavní části funkce send_tick se vstupuje každých 32 tiků, to znamená každých 32*13 µs = 416 µs, což je doba trvání jednoho půlbitu.

Detekce kolizí

Uprostřed každého půlbitu vysoké úrovně se musí kontrolovat jestli nedošlo ke kolizi. Do středu půlbitu se tedy dostáváme každých 32/2=16 tiků.

Kolize se také musí vyloučit do 26 µs po změně úrovně sběrnice z nízké do vysoké úrovně. Tuto změnu indikuje proměnná Edge_to_high a kolize je díky ní zkontrolována hned první tick po přechodu sběrnice do vysoké úrovně.

V normě je ještě definována nutnost detekce kolizí před změnou sběrnice z vysoké do nízké úrovně. Tato kontrola se provádí přímo v proceduře set_DALIOUT pokaždé když je volána s parametrem logické úrovně 0.

Po detekci kolize se pomocí volání funkce set_Timer_delay nastaví počet tiků, které je nutné čekat, než bude možné odeslání zprávy zopakovat. Tato doba je rovna době ustálení, která je dána prioritou odesílané zprávy.

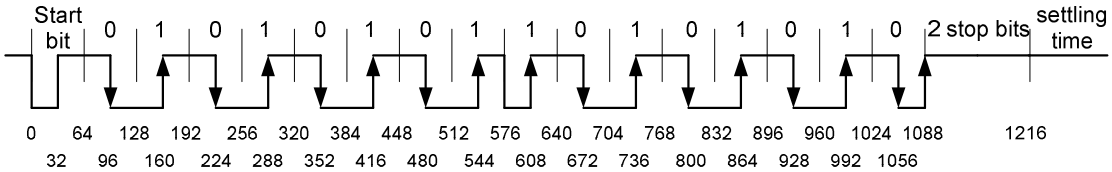
Start bit

Pokud je proměnná tick_count nulová, algoritmus vygeneruje spádovou hranu, která má za úkol připravit adresované zařízení na příjem DALI dat.

Některá zařízení mohou být uvedena do klidového režimu s nízkou spotřebou a tato hrana má za úkol je uvést do aktivního stavu. Dalších 32 tiků poskytuje dostatečnou rezervu, aby se zařízení připravila na příjem DALI signálu a je vygenerována platná hrana start bitu.

Fázová a bitová hrana

Následující obrázek znázorňuje průběh 2-bytového DALI rámce s označenými hodnotami bitů a očíslovanou časovou osou. Jednotky časové osy jsou počty tiků.



Obr. 23: Průběh signálu DALI rámce

Fázová hrana se vyskytuje v polovině bitu, tedy každých 64 tiků. Má za úkol převést úroveň sběrnice do takového stavu, aby mohla být vytvořena a především adresovaným zařízením indikována bitová hrana.

Fázovou hranu je nutné vytvořit, pokud dva po sobě jdoucí bity mají stejnou logickou hodnotu (například logické 1). Pokud bychom nepoužívali fázovou hranu, mezi dvěma takovými bity by nevznikla žádná bitová hrana, protože sběrnice by již byla ve vysoké úrovni od doby předchozího bitu. Příklad fázové hrany je vidět v průběhu signálu DALI rámce u časové značky 576 tiků.

Směr bitové hrany značí logickou hodnotu aktuálního bitu. V bi-fázovém kódování musí mít každý bit svou bitovou hranu. V signálu vidíme bitové hrany pokud od proměnné tick_count odečteme offset 96 tiků (délka spádové hrany a start bitu) a poté každých 64 tiků.

Délky 1, 2 a 3-bytových rámců

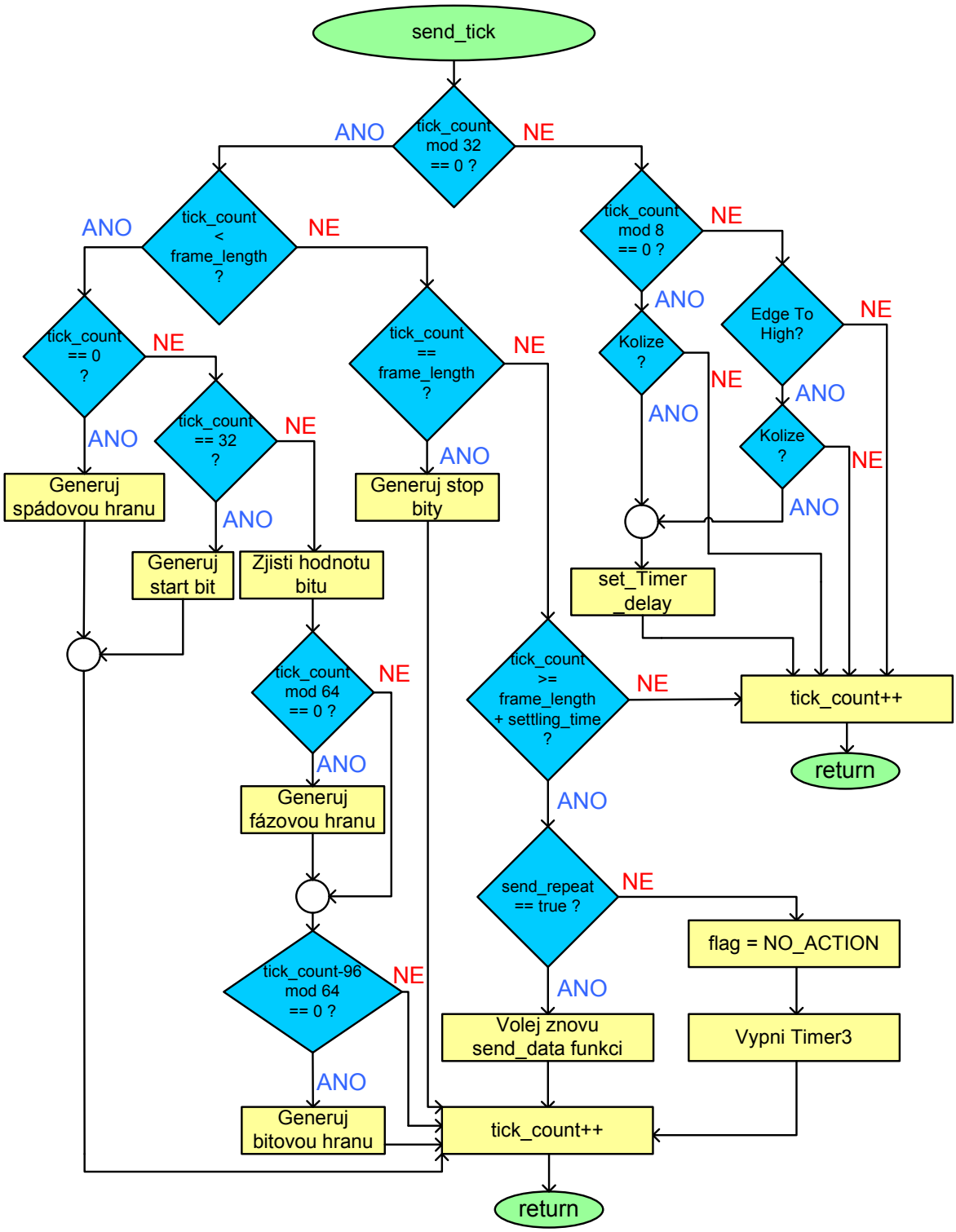
Ve funkci send_data se nastavuje proměnná frame_length podle toho jestli jde o odesílání 1, 2 nebo 3bytového rámce. V našem ukázkovém případě máme 2bytový rámec a jeho délka je 1088 tiků. Pro 1 bytový rámec je délka 576 a pro 3 bytový rámec 1600 tiků.

Stop bity

Potom co je detekována rovnost mezi proměnnou tick_count a délkou rámce frame_length, je vygenerována hrana stop bitu. Normou je určeno, že ačkoliv jsou oba stop bity logické hodnoty 1, není mezi nimi žádná změna fáze, to znamená žádná fázová ani bitová hrana.

Doba ustálení

Po uběhnutí doby určené pro stop bity se čeká po dobu ustálení, při které nejsou prováděny žádné další činnosti. Délka doby ustálení je dána prioritou odesílané zprávy a tato délka je vyjádřená počtem tiků uložených v proměnné settling_time.



Obr. 24: Vývojový diagram metody send_tick (*Master modul*)

Opakované odeslání

Po skončení doby ustálení je odeslání celé zprávy dokončeno a algoritmus zkontroluje proměnnou send_repeat. Pokud je její hodnota rovna true, znamená to že se odeslání celé zprávy musí zopakovat a zavolá se procedura send_data se stejnými parametry jako byly použity pro právě odeslanou zprávu.

Ukončení přenosu

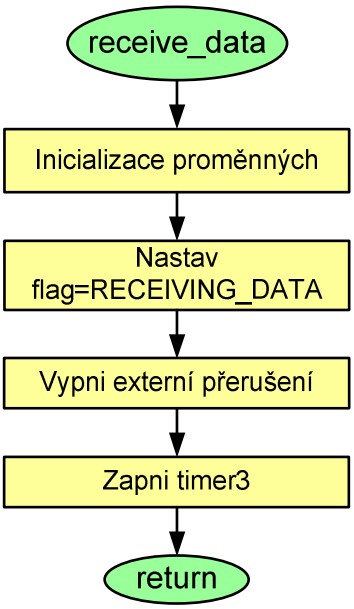
V případě rovnosti proměnné send_repeat hodnotě false se nastaví proměnná flag na NO_ACTION a vypne se časování pomocí timer3. Tím se ukončí odesílání celé zprávy.

5.3.4 Přijímání dat pomocí receive_data funkce

Metoda receive_data je spuštěna z obsluhy externího přerušení, které je vyvoláno příchodem spádové hrany na pin mikroprocesoru, který je určen k přijímání DALI dat.

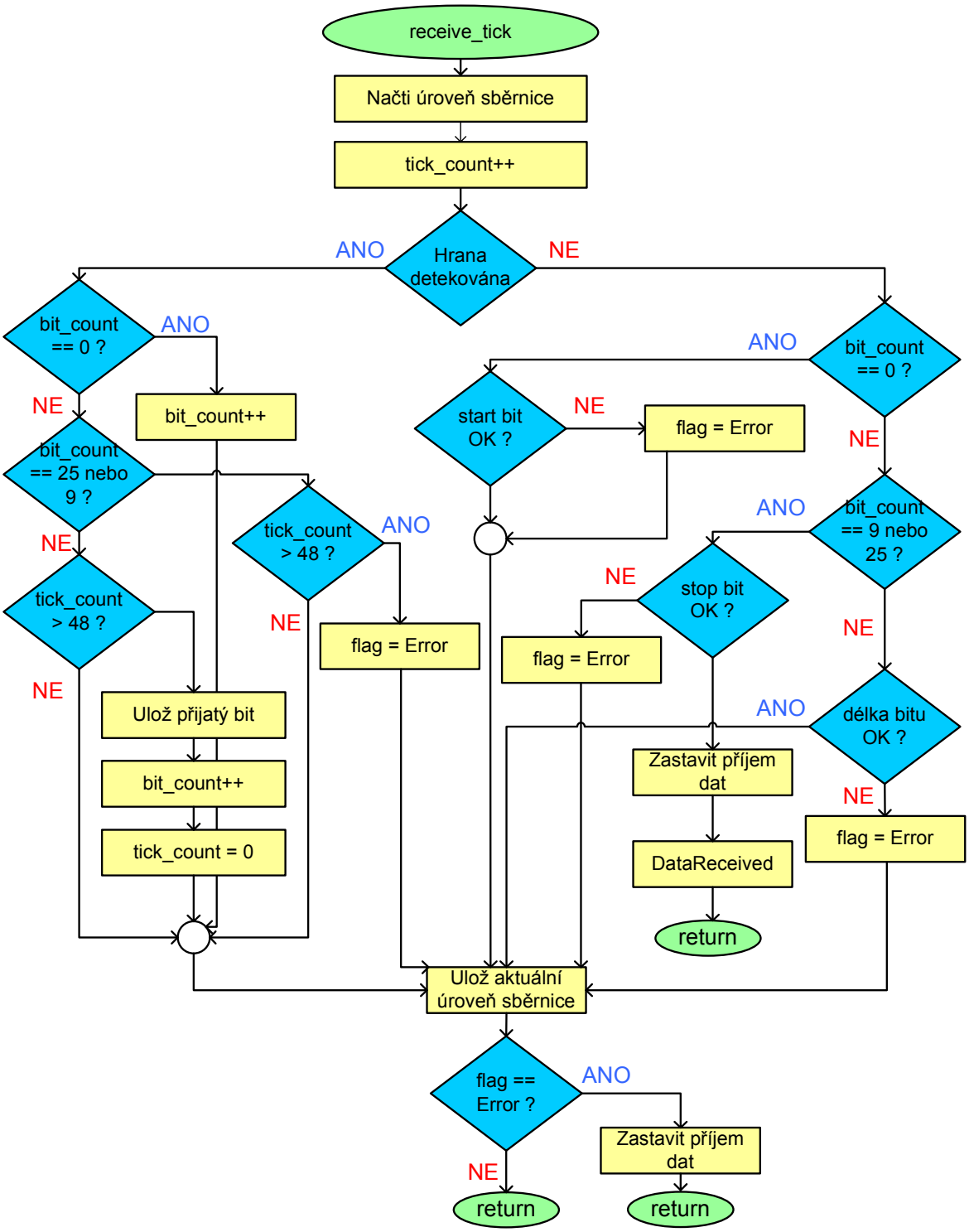
Procedura inicializuje proměnné pro příjem dat, vypne možnost dalšího externího přerušení v průběhu přijímání, nastaví stavovou proměnou flag na RECEIVING_DATA a především zapne časování pomocí timeru3.

