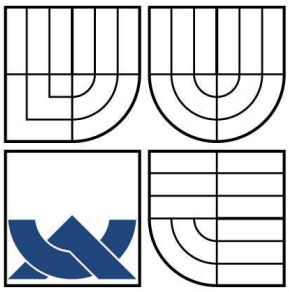
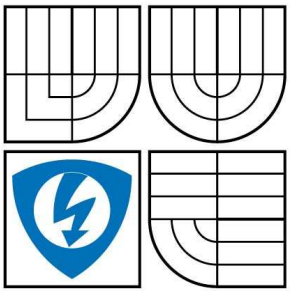


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE KOMUNIKAČNÍHO PROTOKOLU CAN 2.0 V JEDNOČIPOVÝCH MIKROPROCESORECH

IMPLEMENTATION OF THE CAN 2.0 COMMUNICATION PROTOCOL IN SINGLE-CHIP MICROPROCESSORS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. LUKÁŠ MACHÁLKA
AUTHOR

VEDOUCÍ PRÁCE — ING. JAROSLAV KOTON
SUPERVISOR

BRNO 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Lukáš Machálka

Bytem: Trojanovice 797, 744 01 Frenštát pod Radhoštěm

Narozen/a (datum a místo): 11.02.1984, Čeladná

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Kamil Vrba, CSc.

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - ⊠ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP:	Implementace komunikačního protokolu CAN 2.0 v jednočipových mikroprocesorech
Vedoucí/ školitel VŠKP:	Ing. Jaroslav Koton
Ústav:	telekomunikací
Datum obhajoby VŠKP:	

VŠKP odevzdal autor nabyvateli v*:

- ⊠ tištěné formě – počet exemplářů 2
- ⊠ elektronické formě – počet exemplářů 2

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: ……..

…… ……
Nabyvatel Autor

ANOTACE

Tato diplomová práce pojednává o implementaci komunikačního protokolu CAN 2.0 v jednočipových mikroprocesorech. V úvodu této práce je popsána problematika komunikačních protokolů, které se v současné době používají v automobilové technice s důrazem na komunikační protokol CAN 2.0. V další části této práce byly popsány konkrétní vlastnosti komunikačního protokolu CAN 2.0. U tohoto protokolu byl uveden princip přenosu dat, princip arbitráže na sběrnici a také struktura jednotlivých rámců, které se na ní mohou vyskytovat.

Dalším úkolem bylo porovnat dvě základní architektury a to architekturu FIFO a architekturu MailBox. Na základě měření proběhlo srovnání těchto architektur v typických aplikacích. Pro stanovení těchto výsledků byla navržena vývojová deska, která byla použita při návrhu koncepce sítě.

Klíčová slova: automobilová sběrnice, protokol CAN 2.0, architektura FIFO, architektura MailBox, filtrování zpráv

ABSTRACT

This diploma thesis deals with the implementation of the CAN 2.0 communication protocol in single-chip microprocessors. In the first part communication protocols currently used in automotive application and their properties are described, with impact on communication protocol CAN 2.0.

The second part is dedicated to the description of communication protocol CAN 2.0 particular features. The principle of data transmission, principle of bus arbitration and also the structure of individual frames, which can occur on it are mentioned. The final part relates with the task to compare FIFO architecture to MailBox architecture. On the basis of measurement the comparison of these architectures in typical applications was performed. Because of setting these results the prototyping board was designed. This board was utilized to design the net conception.

Keywords: automotive protocol, protocol CAN 2.0, architecture FIFO, architecture MailBox, message filtering

Prohlášení

Prohlašuji, že svou diplomovou práci na téma “Implementace komunikačního protokolu CAN 2.0 v jednočipových mikroprocesorech“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

