Tester pro LIN a CAN sběrnice

LIN and CAN bus tester

Bc. Karel Neřád

Diplomová práce
2013

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Karel NEŘÁD**
Osobní číslo:	**A11900**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Forma studia:	**kombinovaná**
Téma práce:	**Tester pro LIN a CAN sběrnice**

Zásady pro vypracování:

1. **Vypracujte literární rešerši zaměřenou na sběrnice a testery sběrnic.**
2. **Navrhněte datovou komunikaci mezi testerem a uzly sběrnicového systému.**
3. **Vytvořte ovládací stránku v prostředí Freemaster tak, aby umožňovala nastavení způsobu komunikace mezi testerem a uzly.**
4. **Vytvořte hardwarovou podporu pro komunikaci mezi testerem a uzly včetně připojovacího rozhraní se signalizačními diodami.**
5. **Ověřte funkčnost a spolehlivost komunikace testeru s uzly sběrnicového systému.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **LIN Specification Package Revision 2.0ons. Motorola GmbH [online]. 2006, Rev. 2.1, s. 125 [cit. 2013-01-31]. Dostupné z: http://tge.cmaisonneuve.qc.ca/barbaud/R%C3%A9f%C3%A9rences%20techniques/Bus%20Spec_Pac2_1.pdf.**
2. **CANoe/DENoe: Manual. Vector Informatik GmbH [online]. 2008, Rev. 4.1.1, s. 209 [cit. 2013-01-31]. Dostupné z: http://www.kemt.fei.tuke.sk/predmety/KEMT452_ERSA/_materialy/CANoe_Manual_EN.pdf.**
3. **TALBOT, Steave C. a Ren SHANGPING. Comparison of FieldBus Systems, CAN, TTCAN, FlexRay and LIN in Passenger Vehicles. IEEE International Conference on Distributed Computing Systems Workshops [online]. 2009, č. 29 [cit. 2013-01-24]. Dostupné z: http://pdf.marketing5.net/Comparison-of-FieldBus-Systems,-CAN,-TTCAN,-FlexRay-and-LIN-in-download-w14516.pdf.**
4. **IMC9S12XF512 Reference Manual, Freescale PDF document [online]. 2010, [cit. 2013-01-24]., dostupný z http://cache.freescale.com/files/microcontrollers/doc/ref_manual/MC9S12XF512RMV1.pdf?p**
5. **FreeMaster for Embedded Applications: User Manual. Motorola [online]. 2003, Rev. 0.1, s. 103 [cit. 2013-01-31]. Dostupné z: http://www.freescale.com/files/microcontrollers/doc/user_guide/FreeMasterUG.pdf.**

Vedoucí diplomové práce: **doc. Mgr. Milan Adámek, Ph.D.**
Ústav bezpečnostního inženýrství

Datum zadání diplomové práce: **8. února 2013**

Termín odevzdání diplomové práce: **3. června 2013**

Ve Zlíně dne 8. února 2013

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. RNDr. Vojtěch Křesálek, CSc.
ředitel ústavu

ABSTRAKT

Cílem práce je vytvoření funkčního testovacího prostředí pro LIN a CAN sběrnice. Toto prostředí bude využito pro prezentační účely firmy Freescale. Jedná se o ukázkovou aplikaci propojení elektroniky a jednoduchého grafického prostředí. K realizaci je využita deska plošných spojů osazená procesorem MC912SFX navržená ve Freescale, program pro vývoj aplikací CodeWarrior a vývojové prostředí FreeMaster. Cílem je dokázání spolehlivé komunikace po levných sběrnicích používaných v automobilovém průmyslu. Ovládací program je napsán v programovacím jazyku C v prostředí CodeWarrior. K nahrávání a debugování programu bylo využito rozhraní USB Multilink. Monitorování a ovládání sběrnice je realizováno z prostředí FreeMaster, ve kterém je vytvořeno grafické prostředí pomocí HTML. Součástí grafického prostředí jsou skripty napsané ve Visual Basic Scriptu, které zajišťují propojení grafického a textového rozhraní. Součástí práce bylo vytvoření manuálu pro použití grafického prostředí. Tento manuál byl vytvořen za použití programu MS Word. Práce byla úspěšně dokončena a je využívána firmou Freescale.

Klíčová slova:

Sběrnice LIN, sběrnice CAN, FreeMater, Freescale, MC9S12XF

ABSTRACT

The aim of this thesis is to create a functional test environment for LIN and CAN bus. This environment will be used for presentation purposes in company Freescale. This is a sample application of connection of electronics and simple graphical interface. Printed connections board fitted by processor MC912SFX, a program for the development of applications CodeWarrior and development environment FreeMaster are used for realization. The aim is to prove reliable communications for cheap buses used in the automotive industry. The control program is written in the C programming language in the CodeWarrior environment. USB Multilink interface is used for flashing and debugging the code. Monitoring and controlling of bus is implemented in FreeMaster environment, in which graphic user interface is created by Hypertext Markup Language. Connection between text interface and graphic user interface is realized by scripts written in Visual Basic Script language. Creating of manual for graphic environment was part of this work. This manual has been created using MS Word. The work was successfully completed and is being used by Freescale.

Keywords:

LIN bus, CAN bus, FreeMater, Freescale, MC9S12XF

PODĚKOVÁNÍ

Chtěl bych poděkovat doc. Mgr. Adámkovi, Ph.D. za cenné rady a připomínky, kterými přispěl k vypracování této práce. Dále děkuji firmě Freescale a Ing. Cholastovi za poskytnuté informace a konzultace. Také bych chtěl poděkovat svým nejbližším, kteří mě podporovali a motivovali nejen při psaní této práce, ale při celém studiu.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

Obsah

ÚVOD

Práce je zaměřená na vytvoření univerzálního uzlu, který bude schopen monitorovat, modifikovat a testovat komunikaci na sběrnicích LIN a CAN. Tyto sběrnice jsou hojně využívány v automobilovém průmyslu a s jejich pomocí je přenášeno množství informací o stavu vozidla a jeho řízení. V dnešní době je elektronika neodmyslitelnou součástí automobilů. Elektronicky je možno sledovat, ovládat a regulovat téměř všechny části vozu. Jelikož je elektronika součástí kritických systémů jako řízení, airbagy či brzdový systém, je u ní vyžadována vysoká spolehlivost a odolnost. Selhání přenosu informace by mohlo mít fatální následky pro posádku vozidla i pro jeho okolí. Z vysoké spolehlivosti přenosu informace vyplývá zvýšení bezpečnosti vozu. Tuto práci jsem si zvolil proto, abych zvýšil své znalosti v oblasti bezpečnosti automobilového průmyslu a rozvinul své znalosti programování mikroprocesorů. Spolupracoval jsem s firmou Freescale, které mi poskytla hardwarové vybavení a softwarovou podporu. Konzultantem mi byl Ing. Petr Cholasta. K vybudování univerzálního testovacího uzlu bylo využito několik nástrojů. Prvním z nich je prostředí Code Warrior, ve kterém byl napsán hlavní program v programovacím jazyce C. Toto prostředí je propojeno s vývojovým prostředí FreeMaster, které umožňuje sledovat a modifikovat provoz na sběrnici v reálném čase. Pro toto prostředí je vytvořena grafická nástavba, které je vytvořena v HTML s využitím kaskádových stylů. K propojení grafické nástavby a FreeMasteru je použit Visual Basic Script a Javascript. Srdcem uzlu je mikroprocesor MC9S12XF. Nahrávání softwaru a debuggování bylo realizováno přes přípravek USB Multilink PE.

I. TEORETICKÁ ČÁST

1. Automobilový průmysl

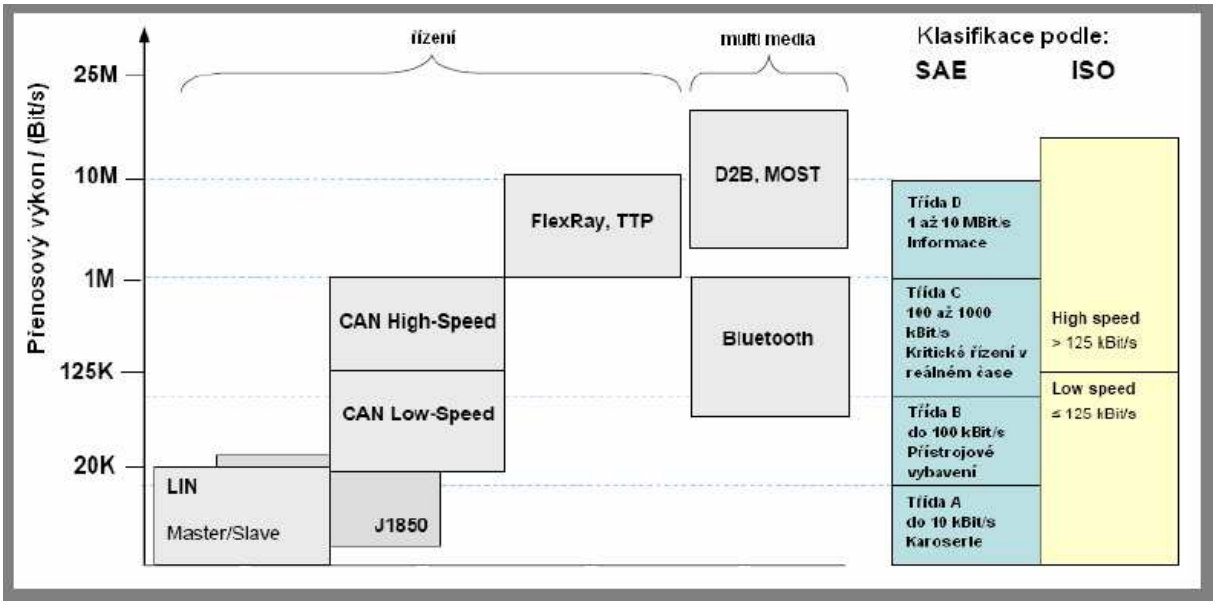
Automobilový průmysl je jedním z největších průmyslových odvětví a pro mnoho průmyslově vyspělých států je základní ekonomickým pilířem. Začátky automobilového průmyslu můžeme hledat již v roce 1769, kdy Nicolas Joseph Cugnot postavil první silniční parovůz. Kapitán francouzské armády, který celý život věnoval novým vynálezům a objevům, dokázal úspěšně zkombinovat poznatky vynálezce Papina a Watta a převedl pohyb pístu na krouživý pohyb. Za podpory francouzského ministra války markýze de Choiseul, sestrojil parní stroj, který byl schopen pohybu o rychlosti až 4km za hodinu. Stroj byl využit jako válečný traktor k tažení děl a bourání zdí. Z dnešního pohledu neohrabané a až úsměvné zařízení, položilo základy automobilového průmyslu, který v dnešní době pohání světovou ekonomiku. [1] Dalším milníkem byl vynález spalovacího motoru, který si patentoval (ale nesestrojil) v roce 1862 francouz Alphonse Beau de Rochas.[2] Až Nicolaus August Otto uvedl do pohybu první spalovací motor na světě. Roku 1886 postavili nezávisle na sobě Karl Benz a také Wilhelm Maybach s Wilhelmem Daimlerem první automobily poháněné benzínovým motorem. V roce 1892 získal německý inženýr Rudolf Diesel patent na vznětový motor, který roku 1897 také jako první na světě postavil. [3] V témže roce byl postaven první automobil na území České republiky v Kopřivnici. Dalším zlomovým bodem byl rok 1913 a zavedení sériové výrobní linky. První sériová výrobní linka byla zprovozněna Henrym Fordem a umožnila masové rozšíření automobilů. [4] Tento vynález se poté rozšířil do Evropy a byl využíván také Baťou a ve Škodových závodech. Ve 20. letech minulého století došlo k prudkému rozvoji průmyslu. Tento rozvoj byl zastaven válkou a následnou hospodářskou krizí. Do 30. let byly vynalezeny téměř všechny mechanické technologie využívané v automobilovém průmyslu. Ve válečném období byl vývoj zpomalen a k rozvoji došlo opět v 50. letech. Celý průmysl se postupně přeorientoval na civilní sféru a kromě výkonu strojů se začaly sledovat i další parametry jako spotřeba, design a bezpečnost. Kvůli nutnosti měření a regulace byla do automobilů zaváděna elektronika. Koncem šedesátých let začala elektronika pronikat do motorů. Společnost Bosch vyvinula technologii elektronicky řízeného vstřikování paliva zážehových motorů. Co se týká podvozku, elektronika se stará především o pohodlí a bezpečnost. Společnost Daimler vyvinula ve spolupráci se společností Bosch antiblokovací systém ABS, který byl uveden na trh v roce 1978. Senzory měří rychlost otáčení jednotlivých kol. Pokud se řídicí jednotka rozhodne, že by se kolo při brzdění mohlo zablokovat, sníží dočasně sílu brzdy a opět ji naplno zapne (i několikrát za sekundu). Systém ABS mimo jiné umožňuje zkrátit brzdnou dráhu a umožňuje zachování lepší stability, ovladatelnosti a řiditelnosti vozidla v mezních situacích. Elektronika zasahuje do řízení stále více. Dalším používaným systémem je ESP (elektronický stabilizační program). ESP má informace o rychlosti jednotlivých kol, krouticím momentu, otáčkách motoru a natočení volantu. Z těchto údajů dokáže zjistit, zda se vozidlo nepohybuje ve smyku. Pokud

ano, může přibrzdit některé z kol a také snížit výkon motoru. [5] Elektronika hraje v automobilu stále důležitější roli. Snižuje průměrnou spotřebu paliva, zvyšuje výkon motoru, bezpečnost cestujících i komfort cestování, který obohacuje o multimediální zařízení a navigační systémy. Moderní polovodičové technologie přispívají ke stále rozsáhlejší náhradě původně mechanicky poháněných agregátů elektrickými. S rostoucím podílem elektroniky v automobilu se dostává do popředí problém integrace řídicích systémů a související architektura sběrnic. Podíl elektroniky na výrobních nákladech automobilu, a tím i na jeho celkové hodnotě, nezadržitelně roste. Zmíněný podíl elektronických systémů se pohybuje mezi 30 až 45 % výrobních nákladů. Role vývojářů automobilové elektroniky, původně zaměřená čistě na elektronické funkce, se postupně mění a spočívá stále více ve vzájemném propojování rozličných elektronických komponent v rámci automobilu. Toto propojování elektronických funkcí bude stále komplexnější, neboť se ukazuje, že téměř všechna měřená data je nezbytné distribuovat do různých míst v automobilu. Teprve výsledná “personalizace” automobilu prostřednictvím inteligentní součinnosti jeho jednotlivých prvků zvyšuje konečnou hodnotu vozu do té míry, že podstatně převyšuje prostou sumu hodnot jednotlivých dílů. [6]

2. Sběrnice v automobilovém průmyslu

Pod pojmem sběrnice obecně rozumíme soustavu vodičů, která umožňuje přenos signálů mezi jednotlivými částmi systému. Pomocí těchto vodičů mezi sebou jednotlivé části systému komunikují a přenášejí data. [7] Nasazení sběrnic bylo logickým krokem, který vyplynul z nutnosti snímat a ovládat stále větší počet prvků v automobilu. Počátky nasazení lze hledat v 80. letech minulého století, kdy byla firmou Bosch vyvinuta sběrnice CAN. Sběrnice používané v automobilech jsou vystaveny značné zátěži rušením a musí také splňovat teplotní odolnost od -40C do +125C. Nepočítá se s komunikací na velké vzdálenosti (více než 100m), ale zato je vyžadována vysoká mechanická odolnost. Tyto faktory odlišují automobilové sběrnice od sběrnic průmyslových. Sběrnice můžeme dělit do čtyř skupin:

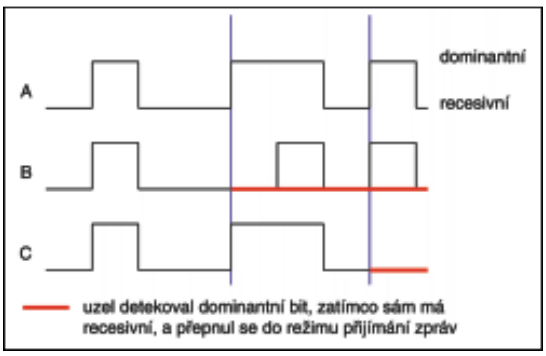
- Třída A: Systém vodičů, který snižuje počet původní kabeláže příjímáním a vysíláním signálů po sběrnici. Určeno pro obecné účely (univerzální asynchronní přijímač/vysílač) s rychlostí do 10kbps
- Třída B: Systém vodičů, který přenáší data mezi uzly. Uzly nahrazují původní samostatné moduly. Určeno pro nekritické aplikace a přenos dat o rychlosti 10kbps až 125kbps.
- Třída C: Systém vodičů, který umožňuje snížení původní kabeláže. Komunikující v reálném čase při rychlosti 125kbps až 1Mbps. Určeno pro kritické aplikace
- Třída X-by-wire: Souhrnný název pro doplňková elektronická zařízení, jejichž úkolem je vylepšení či nahrazení původních mechanických, pneumatických a hydraulických systémů. [8]



Obrázek 1: Rozdělení automobilových sběrnic [9]

2.1. Sběrnice CAN

Historie sběrnice CAN začala již počátkem osmdesátých let minulého století, kdy dostali vývojoví pracovníci firmy Bosch za úkol vyvinout sériovou sběrnici pro použití v automobilech. Do čela vývojové skupiny byl v roce 1983 jmenován Uwe Kiencke. Na vývoji od počátku spolupracovali také technici firmy Mercedes-Benz, budoucího uživatele sběrnice, a vývojoví pracovníci společnosti Intel, potenciálního dodavatele čipů pro sběrnici. Jméno CAN – Controller Area Network dal nové sběrnici prof. Wolfhard Lawrenz z Univerzity aplikovaných věd v Braunschweig-Wolfenbüttelu. Se skupinou od počátku spolupracoval také prof. Horst Wettstein z Univerzity v Karlsruhe. Od samého počátku byla sběrnice CAN navržena jako síť s liniovou topologií pracující v režimu multimaster. To znamená, že žádný z účastnických uzlů není pevně určen jako nadřízená stanice – master. Stanice, která chce ostatním podat zprávu, začne vysílat. V tom okamžiku se stává nadřízenou jednotkou – masterem – a ostatní uzly musí počkat, až je přenos dokončen a linka uvolněna. Není určeno, který uzel má vyslanou zprávu přijmout. Zpráva má identifikátor, z nějž ostatní uzly poznají, co zpráva obsahuje, a podle toho ji přijmou nebo ignorují. O přijetí zprávy tedy nerozhoduje adresa odesílatele ani příjemce, ale její obsah. Je ovšem nutné vytvořit mechanismus, který zajistí, aby byly důležité zprávy doručeny včas a aby se o vysílání nepokoušely současně dva uzly. Stane-li se, že se dva uzly pokoušejí vysílat současně, dostane přednost ten s vyšší prioritou. Priorita je zakódována v identifikátoru. V praxi probíhá celý postup tak, že uzly, které chtějí vysílat, zjistí, zda je volná linka, a v případě, že je, začnou vysílat svůj identifikátor. Přitom kontrolují po jednotlivých bitech shodu vysílané a přijímané zprávy (obrázek 2). [10]



Obrázek 2: Řízení priority zpráv pomocí identifikátorů [10]

Dokud jsou identifikátory shodné, nic se neděje. Teprve v okamžiku, kdy jeden účastnický uzel zjistí, že se v identifikátoru druhého uzlu objevil dominant bit, zatímco on má ve svém identifikátoru ve stejném okamžiku bit recessive, usoudí, že jeho zpráva má nižší prioritu, stáhne se a uvolní sběrnici. Druhý účastník dokončí vysílání a první pokus opakuje, jakmile se linka uvolní. Takové metodě se říká nedestruktivní řízení komunikace a její důležitou předností je to, že se vysílání nezdržuje žádným opakováním úvodní sekvence zprávy. Podmínkou je, že žádné dva identifikátory nesmí být stejné. Postup, který to zabezpečuje, je součástí specifikace

CAN. Mimo jiné také dovoluje připojovat a odpojovat jednotlivé členy i za provozu sběrnice [10].

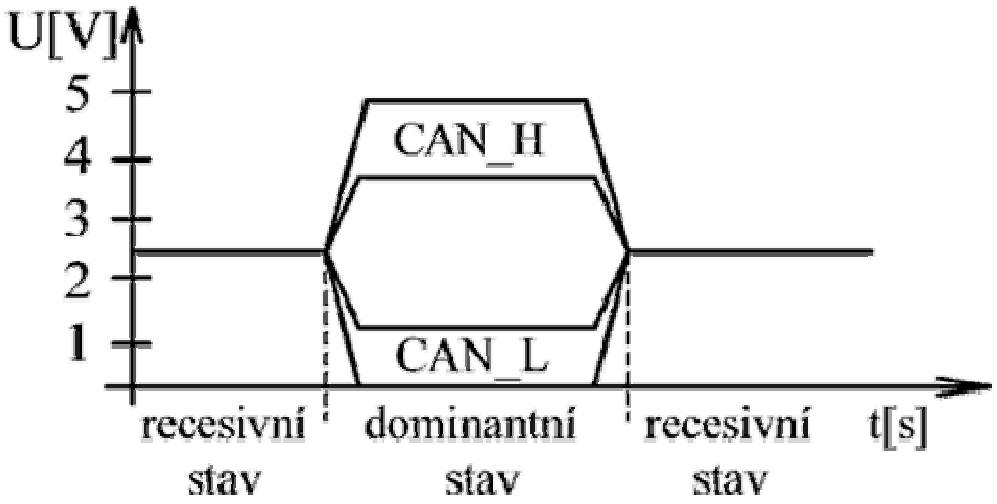
2.1.1. Struktura CAN

CAN je protokol sériové komunikace, určený pro komunikaci v reálném čase s vysokou úrovní bezpečnosti. Je velmi dobře uplatnitelný ve vysokorychlostních sítích stejně jako v levných multiplexových strukturách. V automobilech je použit pro komunikaci s jednotkami ovládajícími vstřikovací systém, senzory pohonu a brzdového systému. Doporučená rychlost přenosu je do 1MBit/s, což umožňuje udržet nízkou cenu komponent a integrování CAN i pro nekritické aplikace. Pro jednodušší implementaci a zachování jednoduchosti je protokol rozdělen do vrstev, a to vrstvu aplikační, vrstvu objektovou, vrstvu přenosovou a vrstvu fyzickou. Objektová a přenosová vrstva zahrnují většinu funkcí a služeb datového přenosu.

Aplikační vrstva
Objektová vrstva - Filtrování zpráv - Obsluha statusu - Obsluha přijetí a odeslání
Přenosová vrstva - Detekce a signalizace chyb - Ověřování zpráv - Potvrzení spojení - Tvoření datových rámců - Rychlost přenosu a časování
Fyzická vrstva - Signálové úrovně - Reprezentace bitů - Přenosová médium

Obrázek 3: CAN ISO/OSI [14]

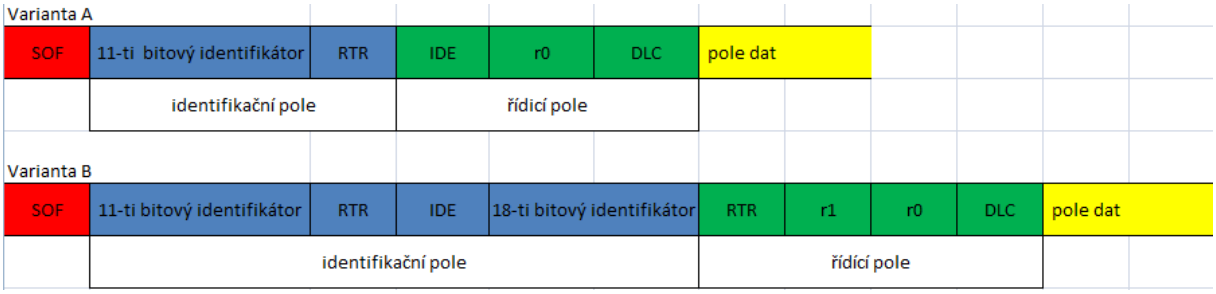
Aplikační vrstva je zaměřená na aplikace, která využívá přenášený signál. Objektová vrstva se stará o filtrování zpráv a předávání datových rámců. Přenosová vrstva představuje jádro CAN protokolu. Stará se o příjem a vysílání. Zprávy, které mají být odeslány, přebírá od objektové vrstvy. Zprávy, které přijme, předává objektové vrstvě k dalšímu zpracování. Je zodpovědná za synchronizaci, časování, tvoření datových rámců a jejich ověřování, detekci a signalizaci chyb a potvrzování příjmu. Fyzická vrstva definuje vlastní přenos signálu po vedení. Definuje přenosové médium a signálové úrovně. V rámci CAN je logická nula definována jako „dominant bit“ a logická jednička jako „recessive bit“.