Timer3 poté, pomocí přerušení, periodicky volá funkci receive_tick, která má na starosti příjem samotného bi-fázového DALI signálu.



Obr. 25: Vývojový diagram funkce receive_data (*Master modul*)

5.3.5 Periodické volání receive_tick



Obr. 26: Vývojový diagram funkce receive_tick (*Master modul*)

Úkolem procedury receive_tick je nejen správně přijmout, zpracovat a odeslat do PC přijatou DALI zprávu, ale také v průběhu přijímání kontrolovat jestli zpráva není

porušena. Tím se myslí například jestli má správné délky bitů, nebo jestli se během přijímání stop bitů nevyskytuje hrana.

Detekce hrany:

Na začátku každého volání procedury receive_tick se načte hodnota vstupního DALI pinu, která udává úroveň DALI sběrnice. Tato hodnota se srovná s úrovní sběrnice v minulé iteraci procedury, která je uložena v proměnné former_val. Pokud jsou tyto dvě hodnoty rozdílné, byla detekována hrana, pokud jsou stejné, je úroveň na sběrnici stabilní.

Hrana detekována:

Pokud byla v signálu detekována hrana, musí algoritmus rozlišit o který bit se jednalo a v případě že šlo o datový bit, uložit takový do k tomu určené proměnné.

Počet již přijatých bitů je uložen v proměnné bit_count. Počty tiků se v této proceduře, narozdíl od procedury send_tick počítají jen pro aktuální bit a po dokončení příjmu bitu je proměnná tick_count vynulována.

Přijaté bity se v případě 8bitové zprávy postupně skládají do proměnné answer. V případě přijímání 3 bytové zprávy se poslední dva byty uloží do 16ti bitové proměnné answerExt.

Pokud má proměnná bit_count již hodnotu 9 (1 bytová zpráva) nebo 25 (3 bytová zpráva), začne se s přijímáním stop bitů. Pokud je v časovém úseku vymezeném pro stop bity detekována hrana, je to vyhodnoceno jako chyba přijímaných dat. Podle normy se v průběhu stop bitů nesmí žádná hrana vyskytovat.

Úroveň sběrnice stabilní:

Tato část algoritmu má za úkol především hlídat správné délky bitů. V případě překročení normou definované délky bitu je do proměnné flag uložena hodnota ERR.

Po přijetí 8 (bit_count = 9) nebo 24 (bit_count = 25) bitů se odešlou přijatá data proceduře DataReceived. Jako parametr je jí předána jak proměnná answer tak answerExt. Počet skutečně přijatých bitů, které jsou určeny k odeslání do PC demonstračnímu programu je odlišen hodnotou v proměnné bit_count-1, která je dalším parametrem funkce DataReceived.

Zastavení příjmu dat

Po odeslání přijatých dat do PC se přijímání zastaví, stavové proměnné flag se přiřadí hodnota NO_ACTION a vypne se časování pomocí timer3. Možnost externího přerušení na vstupním DALI pinu se naopak zapne, aby bylo možné přijímat další DALI zprávy.

Error:

Na konci každé iterace procedury receive_tick se testuje hodnota stavové proměnné flag. Pokud je její hodnota rovna „ERR“, ukončí se přijímání dat stejným způsobem jako je tomu po úspěšném přijetí dat. To znamená, že stavové proměnné flag se přiřadí hodnota NO_ACTION, zastaví se časování pomocí timer3 a povolí se možnost externího přerušení na vstupním DALI pinu.

6 Slave komunikační modul

Tato kapitola se zabývá popisem programu určeného pro implementaci do *slave* zařízení, který jsem vytvořil pro účely této diplomové práce. Principy programu jsou v mnoha ohledech velmi podobné principům *Master modulu*, proto se na jeho popis budu často odvolávat.

Tento modul má za úkol umožnit oboustrannou komunikaci mezi *master* a *slave* modulem. Nemá však již za úkol řídit samotné světelné zařízení ani provádět žádné operace s tím spojené.

6.1 Princip činnosti

Narozdíl od *Master modulu*, *slave* zařízení nekomunikuje po sériové lince, jedinou jeho komunikační sběrnicí je DALI. *Slave* také sám neinicializuje odesílání dat, jediná chvíle kdy odesílá data je v případě odpovídání na dotazový příkaz od *master* zařízení.

Rozdíly oproti *Master modulu* jsou především v délce přijímaných zpráv. *Slave modul* je schopen přijmout pouze 2 bytové zprávy a odpovídat zprávou s 1 datovým bytem. Zprávy jiné velikosti nejsou zpracovány a při jejich příjmu je modul vyhodnotí jako poškozené.

Slave modul také neobsahuje detekci kolizí a metody s ní spojené. Dále také nerozlišuje mezi různými prioritami zpráv a odesílá případnou odpověď vždy po 2,92ms po obdržení *forward rámce*.

6.2 Popis funkcí a stavových proměnných

Funkce a stavové proměnné *modulu Slave* se příliš neliší od těch použitých pro *modul Master*, přesto se zde vyskytují důležité změny, například v parametrech předávaných funkcím.

6.2.1 Rozdělení do souborů

Rozdělení volaných funkcí do jednotlivých souborů je velmi podobné jako v *modulu Master*.

V souboru main.c je umístěno pouze volání inicializace DALI komunikace. Dále sem můžou být umístěny procedury obsluhující řízení světelného zařízení nebo jiné procedury, které bude mít *slave* prvek za úkol vykonávat.

Soubor it.c je určen výhradně pro metody obsluhující přerušení. V tomto případě půjde jen o obsluhu přerušení daného přetečením čítače timer3. Přerušení od sériové komunikace zde není potřeba.

Soubor dalislave.c obsahuje všechny procedury, které jsou použity pro přijímání nebo odesílání dat na DALI sběrnici.

6.2.2 Seznam vytvořených funkcí

Tabulka 3 uvádí seznam funkcí vytvořených pro *modul Slave*

Definice funkcí	Umístění
void **init_DALI**(GPIO_TypeDef* port_out,GPIO_Pin_TypeDef pin_out, GPIO_TypeDef* port_in, GPIO_Pin_TypeDef pin_in);	dalislave.c
void **send_data**(u8 byteToSend);	dalislave.c
void **send_tick**(void);	dalislave.c
void **receive_data**(void);	dalislave.c
void **receive_tick**(void);	dalislave.c
u8 **get_flag**(void);	dalislave.c
void **enable_EXTI**(void);	dalislave.c
void **disable_EXTI**(void);	dalislave.c
BoolT **set_DALIOUT**(BoolT pin_value);	dalislave.c
BoolT **get_DALIOUT**(void);	dalislave.c
BoolT **get_DALIIN**(void);	dalislave.c
void **DataReceived**(u8 address, u8 dataByte)	stm8_it.c

Tabulka 3: Seznam funkcí modulu *Slave*

6.2.3 Funkce volané uživatelem

Tyto funkce musí volat a používat sám uživatel této programové knihovny.

6.2.3.1 init_DALI (dalislave.c)

Funkce init_DALI i zde slouží k inicializaci DALI komunikace a musí být spuštěna před jakýmkoliv použitím odesílacích nebo přijímacích funkcí DALI *Slave modulu*.

Opět se k časování používá timer3 a uživatel si má možnost vybrat vstupní a výstupní porty a piny pro DALI komunikaci. K dispozici jsou porty A-E, na ostatních portech není možné aktivovat externí přerušení nutné pro správný příjem dat.

6.2.3.2 get_flag (dalislave.c)

Tato funkce vrací 8-bitovou hodnotu stavové proměnné flag a i zde vyjadřuje aktuální stav *Slave modulu*.

Uživatel programové knihovny ji bude nejvíce využívat při čekání na stav „NO_ACTION“, kdy jsou všechny operace spojené s odesíláním a přijímáním dat dokončené a může začít přenos dalších dat.

Hodnoty stavové proměnné flag jsou oproti *modulu Master* mírně zredukované, jejich hodnoty uvádí následující tabulka.

Název	Číselná hodnota	Význam
NO_ACTION	0	Právě není prováděna žádná operace
SENDING_DATA	1	Odesílání dat na DALI sběrnici
RECEIVING_DATA	2	Přijímání dat z DALI sběrnice
ERR	3	Detekován error

Tabulka 4: Možné hodnoty proměnné flag (*Slave modul*)

6.2.3.3 send_data (dalislave.c)

Pokud je hodnota stavové proměnné flag rovna „NO_ACTION“, může uživatel zavolat metodu send_data. V případě *modulu Slave* má jediný parametr a to je 8-bitová hodnota, která bude odeslána na DALI sběrnici.

Odeslání odpovědi smí proběhnout nejdříve 2,92ms po obdržení příkazu typu „Query“. Tato doba ustálení je programově zajištěna uvnitř odesílací procedury send_tick.

Stavová proměnná flag je po dobu odesílání dat nastavena na hodnotu SENDING_DATA.

6.2.4 Vnitřně volané funkce

Vnitřně volané funkce volá nadřazený program, který byl spuštěný uživatelem nebo vyvolán přerušením.

6.2.4.1 receive_data (dalislave.c)

Metoda je volána z obsluhy externího přerušení pinu určeného pro příjem DALI dat, které bylo vyvoláno příchodem spádové hrany na tento pin.

Metoda inicializuje stavové proměnné pro příjem dat a spustí timer3, který zajišťuje správné časování a periodické volání procedury receive_tick.

6.2.4.2 receive_tick (dalislave.c)

receive_tick procedura se volá periodicky po každém přetečení čítače timer3. Stavová proměnná flag je nastavena na hodnotu „RECEIVING_DATA“.

Slave modul je schopen přijímat pouze 2 bytové zprávy, ostatní zprávy odmítá jako poškozené a nejsou dále zpracovávány.

Po dokončení příjmu dat se zavolá funkce DataReceived, které se předají oba přijaté byty. Další operace, které se budou s těmito daty provádět záleží již na uživateli programové knihovny.

6.2.4.3 send_tick (dalislave.c)

Pokud je stavová proměnná flag nastavena na hodnotu „SENDING_DATA“, volá se při přerušení timer3 metoda send_tick.

Samotná funkce send_tick již vytváří odpovídající bi-fázový formát protokolu. Signál je poté přes výstupní DALI pin mikroprocesoru odesílán do proudového omezovače a tak na sběrnici.

6.2.4.4 DataReceived (stm8_it.c)

Funkce DataReceived je volána po dokončení přijímání dat z DALI sběrnice. Do této funkce uživatel programové knihovny napíše svůj kód, který bude dál operovat s přijatými daty.