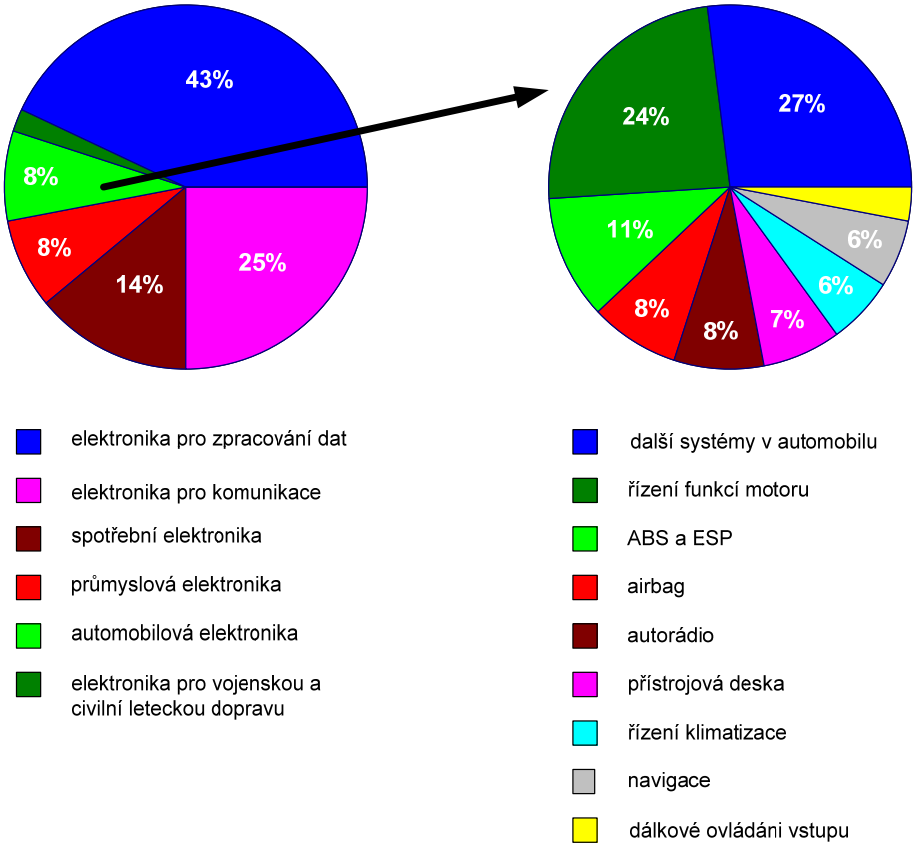
podpis autora

Obsah

1 Úvod:

Automobilový průmysl patří mezi jednu z vůbec nejvíce se rozvíjejících oblastí v současné době (Obr. 1.1). Významným subjektem v této oblasti je také Česká republika, která má velkou tradici ve výrobě automobilů – Škoda Auto Mladá Boleslav a Tatra Kopřivnice, ale také zahraniční výrobci, kteří zde stále častěji budují nové výrobní závody, jako například výrobce automobilů TPCA (Toyota, Peugeot, Citroen). Mezi dalšího velkého investora v oblasti výroby automobilů lze zahrnout Hyunadai Motors, která v současné době dokončuje výstavbu továrny v Nošovicích. V České republice se také nachází velké množství tuzemských i zahraničních firem, jejichž oblast působnosti se zaměřuje na vývoj a výrobu nejrůznějších komponent, ať už elektronického (elektronické komponenty se téměř z 20% až 35% podílejí na výrobní ceně automobilu), mechanického, optického či jiného charakteru. Mezi firmy, které se zaměřují na dříve zmíněné oblasti, patří Freescale Semiconductor, Brose, Dura, Visteon-Autopal a mnoho dalších. Výzkum a vývoj se pak značně soustředí na elektronické prvky, neboť mají vliv na snížení spotřeby paliva, zvyšování výkonu motoru. Dále tyto elektronické prvky nacházejí uplatnění v oblasti bezpečnosti posádky a v neposlední řadě i při komfortu cestování. Nové polovodičové technologie nahrazují původní mechanicky poháněné prvky elektronickými. Vyvstává tedy otázka integrace řídicích systémů a s tím i související implementace sběrnic. Sběrnicí jsou jednotlivá zařízení propojena a data distribuována do elektronických bloků, kde jsou vhodně zpracována.

Tato práce je zaměřena na komunikační protokol CAN 2.0, který se ve velké míře používá v automobilech, pro které byl primárně určen. V práci je diskutována implementace tohoto komunikačního protokolu v jednočipových mikroprocesorech. Tento protokol také nachází velké uplatnění i v jiných průmyslových odvětvích jako automatizaci, textilním průmyslu apod. Sběrnice CAN umožnila propojení různých zařízení a tak významně přispěla k podílu použití elektronických součástek v automobilu.



Obr. 1.1: Podíl jednotlivých průmyslových oblastí na trhu elektroniky a aplikaci v segmentu automobilového průmyslu [16]

2 Cíl této práce

Úvod této diplomové práce je zaměřen na sběrnice, které se používají v automobilové technice a to LIN, FlexRay a CAN 2.0. Hlavní důraz je pak kladen na popis komunikačního protokolu CAN 2.0. Jsou uvedeny oblasti použití a principy této sběrnice. Hlavní pozornost je pak věnována nejčastěji používaným architekturám typu FIFO a MailBox.