Obrázek 4: Napěťové úrovně CAN dle ISO 11898-2

2.1.2.Formát datového rámce

Již v roce 1986 se objevilo mnoho publikací popisujících novou sběrnici. Stále však nebyl k dispozici čip, který by komunikaci prostřednictvím sběrnice CAN zajišťoval. Intel jej pod označením 82526 uvedl na trh až v polovině roku 1987. Brzy po něm následoval další výrobce, Philips, a jeho čip 82C200. Oba čipy se však lišily v poměrně zásadní věci. Zatímco Intel favorizoval tzv. Full CAN, Philips dal přednost konceptu BasicCAN. U provedení Basic CAN je zpráva identifikována a kontrolována v procesoru připojeného zařízení, Full CAN má filtr již v komunikačním čipu a již zde se rozhoduje, zda zprávu přijmout a poslat ji procesoru. Varianta Full CAN tak méně zatěžuje vlastní procesor připojeného zařízení. Tato dvě odlišná provedení se dochovala dodnes, i když ne ve své čisté podobě. Vzhledem k široké nabídce čipů – v současné době je k dispozici více než 50 čipů od přibližně patnácti výrobců – existuje množství přechodových forem. Full CAN má dvě modifikace: modifikace A má identifikátor 11bitový, modifikace B (tzv. extended) 29bitový. Tyto modifikace nejsou kompatibilní. Někteří výrobci (např. Siemens) si s touto potíží poradili zavedením modifikace „pasivní B“, která má 11bitový identifikátor, ale umí pracovat i s identifikátorem 29bitovým. Struktura počátku zprávy obou variant je na obr. 2. Obě modifikace se liší přenosovou rychlostí: modifikace A je pomalejší, do 250 kb/s, modifikace B může dosáhnout rychlosti do 1 Mb/s. Rychlost závisí, kromě varianty protokolu, také na délce sběrnice. [10]



Obrázek 5: Formát datového rámce

SOF – Start of frame, inicializační a synchronyzační pole

RTR – Remote Request, bit, který je dominant u datových rámců, recessive u ostatních

SRR – Substitute Remote Request, nahrazuje RTR u modifikace B, a je vždy recessive

IDE – Identifer Extension, bit, který umožňuje rozlišit obě modifikace (je dominant u modifikace A, recessive u B)

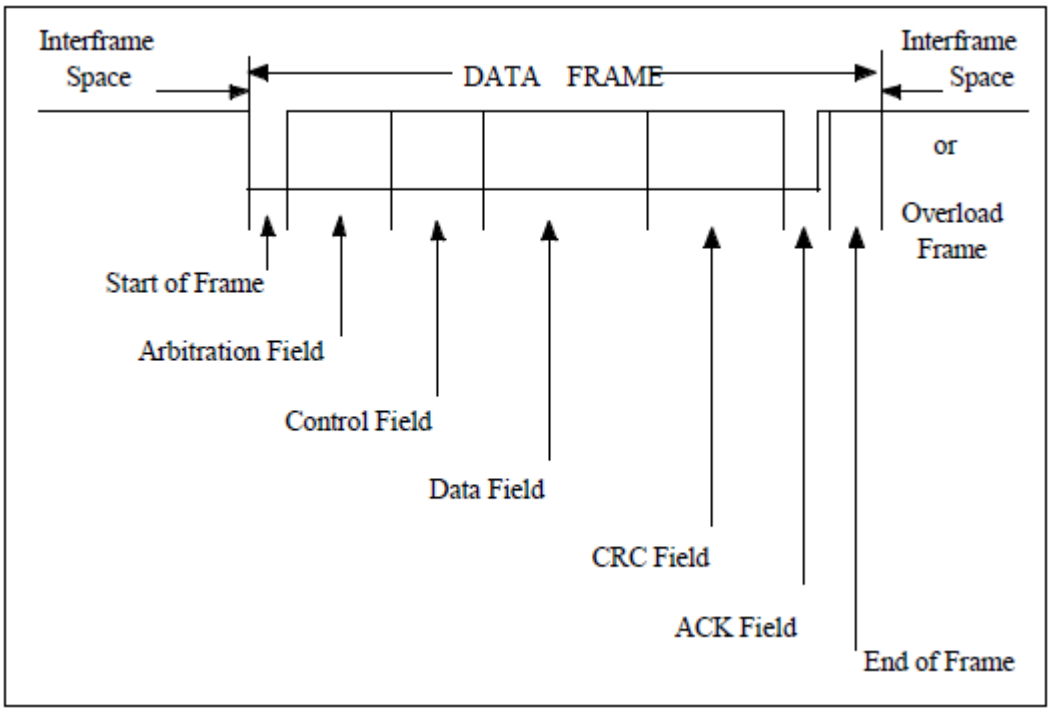
DLC – Data Lenght Code, kód pro délku datového pole

vr1, r0 – rezervované bity (doplňují DLC)

Obě varianty využívají stejných typů datových rámců. Pro zachování jednoduchosti sběrnice jsou využívány pouze 4 druhy rámců.

- Datový rámec (Data frame) – nese data od vysílače k přijímači
- Vzdálený rámec (Remote frame) – je vysílám uzlem, pokud je vyžadován data frame se stejným identifikátorem
- Chybový rámec (Error frame) – je vysílám, pokud některý z uzlů detekuje chybu na sběrnici
- Rámec přetížení (Overload frame) – je vysílám, pokud je potřeba navýšit zpoždění mezi vysílaným rámci

Každý datový rámec se skládá z několika polí: Začátek rámce, rozhodovací pole, kontrolní pole, datové pole, pole kontrolního součtu, potvrzovací pole a konec rámce.



Obrázek 6: Datový rámec CAN [14]

Start of frame udává začátek nového datového nebo vzdáleného rámce. Skládá se z jednoho dominant bitu. Uzel může tento stav vyvolat pouze, pokud je sběrnice v nečinnosti. Všechny uzly se synchronizují podle náběžné hrany.

Rozhodovací pole má rozdílný formát ve standardních a rozšířených datových rámcích. Rozhodovací pole standardních rámců je skládá z 11 identifikačních bitů a RTR bitu. Označení jednotlivých identifikačních bitů je ID-28 … ID-18. V rozšířené verzi je rozhodovací pole složeno z 29 rozhodovacích bitů, SRR bitu, IDE bitu a RTB bitu. Označení jednotlivých identifikačních bitů je ID-28 … ID-0.

2.1.3. Zabezpečení proti chybám

Sběrnice navržená pro použití v dopravních prostředcích musí být velmi spolehlivá a zabezpečená proti možnému výskytu chyb. CAN má celkem pět metod zabezpečení: tři jsou na úrovni zpráv a dvě na úrovni jednotlivých bitů. Na úrovni zpráv se používá detekce chyb pomocí CRC – Cyclic Redundancy Check, ACK a kontrola rámce zprávy.

CAN používá patnáctibitový kontrolní součet, který zaručuje vysokou pravděpodobnost detekce chyby (Hammingova vzdálenost je 6). Pole vyhrazené pro CRC součet obsahuje CRC sekvenci a CRC oddělovač, který ukončuje pole CRC. Aby bylo možné provést výpočet kontrolního polynomu, je nutné provést destuffing datového toku (odstranění stuffed bitů) a získat koeficienty ze SOF, identifikačního pole, řídícího pole a datového pole. Vytvořený polynom je poté vydělen generujícím polynomem ($X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1$)[11]. CRC

oddělovač je tvořen jedním recessivem bitem. Další metodou detekce chyb je kontrola struktury zprávy. Ve zprávě jsou na určitých pevně daných pozicích kontrolní bity. Jednotlivé uzly kontrolují správnost těchto bitů, a nesouhlasí-li jejich hodnota s hodnotou danou strukturou zprávy, hlásí chybu. Třetí metodou zabezpečení je použití ACK – Acknowledgment. V poli ACK jsou dva bity. Vysílající uzel je oba posílá jako recessive. Přijme-li přijímač zprávu korektně, pošle potvrzení s jedním bitem v poli ACK dominant. Obdrží-li vysílající uzel alespoň jedno potvrzení ACK, je vše v pořádku, pokud ne, zkusí zprávu poslat znovu. Na úrovni jednotlivých bitů se používá kontrola shodnosti zpráv. Každý vysílající uzel dostane zpět pro kontrolu kopii odeslané zprávy, která se liší jen v počáteční sekvenci a ACK. Uzel kontroluje bit po bitu, zda se kopie shoduje s originálem. Poslední metodou je vkládání opačného bitu po každém vysílání pěti stejných bitů. Je-li tedy např. pět bitů recesivních, musí být šestý dominantní. Hlášení o chybě obsahuje šest dominantních bitů. Vzhledem k tomu, že poslední uvedená metoda způsobí, že hlášení o chybě je samo detekováno jako chyba, rozšíří se – globalizuje – zpráva o tom, že při komunikaci vznikla chyba, do celé sítě. Postup zpracování zpráv o chybě je dosti složitý a důmyslný. Výsledkem algoritmu, jejž zde pro jeho složitost nebudeme podrobně popisovat (zájemci jej naleznou např. na webové stránce CiA), je to, že většina chyb v komunikaci je spolehlivě detekována, chybná zpráva je automaticky odeslána znovu a uzel, který se opakovaně pokouší o komunikaci a není úspěšný, je v komunikaci omezen prodloužením doby čekání na opakování svého vysílání (k tříbitovému IFS se připočte ještě další osmibitová odmlka), pozbude práv vysílat některá chybová hlášení, která mohou blokovat komunikaci, popř. je při výrazné kumulaci chyb nakonec zcela odpojen, aby nepřetěžoval síť opakovanými neúspěšnými pokusy. Ostatní uzly však mohou komunikovat dále. Částečně omezený uzel (ve stavu error passive) se může dostat do původního stavu, je-li v komunikaci opět úspěšnější, uzel odpojený od sběrnice (bus off) lze připojit jen po resetu softwarového nastavení a opětovném spuštění uzlu. [10]

Obrázek 7: Zakončení zprávy

CRC – Cyclic redundancy Check, 15bitový kontrolní součet

ACK – Acknowledge, dvojbitové pole potvrzení zprávy

EOF – End of Frame, ukončení zprávy

IFS – Intar Frame Space, prodleva mezi zprávami

2.1.4. Standardizace

Dalším důležitým krokem ve vývoji sběrnice CAN byla její standardizace. Stalo se tak v listopadu 1993, po složitých jednáních především s francouzskými výrobci automobilů, prosazujícími svoji sběrnici VAN (Vehicle Area Network). Vznikla tak norma ISO 11898, která definuje fyzickou a datovou vrstvu sběrnice pro rychlosti přenosu do 1 Mb/s. Zásadní změna této normy přišla v roce 1995, kdy byla specifikace rozšířena o již zmíněný 29bitový identifikátor varianty Extended CAN. Kromě ISO 11898 existují ještě např. normy ISO 11992, specifikující aplikační vrstvu pro nákladní automobily a kamióny, a ISO 11783, specifikující totéž pro lesní a zemědělské stroje. Ačkoliv obě normy vycházejí z amerického protokolu J1939, nejsou navzájem kompatibilní. Kromě nich jsou vytvořeny ještě mnohé další standardy a normy pro aplikační vrstvy, testy shody apod., platící obvykle jen v omezené oblasti aplikací (železniční technika, vojenství apod.). Naneštěstí není norma ISO 11898 ani kompletní, ani bez chyb. Může se tedy stát, že zařízení splňující tuto normu nebudou kompatibilní ani v datové a fyzické vrstvě. Firma Bosch proto specifikovala referenční model a metodiku zkoušek kompatibility. Postup testování kompatibility je stanoven normou ISO 16845. Ačkoliv byla sběrnice CAN navržena pro automobily, již od začátku byla používána i v jiných aplikacích. Ve Finsku ji upotřebila firma Kone pro řízení výtahů, švédská konstrukční kancelář Kvaser navrhla použít CAN pro textilní stroje Lindauer Dornier a Sulzer. Tato aplikace byla tak úspěšná, že nakonec vznikla i skupina uživatelů CAN v textilních strojích CAN Textile User's Group. V Nizozemí použila firma Philips Medical Systems CAN pro přenos dat v rentgenových přístrojích. Norma ISO 11898 specifikuje jen dvě nejnižší vrstvy modelu ISO/OSI. To je pro automobilový průmysl, který se snaží spíše o nekompatibilitu a vzájemnou nezaměnitelnost komponent jednotlivých výrobců, zcela vyhovující. Méně to však vyhovuje pro průmyslové použití sběrnice a při výrobě malosériových a kusových zařízení, jako jsou zmíněné medicínské přístroje, zemědělské mechanismy atd. V roce 1992 proto Holger Zeltwanger, v té době šéfredaktor časopisu VMEbus, založil skupinu, které dal do vínku být neutrálním sdružením starajícím se o další technický i marketingový rozvoj sběrnice CAN. Tato skupina se jmenuje CAN in Automation (CiA). Již po několika týdnech vydala CiA první dokument, který se týkal fyzické vrstvy sběrnice. Doporučila v něm jediné – striktně dodržovat ISO 11898. Z trhu tak postupně zmizeli výrobci dodávající komponenty pro CAN pracující na úrovni RS-485. Druhým, obtížnějším úkolem bylo standardizovat aplikační vrstvu CAN. I to se nakonec podařilo, vyšla tzv. Zelená kniha se specifikací CAL – CAN Application Layer. [10]

2.2. Sběrnice LIN

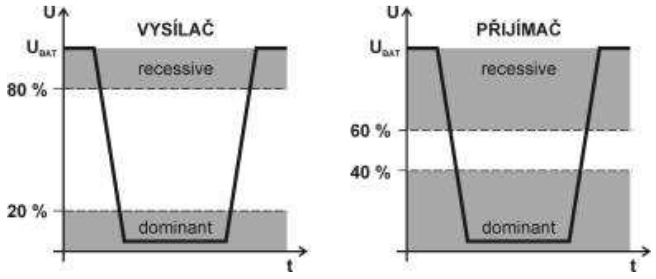
Sběrnice LIN je sériová asynchronní sběrnice používající ke komunikaci jednovodičové spojení připojených zařízení. Je navržena pro použití v automobilové technice s ohledem na minimální cenové náklady spojené s její aplikací. Nemá za cíl nahradit v automobilech dnes hojně

používanou spolehlivou, robustní a rychlou sběrnicí CAN, ale má pokrýt množinu aplikací, pro které je použití sběrnice CAN přílišným luxusem, nebo zatím nebyly z cenových důvodů napojeny na elektronický řídící systém automobilu. Cena vynaložená na propojení s lokální sítí automobilu má být 2 až 3 nižší ve prospěch LINu. Zdálo by se, že je to částka vzhledem k ceně vozu zanedbatelná, musíme si ale uvědomit, že elektronika je dnes v automobilech téměř všude, a že každé ušetřené euro je při sériové výrobě, kdy se vyrábí stotisícové série, velmi důležité a stává se konkurenční výhodou daného výrobce. Výrobce totiž může za stejnou cenu zákazníkovi nabídnout produkt s vyšší užitnou hodnotou. Sběrnice LIN byla původně určena pro automobilový průmysl, brzy však našla uplatnění i v jiných aplikacích jako třeba automatizační technika, měřící technika nebo i spotřební (bílá) elektronika. Dá se předpokládat, že nastane podobná situace jako u sběrnice CAN a sběrnice LIN se stane podobně rozšířenou a používanou. To záleží především na komerčních tlacích na trhu, je však pravděpodobné, že objem počtu vyráběných čipů pro sběrnici LIN a množství implementací driverů této sběrnice, jakožto implementace přenosových protokolů a tím další zlevnění výsledného produktu, bude příznivě působit na používání sběrnice. Tomuto trendu napomáhá i skutečnost, že LIN je otevřený standard sériové automobilové sběrnice třídy A. Současné aplikace vycházejí převážně z oblasti automobilového průmyslu. Jedná se především o ovládání a polohování zrcátek, stahování oken, ovládání zámků dveří a střešního okna, polohování sedadel, ovládání klimatizace, stěračů nebo osvětlení. LIN zde realizuje propojení čidel, ovladačů, akčních členů a indikátorů. Tyto jednotky pak mohou být jednoduše napojeny na síť automobilu a stávají se dostupné pro všechny typy diagnostiky při servisních pracích [12].

2.2.1.Fyzická vrstva

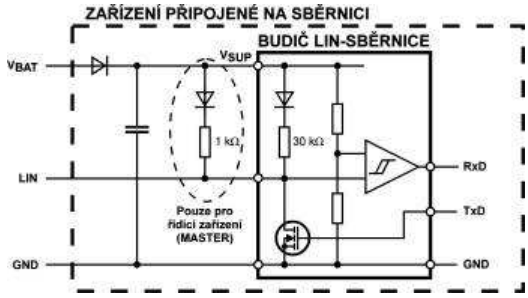
Fyzická vrstva byla odvozena od standardu ISO 9141. Ten byl vyvinut pro diagnostické účely pro použití v servisech. Je zřejmé, že pokud má být fyzická vrstva tohoto standardu použita pro jedoucí automobily, musely být navrhnuty určité změny. Např. strmost náběžných a sestupných hran je záměrně omezena s ohledem na minimalizaci množství vyzařovaného rušení. Také jsou změněny rozhodovací úrovně pro zlepšení odolnosti proti rušení a je pamatováno i na posun zemního potenciálu a různé poruchy. Princip sběrnice LIN je v použití jednoho vodiče pro obousměrnou komunikaci pomocí realizace funkce logického součinu prostřednictvím spínačů a rezistorů zapojených na LIN sběrnici v každém připojeném zařízení. Jsou definovány dvě vzájemně komplementární hodnoty stavů na sběrnici a to dominant a recessive. Velikosti a rozsah jednotlivých úrovní jsou vztaženy relativně k palubnímu napětí v automobilu generovanému akumulátorovou baterií 12V. Hodnoty těchto úrovní jsou ukázány na Obrázek 8: Rozhodovací úrovně vysílače a přijímače LIN [12]. Spínače při sepnutí spojují sběrnici se zemí, a stačí aby byl sepnut alespoň jeden z nich a sběrnice přejde do stavu dominant, což představuje

stav logické nuly. Rezistory zapojené mezi napájecí napětí a sběrnici pak na ní udržují, pokud není žádný spínač sepnutý, stav recessive, tedy logickou jedničku [12].



Obrázek 8: Rozhodovací úrovně vysílače a přijímače LIN [12]

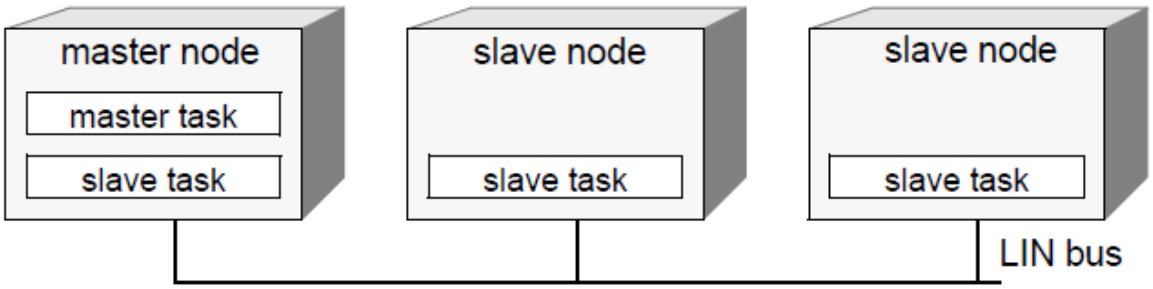
Zjednodušené schéma zapojení budiče sběrnice je na Obrázek 9: Zjednodušené schéma budiče sběrnice v zařízení připojeném na sběrnici LIN [12]. Vodiče VBAT a GND slouží k napájení budiče i vlastního zařízení. Pro případ přerušení dodávky napájecího napětí do zařízení připojeného na sběrnici jsou rezistory definující stav recessive zapojeny v sérii s ochranou diodou. Ta zabrání nedefinovanému napájení jednotky po vodiči LIN sběrnice. Obecně jsou budiče sběrnice LIN konstruovány tak, aby byly odolné proti různým poruchovým stavům, které se mohou vyskytnout. Velikosti těchto rezistorů mají jmenovitou hodnotu 30 kW. Maximální počet zařízení připojených na sběrnici je teoreticky omezen jen počtem volných identifikátorů. Ve skutečnosti je však nutné brát zřetel na elektrické požadavky, které počet striktně omezují. Maximální počet zařízení připojených na sběrnici by neměl překročit 16. Aby se s počtem připojených budičů razantně neměnila velikost výsledného odporu připojující sběrnici na napájecí napětí, je definováno, že u zařízení typu master, které je na sběrnici vždy jen jedno, je kromě interního rezistoru, zapojen navíc externí rezistor o hodnotě 1 kW. Z toho též vyplívá, že budiče sběrnice (integrované obvody) jsou pro master a slave stejné a liší se právě jen externími součástkami. Pro zvýšení odolnosti proti elektromagnetickému rušení se paralelně k vývodu LIN budiče připojují kondenzátory, někteří výrobci dokonce doporučují dolní propusti prvního (RC) nebo druhého řádu (LC). Maximální celková kapacita sběrnice 10 nF, ale nesmí být překročena. Maximální délka sběrnice je udávána 40 m [12].



Obrázek 9: Zjednodušené schéma budiče sběrnice v zařízení připojeném na sběrnici LIN [12]

2.2.2. Operační koncept sběrnice LIN

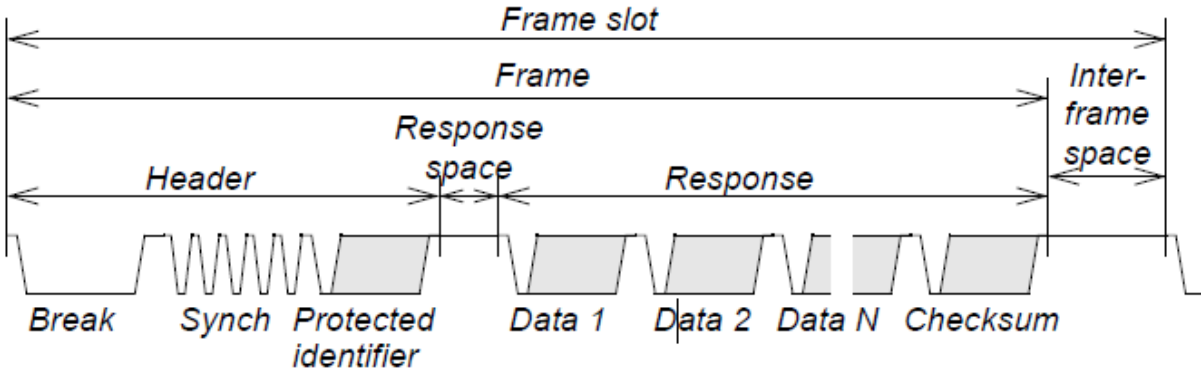
Každý cluster sběrnice se skládá z uzlu typu master a několika uzlů typu slave. Uzel typu master je řídícím uzlem a dokáže obsluhovat jak úlohu master, tak úlohu slave. Úloha master znamená schopnost řídit komunikaci na sběrnice a generovat dotazy na ostatní uzly. Pro bezkolizní provoz je nutné, aby v jednom clusteru byl pouze jeden master. Uzly typu slave obsluhují pouze úlohu slave. Uzly typu slave se můžou účastnit komunikace ve více než jednom clusteru.



Obrázek 10: Uzly sběrnice LIN [13]

2.2.3. Rámec zprávy LIN

LIN používá jednotný formát rámce zprávy, který slouží k synchronizaci, adresaci uzlů a k výměně dat mezi nimi. Formát rámce zprávy je zobrazen na Obrázek 11: Formát rámce zprávy [13]. Řídící jednotka (master) začíná komunikaci, určuje přenosovou rychlost a vysílá hlavičku (Header) rámce zprávy. Ostatní jednotky, ale i jednotka master mohou vysílat odpověď složenou z datových bajtů a kontrolního součtu. Hlavička začíná synchronizačním impulsem (Synch) a následným synchronizačním polem. Toto pole slouží k zasynchronizování podřízených jednotek (slaves) na bitovou rychlost jednotky master. Tyto jednotky tak vystačí s jednoduchým zdrojem časové základny v podobě RC oscilátoru, což má kladný vliv na cenu jednotek slaves. [12]

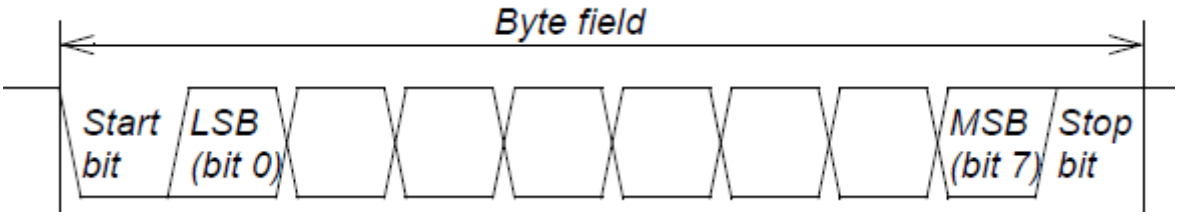


Obrázek 11: Formát rámce zprávy [13]

Pro komunikaci lze použít maximální přenosovou rychlost 20 kbit/s, která představuje horní limit při použití jednodrátového přenosového vedení vzhledem k EMI. Minimální přenosová rychlost je stanovena na 1kbit/s z důvodu předcházení problémům s praktickou implementací

time-out period. Aby se usnadnila implementace do levných LIN zařízení, jsou k použití doporučeny 3 přenosové rychlosti. Pomalá rychlost 2,4kbit/s; střední rychlost 9,6kbit/s; vysoká rychlost 19,2kbit/s [12].