6.2.4.5 Ostatní procedury

Další funkce které se používají pro správný chod komunikace *Slave modulu* byly již popsány v kapitolách 5.2.4.7 až 5.2.4.11 a nebyly v nich, oproti těm použitým v *Master modulu*, prováděny žádné změny. Jedná se o funkce enable_EXTI, disable_EXTI, set_DALIOUT, get_DALIOUT a get_DALIIN.

6.3 Princip přenosu dat v modulu Slave

Tato kapitola je zaměřena na popis činností, které musí software vykonávat při odesílání nebo přijímání dat.

6.3.1 Časování pomocí timer3

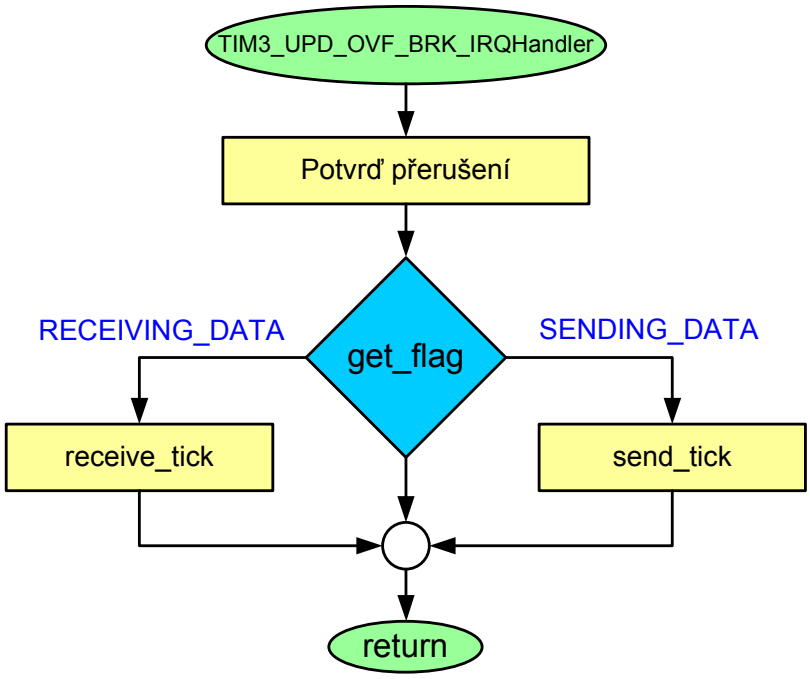
Protože není v *modulu Slave* nutné implementovat detekci kolizí, není potřeba mít možnost načítat úroveň sběrnice maximálně do 26 µs po jejím přechodu z low do high úrovně. Proto pro *modul Slave* volím pomalejší časování, které je zároveň výhodné pro menší zatěžování a menší energetickou spotřebu procesoru.

Perioda přetečení čítače je 104µs. K dosažení této periody jsem zvolil hodnotu předděličky 32 a periody 52. Toto nastavení při rychlosti HSI oscilátoru 16MHz a HSI předděličce rovno 1 podle následujícího vzorce dává:

$$T_{\text{int}\,errupt} = \frac{Přředděliča * \text{Perioda}}{f_{CLK}} = \frac{32 * 52}{16 \cdot 10^{6}\,Hz} = 104 \cdot 10^{-6}\,s = 104\mu s$$

Oba procesy, jak přijímání tak odesílání dat, jsou časovány stejným časovačem (timer3) se stejným nastavením periody 104µs.

f_{CLK} je ovlivněno dalším parametrem, kterým je předděličkа HSI hodinového signálu. Ta musí být rovna jedné a její nastavení se provádí v proceduře init_DALI. Hodnotu HSI předděličky nesmí uživatel změnit, jinak nebude přenos DALI dat správně fungovat.



Obr. 27: Diagram obsluhy přerušení timer3 (*Slave modul*)

Pokaždé když dojde k přerušení, zjistí se hodnota stavové proměnné flag a podle toho jestli se právě data přijímají nebo vysílají se spustí procedura receive_tick nebo send_tick.

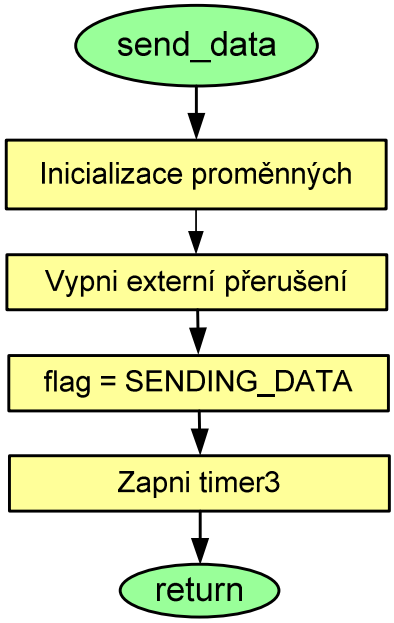
Diagram činnosti timer3 se oproti případu v *Master modulu* zjednoduší, protože *Slave modul* nemá zabudovanou detekci kolizí. Odpadá tedy počítání doby ustálení před opakovaným odesláním zprávy jako v případě *Master modulu*.

6.3.2 Odeslání dat pomocí send_data funkce

Metoda send_data má v případě *Slave modulu* pouze jeden parametr a to je 1byte odesílaných dat.

Oproti *Master modulu* zde také chybí rozhodování o délce zprávy a o opakovaném odeslání zprávy v případě určitých příkazů.

Posledními kroky je vypnutí možnosti externího přerušení (nelze přijímat data v průběhu odesílání dat), nastavení stavové proměnné flag na SENDING_DATA a konečně zapnutí timer3, který bude periodicky volat funkci send_tick a vytvářet tak příslušný bi-fázový signál DALI protokolu.



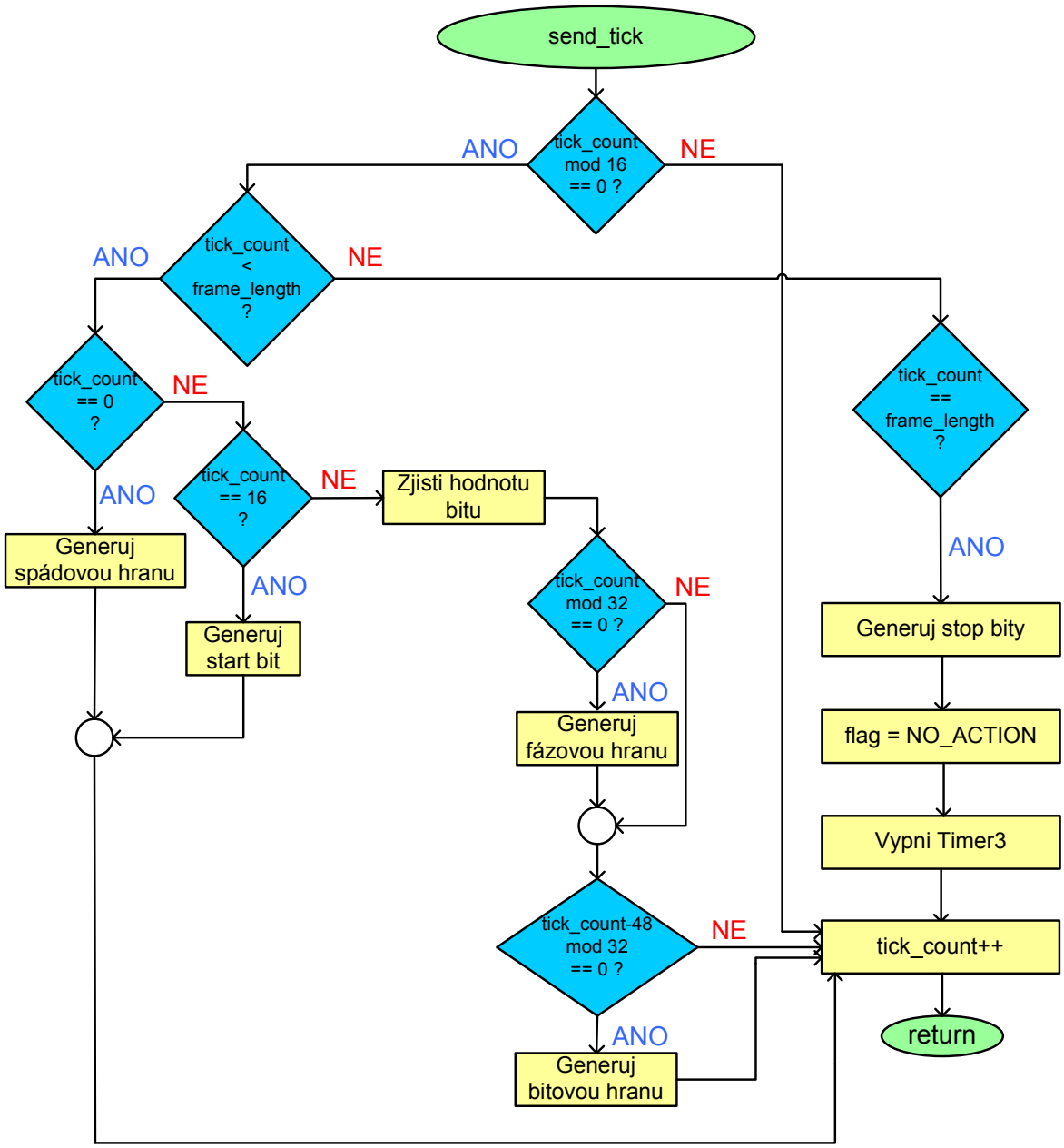
Obr. 28: Diagram funkce send_data *Slave modulu*

6.3.3 Periodické volání send_tick

Procedura send_tick je oproti té v *modulu Master* podstatně zjednodušena. Je to především tím, že *Slave modul* se nerozhoduje o délce zprávy, dále nezajišťuje detekci kolizí a také nerozlišuje prioritu odesílaných zpráv. Další změnou je, že při odesílání

zprávy ze *Slave modulu* se nepoužívá žádná doba ustálení, která v *Master modulu* nastala po ukončujících dvou stop bitech.

Pro podrobné vysvětlení způsobu generování správného bi-fázového tvaru DALI protokolu viz. kapitola 5.3.3.



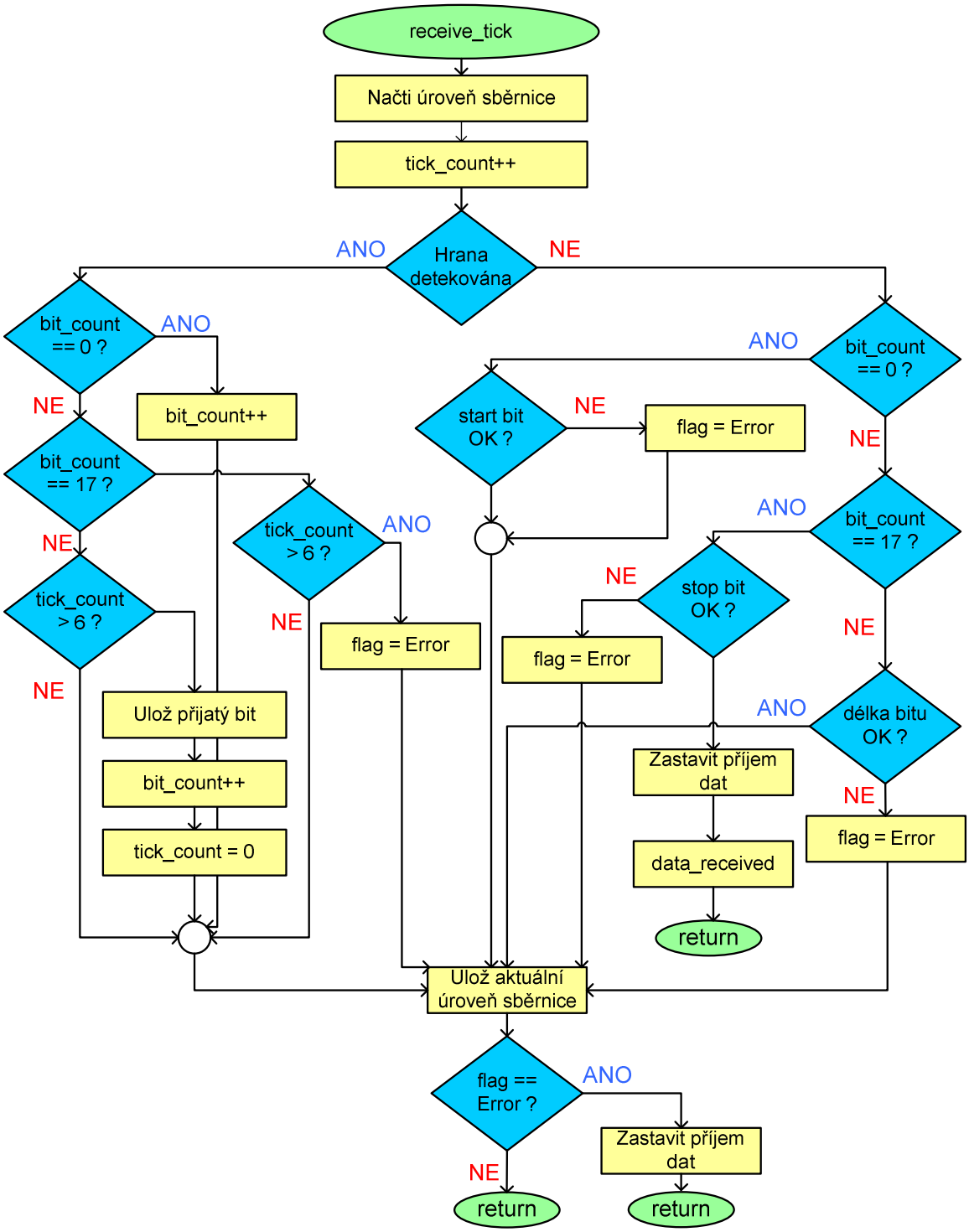
Obr. 29: Vývojový diagram funkce send_tick (*Slave modul*)