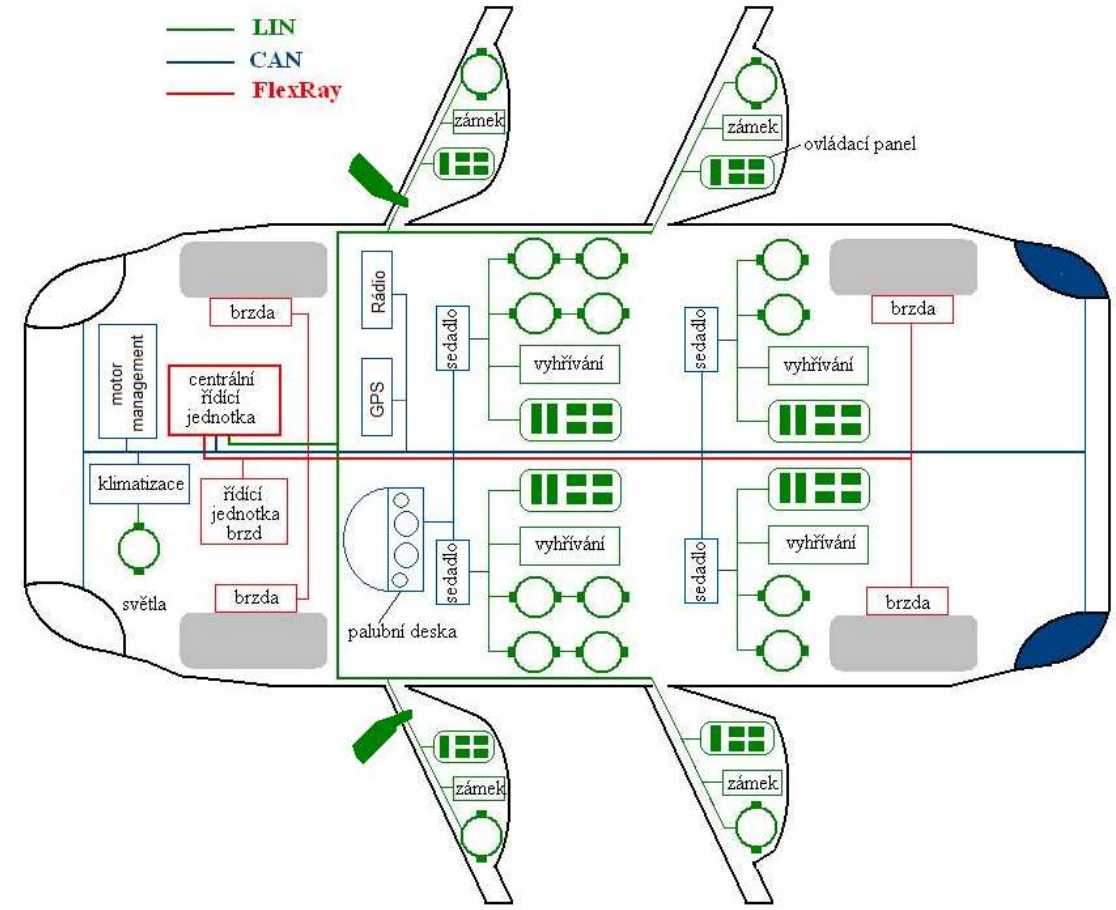
Druhá část diplomové práce je zaměřena na návrh vývojových desek, určených pro srovnání architektury FIFO a MailBox. Dále jsou zde přehledně shrnuty jednotlivé výsledky měření pro výše uvedené architektury v typických aplikacích.

3 Příklady datových sběrnic a jejich oblast použití v automobilech

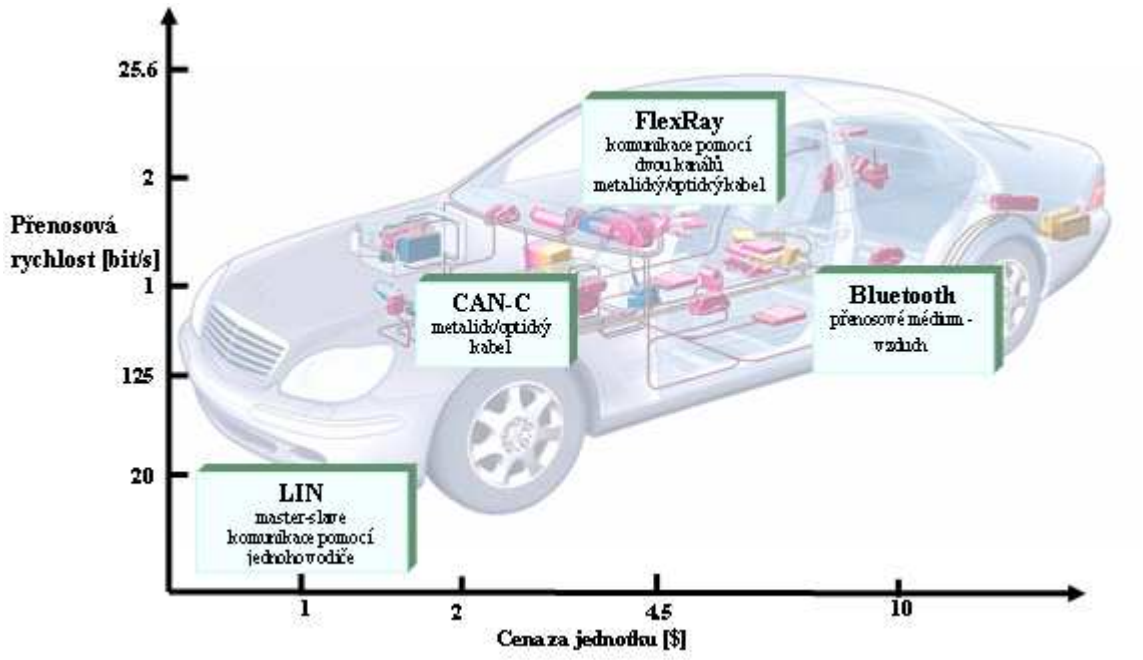
Pro komunikaci mezi jednotlivými zařízeními v automobilu se nejčastěji používají tyto tři typy sběrnic: FlexRay, CAN (Controller Area Network) a LIN (Local Interconnect Network). Využití těchto sběrnic pro jednotlivé typické úlohy v automobilu je vyobrazeno na Obr. 3.1. Mezi tyto úlohy lze zahrnout:

- aktivní tempomaty,
- elektronické stabilizační systémy (ESP),
- ovládání brzdového systému (ABS),
- systém regulace prokluzu kol (ASR),
- nastavení podvozku,
- prvky pasivní bezpečnosti - vzduchové polštáře (Air-Bag),
- motor management, řídící jednotky,
- senzory,
- centrální zamykání,
- elektrické stahování oken, osvětlení interiéru,
- komunikace rádia, GPS navigace s palubní deskou,
- klimatizace,
- elektronicky nastavitelná sedadla.