Jako zdroj signálu pro komunikaci lze použít standardní UART interface s bajtovým polem (viz Obrázek 12: Základní bajtové pole [13]). Jediná výjimka komunikace je synchronizační impuls, který vysílá pouze master. Doba trvání tohoto impulsu odpovídá vyslání minimálně 13 bitů a je většinou generován softwarově. Tento vzor je jednoznačně identifikovatelný, protože jeho délka je větší než jakákoliv standardní sekvence na sériovém kanále. Začátek zprávy je tak bezpečně rozpoznatelný a poskytuje podřízeným jednotkám dostatek času pro zachycení komunikace na sběrnici v libovolném stavu podřízené jednotky [12].



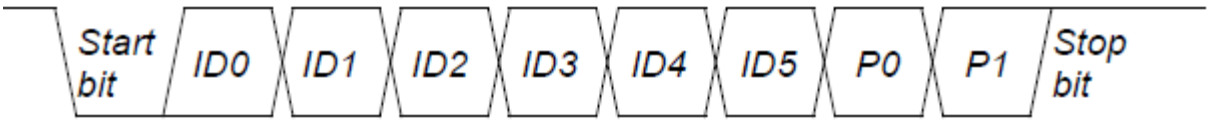
Obrázek 12: Základní bajtové pole [13]

Posledním blokem hlavičky je identifikační pole (Obrázek 11: Formát rámce zprávy [13]). Tento identifikátor v sobě nese informaci o odesílateli, příjemci či příjemcích a délku datové části. Je zabezpečen pomocí dvou paritních bitů a to bitu 6 a bitu 7. Jejich hodnoty jsou získány jako XOR kombinace významových bitů identifikátoru podle rovnic (1) a (2) [12].

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \quad (1)$$

$$P1 = (\overline{ID1 \oplus ID3 \oplus ID4 \oplus ID5}) \quad (2)$$

Vlastní datová část přenosu je pak zabezpečena invertovanou hodnotou součtu s přenosem přes tyto bajty. V nové verzi LIN protokolu je pak možnost započítat do tohoto součtu ještě i bajt identifikátoru a tím dále zvětšit zabezpečení přenosu proti chybám [12].

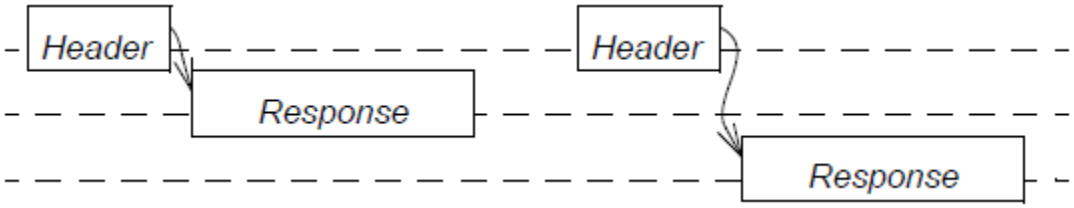


Obrázek 13: Rozmístnění jednotlivých bitů identifikátoru [13]

Z uvedeného vyplývá, že je možné rozlišit kombinace šesti bitů, tedy 64 různých identifikátorů. Ty se pak dělí do čtyř kategorií podle významu a použití. Identifikátory 0 – 59 jsou určeny pro přenos obecných dat, ID 60 a ID 61 jsou rezervovány pro diagnostická data, ID 62 je rezervován pro uživatelem definovaný rámec a ID 63 je rezervován pro budoucí použití.

2.2.4.Komunikace na LIN sběrnici

Na Obrázek 14 je ukázán příklad komunikace řídícího zařízení a dvou podřízených zařízení. Je zde vidět, kdo vysílá jaká data a jaký je výsledný průběh na sběrnici LIN. Každou komunikaci zahajuje master vysíláním hlavičky, která v sobě zahrnuje i informaci o zdroji a příjemci zprávy. V prvním případě je zdrojem dat samotný master a proto po odvysílání hlavičky zahájí i vysílání dat v tomto případě pro jednotku slave 1. V druhém případě jsou data vyžadována od jednotky slave 1, která je začne vysílat po přijetí hlavičky od jednotky master. Pokud jednotka master požaduje data od jiné jednotky slave, musí opět vyslat hlavičku s příslušnou informací o zdroji a příjemci dat. Je také možné, aby příjemci zprávy byly všechny připojené jednotky.



Obrázek 14: Příklad komunikace Master – Slave[13]

Sběrnice LIN je vždy řízena jediným uzlem typu master. Aby nedocházelo ke konfliktům v komunikaci, je nutné vytvořit komunikační rozvrh s definovanými délkami slotů.

2.2.5.Druhy datových rámců

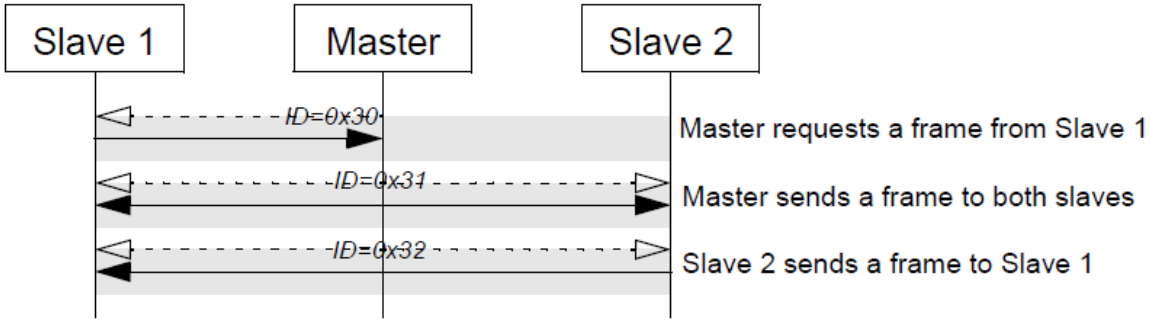
Pro komunikaci se používá několik druhů datových rámců, z nichž každý může být vyslán za určitých podmínek. Druh rámce se určuje podle identifikátoru, který rámec obsahuje (viz tab. 1).

Hodnota identifikátoru	**Význam a název**
0x00 – 0x3B	Přenos obecných dat - Nepodmíněný rámec
0x3C – 0x3D	Přenos diagnostických dat – Diagnostický rámec
0x3E	Uživatelem definovaný rámec – Uživatelský rámec
0x3F	Rezervováno pro budoucí využití

Tabulka 1: Datové rámce LIN

2.2.5.1. Nepodmíněný rámec

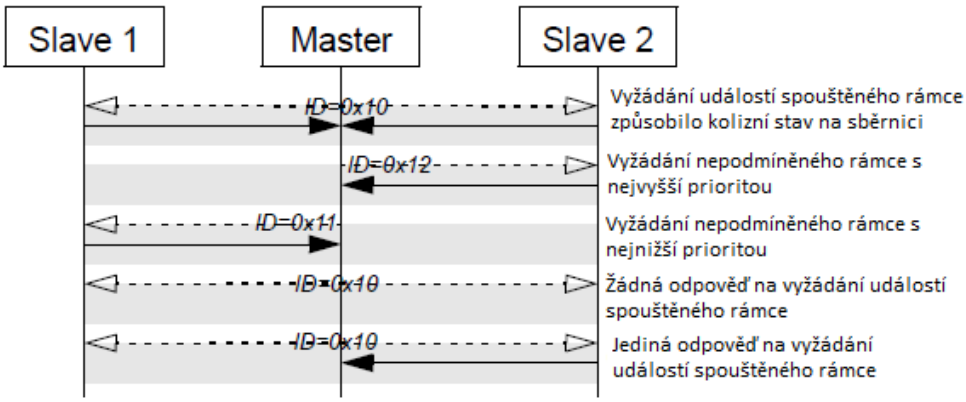
Nepodmíněný rámec vždy nese informaci a jeho identifikátor je v rozmezí 0 až 0x3b. Do této skupiny rámců patří i rámce spouštěné událostí a sporadické rámce. Hlavička tohoto rámce je odvysílány vždy, když je zpracováván slot obsahující tento rámec. Uzel typu slave vždy odpoví na přijatou hlavičku. Nepodmíněné rámce jsou v datové komunikaci nejběžnější a jsou tedy hlavními nositeli informací. Na obrázku 15 je znázorněna komunikace mezi masterem a dvěma uzly typu slave. Je zde znázorněn přenos třech nepodmíněných rámců. Přenos je vždy iniciován uzlem master a má jednoho nebo více odběratelů.



Obrázek 15: Sekvence 3 nepodmíněných rámců [13]

2.2.5.2. Událostí spouštěný rámec

Pro přenos obecných dat lze využít datové rámce spouštěné událostí. Tyto se používají pro zlepšení odpovědí sběrnice na události, které je potřeba hlídat, ale nastávaní zřídka. Identifikátor těchto rámců je v rozmezí 0x00 – 0x3B. Tento druh rámců je schopen přenášet data z jednoho nebo více nepodmíněných rámců. První datový byte je roven chráněnému identifikátoru. To znamená, že lze přenést maximálně 7 datových bajtů. Díky tomuto datovému rámce lze šetřit šířku pásma. Pokud je s tímto rámcem spojen více než jeden nepodmíněný rámec (což je běžné) měly by mít stejnou délku a využívat stejnou funkci výpočtu kontrolního součtu. Hlavička rámce je odvysílána vždy, když je zpracováván daný slot v rozvrhu odesílání. Odpověď je však odvysílána pouze pokud došlo ke změně některého z přenášených signálů. Pokud žádný z uzlů typu slave neodpoví na danou hlavičku, je tato ignorována. Pokud odpoví více uzlů zároveň, musí master node vyřešit kolizní stav vyžádáním všech kolizních rámců před odvysíláním dalšího událostí spouštěného rámce.

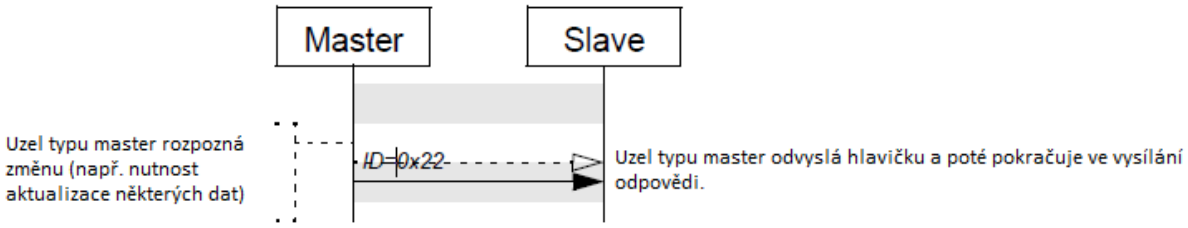


Obrázek 16: Událostí spouštěné rámce [13]

2.2.5.3. Sporadický rámec

Účelem sporadických rámců je vnést trochu dynamického chování do deterministického systému časování sběrnice bez ztráty časové návaznosti na zbytek přenosového rozvrhu. Sporadické rámce nesou vždy data a jejich identifikátor je v rozmezí 0 – 0x3B. Hlavička sporadického rámce je odvysílána pouze tehdy, pokud uzel typu master rozpozná, že data

přenášená rámcem byla modifikována. Po odvysílání hlavičky, pokračuje master node ve vysílání odpovědi. Všichni příjemci sporadického rámce by měli přijmout data, a pokud jsou data validní zpracují je dle dané konfigurace.



Obrázek 17: Sporadický rámec

2.2.5.4. Diagnostický rámec

Diagnostický rámec je použit pro přenos konfiguračních nebo diagnostických dat. Vždy obsahuje 8 datových bajtů. Do této skupiny rámců patří i speciální rámce využité k ovládání napájení uzlů. Každý uzel lze přepnout do sleep modu, který slouží k úspoře energie. K přepnutí se používá go-to-sleep command, pro který je vyhrazen identifikátor 0x3C a jehož první datový bit musí být nulový. K přepnutí do normálního stavu se využívá wake-up command, který má identifikátor 0x3D.

2.2.5.5. Uživatelem definovaný rámec

Pro tento rámec je vyhrazen identifikátor 0x3E a může nést jakoukoli informaci. Hlavička je odvysílána vždy, když je zpracováván slot určený pro tento rámec.

2.2.5.6. Rezervovaný identifikátor

Identifikátor 0x3F by neměl být dle standardu LIN 2.0 využit. Očekává se, že bude v budoucnu využit.

2.2.6. Rozvrhové tabulky

Rozvrhové tabulky jsou základním stavebním kamenem protokolu LIN. Obsahem tabulky je definice rámců, jejich identifikátorů a pořadí. Tabulka jasně určuje provoz na sběrnici, a pokud je správně vytvořena, nemůže dojít k přetížení sběrnice. Protože je komunikace v každém clusteru ovládána jedním uzlem typu master, je tento uzel zcela zodpovědný za provoz na sběrnici. Pokud řídící uzel zpracovává správně sestavenou rozvrhovou tabulku, vzniká záruka stabilního, deterministického přenosu. V rámci jednoho rozvrhu může být definováno několik rozvrhů, mezi kterými lze přepínat. Správně vytvořený rozvrh je také zárukou toho, že nedojde k narušení periodičnosti dotazování. Rozvrh je definován jako časová posloupnost slotů. Délka slotu musí být definována tak, aby byla schopna pojmout rámec při jeho maximální délce.

Při výpočtech se vychází z následujících rovnic:

$$T_{Header\_Nominal} = 34 * T_{Bit} \quad (3)$$

$$T_{Response\_Nominal} = 10 * (N_{Data} + 1) * T_{Bit} \quad (4)$$

$$T_{Frame\_Nominal} = T_{Header\_Nominal} + T_{Response\_Nominal} \quad (5)$$

Vzhledem k protokolu LIN 2.0 může dojít k prodloužení přenosových časů až ok 40% vzhledem k nominální délce. Proto je počítat s maximálními časy:

$$T_{Header\_Maximum} = 1.4 * T_{Header\_Nominal} \quad (6)$$

$$T_{Response\_Maximum} = 1.4 * T_{Response\_Nominal} \quad (7)$$

$$T_{Frame\_Maximum} = T_{Header\_Maximum} + T_{Response\_Maximum} \quad (8)$$

Z těchto vzorců je patrné, že každý slot musí mít délku nejméně $T_{Frame\_Maximum.}$ Čas T_{Bit} vychází z definic fyzické vrstvy. Jedná se o čas potřebný k přenosu jednoho bitu. Proměnná N_{Data} označuje počet bajtů.

3. PROGRAMOVÝ VYBAVENÍ

3.1.FreeMASTER

FreeMASTER je uživatelsky přívětivé realtimové debugovací monitorovací a vizualizační nástroj pro vývoj aplikací a informační management. Pomocí sofistikovaného monitorovacího systému umožňuje grafické či textové znázornění průběhu proměnných. Dále umožňuje přímé ovlivnění proměnných, jejich nastavování a hlídání jejich hodnot. Grafické prostředí je založeno na HTML, čímž umožňuje jednoduché vytvoření přehledného monitorovacího prostředí. Propojení HTML a FreeMASTERu je realizováno pomocí doplňkových skriptů (javascript, VBScript). Je velmi dobře použitelný pro jednoduché demonstrace kódů.

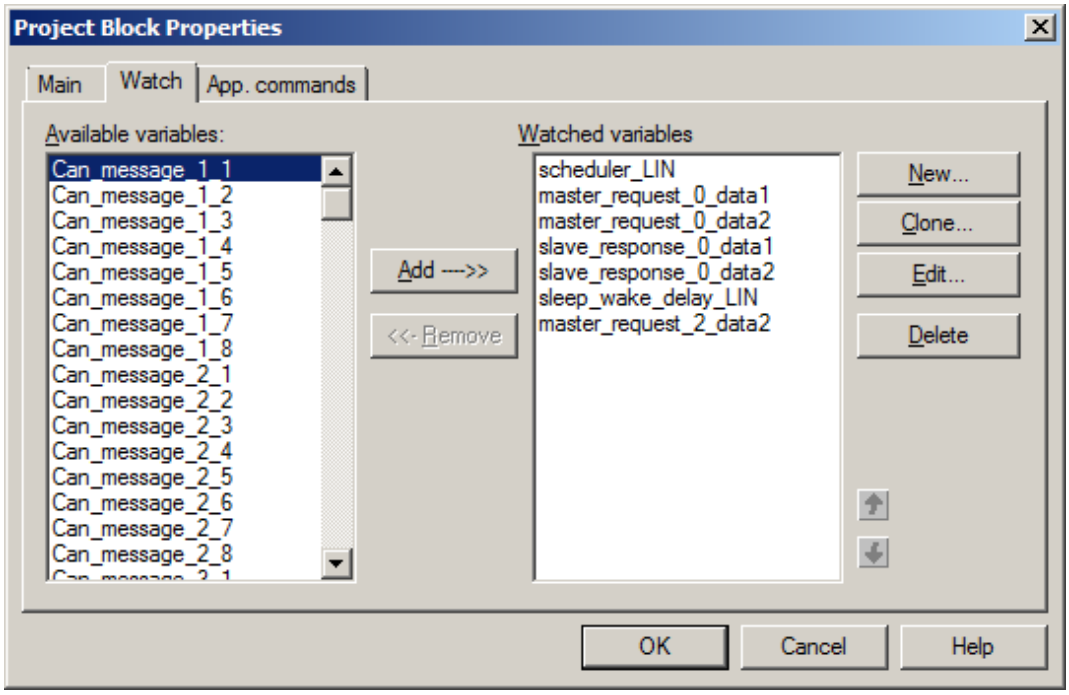
3.1.1. Základní rozhraní

FreeMASTER je volně dostupný na stránkách www.freescale.com. Instalace je velmi jednoduchá a intuitivní. Program vyžaduje pouze 110MB volného místa na disku, Internet Explorer verze 4.5.5. nebo vyšší a operační systém Windows 98 nebo novější. Aplikace je k dispozici jako jediný samostatný samorozbalitelný spustitelný soubor. Propojení elektroniky a počítače může být realizováno pomocí USB nebo RS-232. Pro nastavení komunikace je doporučeno použít některou z ukázkových aplikací publikovaný na webu firmy Freescale. Po instalaci se zobrazí okno základního prostředí. Levá část okna je vyhrazena pro přehled jednotlivých objektů, seřazených do přehledné stromové struktury. V této části lze přidávat objekty (Scope, recorder, subblock), které se hierarchicky skládají. Střední část je vyhrazena pro vložení HTML stránky. Předpokládá se, že tato stránka bude sloužit k rychlému a přehlednému ovládání proměnných. Stránku zle tvořit v libovolném editoru a poté ji nahrát na FreeMASTERu. Dolní část je vyhrazena textovému monitorování proměnných. V této části je zobrazeno symbolické jméno proměnné, její hodnota, jednotky a vzorkovací frekvence. Zobrazení lze jednoduše měnit a upravovat. Tato část je velmi užitečná před zavedením grafického rozhraní a odladěním skriptů, které jsou vykonávány v rámci HTML stránky.

Obrázek 18: Základní prostředí FreeMASTER

3.1.2. Sledování proměnných

Pro zajištění sledování proměnných je potřeba udělat několik snadných kroků. Každá proměnná vytvořená v kódu, který se nahrává do mikroprocesoru, musí mít povolení komunikovat s FreeMASTEREM. Díky tomuto nastavení nepřistupuje FreeMASTER ke všem proměnným. V samotném vývojovém prostředí je pak nutné zařadit monitorovanou proměnnou mezi dostupné. Z dostupných je poté možné vybrat proměnné, které se mají zobrazit v oblasti pro textové sledování proměnných. Pokud je povolena editace těchto proměnných, je možné měnit jejich hodnoty. Výhodou je nastavení zobrazovací soustavy (DEC, BIN, HEX, REAL, ASCII). V podrobném menu lze nastavit mnoho různých voleb povolené týkající se editace a zobrazení každé proměnné.



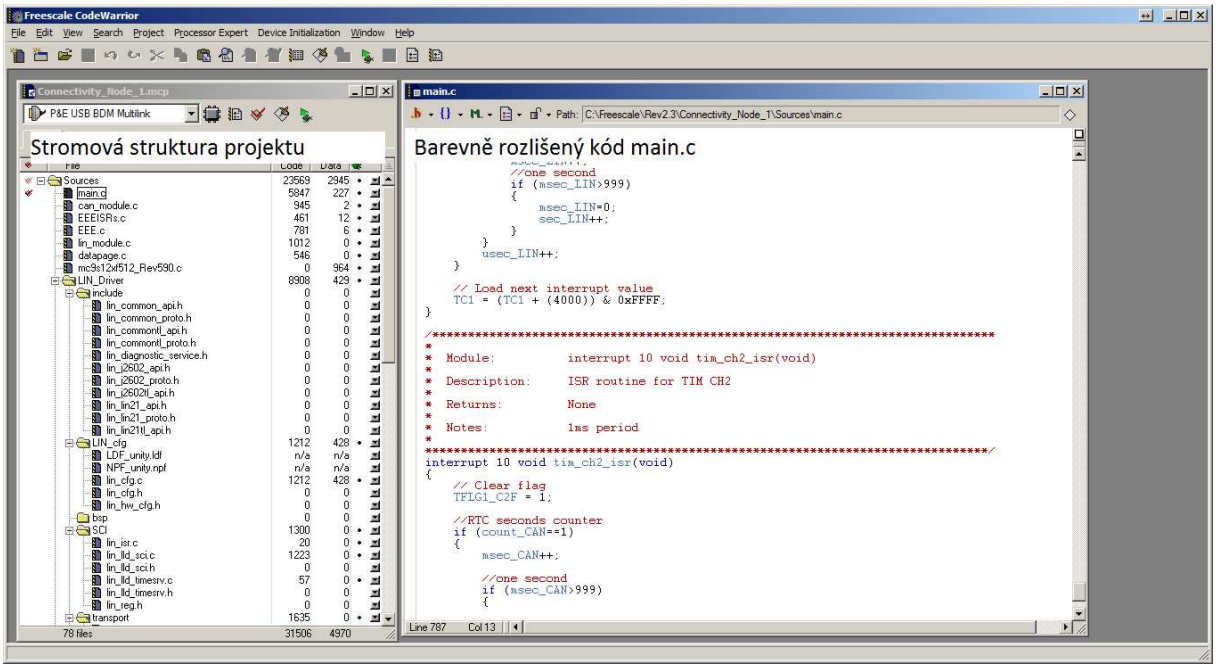
Obrázek 19: Sledované proměnné

3.2. CodeWarrior

CodeWarrior je kompletní vývojový nástroj založený na platformě Eclipse IDE. Je vyvíjen firmou Freescale a pokrývá významnou část výrobního portfolia firmy. Nabízí přehledné prostředí vhodné pro vývoj programů, doplněné širokým spektrem funkcí. V kombinaci s komunikačním rozhraním USB Multilink tvoří velmi účinný a efektivní prvek vývoje aplikací a programovaní mikroprocesorů. Kromě textového editoru a kompilátoru disponuje mnoha nástroji pro zjednodušení vývoje aplikací, jako například nástrojem pro konfigurování periférií mikrokontroleru. Jeho nedílnou součástí je velmi užitečný vestavěný debugger. Software je distribuován v několika verzích od verzí zdarma až do profesionálních vývojových studií za tisíce dolarů. Stejně jako FreeMASTER je tento software dostupný na webu www.freescale.com.