6.3.4 Přijímání dat pomocí receive_data funkce

Tato procedura je v *modulu Slave* zcela identická s procedurou v *modulu Master*, která byla podrobně popsána včetně vývojového diagramu v kapitole 5.3.4.

6.3.5 Periodické volání receive_tick

Úkolem procedury receive_tick je správně přijmout, zpracovat a předat uživateli programové knihovny přijatá data. Dalším úkolem je ale také v průběhu přijímání dat kontrolovat jestli zpráva není porušena. Tím se myslí například jestli má správné délky bitů, nebo jestli se během přijímání stop bitů nevyskytuje hrana.



Obr. 30: Vývojový diagram funkce receive_tick (*Slave modul*)

Princip fungování metody receive_tick je stejný jako v *modulu Master*, kde byl podrobně popsán.

Rozdíly najdeme především v podmínkách používajících proměnnou tick_count, kvůli různému časování. Hodnoty tick_count v podmínkách jsou čtvrtinové díky čtyřikrát pomalejšímu časování timer3. Dalším rozdílem je, že v *modulu Slave* se přijímají pouze 2 bytové zprávy a ne 1 a 3 bytové jako tomu bylo v *modulu Master*.

7 Demonstrační program

Tento demonstrační program jsem vytvořil pro účely testování v předchozích kapitolách popsané programové knihovny DALI *modulů Master* a *Slave*.

Jeho funkce a provedení odpovídají ovládacím programům, které se používají pro řízení osvětlení v budovách. S jeho pomocí lze například ovládat jednotlivé *slave* prvky nebo jejich celé skupiny, graficky zobrazovat úrovně osvětlení jednotlivých osvětlovacích prvků, ukládat provedené příkazy a mnoho dalších funkcí.

Tato kapitola vysvětluje spíše ovládání demonstračního programu než popis jeho vnitřních principů a funkcí.

7.1 Vývojové prostředí a požadavky

Demonstrační program jsem napsal v jazyku C# a jako vývojové prostředí jsem využil Microsoft Visual Studio 2005. Pro jazyk C# jsem se rozhodl, protože má kvalitní dokumentaci a podporu jak od společnosti Microsoft, tak od ostatních vývojářů a myslím si, že se bude rozvíjet i v budoucnosti.

Programy napsané v jazyku C# potřebují ke své správné funkci operační systém Microsoft Windows s .NET platformou. Tento program vyžaduje .NET framework verze alespoň 2.0.

Dalším prostředkem, který je nutné pro funkčnost programu mít je převodník USB/RS232. USB konec převodníku je zapojen do PC a RS232 konec je připojen k vývojové desce pro STM8 mikrokontroler. V případě že má používaný PC svůj vlastní sériový COM port, není převodník potřeba.

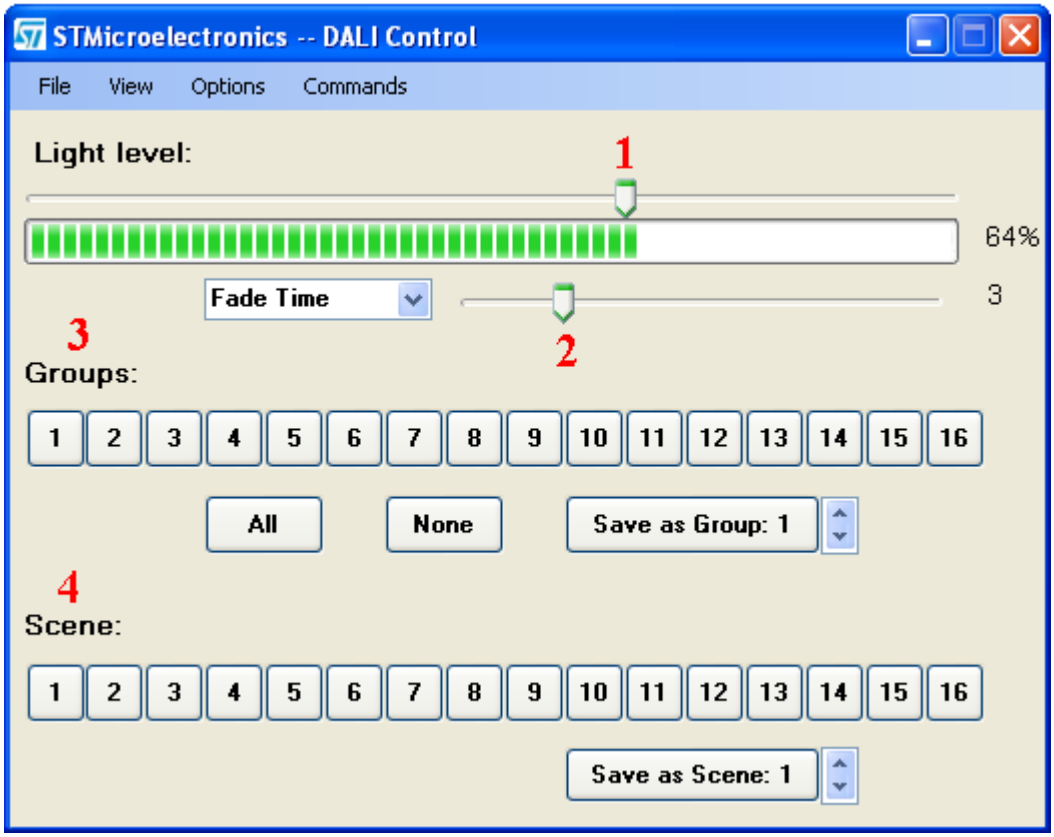
7.2 Stav po spuštění

Po spuštění se program pokusí otevřít virtuální COM port, defaultně nastavený na hodnotu COM3. Pokud tento port neexistuje nebo nejde z jiného důvodu otevřít, je ohlášena chyba. Uživatel má možnost vybrat jiný COM port po spuštění obrazovky Connection. Obrazovka Connection se vyvolá v menu Options/Connection demonstračního programu.

Po úspěšném připojení k virtuálnímu COM portu lze již posílat příkazy *slave* zařízením, ale zatím jen s typem adresace *broadcast*. Jednotlivě adresovat je možné až

když uživatel *slave* prvky vyhledá jedním z vyhledávacích algoritmů. Vyhledávací algoritmy najde uživatel v obrazovce, která se vyvolá v menu Commands/Search Ballasts.

7.3 Hlavní obrazovka



Obr. 31: Hlavní obrazovka demonstračního programu

7.3.1 Posuvník úrovně osvětlení

Pomocí posuvníku nastavuje uživatel úroveň osvětlení adresovaných *slave* prvků. Příkaz nastavující osvětlení je odeslán až po uvolnění tlačítka myši. Pokud tedy uživatel drží levé tlačítko a hýbe s posuvníkem, žádný příkaz se neodesílá.

Hodnota osvětlení má rozsah 0 až 254, zde je ale přepočítána na procenta a tato hodnota je vypsána vedle prvku progressBar, který znázorňuje aktuální úroveň osvětlení.

7.3.2 Fade Time / Fade rate

Rozbalovací menu dává na výběr ze dvou metod postupného rozsvěcení/pohasínání osvětlení.

Fade time udává dobu jak dlouho bude trvat změna z aktuální úrovně osvětlení do požadované úrovně osvětlení. Má 15 stupňů, přičemž samotná doba v sekundách se spočte

podle vzorce $T = \frac{1}{2}\sqrt{2^X}$, $X = 1-15$. Hodnota X = 0 znamená, že je postupné pohasínání vypnuto a úroveň osvětlení se mění skokově.

Fade rate udává o kolik kroků za sekundu se dokáže úroveň osvětlení změnit. Opět má 15 stupňů a počet kroků se vypočte podle vzorce: $F = \frac{506}{\sqrt{2^X}}\, steps/\sec,\ X = 1-15$.

Stupeň X tentokrát hodnoty 0 nenabývá.

Vypočtené hodnoty pro jednotlivé kroky X = 0-15 podle výše uvedených vzorců udává následující tabulka:

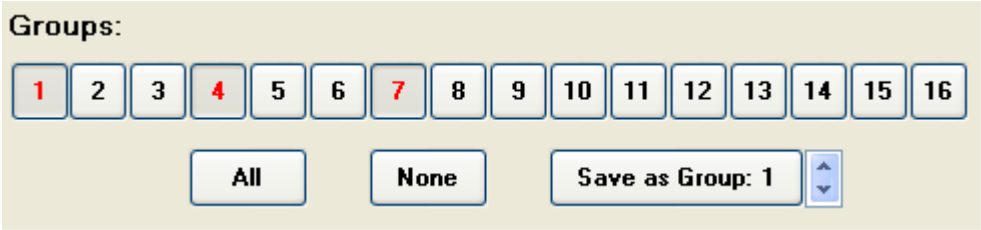
X	**Fade Time (sekund)**	**Fade rate (kroků/s)**
0	<0,707	-
1	0,707	357,796
2	1,000	253,000
3	1,414	178,898
4	2,000	126,500
5	2,828	89,449
6	4,000	63,250
7	5,657	44,725
8	8,000	31,625
9	11,314	22,362
10	16,000	15,813
11	22,627	11,181
12	32,000	7,906
13	45,255	5,591
14	64,000	3,953
15	90,510	2,795

Tabulka 5: Stupně Fade Time/ Fade rate

7.3.3 Groups

Oblast Groups obsahuje 16 tlačítek, kterými může uživatel vybírat skupiny *slave* zařízení, které chce ovládat.

Pokud uživatel zvolí kliknutím myši skupinu, barva textu tlačítka se změní na červenou a tlačítko změní svůj vzhled, viz. obrázek. Skupina se odznačí opětovným kliknutím na tlačítko.



Obr. 32: Výběr skupin *slave* zařízení

Tlačítko All má za úkol označit všechny skupiny najednou, tlačítko None naopak všechny skupiny odznačí.

Pokud chce uživatel uložit vybrané *slave* prvky jako skupinu, musí je nejdříve vybrat v obrazovce Ballasts List, viz Kapitola 7.4. Poté šipkami u tlačítka Save as Group zvolí pod jakou skupinou chce *slave* zařízení uložit a stiskne toto tlačítko. Všem označeným *slave* zařízením se zašle zpráva, že patří do nově definované skupiny.

7.3.4 Scene

Tlačítka při výběru scény fungují podobně jako tlačítka u výběru skupin, jen s tím rozdílem že může být v jedné chvíli zvolena pouze jedna scéna. Proto když zvolíme scénu 1 a poté klikneme například na scénu 3, scéna 1 se automaticky odznačí.