Porovnání cena/výkon zmíněných sběrnic je uvedeno na Obr. 3.2.



Obr. 3.1: Využití jednotlivých sběrnic v architektuře automobilu



Obr. 3.2: Porovnání jednotlivých typů sběrnic v poměru cena/výkon

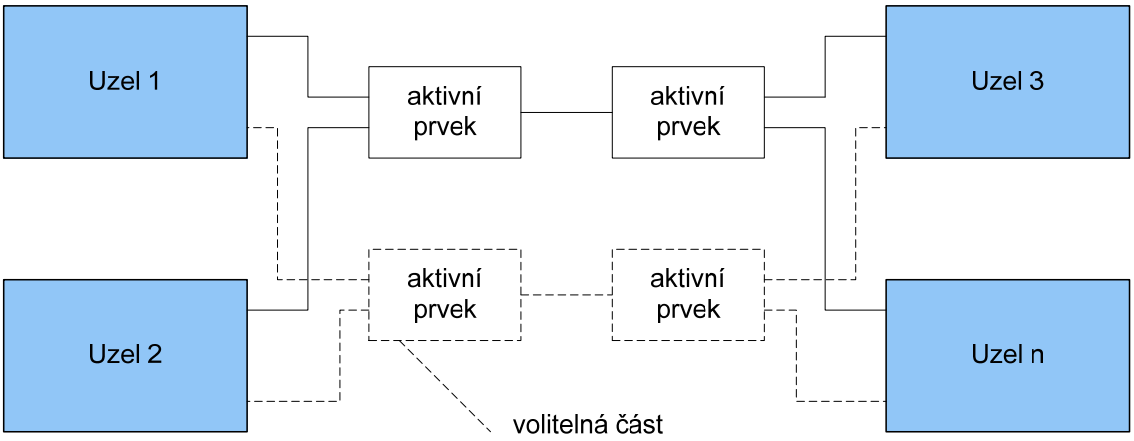
3.1 Datová sběrnice FlexRay

Sběrnice FlexRay byla vyvinuta za účasti významných členů, kteří se zabývají výrobou a vývojem komunikačních technologií pro automobily. Za tímto účelem vzniklo FlexRay konsorcium (www.flexray.com) zastoupené firmami Freescale Semiconductor, DaimlerChrysler, Philips, BMW. Další firmou, která se na vývoji této sběrnice podílela je firma Robert Bosch GmbH, která má mnohaleté zkušenosti se sběrnicí CAN.

Datová sběrnice FlexRay je vysokorychlostní sběrnicí využívající sériový přenos informace, která se využívá pro aplikace vyžadující největší možnou provozní spolehlivost. Tato technologie používá jako přenosové médium metalický, nebo optický kabel.

Mezi aplikace, které vyžadují největší možnou provozní spolehlivost v automobilu lze zahrnout: ovládání brzdového systému (ABS), řízení podvozku, řídící jednotky automobilů apod. Konkrétní využití sběrnice FlexRay v architektuře automobilu je vyobrazeno na Obr. 3.1. Datová sběrnice FlexRay je úzce spjata s pojmem X-by-wire. Použití této technologie znamená nahrazení přenosu informace mechanickými či hydraulickými vazbami, přenosem informace signálem elektrickým. Obdobné technologie se s úspěchem již řadu let využívají v leteckém průmyslu.

Komunikace na této sběrnici probíhá pomocí dvou kanálů viz Obr. 3.3, kde jeden z těchto kanálů může sloužit k navýšení datové rychlosti, nebo pomocí tohoto kanálu může dojít ke zdvojení datové komunikace (z důvodu zabezpečení přenášených dat). V případě poruchy na jednom kanále lze pro přenos informací použít i jediného kabelu. Přenosová rychlost u této koncepce je 10Mb/s. Pokud využijeme druhý kanál pro přenos informací (tj. bez zdvojení komunikace), můžeme dosahovat přenosové rychlosti až 20Mb/s [5], [13].



Obr. 3.3: Základní koncepce sběrnice FlexRay

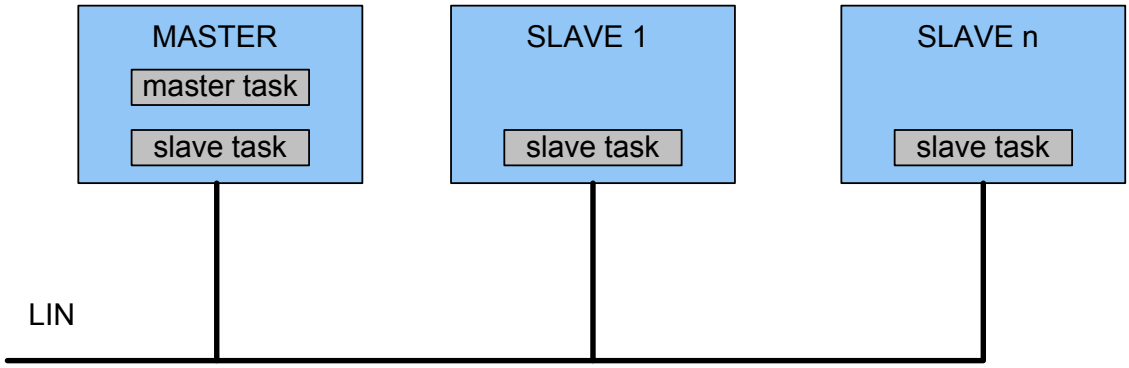
3.2 Sběrnice LIN (Local Interconnect Network)