3.2.1. Základní rozhraní

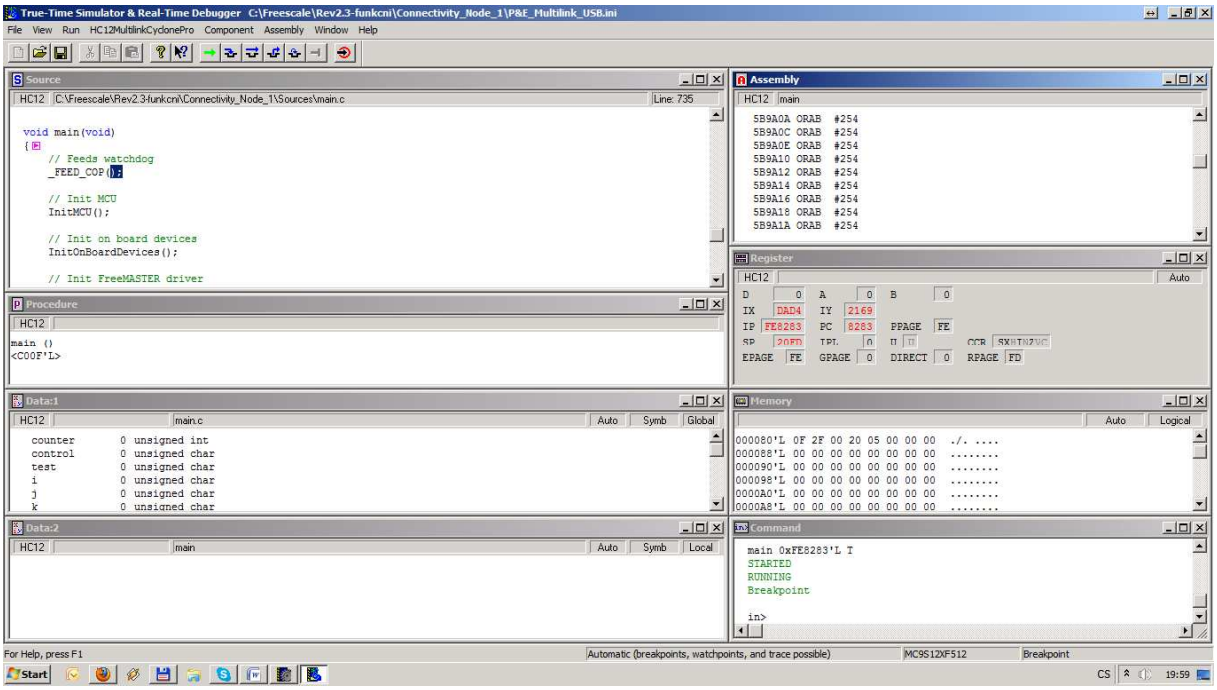
Základem práce s CodeWarriorem je vytvoření projektu, které funguje jako organizační celek. Do projektu lze poté přidávat jednotlivé části kódu, knihovny a konfigurační soubory. Všechny soubory lze prohlížet ve stromové struktuře a otevírat přímo v programovacím prostředí. CodeWarrior má integrované rozpoznání kódu, díky kterému barevně odlišuje názvy proměnných, komentáře a příkazy.



Obrázek 20: CodeWarrior - základní rozhraní

3.2.2. Debuggovací prostředí

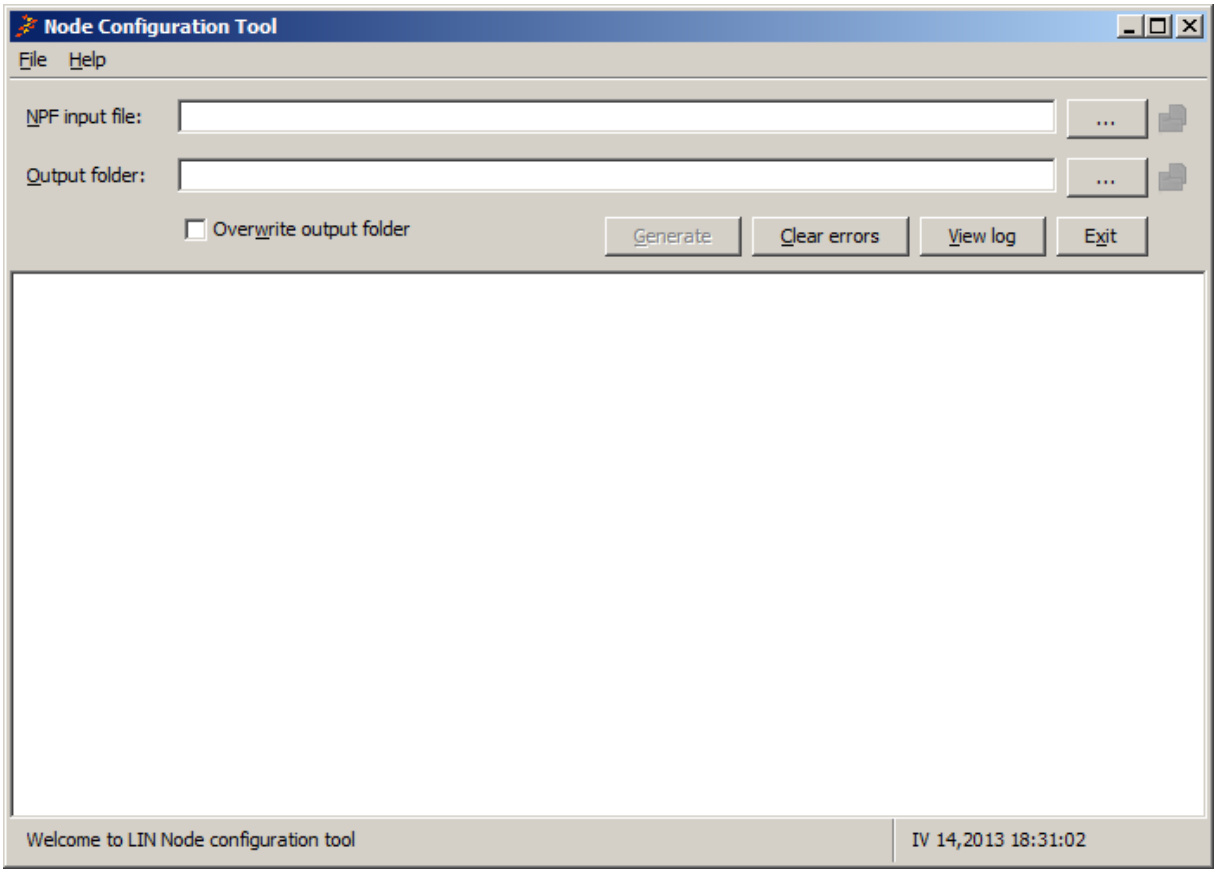
Součástí Codewarrioru jsou kompilovací a debuggovací prostředí pro celou řadu programovacích jazyků. Hlavní zaměření je na jazyky C, C++ a Pascal. Debugger je spuštěn při každém nahrávání programu do mikroprocesoru. Otevře se v novém okně a jeho složen z několika samostatných funkčních celků. Okno zdrojového kódu, okno funkcí, dvě datová okna, okno Assembly, okno registrů, okno paměti a okno příkazů. Každé z oken je možno maximalizovat a pozorovat v něm průběh programu. V horní liště se nachází tlačítka pro ovládání programu. Program můžeme spustit najednou, krokovat po jednotlivých instrukcích, přeskakovat instrukce, vyskakovat z funkcí, atd.



Obrázek 21: Debuggovací prostředí

3.3. Node configuration tool

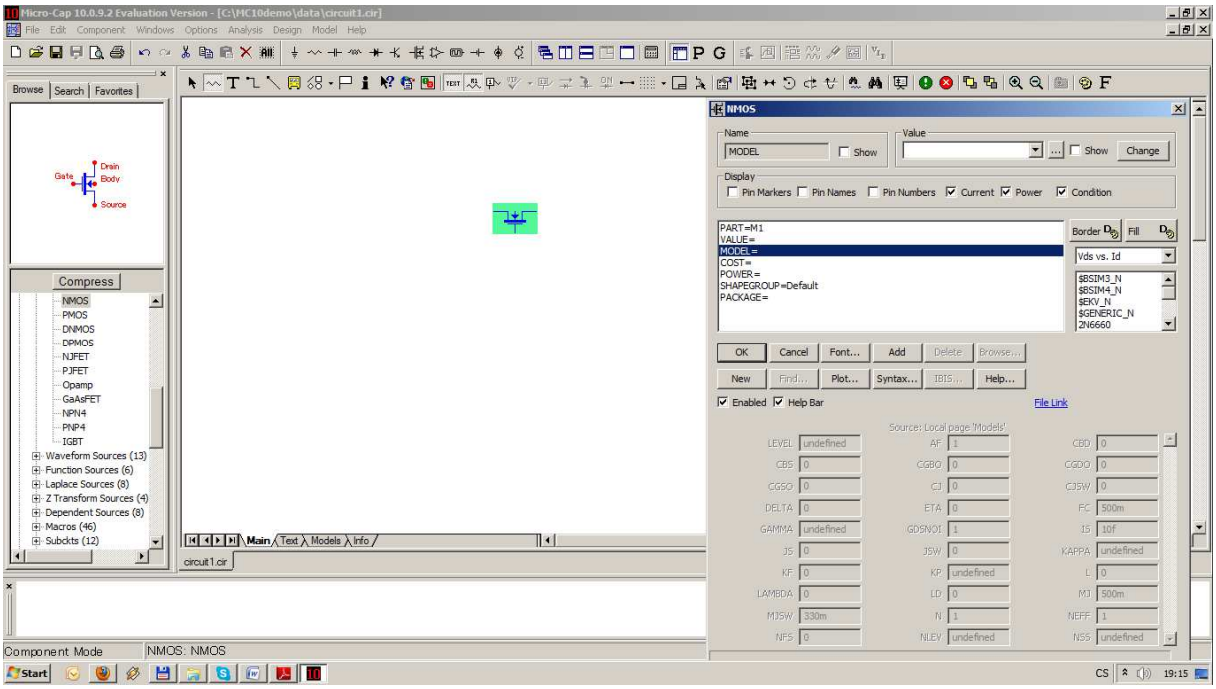
Jedná se o jednoúčelový program vyvinutý firmou Freescale pro generování rozvrhových tabulek protokolu LIN. Vzhledem k jednoúčelovému využití je grafické prostředí velmi jednoduché a přehledné. Program je napsán v programovacím jazyce Java a v současné době je aktuální verze 1.2 vydaná 13. 10 2011. Funkce programu spočívá ve vytvoření souborů lin_cfg.c, lin_cfg.h a lin_hw_cfg.h, které jsou dále využitelné v programování komunikace sběrnice. K úspěšnému vygenerování těchto souborů je potřeba vstupního textového dokument node private file. Tento dokument se poté může odkazovat na LIN Description file, což je soubor specifikací datových rámců a rozvrhových tabulek. Pro generování stačí zadat umístění vstupních a výstupních souborů a stisknout tlačítko „Generate“. Pokud generování skončí chybou je uživatel upozorněn chybovou hlášku, která pomáhá lokalizovat problém. Jsou známy problémy kompatibility programu s 64bitovými operačními systémy. Tyto problémy lze většinou řešit instalací správné verze Javy.



Obrázek 22: Okno Node Configuration Tool

3.4. MicroCap

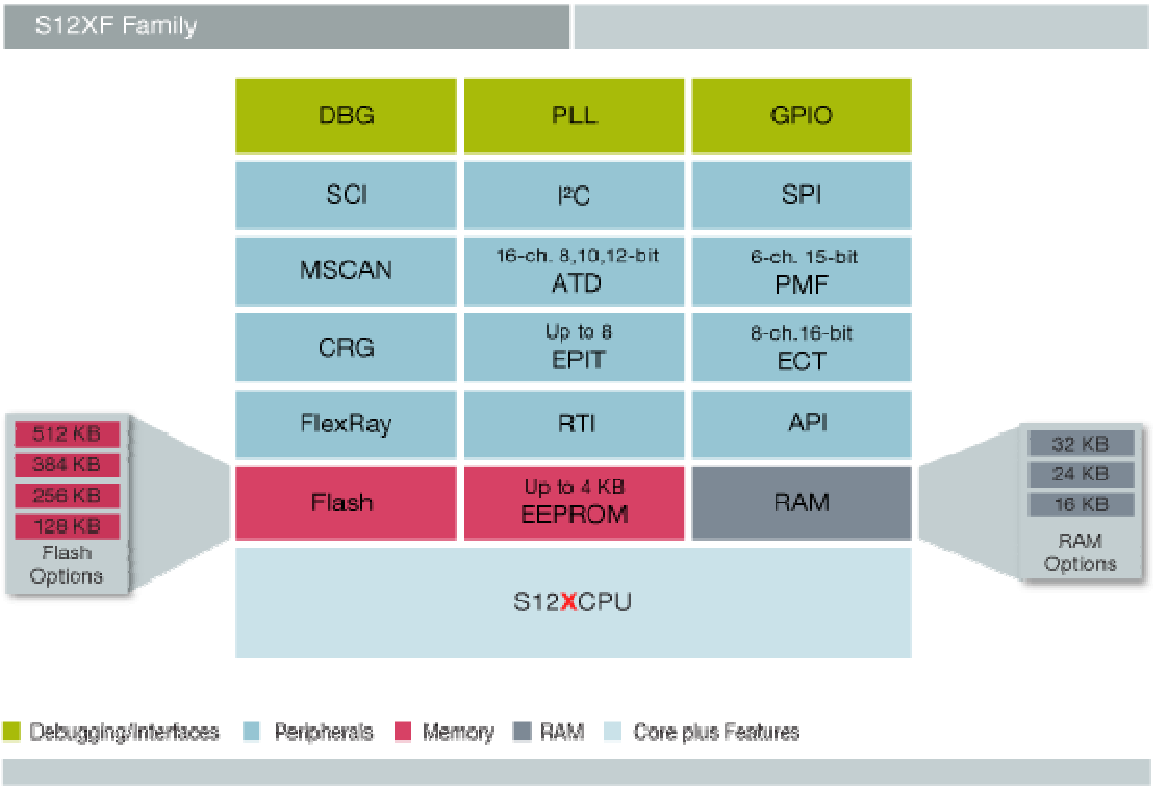
Jedná se o program k simulaci chování analogových a digitálních elektronických obvodů. Přehledné grafické rozhraní a mnoho využitelných funkcí dělají z toho programu silný nástroj pro navrhování elektronických obvodů. Volně stažitelná demo verze má omezení rychlosti analýzy, počtu uzlů a součástek, ale pro jednoduché návrhy je dostačující. MicroCap je vyvíjen od roku 1981, kdy byl zaměřen pouze na analogové součástky. Současná verze 10.0.9.2 je stažitelná z webových stránek http://www.spectrum-soft.com. Program nabízí širokou základnu předdefinovaných součástek a velmi dobré možnosti editování vlastností součástek. Program byl využit pro DC analýzu ochranných obvodů.



Obrázek 23: Okno prostředí MicroCap

4. Mikroprocesor MC9S12XF

Práce je založena na mikroprocesoru MC9S12XF firmy Freescale. Mikroprocesor vychází z architektury S12X, která je navržena pro vysoce výkonné a cenově dostupné procesory. MC9S12XF byl navržen pro satelitní uzly sběrnice FlaxRay™. Díky velkému výkonu je vhodný pro velmi rychlou a spolehlivou komunikaci, což jej předurčuje pro nasazení v kritických aplikacích jako ovládání brzdového systému, airbagů a bezpečnosti. Rodina procesorů S12XF je složena ze čtyř vysoce integrovaných mikroprocesorů, které nabízejí široké spektrum paměťových konfigurací a XGATE koprocesor pro zvýšení výkonu. [15]



Obrázek 24: Vlastnosti S12XF procesorů [15]

Základní parametry:

- Operační frekvence: 50MHz
- Interní Flash paměť: 512KB
- Interní RAM paměť: 32KB
- Emulovaná EEPROM: 4KB
- Provozní napětí 3.13V – 5.5V
- Provozní teplota: -40°C – 125°C

Tento koprocesor je dostatečně výkonný, aby byl schopen tvořit virtuální periferie. Mezi nejdůležitější vlastnosti patří podpora FlaxRay™, flexibilní programovatelná hardwarově emulovaná EEPROM, podpora systémové integrity s jednotkou ochrany paměti (MPU). Mikroprocesor se řadí mezí výrobky, u kterých je garantovaná podpora minimálně 15 let. Mezi další užitečné vlastnosti se řadí: [15]

- rozšířený modul přerušení
- 12-ti bitový A/D konvertor s převodním časem do 3us
- rozšířený modul časovačů (ECT)
- časovač periodických přerušení (PIT)
- real-time přerušení (RTI)
- asynchronní periodické přerušení (API)
- pulzní šířková modulace (PWM)
- sériové komunikační prostředí (SPI)
- debuggovací modul (BDM)
- regulace napětí na čipu

5. Profesionální testery LIN a CAN

5.1. CAN4t

CAN4t je univerzální kompaktní systém pro práci v oblasti Controller Area Network (CAN sběrnice), který je distribuován firmou E4T.

5.1.1. Základní funkce

- záznam dat ze sběrnice CAN
- vysílání dat na sběrnici CAN
- ze zaznamenaných dat
- generování periodických zpráv
- monitor sběrnice CAN

5.1.2. Výhody

- bezdrátový přenos Bluetooth s dosahem do 10m
- nízko- a vysokorychlostní CAN rozhraní (CAN-Highspeed i CAN-Lowspeed)
- ukládání dat až do 32GB (SDHC/SD/MMC)
- ukládání záznamů ve formátu .ASC
- podpora CAN databáze .DBC

5.1.3. Příklady použití

- zjišťování závad, které nelze detekovat běžnou diagnostikou
- vývoj a oživování zařízení na sběrnici CAN
- servisní činnosti
- u techniků v oblasti řízení kvality
- výuka a ostatní kvalifikované činnosti vyžadující přístup ke sběrnici CAN

CAN4t nabízí všechny funkce pro vyhodnocení komunikace a obsahu dat včetně analýzy identifikátorů, snímání a odesílání dat v reálném čase.[16]

Obrázek 25: Tester CAN4t pro sběrnice CAN[16]

5.2. BHT Bus Handheld Terminal

Jedná se o malý univerzální terminál určený pro ovládání, testování a diagnostiku sběrnic.

5.2.1. Základní funkce

- Ovládání a testování zařízení po sběrnicích CAN a LIN
- Diagnostiku pomocí K-vedení, CAN nebo LIN sběrnice
- Různé diagnostické protokoly
- Komfortní ovládání tlačítky
- Zobrazování výsledků a hlášení na podsvíceném 4 řádkovém displeji.
- Malé rozměry a variabilita použiti z něj činí vhodného kandidáta na práci mimo kancelář a jeho robustní koncepce zaručí, že jej lze nasadit v průmyslovém prostředí.
- Provedení terminálu je zákaznicky modifikovatelné.

5.2.2. Parametry

- Sběrnice
  - CAN v jedné z variant
  - high speed
  - low speed
  - single wire
  - bez nebo s galvanickým oddělením
  - LIN nebo diagnostické K vedení
- Sériová linka RS232
  - komunikace s programem na PC apod.
  - update firmware
- Paměti
  - programová paměť FLASH ROM - 64 kB
  - datová paměť RAM – 3 kB
  - datová paměť EEPROM – až 128 kB
- Displej
  - alfanumerický LCD 4 řádky x 20 znaků
  - podsvícený
- Klávesnice
  - 5 ovládacích tlačítek (joystick)
  - až 16 ovládacích tlačítek
- Kontrolky/signalizace
  - až 16 LED

- akustický bzučák

- Přídavný modul
  - možnost rozšíření funkce pomocí dalšího modulu elektroniky umístěného uvnitř terminálu
- Konektory
  - DSUB15 – napájení, CAN, LIN, I/O
  - RJ45 – sériová linka
- Technická specifikace
  - napájení: 9 až 15 V
  - proudový odběr: <100 mA
  - rozměry: 150x82x35 mm
  - provozní teplota: -20 až 60 ºC [17]

Obrázek 26: BHT-MFL tester sběrnic [17]

II. PRAKTICKÁ ČÁST

1. Specifikace cílů

Cílem praktické části práce je navržení a sestrojení funkčního univerzálního testovacího uzlu pro sběrnice CAN a LIN. Tento tester je navržen za podpory firmy Freescale a bude sloužit k testování v automobilovém průmyslu. Požadavkem bylo komplexní řešení jak programové části, tak hardwarové části testeru. Bližší specifikace funkcí vychází z požadavků firmy.

1.1. Sběrnice LIN

- Vytvořte „LDF_unity.ldf“ soubor, obsahující zprávy a konfiguraci provozu sběrnice LIN
- Baudrate: 9600Bd
- Zprávy:
    - Datové zprávy:
        - 5 zpráv od uzlu Master do uzlu Slave (ID: 1,2,3,4,5)
        - 5 zpráv od uzlu Slave do uzlu Master (ID: 11,12,13,14,15)
        - Všechny zprávy mají velikost 8 byte
    - Broadcast zprávy:
        - Příkazový rámec Mater regest
        - Příkazový rámec Slave response
- Možnost výběru komunikačního rozvrhu na ovládací stránce, defaultní Rozvrh 1:
    - Rozvrh 1 (Normal Mode):
        - Každých 100ms odeslat / přijmout 5 zpráv
    - Rozvrh 2 (Periodic Sleep and Wake-up command transmit):
        - Periodické odesílání příkazu Wake – up s možností výběru periody (10s, 20s, 30s)
    - Rozvrh 3 (Periodic wake-up, data transmit, goto sleep):
        - Odeslat příkazu Wake – up
        - Čekat na probuzení uzlu od 1ms do 100ms s možností výběru (rozlišení 1ms, defaultní hodnota 20ms)
        - Odeslat / přijmout 5 zpráv
- Ovládací stránka:
    - Zobrazovat LIN Baud rate
    - Zobrazovat ID rámců a data zpráv s možností úpravy dat
    - Na vyžádání ukládat data zpráv do EEPROM mikroprocesoru (10 pozic uložení)
    - Na vyžádání načítat data zpráv z EEPROM mikroprocesoru (10 pozic načtení)
    - Zobrazit chyby LIN
    - Možnost výběru komunikačního rozvrhu

- Možnost úpravy konfigurace komunikace

1.2. Sběrnice CAN

- Vytvořte „can_application_config.h“ obsahující ID zpráv, délku a konfiguraci komunikace na sběrnici
- Baud rate: 500kBd
- Zprávy:
  - Datové zprávy:
    - 5 zpráv odesílaných (ID: 0x7F1, 0x7F2, 0x7F3, 0x7F4, 0x7F5)
    - 5 zpráv přijatých (ID: 0x7E1, 0x7E2, 0x7E3, 0x7E4, 0x7E5)
    - Všechny zprávy mají velikost 8 byte
  - Speciální příkazy:
    - Příkaz Sleep
    - Příkaz Wake – up
- Možnost výběru komunikačního rozvrhu na ovládací stránce, defaultní Rozvrh 1:
  - Rozvrh 1 (Normal Mode):
    - Periodicky odeslat/přijmout 5 zpráv (výběr periody: nejrychlejší, 20ms, 50ms, 100ms, 200ms, 500ms, 1s)Defaultní hodnota 100ms
  - Rozvrh 2 (Periodic Sleep and Wake-up command transmit):
    - Periodické odesílání příkazu Wake – up s možností výběru periody (10s, 20s, 30s)
  - Rozvrh 3 (Periodic wake-up, data transmit, goto sleep):
    - Odeslat příkazu Wake – up
    - Čekat na probuzení uzlu od 1ms do 100ms s možností výběru (rozlišení 1ms, defaultní hodnota 20ms)
    - Odeslat / přijmout 5 zpráv
    - Čekat určený čas od 1ms do 100ms s možností výběru (rozlišení 1ms, defaultní hodnota 10ms)
    - Uspání uzlu
- Ovládací stránka:
  - Zobrazovat LIN Baud rate
  - Zobrazovat ID rámců a data zpráv s možností úpravy dat
  - Na vyžádání ukládat data zpráv do EEPROM mikroprocesoru (10 pozic uložení)
  - Na vyžádání načítat data zpráv z EEPROM mikroprocesoru (10 pozic načtení)
  - Zobrazit chyby LIN
  - Možnost výběru komunikačního rozvrhu

- Možnost úpravy konfigurace komunikace

1.3. Mechanické požadavky

- Moduly umístěte do černé plastové krabičky, které bude opatřena těmito konektory:
  - 12V nízkoproudový vstup (do 2A)
    - Adaptérový konektor, M, pin 2.1mm
    - Vstup opatřete modulem pro ochranu proti přepólování
      - Maximální napěťový úbytek 100mV při 2A
    - Vstup opatřete proudovou pojistkou PP5-T (Ø 5 x 20 mm)
  - 12V vysokoproudový vstup (do 10A)
    - Banánkové konektory (červený - plus, černý - mínus)
    - Vstup opatřete modulem pro ochranu proti přepólování
      - Maximální napěťový úbytek 100mV při 10A
    - Vstup opatřete proudovou pojistkou. Použijte pojistku pro automobilový průmysl
  - USB: UART <-> USB mini konektor
  - LIN konektor
  - CAN konektor

1.4. Ostatní

Vypracujte grafický návod pro ovládání testovacích modulů.