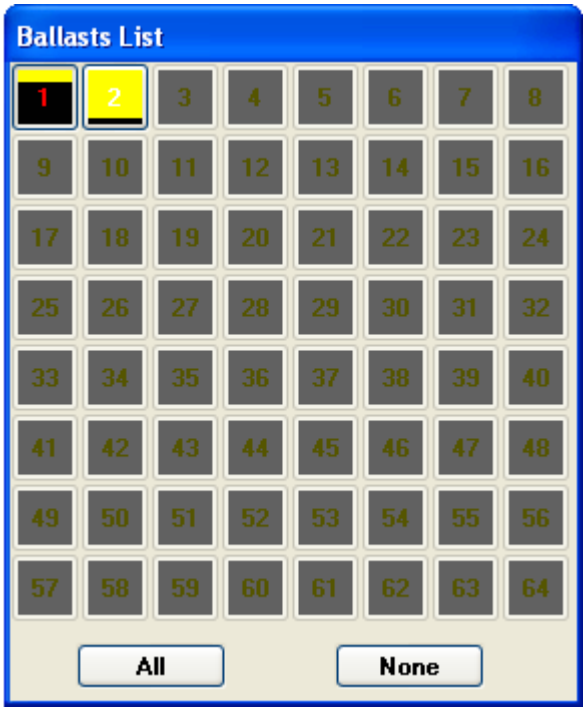
Každé *slave* zařízení má úroveň osvětlení příslušející dané scéně uloženo v paměti. To znamená že každé *slave* zařízení může mít definováno až 16 různých scén. Scénou je v našem případě chápána úroveň osvětlení, kterou uživatel do *slave* prvku uložil jako scénu s určitým pořadovým číslem.

Ukládání scén probíhá podobně jako u ukládání skupin. Nejdříve zvolíme v obrazovce Ballasts List *slave* prvky, kterých se příkaz bude týkat. Poté nastavíme úroveň osvětlení, která bude odpovídat nové scéně. Dále nastavíme číslo scény šipkami a potvrdíme tlačítkem Save as Scene.

7.4 Ballasts List obrazovka

Hlavním úkolem Ballasts List obrazovky je zobrazovat seznam aktivních *slave* zařízení. Ty prvky které jsou právě do sítě DALI připojeny se zviditelní, zobrazí se jejich úroveň osvětlení a indikuje se jestli jsou právě vybrány.

Obrazovka se zobrazuje automaticky při zapnutí demonstračního programu. Zviditelnit nebo skrýt lze tuto obrazovku v záložce View.



Obr. 33: Ballasts List obrazovka

Prohledání DALI sítě

Abychom zjistili, které *slave* prvky jsou do sítě právě připojeny, musíme síť prohledat jedním z algoritmů v menu Commands/Search Ballasts. Nejlépe nám poslouží Quick Search, který vyzkouší přítomnost všech *slave* prvků. To znamená, že se prohledávají zařízení se *short adresami* 0 až 63 a ty které jsou právě aktivní se zvýrazní v Ballasts List obrazovce.

Zároveň se s indikací přítomnosti zobrazí i aktuální úroveň osvětlení *slave* prvku. V našem případě vidíme, že *slave* prvek se *short adresou* 2 je mnohem více rozsvícený než prvek s adresou 1.

Výběr slave prvků

V této obrazovce se také vybírají *slave* prvky, které budou adresovány příkazy z Hlavní obrazovky. Pro označení *slave* prvku stačí kliknout na příslušné políčko a barva textu políčka se změní na červenou. Pro odznačení stačí opět kliknout na příslušné políčko. V našem případě je označen pouze 1. prvek.

Příkazy zvolené v Hlavní obrazovce se tedy budou týkat jen takto zvolených prvků. Při volbě skupin v hlavní obrazovce se políčka v obrazovce Ballasts List označí automaticky, aby bylo zřejmé které zařízení patří do zvolených skupin.

Tlačítko All označí všechny aktivní *slave* prvky, tlačítko None odznačí všechny předtím označené prvky.

7.5 Connection Setup obrazovka

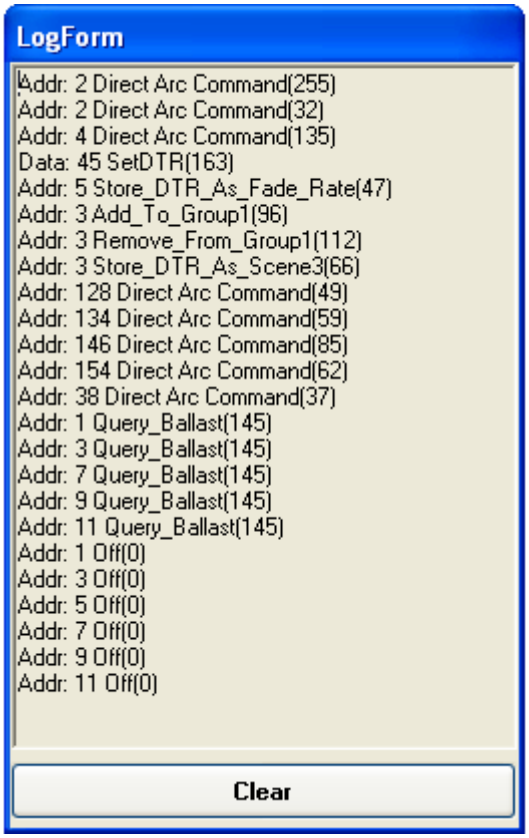
Obrazovka Connection Setup se vyvolá v záložce Options/Connection v menu hlavní obrazovky.

Obr. 34: Obrazovka nastavení sériové komunikace

Pomocí této obrazovky lze změnit číslo COM portu, ke kterému je připojen sériový kabel, případně převodník z USB na RS232, který propojuje PC a vývojovou desku. Uživatel má možnost měnit pouze číslo portu a žádné jiné parametry, ty jsou nastaveny automaticky.

7.6 Log obrazovka

Obrazovka Log slouží k ukládání záznamů o provedených příkazech vzhledem k DALI síti. Log lze vyvolat nebo skrýt v menu hlavní obrazovky, záložce View/Log.



Obr. 35: Obrazovka log (ukládání záznamů)

Formát výpisu do logu se liší podle typu odeslaného příkazu, například speciální příkazy nepoužívají adresu, takže by nemělo smysl používat pro všechny příkazy stejný výpis.

Formáty výpisu příkazů do Log obrazovky:

- Příkazy přímo ovlivňující úroveň osvětlení

  Addr: <adresový byte> Direct Arc Commands(<úroveň osvětlení>)
- Speciální příkazy

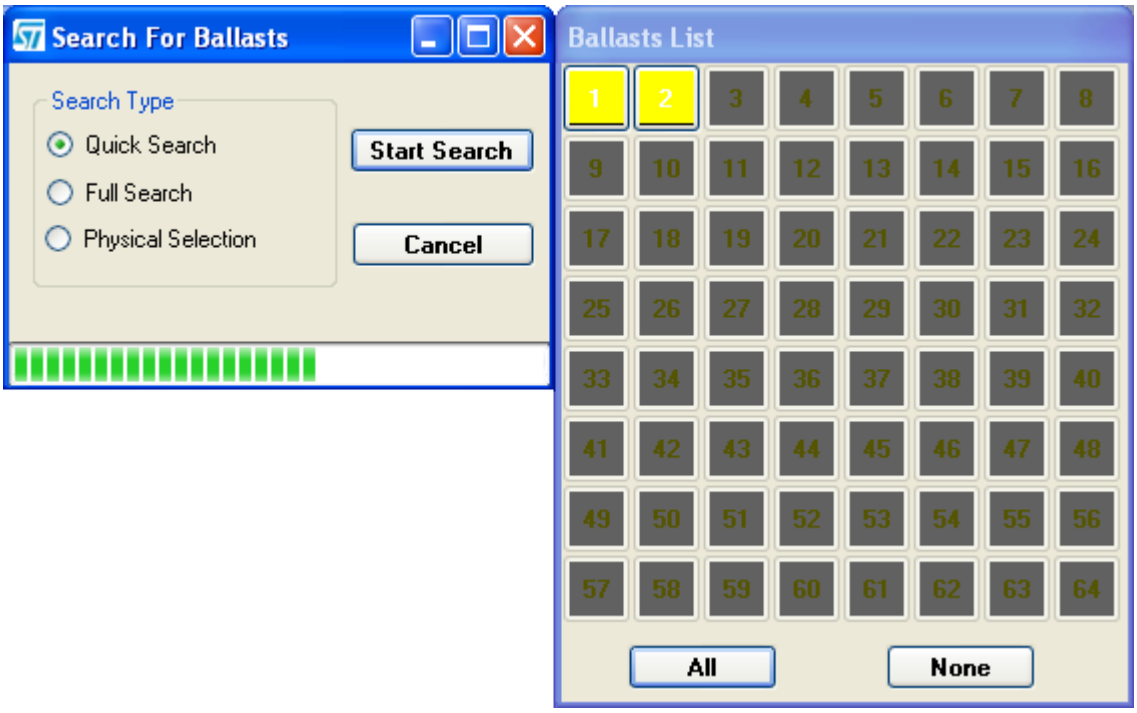
  Data: <datová hodnota> <název příkazu>(<číslo příkazu>)
- Příkazy nepřímo ovlivňující úroveň osvětlení

  Addr: <adresový byte> <název příkazu>(<číslo příkazu>)

Výpisy v obrazovce Log lze smazat použitím tlačítka Clear ve spodní části obrazovky.

7.7 Search Ballasts obrazovka

Obrazovka Search Ballasts umožňuje použít 3 algoritmy pro vyhledávání *slave* zařízení. Stavový pruh na spodní hraně obrazovky znázorňuje průběh prohledávání. Uživatel si vybere z jednoho algoritmu a zapne prohledávání DALI sítě stiskem tlačítka Start Search. Ukončit algoritmus lze i v průběhu prohledávání stiskem tlačítka Cancel.



Obr. 36: Search Ballasts obrazovka a Ballasts List

Quick Search

Algoritmus Quick Search prohledá DALI síť a vyzkouší přítomnost *slave* zařízení se *short adresami* 0-63. Pokud se takový prvek nalezne, aktivuje se v obrazovce Ballasts List políčko s číslem stejným jako je *short adresa* prvku. Dále se zjistí aktuální úroveň osvětlení *slave* prvku a podle toho se změní vzhled tlačítka v Ballasts List obrazovce. V našem případě jsou oba *slave* prvky rozsvíceny téměř na maximum.

Full Search

Full Search algoritmus byl již podrobně popsán v kapitole 3.4.1 pod názvem Náhodné přiřazení adres.

Algoritmus víceméně znovu náhodně přiřadí *short adresy slave* prvkům a jejich pořadí může být tedy jiné. Zaručeno je ale, že očíslování bude začínat od *short adresy* 0 až po počet aktivních *slave* prvků – 1.

Physical Selection

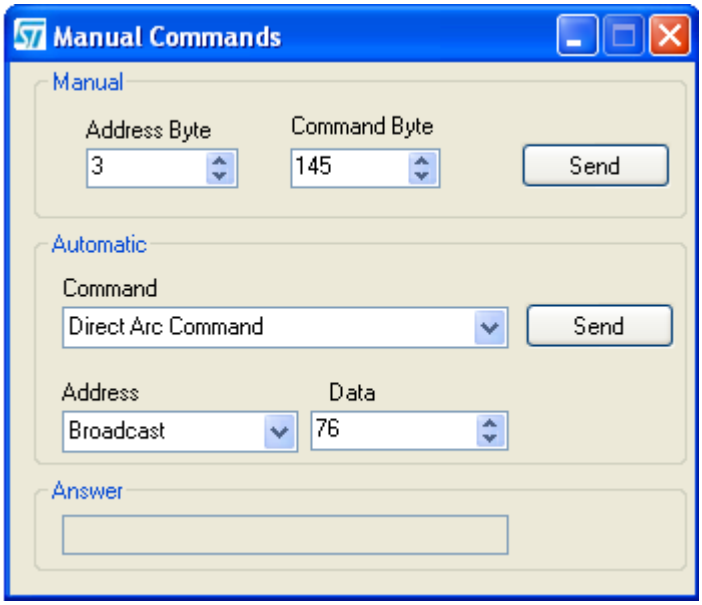
Algoritmus Physical Selection byl podrobněji popsán v kapitole 3.4.2 pod názvem „Přiřazení fyzickým zvolením“.