Volvo a Volcano Communication (VCT) v roce 1996 vyvinuli pro Volvo S80 protokol založený na UART/SCI komunikaci, který nazvali Volcano Lite. V prosinci roku 1998 Audi, BMW, DaimlerChrysler a Wolksvagen vytvořili LIN konsorcium (www.lin-subbus.org) ve spolupráci s firmou VCL a firmou Freescale. V roce 2002 LIN konsorcium vydalo specifikaci s označením LIN 1.3. Tato specifikace v sobě zahrnovala změny hlavně ve fyzické vrstvě. Tyto změny vedly k lepší komunikaci mezi jednotlivými uzly v síti. V září roku 2003 byla vydána specifikace LIN 2.0, která přináší podstatné rozšíření diagnostických funkcí protokolu [11], [12]. V současné době je na trhu specifikace, která nese označení LIN 2.1.

Sběrnice LIN (Local Interconnect Network) se používá v automobilech pro aplikace, které nevyžadují spolehlivé doručení. Výhody této sběrnice nacházejí uplatnění v aplikacích jako jsou stahování oken v automobilu, dále pro centrální zamykání, vyhřívání a ovládání zrcátek, ovládání klimatizace, posuv sedadel, elektronické ovládání střešního okna atd. Tato sběrnice najde široké uplatnění i v jiných odvětvích, než je automobilový průmysl. Například může být použita ve spotřební a bílé elektronice. Možnost využití této sběrnice spolu s jejím propojením na nadřazené systémy je na Obr. 3.1 [8].

Při definici LIN koncepce se vycházelo z předpokladu, aby cena za koncové zařízení se pohybovala v oblasti 1€. Hlavní koncepce této sběrnice je jednovodičový asynchronní sériový přenos informace. Dále jsou potřebné dva vodiče a to napájecí vodič +12V a vodič pro společnou zem GND. Jak už vyplývá z topologie sítě single

master/multiple slave, tak na tuto sběrnici může být připojeno jedno *Master* zařízení a několik *Slave* zařízení. Celkový počet zařízení na jedné sběrnici typu LIN může být 16. Vždy je třeba použít jedno hlavní řídící zařízení typu *Master*. Z důvodu maximální přípustné kapacity a úbytku napětí na sběrnici je její maximální délka 40m. Přenosová rychlost u této sběrnice se pohybuje v rozmezí 1kb/s až 20kb/s [5], [11] a [12].



Obr. 3.1: Základní koncepce sběrnice LIN

3.3 Bezdrátová technologie Bluetooth

Technologie Bluetooth byla vyvinuta firmou Ericsson. Za účelem vybudování této technologie vznikla organizace s označením SIG (www.bluetooth.com) [15]. Na rozvoji Bluetooth se podílejí zejména firmy SonyEricsson, Freescale, Nokia, IBM a Toshiba.

Této technologie se v automobilové technice využívá především pro komunikaci mezi zařízeními jako jsou mobilní telefony (připojení k handsfree sadě), zařízení PDA, navigační systémy apod., vždy však pro aplikace komfortního charakteru.

Technologie Bluetooth využívá jako přenosové médium vzduch, lze tedy velmi rychle budovat lokální sítě bez nutnosti mechanického propojení komunikujících zařízení. Přenosová rychlost této technologie dosahuje až 723kb/s v závislosti na překážkách, které by mohly bránit přenosu. Pro přenos je použito bezlicenční kmitočtové pásmo 2,4 GHz [8], [14], [15].

4 Komunikační protokol CAN 2.0

4.1 Historie komunikačního protokolu CAN (Controller Area Network)

Vývoj datové komunikační sítě pod názvem CAN (Controller Area Network) byl zahájen Německou firmou Robert Bosch GmbH v roce 1983. V únoru 1986 tato firma představila sběrnici CAN na kongresu Společnosti automobilových inženýrů SAE (Society of Automotive Engineers) v Detroitu. Následující rok byly uvedeny první obvody, které obsahovaly CAN řadiče. V roce 1991 vydala společnost BOSCH specifikaci CAN 2.0 [2].

V roce 1992 byla ustanovena komise CiA (CAN in Automation), jejíž cílem je marketingový a technický rozvoj sběrnice CAN. V současné době je součástí této organizace 519 společností [17]. Po krátké době vzniku tohoto sdružení byl vydán dokument týkající se fyzické vrstvy, kde bylo doporučeno striktně držet se normy ISO 11898 [1]. Díky tomuto opatření za krátkou dobu vymizely z trhu produkty pracující na standardu RS-485. Tato organizace vydala tzv. *Zelenou knihu* se specifikací CAL (Can Application Layer), která standardizuje aplikační vrstvu.