2. Oživení komunikace

2.1. Program FreeMaster

Program FreeMaster je vyvíjen firmou Freescale a slouží zejména k realtimovému debuggování monitorování a vizualizaci. Nástroj je přizpůsoben pro propojení s vývojovým prostředím CodeWarrior. Pro propojení s testovacím uzlem je nutné provést krátké nastavení programu. V nastavení projektu (Project -> Option) je nutné nastavit propojení projektu s portem PC, na který je připojen komunikační uzel. Připojení lze nastavit pro USB nebo COM porty a nabízí detailní konfiguraci. Při propojení s prostředím CodeWarrior je nutné nastavit umístění souboru „Project.abs“, který slouží k synchronizaci proměnných v CodeWarrioru a FreeMasteru. Vhodné je nastavit synchronizaci při každém načtení. Po tomto nastavení a načtení souboru „Project.abs“, můžeme přistoupit k přidání proměnných, které chceme monitorovat či ovlivňovat (viz. 3.1.2). Pro prvotní aplikace stačilo monitorování několika proměnných, které dokazovaly funkčnost sběrnice. Ve finální části práce je pomocí skriptů monitorováno a vizualizováno přes 200 proměnných na každém uzlu.

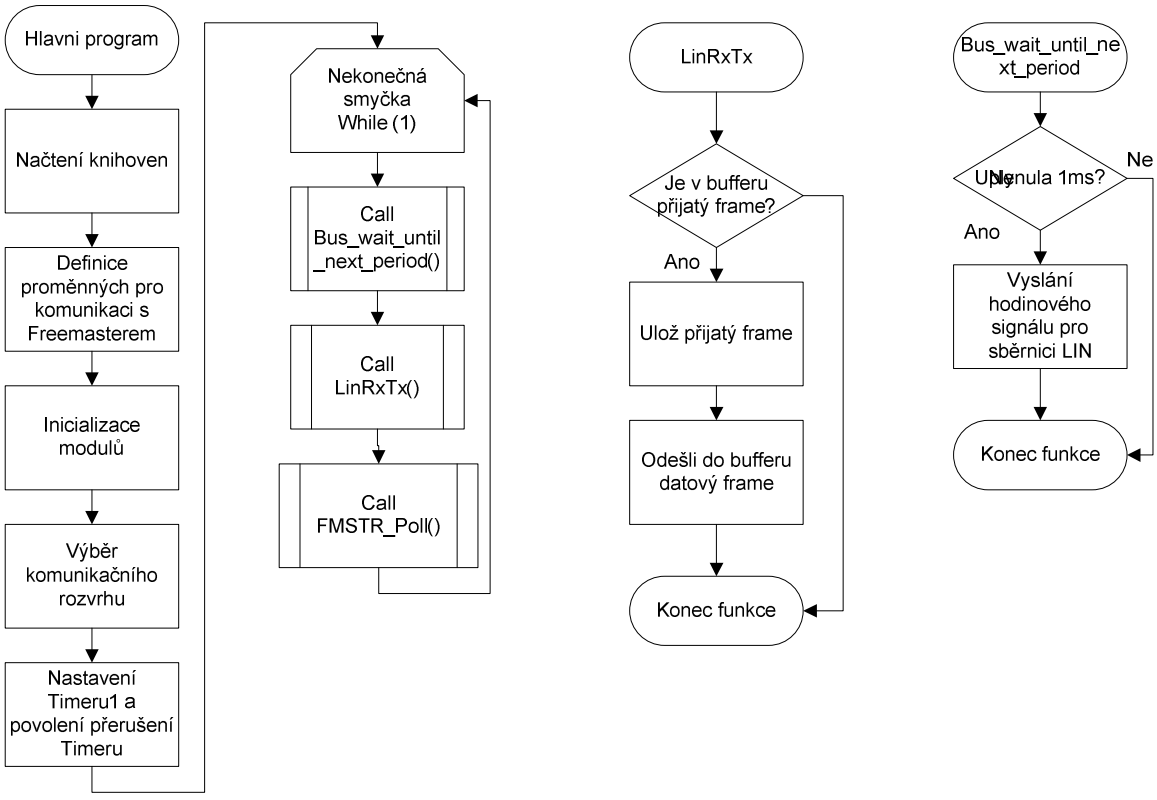
K propojení vývojového prostředí CodeWarriod s programem FreeMaster slouží driver, který je aktuálně ve verzi 1.0.16.0. Driver je nutné zakomponovat do struktury projektu. Po zavedení driveru je nutné definovat, které proměnné se účastní komunikace a udat jejich datový typ. Tato definice je součástí definice proměnných v kódu. Proměnné se zapisují do tabulky, kde je určeno zda FreeMaster může proměnnou číst nebo do ní i zapisovat, název proměnné a její datový typ. V následujícím příkladu je uvedena proměnná s názvem „test“, kterou může FreeMaster číst i do ní zapisovat, datový typ proměnné je osmibitový unsigned integer.

```
FMSTR_TSA_TABLE_BEGIN(table)

    FMSTR_TSA_RW_VAR(test, FMSTR_TSA_UINT8)

FMSTR_TSA_TABLE_END()

FMSTR_TSA_TABLE_LIST_BEGIN()

    FMSTR_TSA_TABLE(table)

FMSTR_TSA_TABLE_LIST_END()
```

2.2. Sběrnice LIN

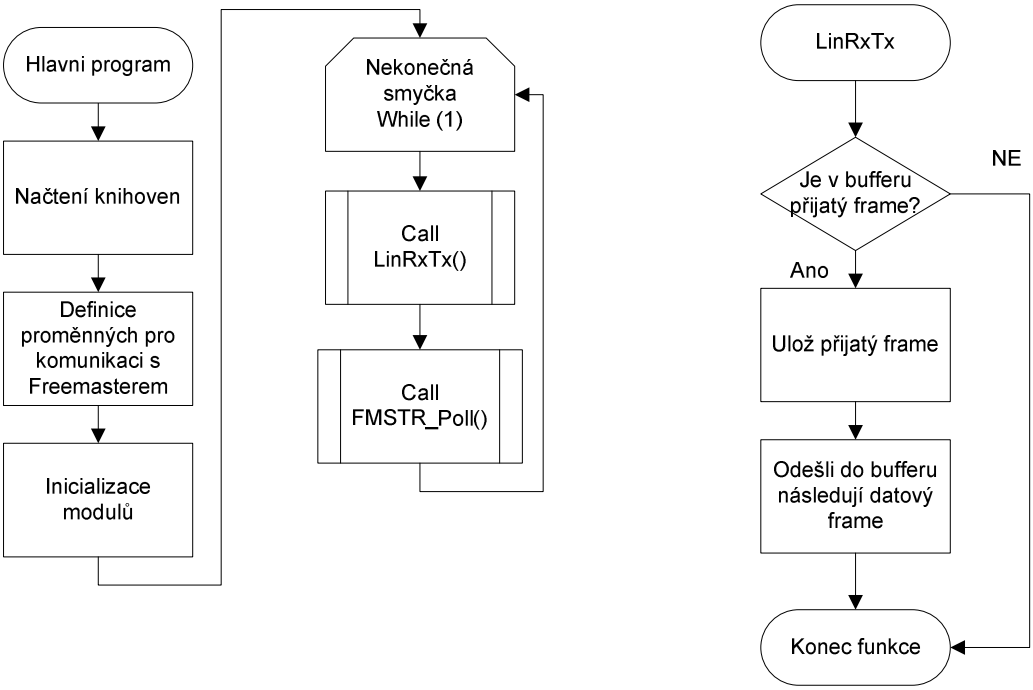
První části práce bylo oživení komunikace LIN. Desky plošných spojů a zkušební zdroj byl zapůjčen zadavatelem práce. Ověření funkčnosti desky a driverů bylo provedeno konzultantem

Ing. Cholastou. První zkušební program ovlivňující provoz na sběrnici měl za úkol pouze odeslání požadavku uzlem Master a zobrazení odpovědi přijaté od uzlu Slave. Pro jednoduchost aplikace nebylo potřeba počítat délku vysílání datových rámců, ani se příliš zabývat rozvrhem sběrnice. První datový rámec byl odeslán uzlem Master po inicializaci a poté byl sledován příznak přijetí zprávy do bufferu. Tento příznak byl nastaven pouze pro jeden identifikátor zprávy, který byl pevně určen v komunikačním rozvrhu. Časování sběrnice bylo prováděno pomocí vestavěného časovače, který vyvolával přerušení každých 100us. Obsluha přerušení poté počítala počet přerušení, čímž byl určen čas 1ms, kterým jsou generovány hodinové pulzy sběrnice. Funkce FMSTR_Poll() slouží k posílání dat do programu FreeMaster a je definována v driveru FreeMasteru, který je vlastnictvím firmy Freescale. Následující diagramy zjednodušeně popisují funkci programu uzlu Master.



Obrázek 27: Vývojový diagram oživovacího programu komunikace LIN uzlu Master

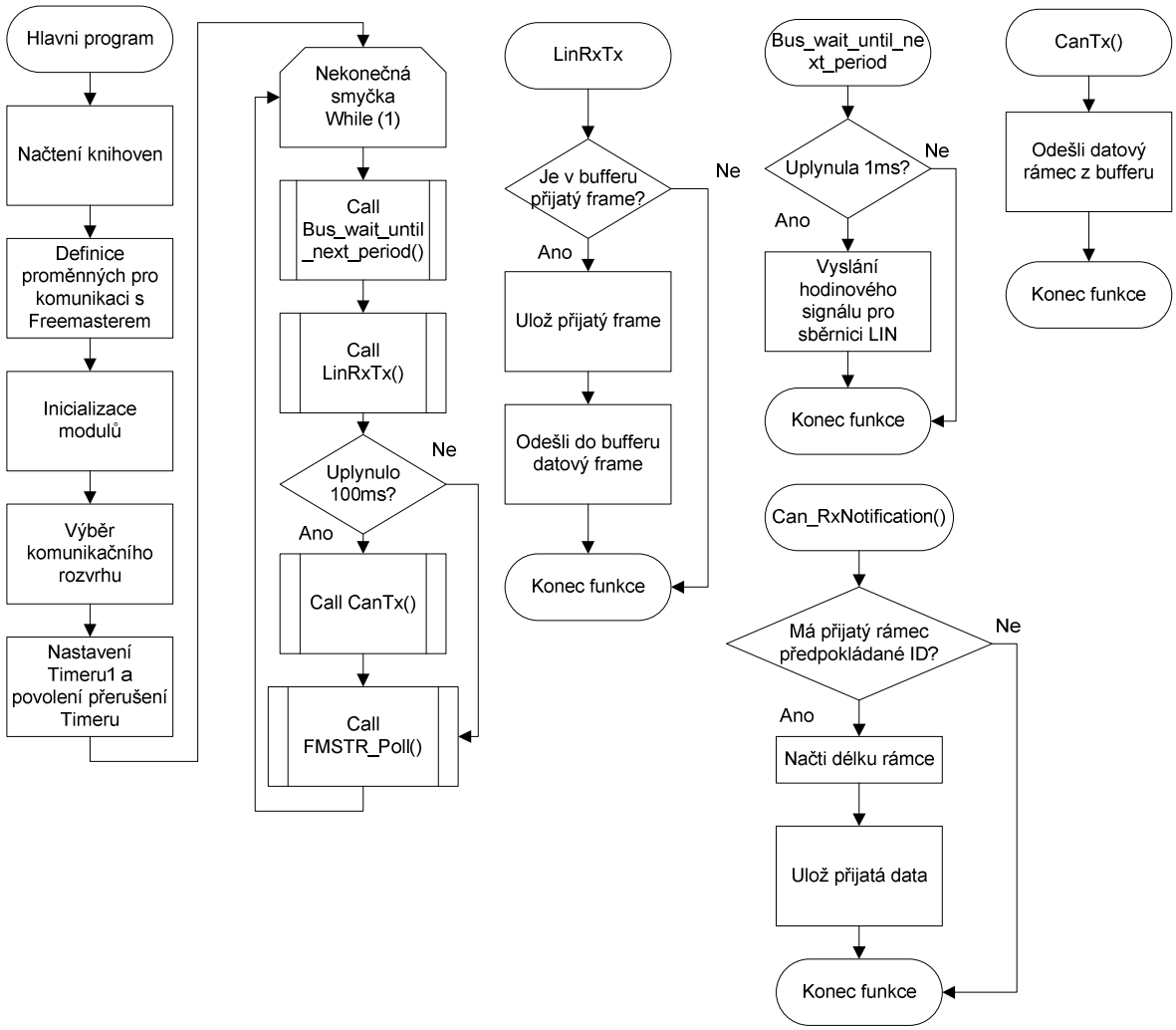
U uzlu Slave byla situace jednodušší, jelikož komunikace sběrnice LIN je řízena uzlem Master a uzly Slave pouze odpovídají na dotazy. Z toho vyplývá, že není potřeba řešit rozvrh datových rámců, hodinové pulzy ani obsluhu přerušení od časovače. Uzel Slave čeká na přijetí datového rámce a sleduje příznak přijetí. Při přijetí hlavičky rámce je vyhodnoceno, zda má uzel odpovědět nebo ne. V případě že ano, je okamžitě vyslán datový rámec odpovědi. Pokud vysílá data uzel typu Master, jsou tato data přijata uzle Slave, uložena do paměti a odeslána do programu FreeMaster. Následující diagramy zjednodušeně popisují funkci programu Slave.



Obrázek 28: Vývojový diagram oživovacího programu komunikace LIN uzlu Slave

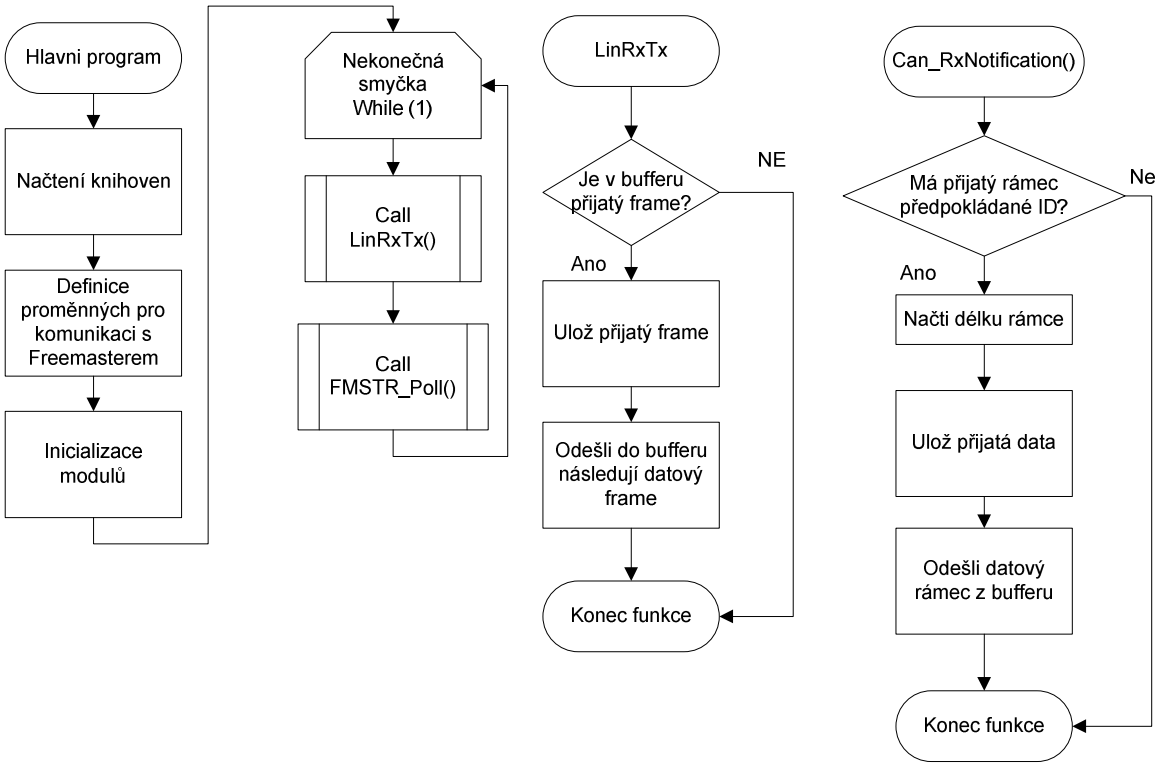
2.3. Sběrnice CAN

Stejně jako u komunikace LIN bylo potřeba ověřit základní funkčnost sběrnice CAN. První program posílal pouze pevně definované data z jednoho uzlu do druhého. K časování sběrnice CAN byl využit vnitřní časovač, který vyvolával přerušení každých 100us. V obsluze přerušení byl inkrementován čítač, který dále sloužil k měření času 100ms. Každých 100ms byl odeslán datový rámec CAN. Přijetí datového rámce je obslouženo driverem CAN. Po úspěšném přijetí je spuštěna funkce „Can_RxNotification“, která uloží přijatá data a oznámí přijetí rámce. Při úspěšném přijetí jsou data uložena do datového pole a posílána programu FreeMaster. Funkce „Can_RxNotification“ je volána driverem. Jelikož komunikace na LIN a CAN musí běžet zároveň, bylo nutné vycházet z fungujícího programu, který byl popsán v předchozí kapitole (Kapitola 2.2). V následujícím vývojovém diagramu je stručně popsán způsob řízení komunikace jednoduché komunikace na sběrnici LIN a CAN na uzlu Master.



Obrázek 29: Vývojový diagram oživovacího programu komunikace LIN a CAN uzlu Master

Řízení druhého uzlu je zjednodušeno o časování sběrnice. Frekvence odesílání zpráv je řízena na prvním uzlu a uzel druhý odpovídá okamžitě po přijetí a zpracování datového rámce. Z tohoto důvodu je funkce „CanTx()“ součástí funkce „Can_RxNotification“. Poté, co je přijat a zpracován datový rámec, je buffer okamžitě naplněn novými daty a nově vytvořený datový rámec je odeslán. Vývojový diagram je zjednodušeně zakreslen na následujícím obrázku.



Obrázek 30: Vývojový diagram oživovacího programu komunikace LIN a CAN uzlu Slave

3. Nastavení rozvrhových tabulek

Jak bylo uvedeno výše, provoz sběrnice LIN a velmi závislý na rozvrhových tabulkách. Tyto tabulky zajišťují bezproblémový provoz a definují, jak bude komunikace probíhat. Tabulka je složena ze slotů definované délky. Slot je časově ohraničený interval, ve kterém musí dojít k odvysílání daného datového rámce. Z toho vyplývá, že každý slot musí mít dostatečnou velikost, i pro případ, že by došlo k nejhorší možné variantě a délka vysílání by se prodloužila. Pokud by velikost slotu nebyla dostatečná, došly k nežádoucí ztrátě informací a konfliktům na sběrnici. Postup výpočtu délky časového rámce je uveden v kapitole 2.2.6. Nastavení komunikačních rozvrhů je obsaženo v textovém dokumentu LDF_unity.ldf. Tento dokument definuje základní komunikační požadavky a definice. Obsahuje informace o verzi komunikačního protokolu, rychlosti, signálech, okolních uzlech a jednotlivých slotech. Soubor může být modifikován v libovolném textovém editoru. Na základě tohoto souboru bylo možné vygenerovat soubory potřebné pro programování v jazyku C. Pro generování dalších potřebných souborů byl využit nástroj Node configuration tool od firmy Freescale. Na Obrázek 31: Ukázka definice rozvrhové tabulky je ukázka definice rozvrhové tabulky, která definuje rozvrh s názvem normal_mode. V tomto rozvrhu je definováno 10 slotů stejné velikosti 100ms. Takovéto rozvrhové tabulce musí předcházet definice uzlů, které se účastní komunikace, jednotlivých rámců a signálů.

```
// Schedule Table Definition
Schedule_tables
{
  normal_mode
  {
        master_frame_0 delay 100 ms;
        slave_frame_0 delay 100 ms;

        master_frame_1 delay 100 ms;
        slave_frame_1 delay 100 ms;

        master_frame_2 delay 100 ms;
        slave_frame_2 delay 100 ms;

        master_frame_3 delay 100 ms;
        slave_frame_3 delay 100 ms;

        master_frame_4 delay 100 ms;
        slave_frame_4 delay 100 ms;


  }
```

Obrázek 31: Ukázka definice rozvrhové tabulky

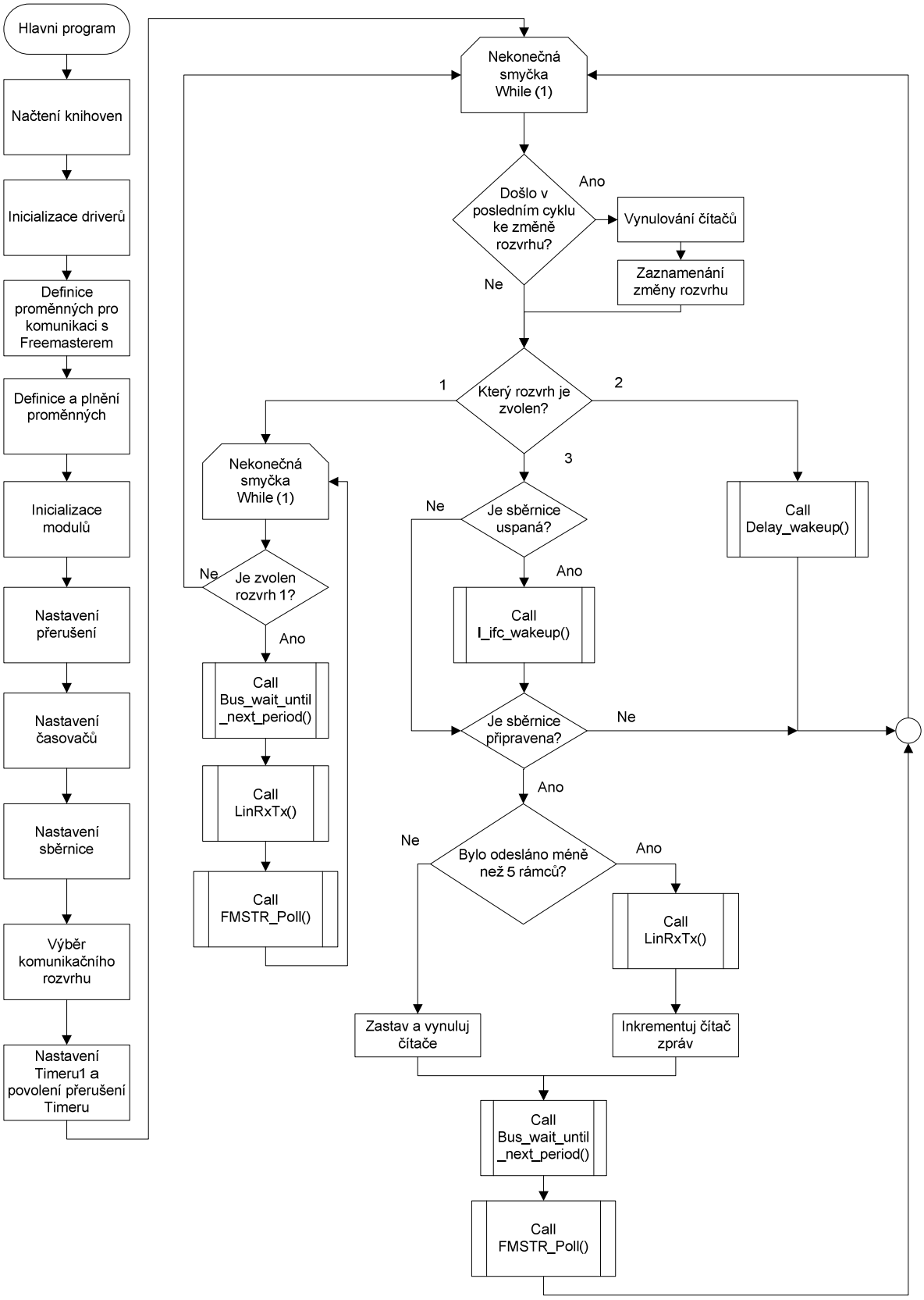
Pro účely této práce byly definovány dva uzly, tři rozvrhy, deset slotů a osmdesát signálů. Výhodou je, že počet slotů a počet bajtů zprávy byl dopředu zadán a pevně stanoven. Pro testování byly definovány i dva diagnostické rámce, které ale nejsou ve finální verzi programu využity.

4. Program pro komunikaci na sběrnicích CAN a LIN pro řídící uzel

Program byl vytvořen v programovacím jazyku C v programu CodeWarrior. V tomto vývojovém prostředí byl také debuggován. Nahrávání programu do mikroprocesoru bylo realizováno přímo z vývojového prostředí. K nahrávání (také označováno jako flashování) byl využit přípravek P&E micro USB Multilink, který umožňuje přístup v BDM (background debugger mode). Přípravek se připojí k PC přes USB a k programovanému zařízení přes jeho debuggovací vstupy. Celý program se skládá ze 73 souborů a jeho velikost je přes 31KB. Pro jednoduchost jsou jeho základní funkce zakresleny do vývojových diagramů, které popisují základní funkce programu. V následující kapitole je schematicky naznačena funkce programu a jsou popsány některé základní funkce.