Aby bylo možno vyhledat *slave* prvky tímto algoritmem, musí být opatřeny zařízením, kterým je možno nastavit adresu *slave* prvku (například přepínač). Algoritmus tedy periodicky hledá *slave* prvek se *short adresou* 0. Potom co jej najde, začne hledat prvek se *short adresou* 1 atd. Hledání lze zastavit stiskem tlačítka Cancel.

Tento algoritmus uživateli umožňuje zvolit pořadí *slave* prvků, ve kterém jim budou přiřazeny *short adresy*.

7.8 Manual Commands obrazovka

Obrazovka Manual Commands má za úkol umožnit uživateli ručně zadávat jakékoliv příkazy, které DALI norma umožňuje a odesílat je DALI zařízením.



Obr. 37: Obrazovka Manual Commands

Manual příkazy

V první části obrazovky má uživatel možnost vybrat číselné hodnoty Address byte a Command byte. Po nastavení číselných hodnot se zpráva odešle stiskem horního tlačítka Send.

Automatic příkazy

V druhé části obrazovky Manual Commands lze vybrat příkaz pomocí prvního roletového menu a nastavit adresování *slave* prvku v druhém roletovém menu. Tento způsob je výhodný proto, že uživatel nemusí znát číselné označení příkazu a program sám vybere správný číselný kód pro adresaci *slave* zařízení.

Answer textové pole

V dolní části obrazovky se nachází textové pole pro výpis odpovědi, která byla přijata od *slave* prvku.

Pokud *slave* prvek neodpověděl, *master* a tedy ani demonstrační program žádná data nepřijme a do textového pole je vypsán údaj „NO ANSWER“. Tento údaj může znamenat jak žádnou odpověď, tak odpověď typu „NO“, kdy *slave* zařízení také neodpovídá.

8 Závěr

Výsledkem této diplomové práce je programová knihovna, kterou jsem vytvořil pro implementaci do zařízení *master* a *slave* s mikroprocesorem STM8, umožňující jejich komunikaci po DALI síti. Pro účely demonstrace funkčnosti programové knihovny jsem *modul Master* navrhnul tak, že slouží jako interface mezi demonstračním řídicím programem na PC a DALI sítí. Uživatel zadává své příkazy v grafickém prostředí demonstračního programu. Příkazy se poté po sériové lince posílají *Master modulu*, který je upraví do předepsaného bi-fázového formátu a odesílá na DALI sběrnici. Zde příkazy zachytí příslušné *slave* zařízení a vykoná příkaz, typicky ve formě operace spojené se změnou úrovně osvětlení světelného zařízení.

Pro mezinárodní zákazníky společnosti STMicroelectronics jsem navíc vytvořil zkrácený uživatelský manuál programové knihovny v anglickém jazyce, který je uveden v příloze A.

Podařilo se mi splnit všechny body zadání diplomové práce a vytvořit tak produkt, který je plně připravený pro komerční využití nebo k propagaci možností využití sítí DALI. Mnou vytvořený demonstrační řídicí program poskytuje možnost vyvolat kterýkoliv příkaz, který je definován normou protokolu DALI. Navíc svým grafickým formátem umožňuje přehledné ovládání světelných prvků, jejich spojování do skupin, nastavení parametrů, vyvolání přednastavených scén a jiných funkcí spojených s ovládáním světelných systémů.

V rámci mé diplomové práce jsem se seznámil, poznal výhody a široké možnosti dnešních 8-bitových mikroprocesorů, které jsou schopny velice rychle zvládat i poměrně složité operace. Práce pro mě byla mimo jiné přínosem v tom, že jsem si rozšířil obzory v oblasti domovní automatizace. Zjistil jsem také, že v případě řízení osvětlovacích systémů jde o oblast, která není zatím příliš rozšířená a je zde velký prostor pro další výrobce a společnosti, které by se chtěly zabývat vývojem zařízení pro DALI sítě.

Díky docházení, měření a konzultacím ve společnosti STMicroelectronics jsem si také udělal představu o fungování a evoluci projektu ve velké společnosti.

Hlavním úkolem této diplomové práce bylo vytvořit programové řešení pro oba základní typy komunikačních modulů DALI. Jako vhodné rozšíření mé práce by mohla být hardwarová realizace těchto modulů. V mém případě byl *modul master* i *slave* testován na vývojové desce pro STM8S2xx a proudový omezovač byl sestaven na nepájivém poli.

Další možností zajímavého pokračování mé práce by mohlo být rozšíření demonstračního řídicího programu o možnost spouštění příkazových skriptů, vhodné například pro programové řízení sestav jevištních reflektorů. Dále se nabízí možnost převedení demonstračního programu pro využití na přenosném mobilním zařízení typu PDA, včetně možnosti ovládání demonstračního programu přes internet. Vzdálené ovládání osvětlení budovy, například rodinného domku, může být výhodné z bezpečnostních důvodů, kdy majitel simuluje přítomnost osob v jinak prázdném domě.

Vzhledem k ochotě mého vedoucího Ing. Jana Fischera, CSc. a široké podpoře, které se mi dostalo ve společnosti STMicroelectronics pro mě byla tato práce velkým přínosem a pokud k tomu budou vhodné podmínky, chtěl bych se programování embedded zařízení určených pro automatizaci věnovat i ve své profesní kariéře.

Literatura

[1] AG-DALI Group [online]
http://www.dali-ag.org/

[2] Nema Standard Publication 243 – 2004 (Digital Addressable Lighting Interface (DALI) Control Devices Protocol), General Requirements, part 1-2004 [online]
http://www.archenergy.com/lrp/lightingperf_standards/DALI%20V1_13.pdf

[3] Nema Standard Publication 243 – 2004 (Digital Addressable Lighting Interface (DALI) Control Devices Protocol), Specific Commands for Control Devices, part 2-2004 [online]
http://www.archenergy.com/lrp/lightingperf_standards/DALI%20Pt2%20V1_3.pdf

[4] STMicroelectronics (2004) Application Note AN1756: Choosing a DALI implementation strategy with ST7DALI [online]
http://www.st.com/stonline/books/pdf/docs/10132.pdf

[5] STMicroelectronics (2004) Application Note AN1900: Hardware implementation for ST7DALI Eval [online]
http://www.st.com/stonline/books/pdf/docs/10252.pdf

[6] STMicroelectronics (2004) Application Note AN1601: Software implementation for ST7DALI-eval [online]
http://www.st.com/stonline/books/pdf/docs/10307.pdf

[7] STMicroelectronics (2008) Programming manual PM0044: STM8 CPU programming manual [online]
http://eu.st.com/stonline/products/literature/pm/13590.pdf

[8] STMicroelectronics (2008) STM8AF61xx, STM8AF51xx Datasheet [online]
http://www.st.com/stonline/products/literature/ds/14395.pdf

[9] STMicroelectronics (2008) Reference manual RM0016: STM8S microcontroller family [online]
http://www.st.com/stonline/products/literature/rm/14587.pdf

[10] STMicroelectronics (2008) Application note AN2752: Getting started with the STM8S [online]
http://www.st.com/stonline/products/literature/an/14651.pdf

[11] Motorolla (2002) Designer Reference Manual: Digirally Addressable Lighting Interface (DALI) Unit using the MC68HC908KX8 [online]
http://www.freescale.com/files/microcontrollers/doc/ref_manual/DRM004.pdf

[12] STMicroelectronics (2001) ST7DALI evaluation kit: Usermanual

[13] STMicroelectronics (2001) ST7DALI evaluation kit: DALI draft (Annex E: Control interface for controllable ballasts, Annex E4: Control by digital signals)

[14] STMicroelectronics (2008) Application note AN2781: UART emulation software in STM8S microcontrollers [online]
http://www.st.com/stonline/products/literature/an/14762.pdf

[15] STMicroelectronics (2008) ST7 8-bit MCU family: User guide
http://www.st.com/stonline/products/promlit/pdf/brstm8s0908.pdf

[16] International Light Fair 2002 Presentation June 4, 2002: A Specifier´s View of DALI [online]
http://www.dalibydesign.us/documents/Lightfair_specifier_view_of_dali_presentation.ppt

[17] International Light Fair 2003 Presentation, May 7 2003: Taking the Byte out of Bits: Lighting Desing, Installation, and Operation in the Digital Age [online]
http://www.dalibydesign.us/documents/Lightfair_taking_byte_out_of_bits_presentation.ppt

[18] Phillips (2003) Application note AN10225: P87LPC760 based DALI slave [online] http://www.nxp.com/acrobat_download/applicationnotes/AN10225_2.pdf

[19] Herout, Pavel; Učebnice jazyka C, 1.díl (III. upravené vydání): Vydavatelství Kopp 2003 269s., ISBN 80-85828-21-9

[20] Herout, Pavel; Učebnice jazyka C, 2. díl: Vydavatelství Kopp 2002 236s., ISBN 80-85828-50-2

[21] Gunnerson Eric; Začínáme programovat v C#, Vydavatelství Computer Press 2001, vydání první, ISBN 80-7226-525-3

[22] The Code Project (2006) Multiple Matrices With ColorMatrix in C# [online] http://www.codeproject.com/KB/graphics/multiple_color_matrices.aspx

[23] The Code Project (2002) Threading in C# [online] http://www.codeproject.com/KB/cs/threader.aspx

[24] The Code Project (2008) Asynchronous Method Invocation [online] http://www.codeproject.com/KB/cs/AsyncMethodInvocation.aspx

[25] Wikipedia (2008) Manchester coding [online] http://en.wikipedia.org/wiki/Manchester_coding

[26] Allbusiness (2003) Turned on by DALI [online] http://www.allbusiness.com/manufacturing/electrical-appliance-equipment-manufacturing/6341034-1.html

Příloha A

Uživatelský manuál programové knihovny v angličtině

D.A.L.I. Protocol implementation on STM8 microcontroller

1 Introduction

This document describes software libraries designed for DALI master and DALI slave devices and instructions for the user of this solution. The software libraries enable both devices to communicate over the DALI bus. It is required for STM8 microcontroller because it doesn't have support for DALI protocol in form of peripheral. Therefore, the DALI protocol has to be emulated by the software.

Master is able to exchange data with both other master devices and slave devices. Slave can communicate just with master devices and cannot initiate communication on its own. Slave can just respond to the questions sent by the master in a defined way.

Sending 1-,2- and 3-byte frames is supported for master device, slave device can send just 1-byte frames. Master can receive 1-byte answer frame from slave device and 3-byte command frames from another master connected to the DALI bus.

On the STM8 processor, two pins are used for DALI communication and user has the possibility to choose both of them. The only restriction is that DALI input pin must be external interrupt capable.

A firmware library is delivered with this document. It contains the complete C source code so that the user can compile, link or modify it as necessary.

2 Principle of DALI protocol

DALI protocol is bi-phased (Manchester) asynchronous serial protocol. Bi-phased coding is used for all bits in the data frame with exception of two stop bits (no change in phase). This type of coding is also used in Ethernet networks and corresponds to the IEEE 802.3 standard.

The following figure illustrates the principle of the bi-phased coding:

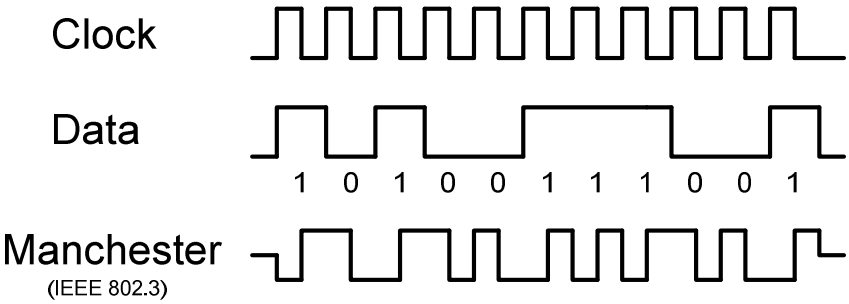


Figure 1: Bi-phased coding principle

3 Principle of the software solution

Timing of the transmission is based on the processor's timer3 overflow interrupts for both sending and receiving data. Therefore, timer3 should not be used for other purposes and should be reserved for DALI communication only.