V následujícím období došlo k velkému rozvoji této sběrnice. Významní výrobci (Motorola, Siemens Semiconductors, NEC a další) začali vyrábět velké množství čipů, které obsahovaly CAN řadiče. Prvním výrobcem, který implementoval tuto sběrnice byla firma Mercedes–Benz (1992), která tento protokol použila u vozů vyšší střední třídy. V tomto trendu následovali další významní výrobci automobilů jako Volvo, Saab, BMW, Škoda-Auto apod. [1], [3] a [7].

Další důležité časové mezníky vývoje sběrnice CAN [2], [4], [17]:

- 1994 – první mezinárodní CAN konference organizována sdružením CiA,
- 1994 – firma Allen-Bradley uvádí high-level protokol DeviceNet (optimalizován pro průmyslovou automatizaci),
- 1995 – sdružení CiA publikuje specifikaci protokolu CANopen – je určen pro výrobce vestavných systémů a řídicích systémů strojů,
- 2000 – vývoj Time Triggered CAN (TTCAN).

4.2 Úvod k sběrnici CAN

Jedná se o sériový komunikační protokol, jehož maximální přenosová rychlost je 1Mbit/s. Hodnoty jednotlivých přenosových rychlostí a dostupné vzdálenosti jsou uvedeny v Tab. 4.1 [1]. Z této tabulky vyplývá, že maximální přenosové rychlosti 1Mbit/s lze dosáhnout při maximální délce sběrnice 40m. Počet uzlů na sběrnici není teoreticky omezen, ale je nutno brát v úvahu možné zpoždění a výkonové zatížení sítě. Doporučený počet uzlů na sběrnici byl stanoven na 30. Pro přenos dat se používá metalické vedení, nebo stále častěji optický kabel. V CAN specifikaci není předepsáno fyzické médium ani jejich úrovně, z tohoto plyne, že může vytvořit síť s různým napětím, proudem, nebo světelným paprskem apod. Tento protokol se vyznačuje spolehlivým doručením zpráv. V automobilech nachází uplatnění u aplikací bezpečnostního charakteru jako ovládání brzdového systému, řízení podvozku, nebo se také používá k propojení jednotlivých podsítí, které jsou založeny na komunikačním protokolu LIN (Local Interconnect Network) viz Kap. 3.2. Další oblasti byly popsány v Kap. 3. [1], [3].

Mezi hlavní výhody tohoto komunikačního protokolu patří:

- vysoká přenosová rychlost,
- nízká cena polovodičových součástek,
- prioritní přístup zabezpečující doručení přednostních zpráv,
- snadná implementace nových uzlů,
- vysoká spolehlivost.

Mezi nevýhody tohoto komunikačního protokolu patří:

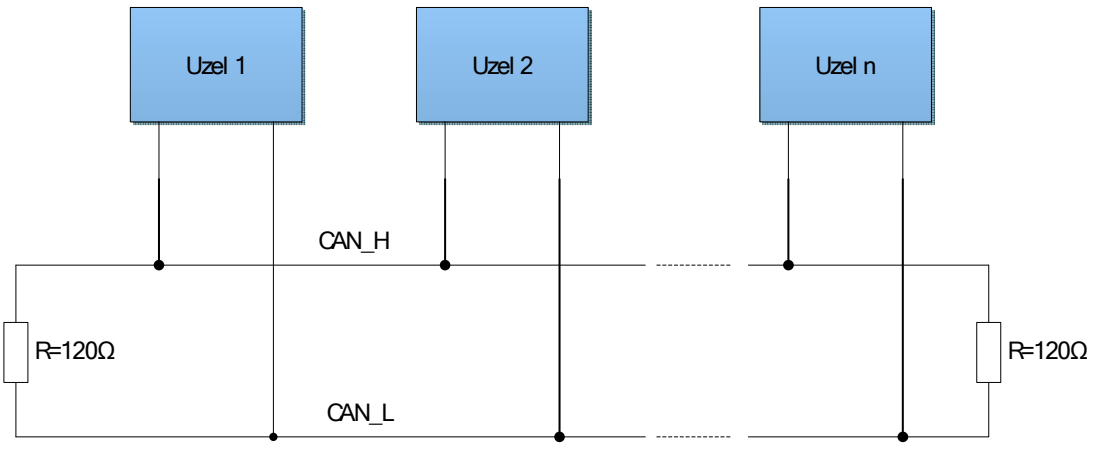
- malé množství přenášených dat v jedné zprávě (8B).

Tab. 4.1: Přenosová rychlost sběrnice v závislosti na délce vedení

délka sběrnice [m]	přenosová rychlost [Kbit/s]
40	1000
112	500
200	300
640	100
1340	50
2600	20
5200	10

4.3 Základní vlastnosti sběrnice CAN

Jedná se o multi-master komunikační protokol, tj. každý uzel v síti může být v určitém okamžiku hlavním uzlem (master) a tak řídit provoz na sběrnici. V případě, že se na sběrnici vyskytne uzel, který chybně pracuje, nebo je nefunkční, tak je vyřazen z komunikace, aniž by došlo k přerušení provozu na sběrnici. Základní koncepce sběrnice CAN je vyobrazena na Obr. 4.1. Pro komunikaci se používají dva datové vodiče, které nesou označení CAN_H a CAN_L. Vedení je zakončeno impedancí 120Ω [7], kvůli odrazům na vedení. Protokol CAN je definován normou ISO 11898. V současné době existuje specifikace CAN 2.0A a CAN 2.0B, které se liší ve standardním a rozšířeném formátu zprávy (odlišná délka pole identifikátoru). Specifikace CAN 2.0A používá standardní identifikátor o délce 11b a specifikace CAN 2.0B, která používá rozšířený identifikátor s délkou 29b. Podrobnější popis těchto vlastností je uveden v Kap. 4.4 [3].