4.1. Main()

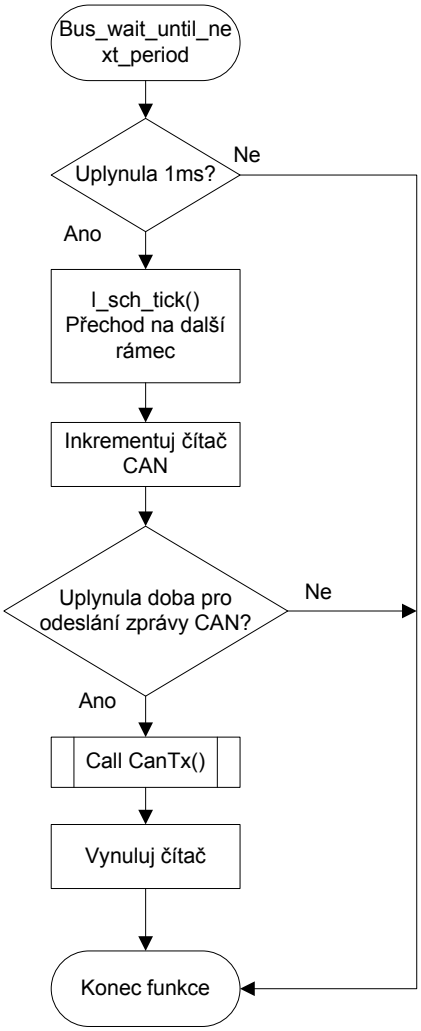
Hlavní funkce main.c spočívá v načtení knihoven, nastavení mikroprocesoru, načtení driverů, inicializaci modulů a nastavení přerušení. Dále pak řeší větvení programu vzhledem ke zvolenému komunikačnímu rozvrhu sběrnice LIN. Dle vytyčených cílů byly naprogramovány tři různé rozvrhy pro LIN a tři pro CAN, mezi kterými lze softwarově přepínat. Načtení knihoven, nastavení mikroprocesoru, načtení driverů, inicializaci modulů, inicializace a plnění proměnných a nastavení přerušení je vykonáno na začátku programu. Dále pak následuje nekonečná smyčka, ve které se program větví podle zvoleného komunikační rozvrhu sběrnice LIN. Rozvrh je periodicky kontrolován a při jeho změně jsou vždy resetovány čítače, které se používají k určení délky intervalů, kdy se čeká na probuzení uzlu nebo času kdy je uzle uspán. Funkce „l_ifc_wakeup()“ patří do komunikačního standardu LIN 2.0 a je definována v rámci jeho driveru. Tato funkce vygeneruje příkaz wake – up a pošle ho na sběrnici, čímž dojde k probuzení vzdáleného uzlu. Komunikace na sběrnici CAN je spouštěna v rámci funkce „Bus_wait_until_nex_period()“ ve které je periodicky volána funkce Tx_Can_period().



Obrázek 32: Vývojový diagram funkce main() řídícího uzlu

4.2. Bus_wait_until_next_period()

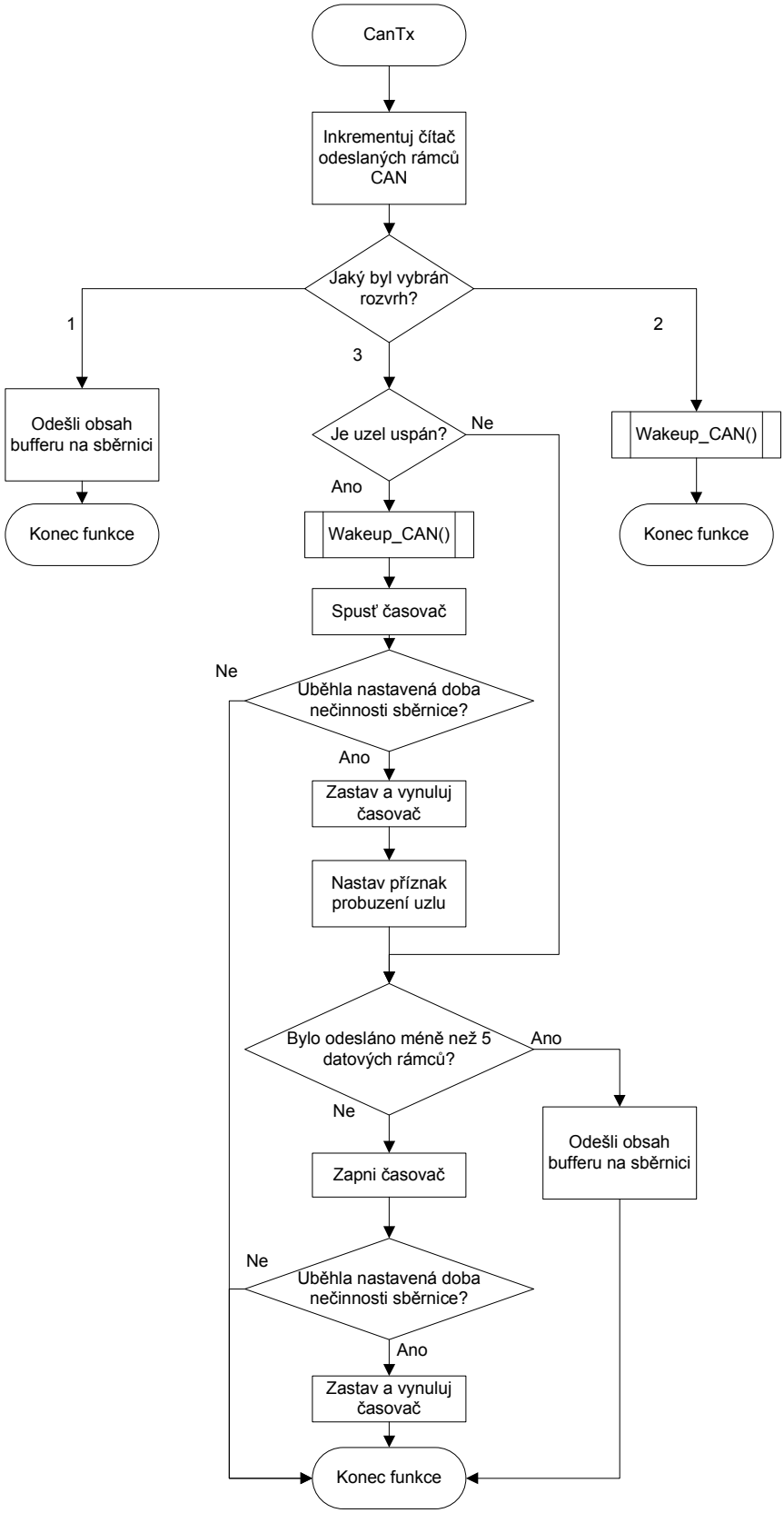
Ve funkce „Bus_wait_until_next_period()“, která je periodicky volána z funkce main(), je volána funkce l_sch_tick().Tato funkce přímo řídí posuny v komunikačním rozvrhu sběrnice LIN. Může být volána pouze z uzlu Master (Slave neřídí komunikaci, ale pouze odpovídá). Když je funkce zavolána, dojde k aktualizaci rozvrhových signálů a přechodu k dalšímu rámci rozvrhu. Pokud je zjištěn konec rozvrhu, přejde funkce na jeho začátek. Bus_wait_until_next_period() je funkce, která porovnává stav čítačů a požadovanými hodnotami a na základě jednoduchých nerovností rozhoduje, zda se má přistoupit k další komunikaci.



Obrázek 33: Vývojový diagram funkce Bus_wait_until_next_period() řídícího uzlu

4.3. CanTx()

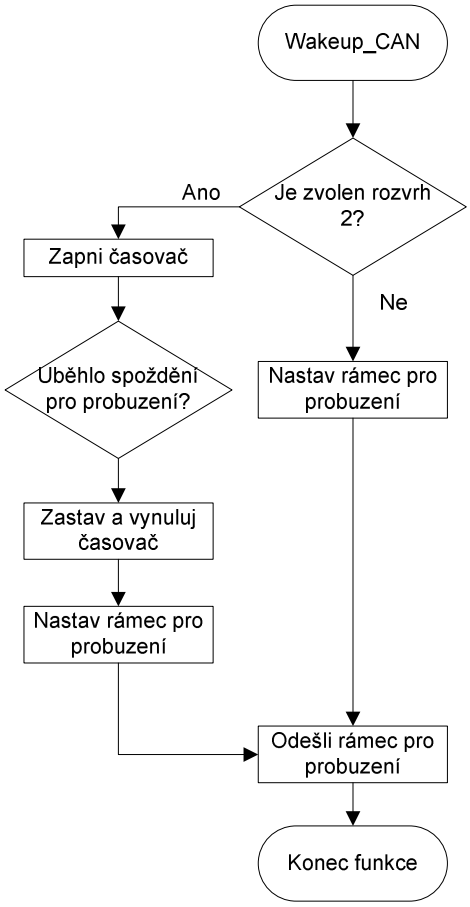
Funkce CanTx() se stará o vysílání datových rámců na sběrnici CAN. Datové rámce na sběrnici CAN jsou vysílána v nastavitelné periodě od 1ms do 1s. Zda uplynula tato perioda je kontrolováno ve funkci Bus_wait_until_next_period(). Po uplynutí vysílací periody je volána funkce CanTx(). V této funkci se program větví podle zvoleného komunikačního rozvrhu CAN. Pokud je zvolen rozvrh 1, je pouze odeslán obsah bufferu, který je konfigurován při obsluze přijetí zprávy. Pokud je zvolen rozvrh 2, je volána funkce Wakeup_Can, která se dále větví podle rozvrhu 2 a rozvrhu 3. Jestliže je nastaven rozvrh 3, pak je nejdříve ověřeno, zda je sběrnice vzdálený uzel uspán. Pokud ano, je odvysílán wake-up signál a je spuštěn časovač, který odpočítává dobu nečinnosti sběrnice, která je nutná k probuzení uzlu. Po uplynutí této doby nastaví příznak probuzení. Pokud uzel nebyl uspán, pak přejde přímo k vysílací části. Vy vysílání části je ověřeno kolik datových rámců bylo odesláno a pokud je to méně než pět, odešle se další rámec. Pokud je počet odeslaných rámců pět nebo více, zapne se časovač, který měří zpoždění pro ukončení komunikace a poté přejde vzdálený uzel do spánkového režimu.



Obrázek 34: Vývojový diagram funkce CanTx() řídícího uzlu

4.4. Wakeup_CAN()

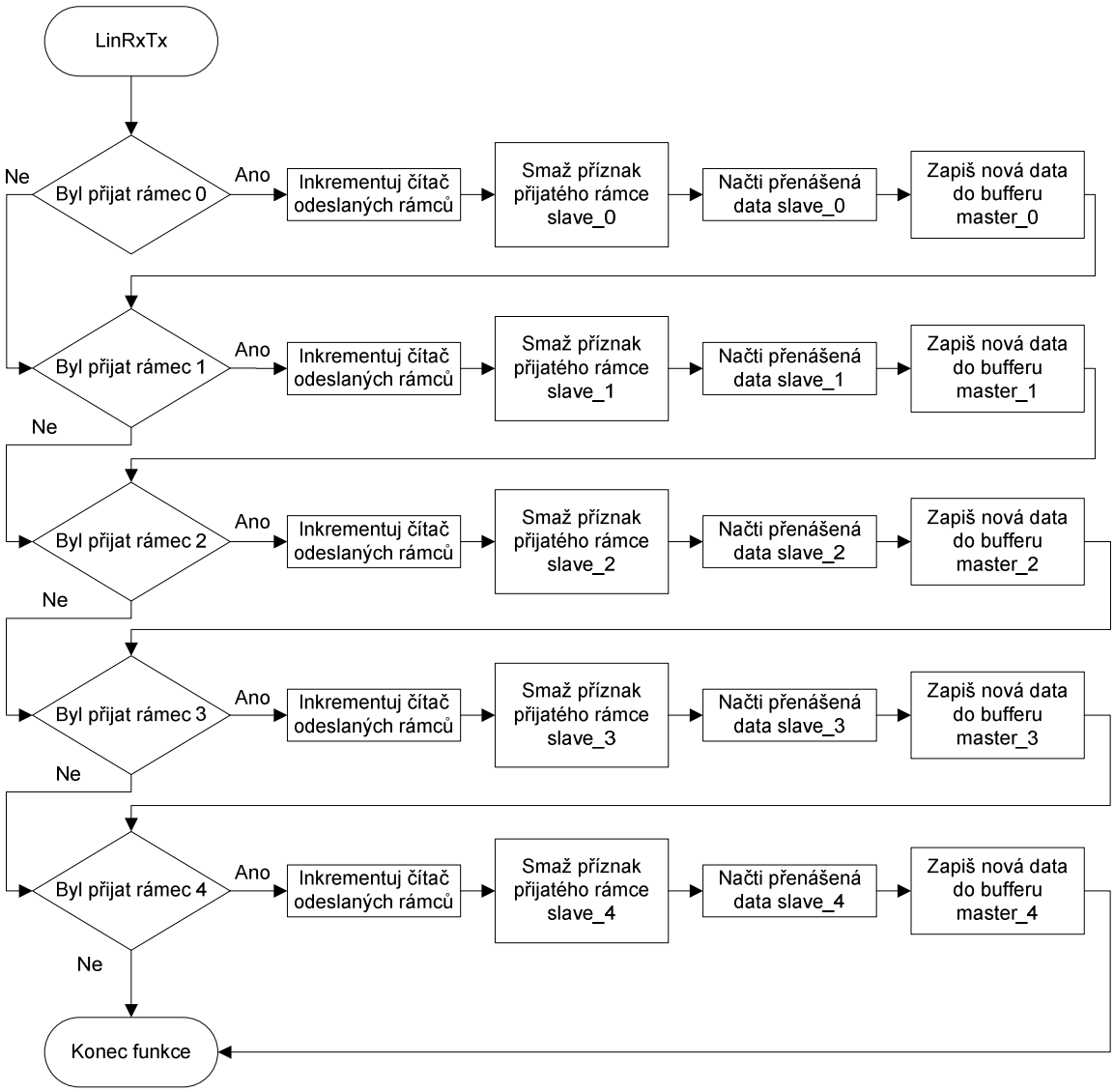
Jednoduchá funkce, která nastavuje speciální signál pro probuzení uzlů (wake-up signál). Pokud je funkce zavolána, nejdříve ověří jaký je komunikační rozvrh. Pokud je zvolen rozvrh 2 nejprve se zapne časovač a odpočítává se zpoždění před probuzením sběrnice. Po uplynutí stanoveného času je do bufferu připraven wake-up signál, čítač zpoždění je vynulován a buffer je odeslán na sběrnici.



Obrázek 35: Vývojový diagram funkce Wakeup_CAN() řídícího uzlu

4.5. LinRxTx()

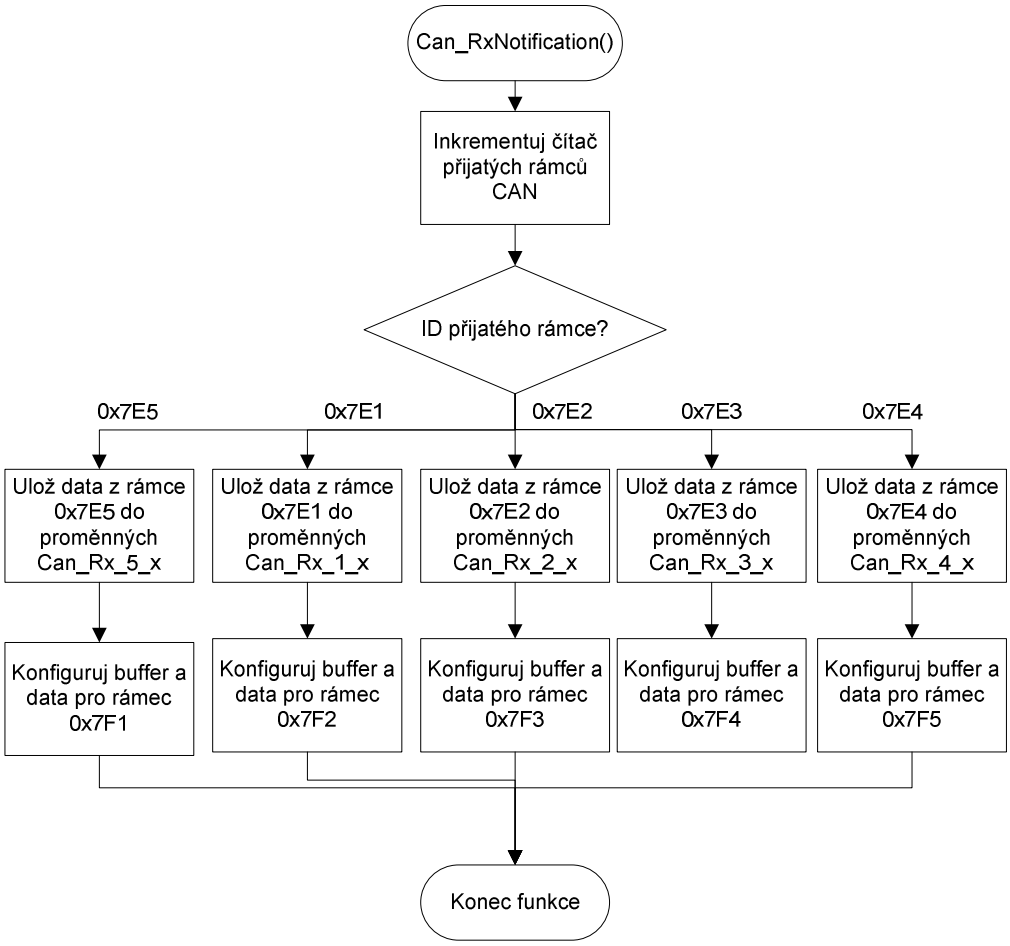
Tato funkce se stará o zpracování přijatých datových rámců a o plnění bufferu novými daty. Pokud je přijat nový datový rámec, je nastaven příznak přijetí, který odpovídá danému rámci. Rámce jsou rozděleny pomocí ID a to tak, že rámce odeslané uzlem Master mají ID 1,2,3,4,5 a rámce odeslané uzlem Slave mají ID 11,12,13,14,15. Dle této identifikace jsou schopni přesně určit, o který rámec se jedná a známe i jeho směr. Funkce LinRxTx() vychází z jednoduchých podmínek, kterými určí, jaká data nají být uložena a nachystána do bufferu. Každý byte každého rámce má svůj příznak přijetí, což umožňuje práci pouze s vybranými částmi dat. Funkce určí, který rámec byl přijat, resetuje příznak přijetí, inkrementuje čítač přijatých rámců, uloží přijatá data do určených umístění a nachystá nová data do bufferu. Program je nastaven tak, aby posílal 5 datových rámců stále dokola.



Obrázek 36: Vývojový diagram funkce LinRxTx() řídícího uzlu

4.6. CAN_RxNotification()

Přijetí rámce na sběrnici CAN je obsluhováno driver. Pokud je rámec správně přijat je zavolána funkce CAN_RxNotification(). Tato funkce pracuje s označením modulu CAN, ID rámce, jeho délkou a samotnými daty. Po zavolání funkce rozpozná podle ID, který rámec byl přijat, inkrementuje čítač přijatých zpráv, uloží data do přidělených proměnných a nastaví buffer pro další komunikaci. Jelikož u komunikace CAN nemáme uzel typu Master, jsou si všechny uzly rovny. Pro odlišení hlavního uzlu jsou mu přidělena ID zpráv 0x7F1, 0x7F2, 0x7F3, 0x7F4, 0x7F5. Podle ID jsme schopni určit, který uzel odvysílal datový rámec. Pokud je odvysílán rámec s ID 0x7F1, očekávaný další rámec na sběrnici bude 0x7E1, což znamená, že druhý uzel přijal odvysílaný datový rámec a posílá další data.



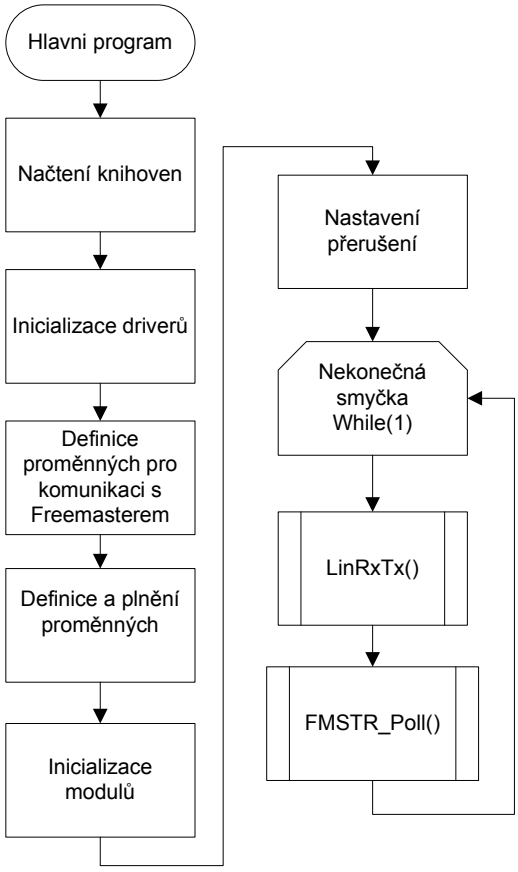
Obrázek 37: Vývojový diagram CAN_RxNotification() řídícího uzlu

5. Program pro komunikaci na sběrnicích CAN a LIN pro řízený uzel

Program pro řízený uzel je o poznání jednodušší. Počítá se s tím, že veškeré řízení komunikace a časování bude ovládáno na řídícím uzlu a uzel řízený bude pouze posílat datové rámce. Ačkoli se program pro řízený uzel skládá také ze 73 souborů, je jeho velikost jen 25KB. Proto, aby byl dodržen koncept testeru, je nutné, aby co možná nejvíce úloh řešil řídící uzel. Řízený uzel by měl být zatěžován co nejméně. V následujících kapitolách jsou zobrazeny vývojové diagramy, které popisují základní funkce ovládacího programu.

5.1. Main()

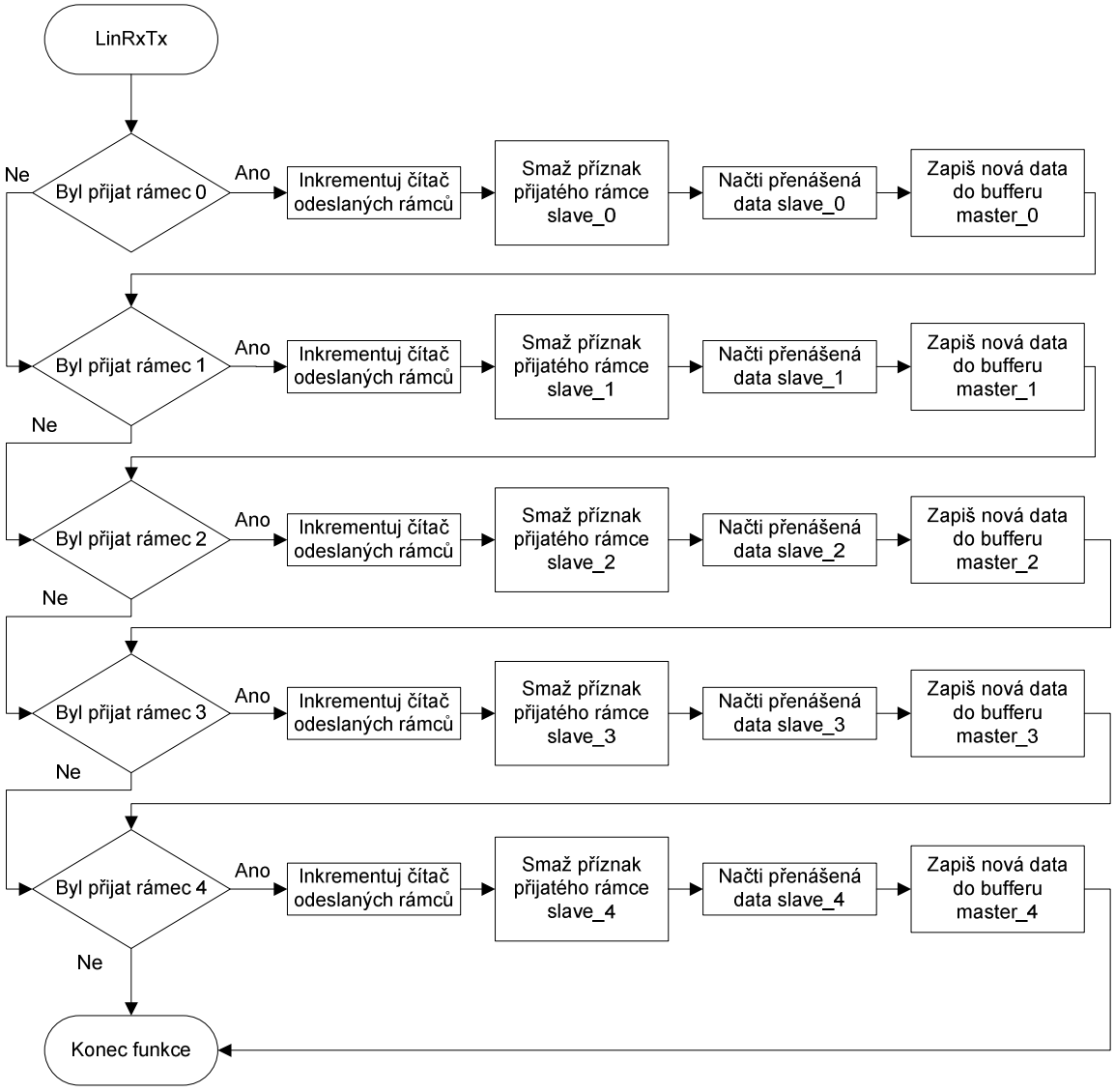
Hlavní funkce má za úkol zejména inicializaci jednotlivých modulů, nastavení přerušeních, inicializaci proměnných a jejich naplnění daty. Součástí funkce je nekonečná smyčka, ve které se neustále testují příznaky přijetí jednotlivých datových rámců. Dále je volána funkce FMSTR_Poll(), která slouží ke komunikaci s programem FreeMaster. Po inicializaci a nastavení jsou tedy volány pouze dvě funkce.