3.1 Master transmission

Principle of the transmission is based on the overflow interrupts of timer3. In the case of master module, the interrupt takes place every 13µs. This period is set up in the initialization procedure that has to be run by the user before using send or receive procedures. In the initialization, the source of the clock is chosen the 16 MHz HSI RC oscillator and its divider is set to 1. The prescaler of the timer3 is set to 4 and period of timer3 is 52. The period of the overflow interrupts is then:

$$T_{\text{int}\,errupt} = \frac{\text{Pr}\,escaler * \text{Period}}{f_{CLK}} = \frac{4*52}{16Mhz} = 13\mu s$$

This period seems to be quite short but it is necessary because of the collision detection described further. The advantage of this setup is that the latency for transmission start is just 13 µs compared with 104µs in case of slave module. In case of transmitting several frames one after another, the delay before start of the transmission becomes important.

User should not change this setup otherwise the software will not work properly.

The DALI is bi-phased protocol and its generation is carried out according to the following timing diagram:

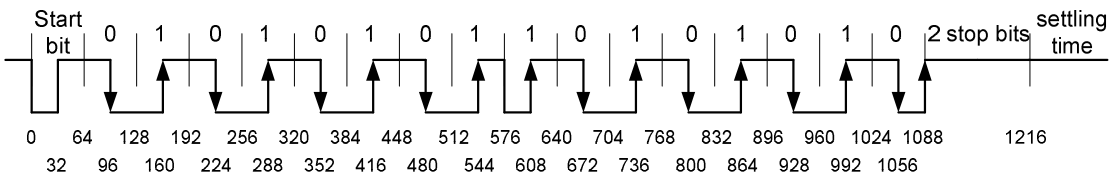


Figure 2: DALI frame structure

Units of the both axis are numbers of ticks. One tick is period between two timer3 overflow interrupts, in our case 13µs.

Upper time axis is the time of the phase edge. It is situated in the middle of the bit (every 64 ticks). The task of the phase edge is to prepare the level of the bus into the state that the bit edge could be created. Phase edge has to be created just in the case of the two following bits with the same logical value. We can see this situation between ticks 512 and 640. At the time of 576 ticks, the phase edge to low has to be created in order to allow the meaningful bit edge at 608 ticks to be formed. If we didn't use phase edge, there would be no bit edge between following 1's bits because the level of the bus would be already in the high state from the previous bit generation.

Lower time axis is the time of the bit edge. The direction of the bit edge marks logical value of the bit. As we can see from the above stated scheme, the length of one bit is 64 ticks (64*13µs = 832 µs).

Collision detection

In the case of master device, the collision detection was implemented. Detection is provided in three situations:

- Middle of every high half bit (M)

  If the level of the bus is low during transmission of high bit, the collision is detected.
- Edge from high to low (E)

  During maximum of 26 µs before the edge from high to low, the level of the bus has to be checked. If the level of the bus is already in the low level, the collision is detected.
- Edge from low to high (B)

  During maximum of 26 µs after the edge from low to high, the level of the bus has to be checked. Low level of the bus indicates collision.

This type of collision detection (B) is the reason why timing of the master module has to be so fast. The detection is provided in 13 µs + time to execute control mechanism. Another solution without increasing speed of the timing is to control the detection immediately after the edge to high was generated. However, there is a problem of the insufficient speed of the circuits dedicated to the galvanic isolation. Therefore, the first mentioned solution was selected.

Situations in which the collision detection is needed are highlighted on the following figure:

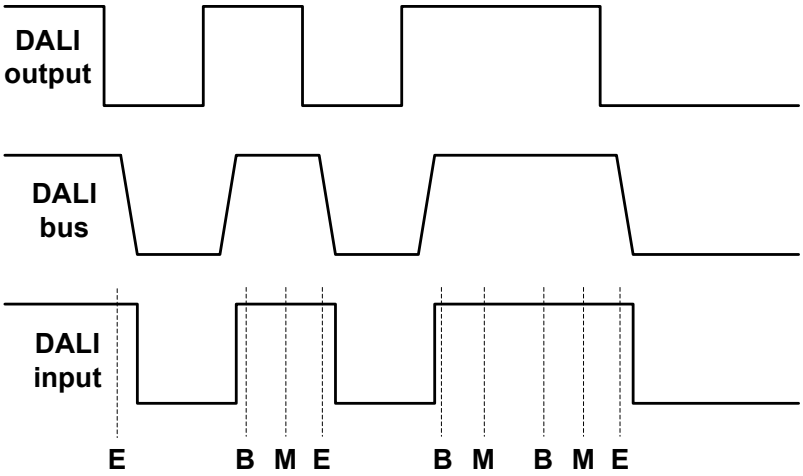


Figure 3: Collision detection situations

Settling time

Settling time between two forward frames (frames sent by the master) is set according to the priority of the frame. Priority is set in the send procedure by the user. There are 5 priorities defined in the standard. Following figure shows dependence of the lengths of the settling time on the frame priority:

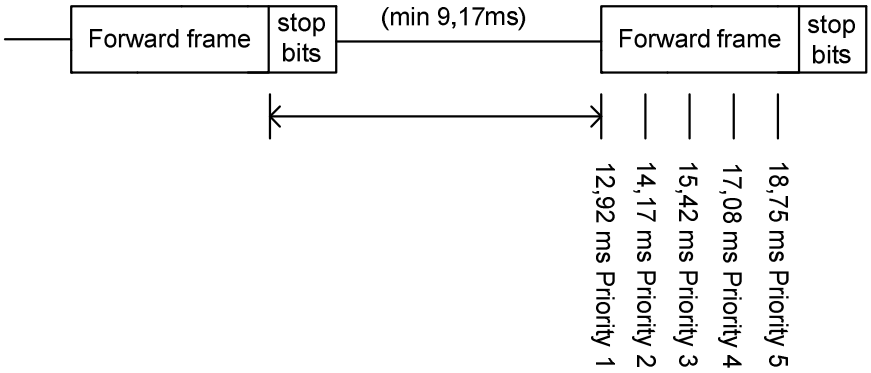


Figure 4: Settling times and priority

3.2 Slave transmission

Slave is using the same timer for timing as master it means timer3. The difference lies in the period between overflow interrupts of the timer. There is no collision detection in the slave module. Therefore, there is no need for the fast timing as it was in the master module. As the clock source has been chosen HSI RC oscillator with 16 MHz and HSI divider set to 1. Timer3 prescaler is set to 32 and period to 52. Therefore the period between two overflow interrupts is:

$$T_{\text{int}\,errupt} = \frac{\text{Pr}\,escaler * \text{Period}}{f_{CLK}} = \frac{32 * 52}{16Mhz} = 104\mu s$$

The timing diagram is then changed accordingly and the length of one bit is 8 ticks, that is 8*104µs = 832 µs.

There are also no priorities of the frames in the slave module and the settling time between the reception of the forward frame and transmission of the answer is always 2,92ms.

3.3 Master and slave reception

The principle of frame reception is the same for both master and slave, difference is just in the length of the frames that are being received. Timing period is the same for receiving and for transmitting.

The receiving process starts with falling edge occurring on the DALI input pin. The external interrupt on the corresponding port is invoked and receiving begins. That is the reason why DALI input pin has to be external interrupt capable. In the case of STM8, microprocessor ports A to E are capable to meet the requirement.

The structure of the galvanic isolation circuit caused negation of the input signal. Therefore, the reading of the DALI input pin was negated in the code. If the user employs another circuit that is not negating the value of the DALI bus, the user has to change code from actual_val = !get_DALIIN(); to actual_val = get_DALIIN(); at the beginning of the procedure *receive_tick()* located in the dalislave.c or dalimaster.c file.

4 User interface

All further described functions are defined in dalimaster.c and dalislave.c files depending on which module user is using.

4.1 Initialization

The interface has to be initialized before the usage by calling the init procedure. This ensures the needed hardware and software is set to initial values.

Definition (both master and slave):

```
void init_DALI(GPIO_TypeDef* port_out,GPIO_Pin_TypeDef
pin_out, GPIO_TypeDef* port_in,GPIO_Pin_TypeDef pin_in);
```

User can choose any general-purpose port and pin for DALI output. Port with the external interrupt capability has to be chosen for DALI input.

4.2 Sending data

Definition (master):

```
void send_data(u8 addrByte, u8 cmdByte, u8 dataByte, u16
priority, u8 BytesToSend);
```

Parameter *BytesToSend* is used to specify how many bytes the message contains. Permitted values are 1,2 or 3. According to this parameter *addrByte*, *cmdByte* and *dataByte* parameters are used. For *BytesToSend* = 1 just *addrByte* is used and others can be random. For *BytesToSend* = 2 *addrByte* and *cmdByte* are used and for *BytesToSend* = 3 all data parameters are used.

Parameter *priority* specifies level of priority and the corresponding settling time that is used after the completition of the frame. Values of the priority can vary from „PRIORITY1“ to „PRIORITY5“. „PRIORITY1“ is the highest one.

In the case of *BytesToSend* = 1, *priority* parameter can be random.

Definition (slave):

```
void send_data(u8 byteToSend);
```

Slave device is able to send just one-byte message as an answer to the master device. This data byte is stored in *byteToSend* parameter.

State variable flag condition

All transmit and receive processes are self-operated and run in the background. User should check value of the variable *flag* before sending data. User should wait until the

value of state variable *flag* is „NO_ACTION“ otherwise calling send_data function can cause disruption of previous send or receive process.

Possible *flag* variable values for master:

Name	Value	Definition
NO_ACTION	0	No operation is in progress
SENDING_DATA	1	Sending data in progress
RECEIVING_DATA	2	Receiving data in progress
REPEAT_SEND	3	Repeated sending data in progress
ERR	4	General error detected
CD_ERR	5	Collision detected

Table 1: State variable *flag* values (master)

Possible *flag* variable values for slave:

Name	Value	Definition
NO_ACTION	0	No operation is in progress
SENDING_DATA	1	Sending data in progress
RECEIVING_DATA	2	Receiving data in progress
ERR	3	General error detected

Table 2: State variable *flag* values (slave)

4.3 Data reception

After the reception of data is completed, procedure DataReceived is called. This procedure is located in it.c file and user should implement further processing of received data in this function.

Definition (master):

```
void DataReceived(u8 answer,u16 answerExt, u8 bit_count)
```

Length of received data is defined by *bit_count* parameter. For *bit_count* = 8 the answer from slave device was received. If another master device sent the received message, *bit_count* parameter should be 24 and both *answer* and *answerExt* parameters carry information.

Definition (slave):

```
void DataReceived(u8 address, u8 dataByte)
```

Data received from the master device are stored in parameters *address* (address byte) and *dataByte* (data byte).

5 Package content

This software package is divided into four parts.

- *DALImaster* - directory for master device software.
- *DALIslave* - directory for slave device software.
- *MasterExample* – example application of master device software.
- *DALIdemo* – demonstration application cooperating with the *MasterExample* software

MasterExample firmware should be loaded into the STM8 evaluation board. STM8 evaluation board should be connected by its RS232 serial bus connector to the PC where the demonstration application should be run. If the PC has no serial port, user can use USB/RS232 convertor.

User places orders by the help of the graphical user interface of the demonstration application and these data are sent over the USB/RS232 convertor into the STM8 evaluation board. Demonstration application supports every command and operation defined in the DALI standard.

Master running on STM8 evaluation board works as an interface between user commands sent from the PC and the DALI bus in this example. Port PD0 is chosen to be DALI input pin and PD5 is DALI output pin, however this setting can be easily changed in init procedure called from main.c file.