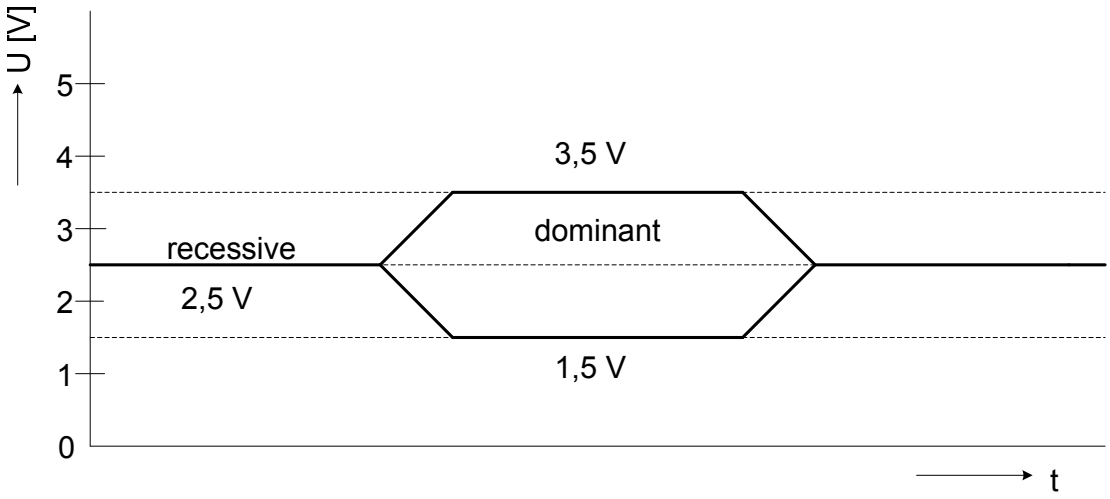


Obr. 4.1: Základní koncepce sběrnice CAN

4.3.1 Fyzická vrstva

Pro přenos informace se nejčastěji používají dva datové vodiče CAN_H a CAN_L v diferenciálním zapojení, které se řídí normou ISO 11898-2[1] [1]. Tato norma se nejčastěji používá ve spojení s protokolem CAN v automatizaci a automobilovém průmyslu. Protokol CAN používá další normy, např. ISO 11992, SAE J2411 [1], které se liší v rozdílných parametrech (elektrické vlastnosti, principy časování apod.). Na sběrnici jsou definovány dvě úrovně a to recessive a dominant viz Obr. 4.2. Rozdílové napětí vodičů CAN_H a CAN_L, dle normy ISO 11898 je pro stav recessive U_{DIFF} = 0V a pro stav dominant U_{DIFF} = 2V [1].

- Stavu recessive odpovídá hodnota log „0“.
- Stavu dominant odpovídá hodnota log „1“.



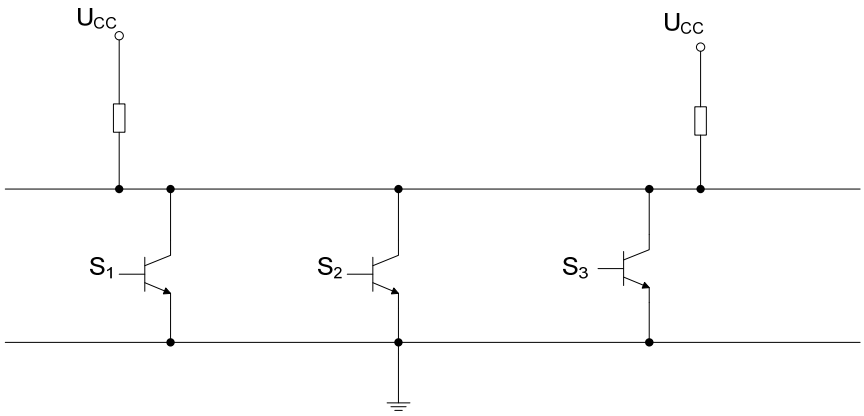
Obr. 4.2: Napěťové úrovně stavu recessive a dominant, dle normy ISO 11898-2 [1]

4.3.1.1 Realizace fyzické vrstvy

Fyzická vrstva může být například realizována pomocí hradel, jak je uvedeno na Obr. 4.3. Z tohoto obrázku vyplývá, že pokud je na sběrnici jeden tranzistor v sepnutém stavu, pak na sběrnici je stav dominant[2]. Ostatní tranzistory jej neovlivní i v případě, budou-li sepnuty či rozepnuty. Pokud na sběrnici není sepnut žádný tranzistor, tak na sběrnici je úroveň log „1“, což odpovídá stavu recessive viz Tab. 4.2. Hodnoty stavu

[1] Definuje parametry na sběrnici.
[2] Odpovídá hodnotě log „0“.

recessive a dominant nejsou striktně určeny a skutečná reprezentace záleží na konkrétní realizaci fyzické vrstvy [1], [4].