Obrázek 38: Vývojový diagram funkce main() řízeného uzlu

5.2. LinTxRx()

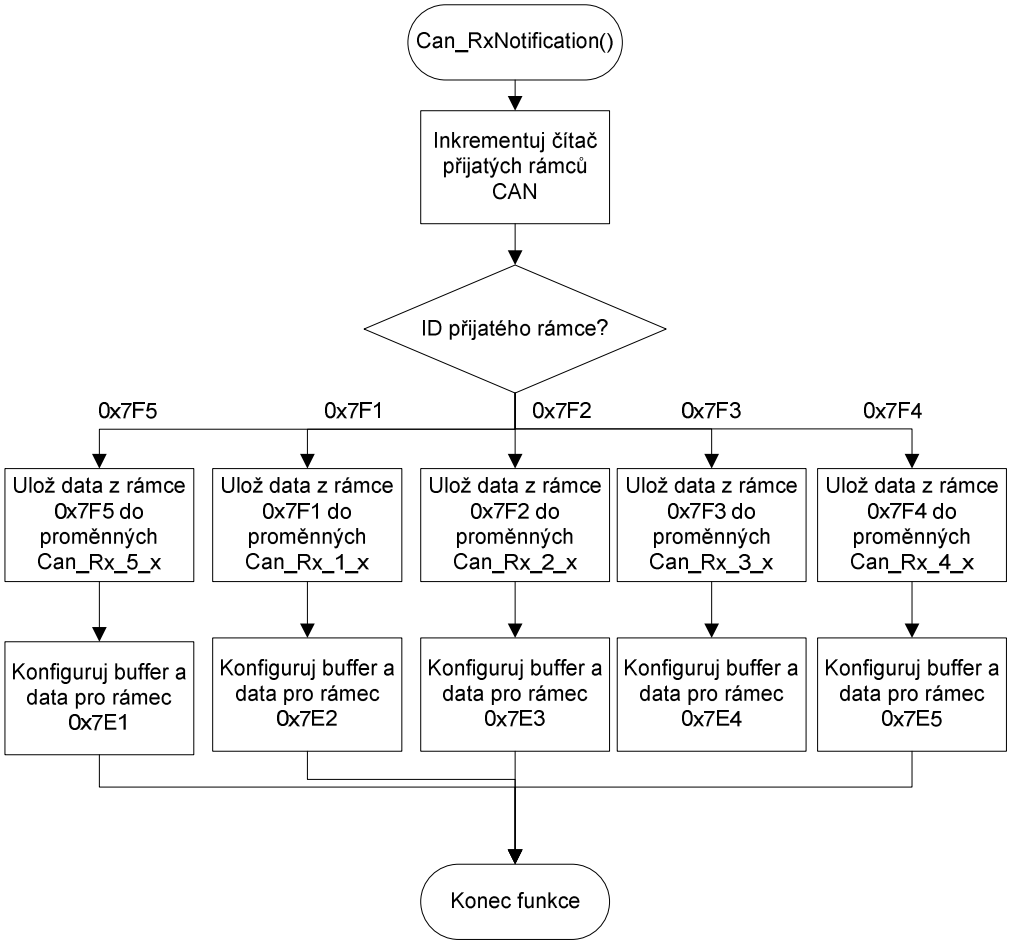
Funkce LinTxRx() je podobná jako u řídícího uzlu. Uzel typu Slave přijímá data, které vysílá Master a periodicky kontroluje příznaky přijetí datových rámců. Každý příznak má svůj příznak, takže je po přijetí možné určit, který rámec byl přijat a nastavit, jak se na něj má odpovědět. Funkce se také stará o počítání přijatých datových rámců. Tento čítač je propojen s FreeMasterem a dále pomocí skriptů i s grafickým rozhraním.



Obrázek 39: Vývojový diagram funkce LinTxRx() řízeného uzlu

5.3. Can_RxNotification()

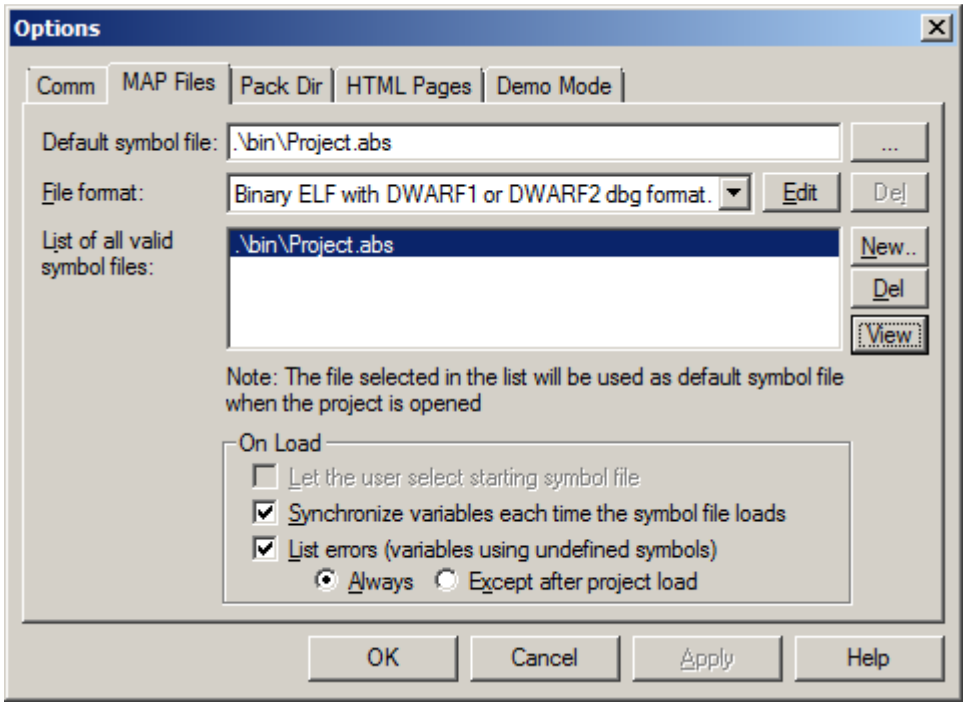
Stejně jako u řídícího uzlu je tato funkce volána při přijetí datového rámce. Volání je obsluhováno driverem CAN. Datové rámce odvysílané řízeným uzlem mají ID 0x7E1, 0x7E2, 0x7E3, 0x7E4, 0x7E5. Díky odlišení ID jednotlivých uzlů, můžeme pozorovat datovou výměnu mezi uzly a případně zjistit chyby ve vysílání. V rámci nastavování datového rámce můžeme nastavit délku rámce, jeho prioritu, formát ID, samotné ID a přenášená data. Vývojový diagram popisuje přijetí dat a nastavení nového datového rámce.



Obrázek 40: Vývojový diagram funkce Can_RxNotification() řízeného uzlu

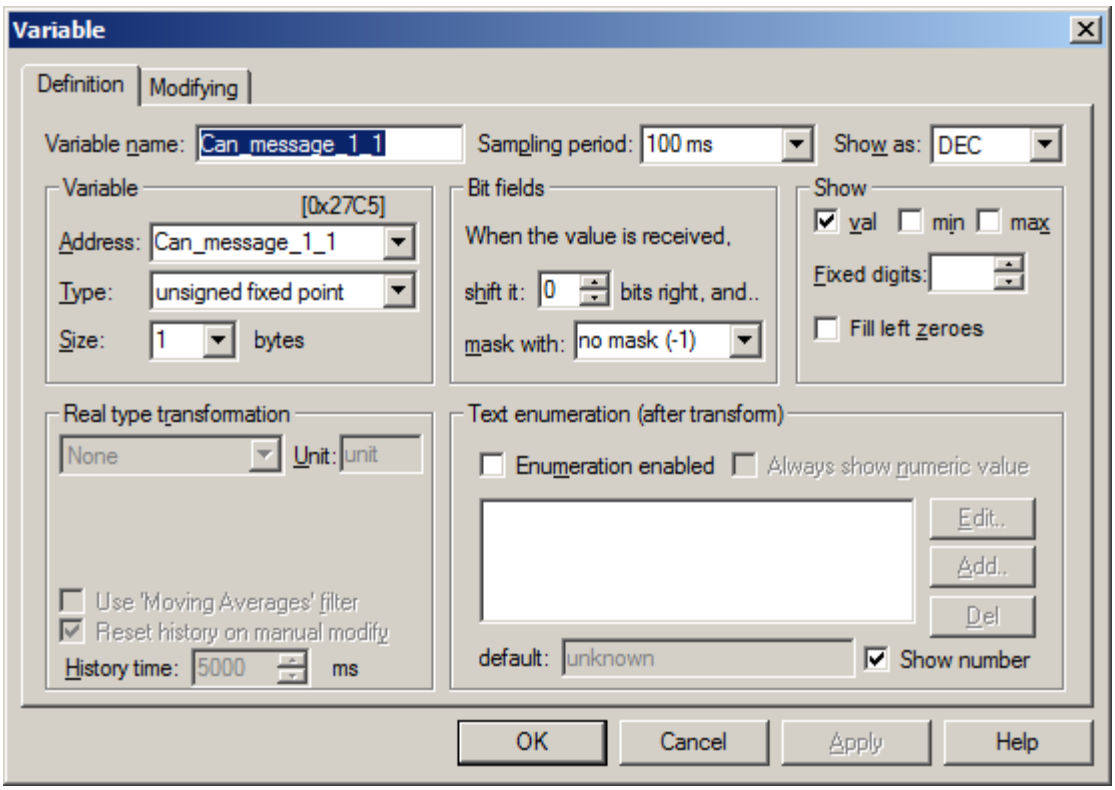
6. Nastavení programu FreeMaster

Jak už bylo uvedeno výše, FreeMaster je diagnostický program vyvíjený firmou Freescale(viz Kapitola 3.1). Pro propojení FreeMasteru a uzlů sběrnice je využito rozhraní USB. Dříve bylo často využíváno sériové rozhraní RS-232, ale vzhledem k tomu, že většina moderních zařízení není vybavena sériovým portem, je USB lepší variantou. Po připojení uzlu k PC bylo nutné nastavit MAP soubor, který byl vytvořen ve vývojovém prostředí CodeWarrior. FreeMaster podporuje několik různých formátů, ale v tomto konkrétním případě se jedná o soubor Project.abs. V tomto souboru je obsažen soupis proměnných, jejich umístění v paměti a jejich velikost. Soubor je generován automaticky. Nastavení souboru se provádí v okně FreeMasteru, ve volbách Project->Option. Spolu s určením cesty je možné nastavit další položky jako například, zda se má Project.abs synchronizovat při každém načtení. V tom samém okně, ale pod záložkou Comm je nutné nastavit komunikační port a rychlost komunikace. Po uložení těchto nastavení je možné pokračovat s přidáním proměnných, které chceme sledovat popř. upravovat.



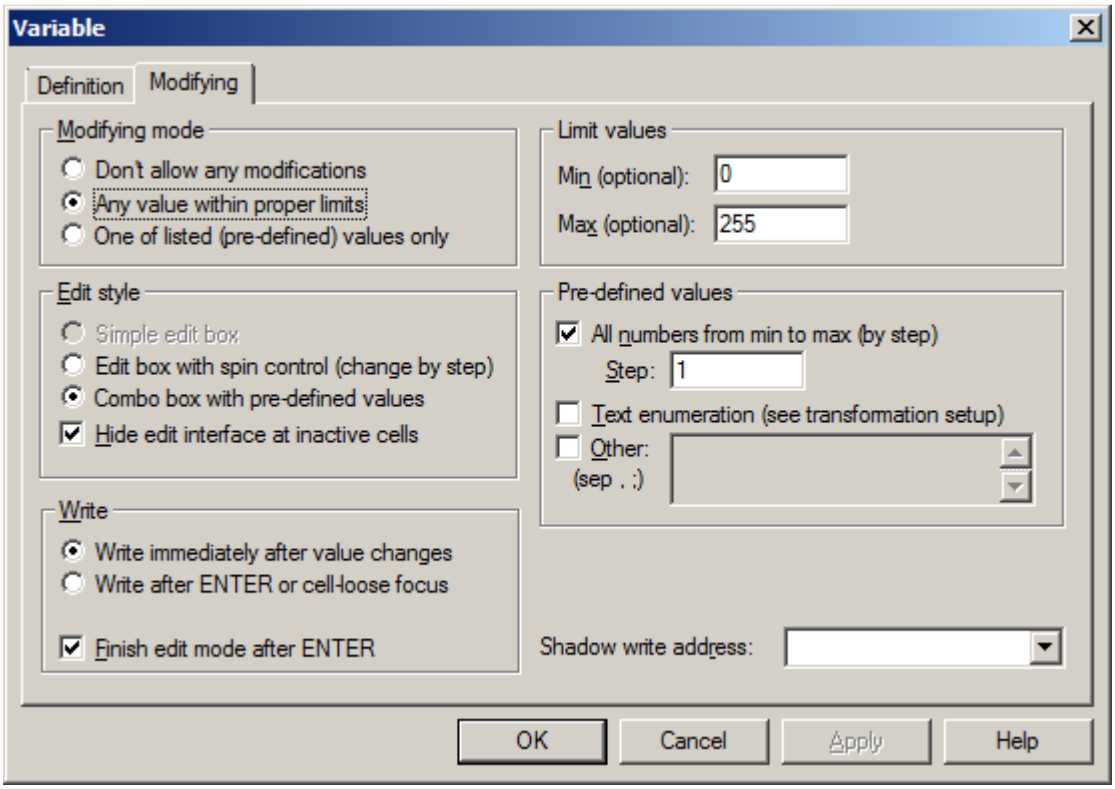
Obrázek 41: Ukázka okna pro nastavení Freemasteru

Pro to, abychom mohli sledovat obsah proměnných je nutné je přidat do skupiny Dostupné proměnné (Available variables). Freemaster snímá hodnoty pouze vybraných proměnných. Pokud chceme proměnou sledovat v textovém okně, můžeme ji přidat do skupiny Sledované proměnné (Watched variables). Každá proměnná má vlastní nastavení, ve kterém můžeme definovat její název, adresu v paměti, datový typ, velikost, testovací periodu, číselnou soustavu pro zobrazení atd. Následující obrázek zobrazuje okno pro definici proměnné.



Obrázek 42: Okno pro definici proměnné v programu FreeMaster

Důležitá nastavení, týkající se možnosti změny hodnoty proměnné v prostředí FreeMaster, jsou v pod záložkou „Modifying“. V této záložce je možné nastavit, zda se proměnná může pouze číst nebo se do ní může i zapisovat. Velkou výhodou je možnost nastavení limitů změny, krokování změn nebo povolení pouze určitých hodnot. Díky tomuto nastavení není potřeba hlídat obsah proměnných v kódu. Pro zjednodušení nastavení proměnných lze nastavení „klonovat“. Klonování znamená kopírování nastavení z jedné proměnné na další. V tomto programu je pracováno s téměř dvěma sty proměnnými. Na následujícím obrázku je zobrazeno nastavení proměnné Can_message_1_1, která obsahuje první byte datového rámce s ID 0x07F1. Jelikož se jedná o jeden byte, jsou nastaveny limitní hodnoty 0 až 255 s rozlišením 1. Hodnota proměnné se tedy může měnit pouze v těchto limitech a zápis je proveden okamžitě po změně.



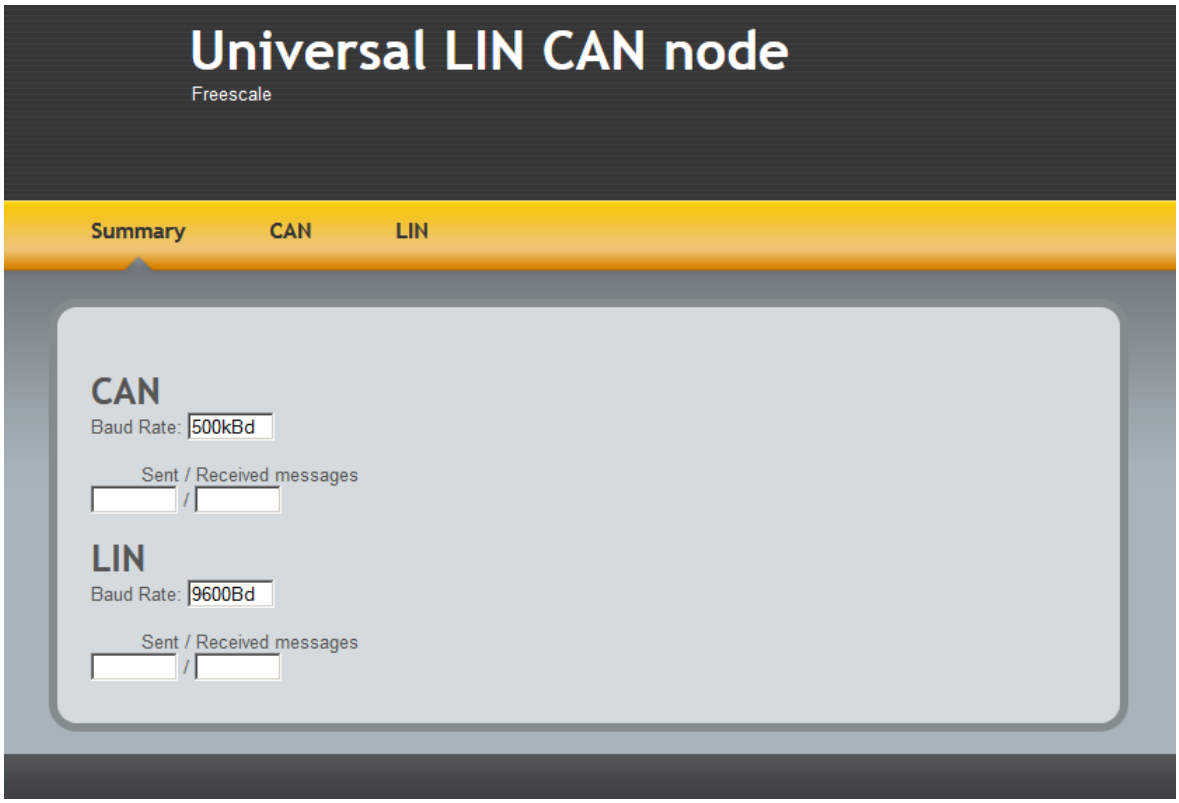
Obrázek 43: Ukázka nastavení možností proměnné v programu FreeMaster

7. Grafické rozhraní

V rámci diagnostického prostředí FreeMaster je možné vytvořit grafické prostředí v HTML. HTML stránky lze vytvořit v libovolném editoru a poté nastavit přístup k hotovým stránkám. Grafické prostředí bylo vytvořeno v základním textovém editoru Notepad. K formátování byly využity kaskádové styly. Cílem bylo vytvořit velmi jednoduché grafické rozhraní, které by umožnilo jednoduše ovládat sběrnici a zobrazovalo data. Všechny stránky obsahují hlavičku s nadpisem Universal LIN CAN node. Pod hlavičkou je řádek s menu. Většina stránky je vyhrazena zobrazení přijatých a odeslaných data a nastavení sběrnice. Grafické prostředí řídícího uzlu je složitější o některé nastavení, které se netýkají řízeného uzlu. Komunikace s prostředím FreeMaster je založena na skriptech v jazyce VBS (Virtual Basic Script) a pomocného skriptu v Javascriptu. Prostředí se skládá z třech stránek. První strana obsahuje přehled o přijatých a odeslaných rámcích jak na sběrnici CAN, tak na sběrnici LIN. Druhá stránka je věnována CAN sběrnici a třetí LIN.

7.1. Stránka Summary

První stránka zobrazuje informace o odeslaných a přijatých datových rámcích a o rychlosti komunikace sběrnic. Žádné s formulářových oken neumožňuje zápis a jsou určena pouze pro změny vyvolané skriptem. Rychlost sběrnice není nijak sledována, ale je nastavena v HTML kódu.



Obrázek 44: Ukázka vzhledu hlavní stránky grafického rozhraní

Skript komunikuje se čtyřmi proměnnými a je aktualizován každých 500ms. Tato hodnota byla zvolena proto, aby nedocházelo k velké zátěži procesoru zobrazovacího zařízení. Načítání jednotlivých proměnných je realizováno pomocí Virtual Basic Scriptu. Skript je napsán tak, aby zobrazoval případné chybné načtení. Pokud čtení neproběhne správně, je místo hodnoty zobrazeno slovo „Err“, což znamená Error. Na následujícím obrázku je zobrazena ukázka skriptu pro načtení proměnné counterRxLin. Na začátku je nutné definovat skriptovací jazyk příkazem <skript langure =“VBScript“>, dále následuje definice funkce „read()“. Funkci definujeme proto, abychom mohli skript opakovat v určitých časových intervalech. Ve funkci read() jsou definovány proměnné v, tv, retMsg a succ. Tyto proměnné jsou platné pouze ve funkci read(). V následujícím řádku je načtena proměnná counterRxLin. Do proměnné „v“ je uložena její numerická hodnota, do „tv“ je načtena hodnota ve formátu řetězce (string) a do „retMsg“ je v případě chyby uložena chybová hláška. Pokud proběhne čtení úspěšně, je proměnná „succ“ nastavena na hodnotu „true“ a vykoná se první část podmínky. Pokud při čtení dojde k chybě, nabude „succ“ hodnotu „false“ a vykoná se druhá část podmínky.

```
<script language="VBScript">
Function read()
        dim v, tv, retMsg
        succ = false

        succ = pcm.ReadVariable ("counterRxLin",v, tv, retMsg)
        If succ then
                TxLin.value = tv
        Else
                TxLin.value = "Err"
        End If
```

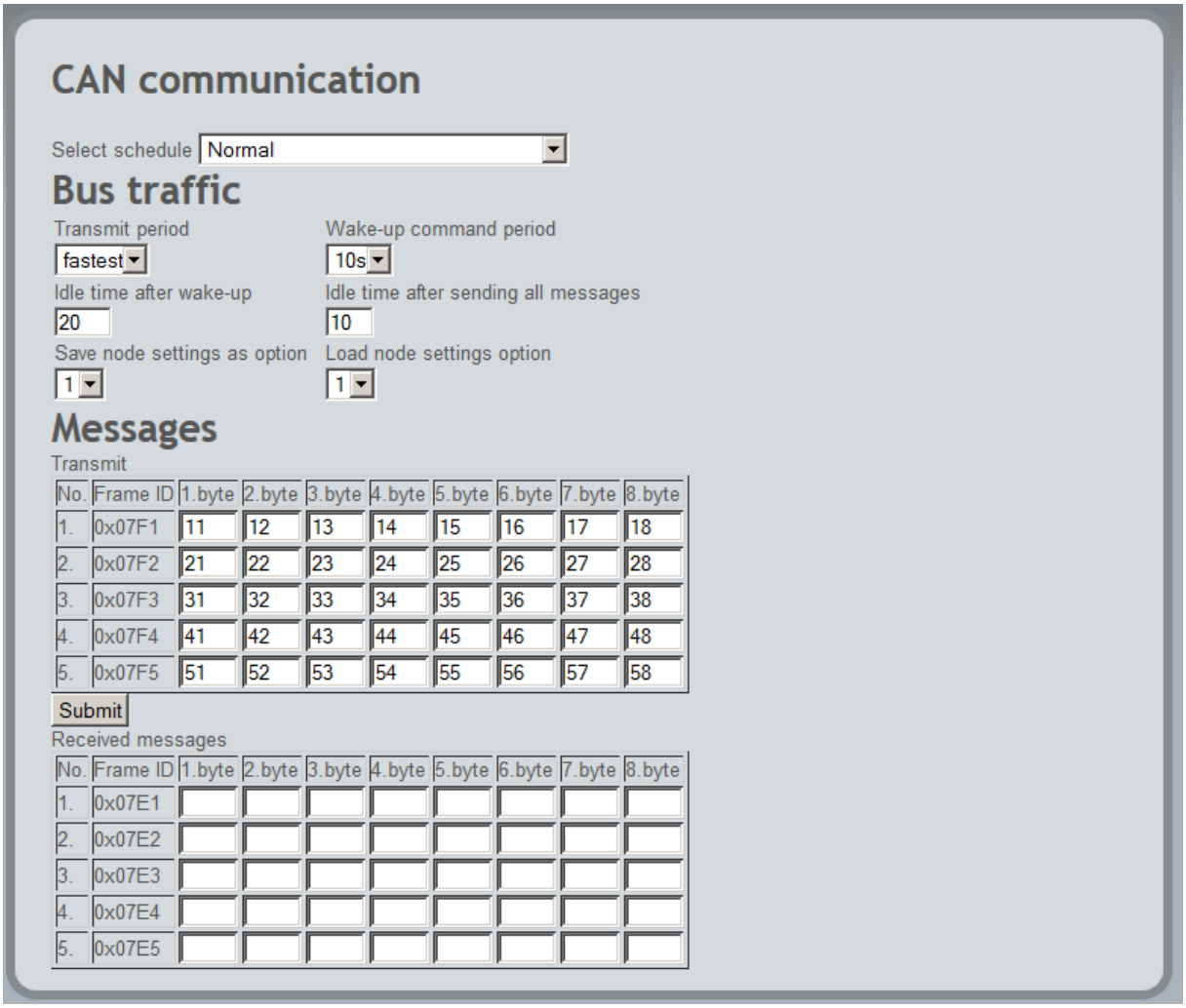
Obrázek 45: Ukázka skriptu pro načtení proměnné

Takto zapsaný skript se provede pouze jednou, při načtení stránky. K jeho opakování byl využit krátký skript v jazyce Javascript. Ve skriptu je využit časovač, který spouští funkci „read()“ každých 500ms.