6 Firmware package

There are complete projects included in the directories of the software stated in chapter 5. To run the project example, STVD (ST Visual Develop environment) has to be installed and the Cosmic compiler for the STM8S family used. Both tools are free for download from the STMicroelectronics and Cosmic websites. The following table gives the list of the files included in the packages

File name	Module	Content
main.c	master, slave	Main file of the software
dalimaster.c	master	User and system functions of DALI master
dalimaster.h	master	User and system definitions of DALI master
dalislave.c	slave	User and system functions of DALI slave
dalislave.h	slave	User and system definitions of DALI slave
stm8_it.c	master, slave	User interrupt services
stm8_it.h	master, slave	Interrupt declaration header
stm8_conf.h	master, slave	STM8S configuration header
stm8_interrupt_vector.c	master, slave	STM8S interrupt vector table
stm8_clk.c	master, slave	Included library peripheral file
stm8_exti.c	master, slave	Included library peripheral file
stm8_gpio.c	master, slave	Included library peripheral file
stm8_spi.c	master example	Included library peripheral file
stm8_tim3.c	master, slave	Included library peripheral file
stm8_usart.c	master example	Included library peripheral file
stm8_clk.h	master, slave	Peripheral library header file
stm8_exti.h	master, slave	Peripheral library header file
stm8_gpio.h	master, slave	Peripheral library header file
stm8_spi.h	master example	Peripheral library header file
stm8_tim3.h	master, slave	Peripheral library header file
stm8_usart.h	master example	Peripheral library header file
stm8_lib.h	master, slave	Library peripheral header files
stm8_macro.h	master, slave	Macro header file
stm8_map.h	master, slave	STM8S hardware mapping header
stm8_type.h	master, slave	Common data type header

Table 3: Files included in the firmware packages

Příloha B

Instrukční sada master → slave komunikace

Číslo	Zakódování příkazu	Jméno příkazu
-	YAAA AAA0 XXXX XXXX	DIRECT ARC POWER CONTROL
0	YAAA AAA1 0000 0000	OFF
1	YAAA AAA1 0000 0001	UP
2	YAAA AAA1 0000 0010	DOWN
3	YAAA AAA1 0000 0011	STEP UP
4	YAAA AAA1 0000 0100	STEP DOWN
5	YAAA AAA1 0000 0101	RECALL MAX LEVEL
6	YAAA AAA1 0000 0110	RECALL MIN LEVEL
7	YAAA AAA1 0000 0111	STEP DOWN AND OFF
8	YAAA AAA1 0000 1000	ON AND STEP UP
9-15	YAAA AAA1 0000 1XXX	RESERVED
16	YAAA AAA1 0001 XXXX	GO TO SCENE
32	YAAA AAA1 0010 0000	RESET
33	YAAA AAA1 0010 0001	STORE ACTUAL LEVEL IN THE DTR
34-41	YAAA AAA1 0010 XXXX	RESERVED
42	YAAA AAA1 0010 1010	STORE THE DTR AS MAX LEVEL
43	YAAA AAA1 0010 1011	STORE THE DTR AS MIN LEVEL
44	YAAA AAA1 0010 1100	THE DTR AS SYSTEM FAILURE LEVEL
45	YAAA AAA1 0010 1101	STORE THE DTR AS POWER ON
46	YAAA AAA1 0010 1110	STORE THE DTR AS FADE TIME
47	YAAA AAA1 0010 1111	STORE THE DTR AS FADE RATE
48-63	YAAA AAA1 0011 XXXX	RESERVED
64-79	YAAA AAA1 0100 XXXX	STORE THE DTR AS SCENE
80-95	YAAA AAA1 0101 XXXX	REMOVE FROM SCENE
96-111	YAAA AAA1 0110 XXXX	ADD TO GROUP
112-127	YAAA AAA1 0111 XXXX	REMOVE FROM GROUP
128	YAAA AAA1 1000 0000	STORE DTR AS SHORT ADDRESS
129	YAAA AAA1 1000 XXXX	RESERVED
144	YAAA AAA1 1001 0000	QUERY STATUS
145	YAAA AAA1 1001 0001	QUERY BALLAST
146	YAAA AAA1 1001 0010	QUERY LAMP FAILURE
147	YAAA AAA1 1001 0011	QUERY LAMP POWER ON
148	YAAA AAA1 1001 0100	QUERY LIMIT ERROR
149	YAAA AAA1 1001 0101	QUERY RESET STATE
150	YAAA AAA1 1001 0110	QUERY MISSING SHORT ADDRESS
151	YAAA AAA1 1001 0111	QUERY VERSION NUMBER

152	YAAA AAA1 1001 1000	QUERY CONTENT DTR
153	YAAA AAA1 1001 1001	QUERY DEVICE TYPE
154	YAAA AAA1 1001 1010	QUERY PHYSICAL MINIMUM LEVEL
155	YAAA AAA1 1001 1011	QUERY POWER FAILURE
156-159	YAAA AAA1 1001 11XX	RESERVED
160	YAAA AAA1 1010 0000	QUERY ACTUAL LEVEL
161	YAAA AAA1 1010 0001	QUERY MAX LEVEL
162	YAAA AAA1 1010 0010	QUERY MIN LEVEL
163	YAAA AAA1 1010 0011	QUERY POWER ON LEVEL
164	YAAA AAA1 1010 0100	QUERY SYSTEM FAILURE LEVEL
165	YAAA AAA1 1010 0101	QUERY FADE TIME / FADE RATE
166-175	YAAA AAA1 1010 XXXX	RESERVED
176-191	YAAA AAA1 1011 XXXX	QUERY SCENE LEVEL (SCENES 0-15)
192	YAAA AAA1 1100 0000	QUERY GROUPS 0-7
193	YAAA AAA1 1100 0001	QUERY GROUPS 8-15
194	YAAA AAA1 1100 0010	QUERY RANDOM ADDRESS (H)
195	YAAA AAA1 1100 0011	QUERY RANDOM ADDRESS (M)
196	YAAA AAA1 1100 0100	QUERY RANDOM ADDRESS (L)
197-223	YAAA AAA1 110X XXXX	RESERVED
224-255	YAAA AAA1 11XX XXXX	APPLICATION EXTEND. COMMANDS
256	1010 0001 0000 0000	TERMINATE
257	1010 0011 XXXX XXXX	DATA TRANSFER REGISTER (DTR)
258	1010 0101 XXXX XXXX	INITIALISE
259	1010 0111 0000 0000	RANDOMISE
260	1010 1001 0000 0000	COMPARE
261	1010 1011 0000 0000	WITHDRAW
262	1010 1101 0000 0000	RESERVED
263	1010 1111 0000 0000	RESERVED
264	1011 0001 HHHH HHHH	SEARCHADDRH
265	1011 0011 MMMM MMMM	SEARCHADDRM
266	1011 0101 LLLL LLLL	SEARCHADDRL
267	1011 0111 0AAA AAA1	PROGRAM SHORT ADDRESS
268	1011 1001 0AAA AAA1	VERIFY SHORT ADDRESS
269	1011 1011 0000 0000	QUERY SHORT ADDRESS
270	1011 1101 0000 0000	PHYSICAL SELECTION
271	1011 1111 XXXX XXXX	RESERVED
272	1100 0001 XXXX XXXX	ENABLE DEVICE TYPE X
273	110X XXX1 XXXX XXXX	RESERVED

Příloha C

Hodnoty logaritmické křivky tlumení

n	X	n	X	n	X	n	X	n	X
1	0,100	52	0,402	103	1,620	154	6,520	205	26,241
2	0,103	53	0,414	104	1,665	155	6,700	206	26,967
3	0,106	54	0,425	105	1,711	156	6,886	207	27,713
4	0,109	55	0,437	106	1,758	157	7,076	208	28,480
5	0,112	56	0,449	107	1,807	158	7,272	209	29,269
6	0,115	57	0,461	108	1,857	159	7,473	210	30,079
7	0,118	58	0,474	109	1,908	160	7,680	211	30,911
8	0,121	59	0,487	110	1,961	161	7,893	212	31,767
9	0,124	60	0,501	111	2,015	162	8,111	213	32,646
10	0,128	61	0,515	112	2,071	163	8,336	214	33,550
11	0,131	62	0,529	113	2,128	164	8,567	215	34,479
12	0,135	63	0,543	114	2,187	165	8,804	216	35,433
13	0,139	64	0,559	115	2,248	166	9,047	217	36,414
14	0,143	65	0,574	116	2,310	167	9,298	218	37,422
15	0,147	66	0,590	117	2,374	168	9,555	219	38,457
16	0,151	67	0,606	118	2,440	169	9,820	220	39,522
17	0,155	68	0,623	119	2,507	170	10,091	221	40,616
18	0,159	69	0,640	120	2,577	171	10,371	222	41,740
19	0,163	70	0,658	121	2,648	172	10,658	223	42,895
20	0,168	71	0,676	122	2,721	173	10,953	224	44,083
21	0,173	72	0,695	123	2,797	174	11,256	225	45,303
22	0,177	73	0,714	124	2,874	175	11,568	226	46,557
23	0,182	74	0,734	125	2,954	176	11,888	227	47,846
24	0,187	75	0,754	126	3,035	177	12,217	228	49,170
25	0,193	76	0,775	127	3,119	178	12,555	229	50,531
26	0,198	77	0,796	128	3,206	179	12,902	230	51,930
27	0,203	78	0,819	129	3,294	180	13,260	231	53,367
28	0,209	79	0,841	130	3,386	181	13,627	232	54,844
29	0,215	80	0,864	131	3,479	182	14,004	233	56,362
30	0,221	81	0,888	132	3,576	183	14,391	234	57,922
31	0,227	82	0,913	133	3,675	184	14,790	235	59,526
32	0,233	83	0,938	134	3,776	185	15,199	236	61,173
33	0,240	84	0,964	135	3,881	186	15,620	237	62,866
34	0,246	85	0,991	136	3,988	187	16,052	238	64,607
35	0,253	86	1,018	137	4,099	188	16,496	239	66,395
36	0,260	87	1,047	138	4,212	189	16,953	240	68,233
37	0,267	88	1,076	139	4,329	190	17,422	241	70,121
38	0,275	89	1,105	140	4,449	191	17,905	242	72,062
39	0,282	90	1,136	141	4,572	192	18,400	243	74,057
40	0,290	91	1,167	142	4,698	193	18,909	244	76,107
41	0,298	92	1,200	143	4,828	194	19,433	245	78,213
42	0,306	93	1,233	144	4,962	195	19,971	246	80,378
43	0,315	94	1,267	145	5,099	196	20,524	247	82,603
44	0,324	95	1,302	146	5,240	197	21,092	248	84,889
45	0,332	96	1,338	147	5,385	198	21,675	249	87,239
46	0,342	97	1,375	148	5,535	199	22,275	250	89,654
47	0,351	98	1,413	149	5,688	200	22,892	251	92,135
48	0,361	99	1,452	150	5,845	201	23,526	252	94,686
49	0,371	100	1,492	151	6,007	202	24,177	253	97,307
50	0,381	101	1,534	152	6,173	203	24,846	254	100,000
51	0,392	102	1,576	153	6,344	204	25,534		

Příloha D

Seznam součástek použitých při stavbě proudového omezovače

Číslo	Součástka	Hodnota	Omezení	Přesnost	Popis
1	R_1, R_5	0	-	-	-
2	R_2	220	0,5W	5%	Rezistor
3	R_3	10K	0,1W	1%	Rezistor
4	R_6	11K	0,1W	0,1%	Rezistor
5	R_9	1K21	0,1W	0,1%	Rezistor
6	R_{10}	332R	0,1W	0,1%	Rezistor
7	R_{11}	4R7	0,1W	0,1%	Rezistor
8	R_{12}	3k16	0,1W	0,1%	Rezistor
9	C_8, C_9	1µF	20V	10%	Tantalový kondenzátor
10	C_{14}	22 µF	20V	10%	Tantalový kondenzátor
11	D_1	BZX85C22	22V	-	Zenerova dioda
12	D_2	MB2S	200V	-	Diodový můstek
13	D_5	BZ84C2V7	2,7V	-	Zenerova dioda
14	D_6, D_7	BAS16	-	-	Dioda
15	Q_1	BC817-25	-	-	NPN tranzistor
16	U_3, U_4	SFH6156-2	-	-	Optočlen
17	U_5	L78M15CDT	-	-	Napěťový regulátor 12V

Příloha E

Obsah přiloženého CD

- /DaliMaster – softwarová knihovna pro master zařízení
- /DaliSlave – softwarová knihovna pro slave zařízení
- /MasterDemo – softwarová knihovna pro master zařízení upravená pro komunikaci s demonstračním programem po sériové lince
- /DALIdemoPC – demonstrační program pro PC
- /Datasheets – základní dokumenty použité při vývoji této práce
- DP_2009_Knotek_Martin.pdf – diplomová práce v .pdf formátu