7.2. Stránka CAN

CAN stránka zobrazuje bližší informace o datovém provozu a nastavení na sběrnici CAN. Stejně jako na hlavní straně je zde hlavička a menu v podobě pásu s odkazy. V horní části stránky je dialogové okno, které umožňuje nastavení jednoho ze tří rozvrhů. Rozvrh Normal označuje periodické odesílání pěti zpráv v určitých časových intervalech. Rozvrh Periodic Wake-up/Sleep neodesílá žádné datové rámce, ale pouze v nastavených časech probouzí a uspává vzdálený komunikační uzel. Poslední rozvrh Periodic Wake-up/send/recese/sleep simuluje reálnou komunikaci, kdy je uzel probuzen, jsou odeslány a přijaty datové rámce a poté je uzel uspán. Podrobnější nastavení sběrnice jako perioda vysílání, perioda probouzení uzlu, čas nečinnosti po probuzení uzlu a čas nečinnosti po odeslání všech zpráv, se nachází v prostřední části stránky označené jako Bus trefíc. Nastavení hodnost je prováděno buď

výběrem z roletkového menu, nebo zapsáním numerické hodnoty do příslušného formulářového okna. Ukládání nastavení uzlů do vnitřní paměti EEEPROM zatím není funkční. V dolní části stránky je část označená jako Messages. V této části je v tabulkách zobrazena datová komunikace. V horní tabulce označené jako Transmit je možné měnit data, která jsou uzlem vysílána. Každé pole tabulky představuje jeden byte. Data jsou zapsána do příslušných proměnných po stisku tlačítka Submit. V dolní tabulce jsou zobrazena přijatá data. Tato tabulka je určena pouze pro čtení.



Obrázek 46: Grafické rozhraní stránka CAN – řídící uzel

Po načtení stránky je vykonán skript, který načte skutečný stav proměnných a zobrazí je ve formulářových polích. Tak je zaručeno, že v grafickém rozhraní jsou zobrazeny aktuální hodnoty nastavení sběrnice a datových rámců. Tento skript je spuštěn po každém načtení stránky. Dále je definována funkce read(), která je periodicky spouštěna opakovacím skriptem v Javascriptu. Funkce read() načítá hodnoty datových rámců a zobrazuje je v tabulce. Perioda čtení je 1,5s. Pro zápis datového rámce na sběrnici je definována funkce send(). Tato funkce je spouštěna po stisknutí tlačítka Submit. O změny nastavení ostatních možností sběrnice se stará

pět krátkých skriptů, které jsou spouštěny při změně některého nastavení. Krátký skript pro nastavení periody vysílání datových rámců. Funkce nese název Tx() a jsou v ní definovány dvě proměnné. Pokud je korektně proveden zápis hodnoty z políčka Tx_period do proměnné Tx_CAN_period, je do proměnné succ uložena hodnota „true“. Pokud zápis není proveden korektně, je do proměnné retMsg uložena chybová hláška a do proměnné succ je uloženo „false“.

```
Function Tx()
        dim retMsg
        succ = false

        succ = pcm.WriteVariable("Tx_CAN_period",Tx_period.value,retMsg)

End function
```

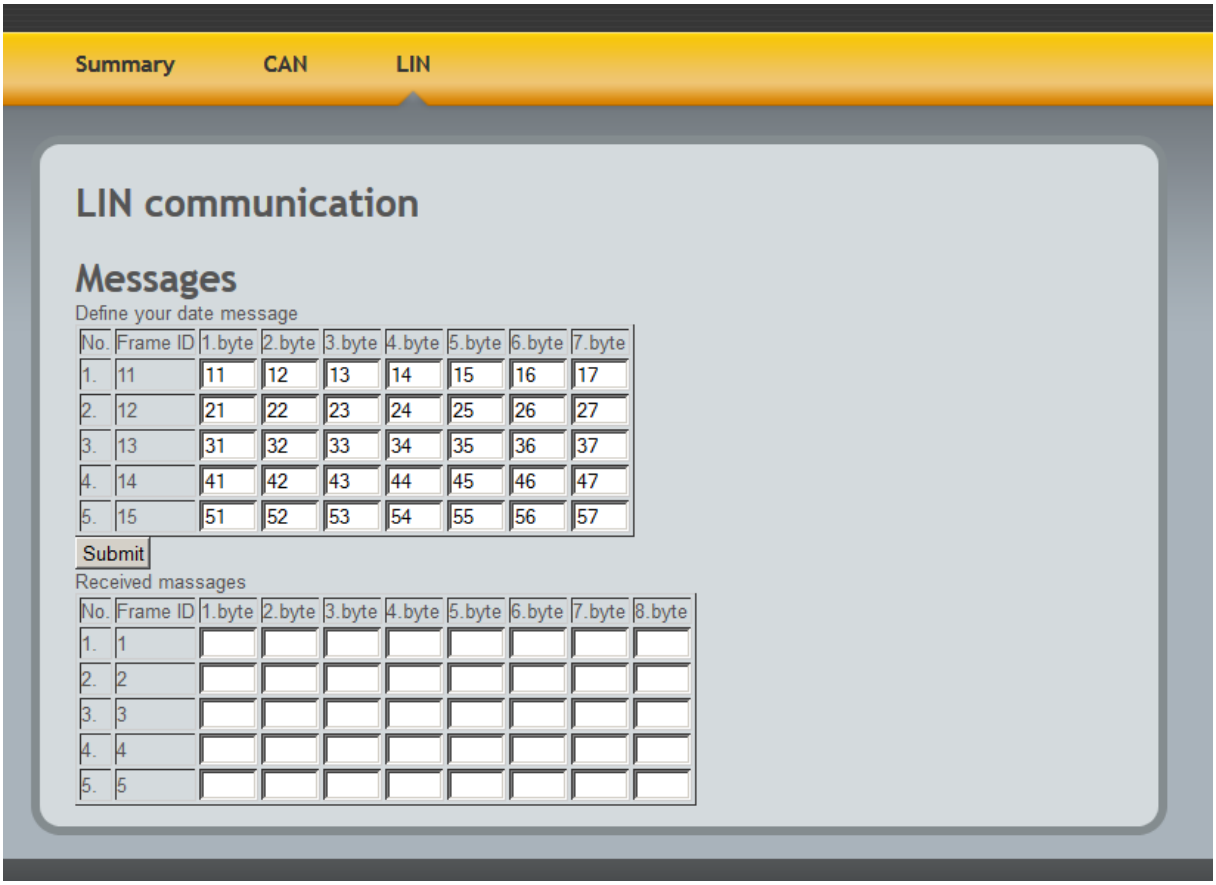
Obrázek 47: Ukázka skriptu pro nastavení periody vysílání

Stránka CAN je obdobně navržena i pro řízený uzel. Na řízeném uzlu však chybí možnosti nastavení sběrnice a rozvrhu. Stránka se skládá pouze ze dvou tabulek, z nichž jedna zobrazuje přijatá data a druhá odesílaná. Odesílaná data můžeme měnit stejně jako na uzlu řídícím. Vzhledem k chybějícím možnostem nastavení chybí i některé skripty. Skript pro odesílání je spouštěn tlačítkem Submit. Skript pro čtení je spouštěn opakovaně s periodou 1,5s.

7.3. Stránka LIN

LIN stránka zobrazuje bližší informace o provozu a nastavení na sběrnici LIN. Pro jednodušší orientaci v grafickém prostředí, je navržena tak, aby se co nejvíce podobala stránce CAN. Struktura stránky je zachována. Pod hlavičkou a řádkovým menu je roletka pro výběr vysílacího rozvrhu. Stejně jako u sběrnice CAN je možné vybírat ze třech možností rozvrhu. Rozvrh Normal označuje periodické odesílání pěti zpráv. Rozvrh Periodic Wake-up/Sleep neodesílá žádné datové rámce, ale pouze v nastavených časech probouzí vzdálený komunikační uzel. Poslední rozvrh Periodic Wake-up/send/recese/sleep simuluje reálnou komunikaci, kdy je uzel probuzen, jsou odeslány a přijaty datové rámce a poté je uzel automaticky uspán. Další nastavení sběrnice určují, jak dlouho bude uzel uspán a délku nečinnosti po probuzení uzlu. Ukládání a načítání nastavení uzlu do vnitřní EEEPROM paměti zatím nefunguje. Stejně jako u stránky CAN jsou formulářová okna ovládána skripty. Po načtení stránky je spuštěn skript, který načte hodnotu všech proměnných a zapíše ji do jednotlivých formulářových oken a tabulek. Poté je opakovaně volána funkce pro čtení přijatých dat. Perioda volání je 1,5s. Všechna nastavení sběrnice jsou použita okamžitě po jejich změně. Tabulka pro zobrazení přijatých dat zobrazuje pouze sedm bajtů. Osmý byte je vyhrazen pro chybová hlášení, a proto není zobrazován.

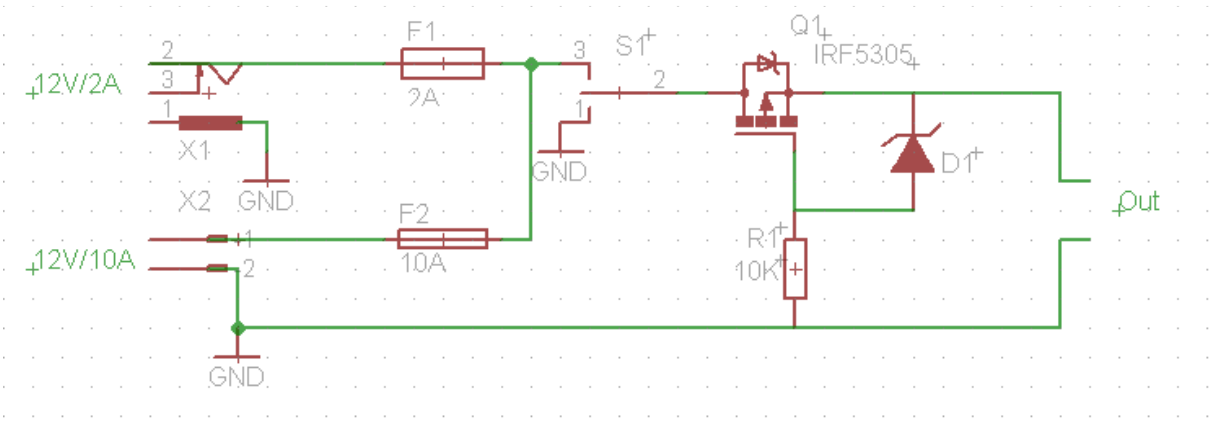
Grafické rozhraní pro řízený uzel je velmi podobné. Stejně jako u stránky CAN však chybí ovládání sběrnice. Stránka je složena ze dvou tabulek, z nichž jedna pouze zobrazuje přijatá data a druhá umožňuje zobrazení a modifikaci odesílaných dat. Tabulka pro zobrazení přijatých dat zobrazuje pouze sedm bajtů. Osmý byte je vyhrazen pro chybová hlášení, a proto není zobrazován.



Obrázek 48: Ukázka grafického rozhraní stránky LIN řízeného uzlu

8. Proudová ochrana a ochrana proti přepólování

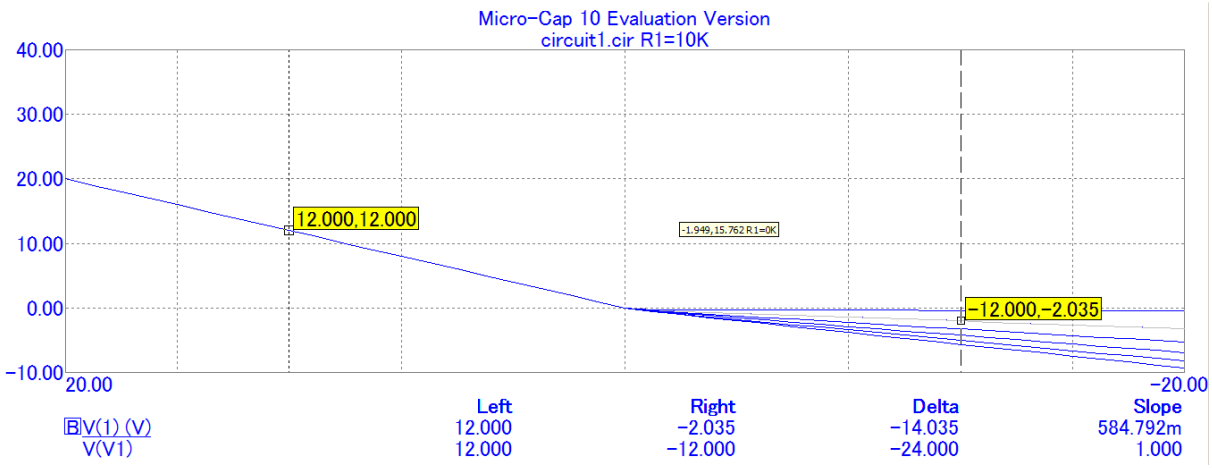
Při návrhu proudové a přepěťové ochrany bylo postupováno dle zadaných parametrů. K napájení je možné využít dva vstupy. První konektor je určen pro napájení do 2A a je realizován zdířkou Ø2.1x5.5mm. Ke kladnému pólu je připojena pojistka umístěná do držáku, který umožňuje jednoduchou výměnu. Dle zadání byla použita pojistka PP5 (Ø 5 x 20 mm) do 2A. Druhá možnost napájení je určena pro proud do 10A. Jsou použity barevně odlišené zdířky (červená pro kladný pól, černá pro záporný) kladný pól je napojen na pojistku automobilového typu IMP10 do 10A. Pojistka je umístěna do držáku, který umožňuje její rychlou výměnu. Obě napájecí cesty jsou spojeny na spínači. Sepnutí spínače je signalizováno LED diodou modré barvy. Pro dosažení malé napěťové ztráty (do 100mV) bylo nutné využít ochranu s minimálním vstupním odporem. Přepěťová ochrana je realizována tranzistorem IRF5305. Jedná se o tranzistor typu MOSFET s kanálem typu P. Mezi velké výhody patří již zmiňovaný malý vstupní odpor a možnost přenosu velkých proudů. Vstup tranzistoru gate je připojen k zemi přes rezistor s odporem 10KΩ. Mezi gate a drain je připojena Zenerova dioda, která chrání tranzistor před zničením přepětím. Rezistor (R1) byl volen tak, aby nemohlo dojít ke zničení diody a zároveň, aby nedocházelo k přenosu napětí v případě přepólování. K zjištění odpovídající velikosti byla využita analýza obvodu v programu Micro-Cap 10. Z hodnot zjištěných analýzou byla určena hodnota odporu R1 10KΩ. Při této hodnotě je při přepólování výstupní hodnota napětí do -2.035V a proud procházející Zenerovou diodou při 20V je 2mA.



Obrázek 49: Schéma proudové ochrany a ochrany proti přepólování

Prvky v programu byly definovány dle reálných parametrů, aby byla analýza, co nejpřesnější. Odpor vodičů byl pro jejich krátkou délku zanedbán. Z analýzy je patrné, že při správném zapojeném napětí nedochází ke ztrátám a napětí na vstupu obvodu je rovno napětí na výstupu. Toto je znázorněno v levé části charakteristiky. Při zapojení přepólovaného napětí (záporný pól na kladnou svorku a kladný pól na zápornou svorku) je při vstupních -12V na výstupu pouze -2.035V. Z pravé části charakteristiky je patrné, že záporné výstupní napětí je tím větší, čím je větší odpor R1. Pokud by nebyl použit žádný odpor, bylo by záporné napětí pouze -0.382V, ale

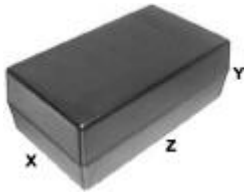
při zapojení napětí většího než -15V by hrozilo zničení diody a následně i tranzistoru. Záporné napětí -2.035V neumožní spouštění mikrokontroleru a zároveň díky rezistoru R1 nedojde k poškození diody.



Obrázek 50: Grafické znázornění simulace chování ochranného obvodu

9. Mechanické řešení krabiček a konektorů

Aby nedošlo k poškození či nedovolené manipulaci s deskami plošných spojů a konektory, byly oba uzly uloženy do černých, plastových krabiček. K upevnění byly použity kovové distanční sloupky, které byly navrtány do plastu a následně zajištěny maticí. Rozměry plastových krabiček jsou 140x50x190(x,y,z).



Obrázek 51: Plastová krabička pro uložení DPS

Konektory jsou umístěny na delší straně krabičky. U řídícího uzlu jsou na jedné straně konektory napájecí a na druhé konektory datové. Řízený uzel je napájen po sběrnici.

Obrázek 52: Ukázky hardwarového řešení a konektorů

Závěr

Teoretická část práce přináší shrnutí základních informací o protokolech LIN a CAN. Jelikož se jedná o složité standardy, bylo nutné vybrat pouze informace, které byly nezbytné pro tuto práci. Důraz byl kladen na formát přenášených zpráv a na popis komunikační struktury. Pro zvládnutí této práce bylo nutné hlubší pochopení standardů LIN a CAN. Déle v teoretické části najdeme stručný popis vývojových prostředí a programového vybavení, které bylo zapůjčeno společností Freescale. Vývojové prostředí CodeWarrior bylo využito pro programování v jazyce C a pro debuggování kódu. Prostředí FreeMaster bylo použito pro monitorování, ovládání a řízení sběrnice. V rámci teoretické části byly uvedeny i dva testery sběrnic. Jedná se o přístroje, které jsou dostupné na českém trhu. První část praktické práce je zaměřena na tvoření programu pro obsluhu komunikace na sběrnicích LIN a CAN. Programy vychází ze softwarového vybavení, které poskytla společnost Freescale. Jelikož jsou drivery vlastnictvím této společnosti, nejsou v práci podrobněji popsány. Popsány jsou důležité části kódu, které byly napsány autorem. Debuggování kódů bylo provedeno v prostředí CodeWarrior za použití přípravku USB Multilink PE. Propojení komunikačních uzlů s prostředím FreeMaster funguje spolehlivě. Grafické prostředí vytvořené v HTML kódu je přehledné a jednoduché. Ukázalo se, že použití Visual Basic Scriptu ve větší míře, může mít za následek prodloužení reakční doby monitorovacího softwaru. Ochranné obvody byly simulovány a jejich funkčnost byla odzkoušena úmyslným přepólováním napájecího napětí. Byly splněny všechny definované cíle až na ukládání do vnitřní paměti EEEPROM. Tato část bude pravděpodobně nahrazena ukládáním do přídavné paměti. K ověření spolehlivosti sběrnice je možné využít tři různé módy, tak jak bylo definováno v cílech praktické práce.

Literatura

[1] Nicolas Joseph Cugnot: Otec automobilu. *Eurooldtimers.com: The world of historic vehicles and classic cars* [online]. 2012 [cit. 2013-02-09]. Dostupné z: http://www.eurooldtimers.com/cze/historie-clanek/773-nicolas-joseph-cugnot--otec-automobilu.html.

[2] Moteur à explosion à 4-temps. *Eureka* [online]. 2012 [cit. 2013-02-09]. Dostupné z: http://eurekaweb.free.fr/sh1-moteur_explosion_4T.htm.

[3] Rudolf Diesel. *About.com: Inventors* [online]. 2012 [cit. 2013-02-09]. Dostupné z: http://inventors.about.com/library/inventors/bldiesel.htm.

[4] EuroEkonom. *Henry Ford* [online]. 2012 [cit. 2013-02-09]. Dostupné z: http://www.euroekonom.cz/osobnosti-clanky.php?type=jz-ford.

[5] Elektronika v automobilech. *Chip online CZ* [online]. 2012 [cit. 2013-02-09]. Dostupné z: http://earchiv.chip.cz/cs/earchiv/vydani/r-2010/casova-osa-03-10.html.

[6] Sběrnice. *Autodiagnostika* [online]. 2012 [cit. 2013-02-09]. Dostupné z: http://www.carmotor.cz/perspektivy-automobilove-elektroniky/.

[7] Sběrnice (bus). *Fakulta informatiky Masarykovy univerzity* [online]. 2005 [cit. 2013-02-09]. Dostupné z: http://www.fi.muni.cz/usr/pelikan/ARCHIT/TEXTY/BUS.HTML.

[8] Automotive buses. *Automotive buses and Vehicle bus description* [online]. 2010 [cit. 2013-02-09]. Dostupné z: http://www.interfacebus.com/Design_Connector_Automotive.html.

[9] *VÝVOJ A MOŽNOSTI SOUASTNÝCH OBD SYSTÉM U OSOBNÍCH AUTOMOBILA MOTOCYKL* [online]. Brno, 2007 [cit. 2013-02-10]. Dostupné z: http://old.uk.fme.vutbr.cz/zobraz_soubor5717.pdf?id=257. Bakalářská práce. FAKULTA STROJNÍHO INŽENÝRSTVÍ ÚSTAV KONSTRUOVÁNÍ. Vedoucí práce DOC. ING. IVAN MAZREK, CSC.

[10] *Automa: časopis pro automatizační techniku* [online]. Praha: FCC Public, 2001 [cit. 2013-02-10]. ISSN 1210-9592. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=33717.

[11] CANoe/DENoe: Manual. Vector Informatik GmbH [online]. 2008, Rev. 4.1.1, s. 209 [cit. 2013-02-10]. Dostupné z: http://www.kemt.fei.tuke.sk/predmety/KEMT452_ERSA/_materialy/CANoe_Manual_EN.pdf.

[12] *ElektroRevue: LIN (Local Interconnect Network)* [online]. 2004 [cit. 2013-02-10]. ISSN 1213-1539. Dostupné z: http://www.elektrorevue.cz/clanky/04012/index.html.

[13] LIN Specification Package Revision 2.0ons. Motorola GmbH [online]. 2006, Rev. 2.1, s. 125 [cit. 2013-01-31]. Dostupné z: http://tge.cmaisonneuve.qc.ca/barbaud/R%C3%A9f%C3%A9rences%20techniques/Bus%20LIN/LIN-Spec_Pac2_1.pdf.

[14] CAN in Automotive. In: *Can 2.0a* [online]. 2010 [cit. 2013-03-27]. Dostupné z: http://www.can-cia.org/fileadmin/cia/specifications/CAN20A.pdf.

[15] Freescale. S12XF: 16-Bit Automotive Microcontroller [online]. 2010 [cit. 2013-03-27]. Dostupné z: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=S12XF .

[16] Electronics for transportation E4T [online]. 2010 [cit. 2013-04-28]. Dostupné z: http://e4t.cz/Vyrobky/CAN4t.aspx

[17] EHL Elektronika [online]. 2012 [cit. 2013-04-28]. Dostupné z: http://www.ehl.cz/index.php?page=bht-bus-handheld-terminal

Seznam obrázků

Seznam tabulek