Obr. 4.3: Příklad realizace fyzické vrstvy

Dalším příkladem pro přenos informace může být použito optické vlákno, kde stavu dominant odpovídá stav svítí a stavu recessive nesvítí.

Tab. 4.2: hodnoty jednotlivých spínačů a jejich výsledný stav na sběrnici za použití tří uzlů viz Obr. 4.3

	Hodnoty jednotlivých spínačů							
S_1	D	R	D	R	D	R	D	R
S_2	D	D	R	R	D	D	R	R
S_3	D	D	D	D	R	R	R	R
Stav na sběrnici	**D**	**D**	**D**	**D**	**D**	**D**	**D**	**R**

R – recessive log „1“

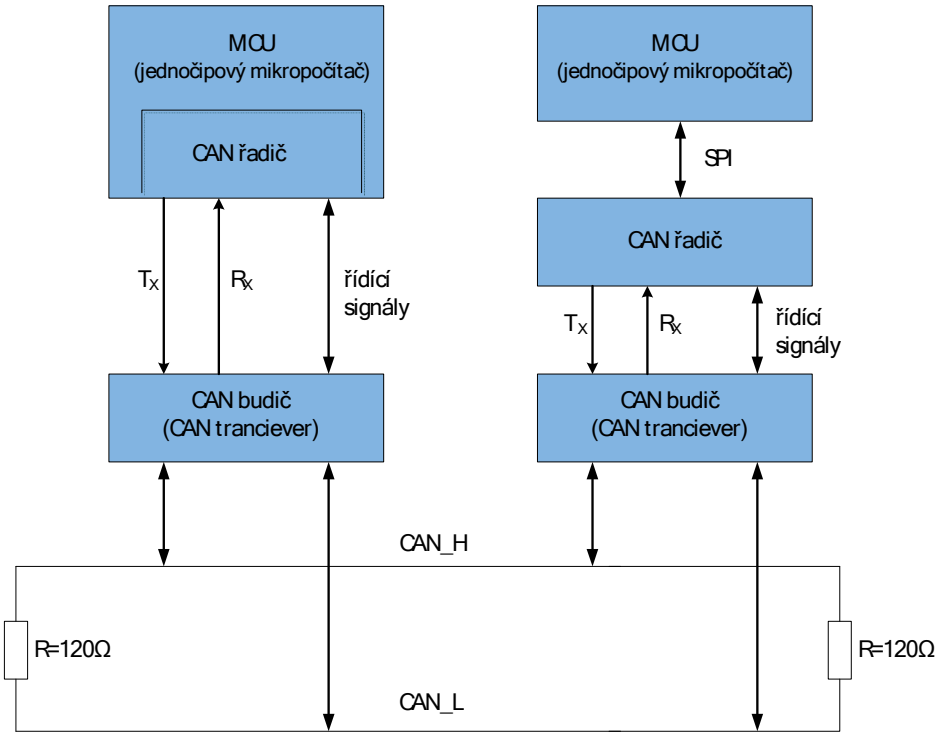
D – dominant log „0“

4.3.1.2 Hardwarové řešení sběrnice CAN

V současné době výrobci čipů nabízí dvě základní řešení, která umožní připojení jednotlivých uzlů na sběrnici. Na Obr. 4.4. jsou vyobrazena možná hardwarová řešení sběrnice:

- kompaktní řešení – mikroprocesor má na čipu implementován řadič protokolu CAN, ke kterému je dále připojen budič sběrnice viz Obr. 4.4.

  - MCU s řadičem např. Freescale MC9S12XDP512 (MSCAN řadič) [9],
  - Budič sběrnice CAN např. Freescale MC33989 [18],
- kombinované řešení – k mikroprocesoru je pomocí SPI (Serial Peripheral Interface) sériového periferního rozhraní připojen CAN řadič, ke kterému je dále připojen budič sběrnice viz Obr. 4.4.
  - MCU např. Infineon SAB 80C166 [19],
  - CAN řadič např. Infineon SAE 81C90 [20].



Obr. 4.4: Hardwarová řešení sběrnice

4.3.2 Přístup na sběrnici CAN

Protokol CAN umožňuje v jednom časovém okamžiku přenášet jednu zprávu na sběrnici, používá metodu CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Pokud by na sběrnici vysílalo současně více uzlů, tak by docházelo ke kolizím a ztrátám, které jsou nežádoucí. Proto každá zpráva obsahuje pole identifikátoru viz Kap. 4.4.1, ve kterém je obsažena priorita zprávy. Zprávy s vyšší prioritou (nižší identifikátorem) budou přeneseny přednostně. Pokud ve stejný okamžik požádá více uzlů o přenos, dojde k arbitráži a zpráva s vyšší prioritou je přenesena přednostně.

Ostatní uzly, které chtěly vysílat, mohou svůj přenos uskutečnit až po odeslání zprávy s vyšší prioritou. Z tohoto plyne, že žádná zpráva nebude ztracena. Tomuto se říká nedestruktivní řízení komunikace [1]. Podmínkou této bezchybné komunikace je, aby se na sběrnici nevyskytly zprávy se stejným identifikátorem zprávy. Postup zabezpečení proti tomuto nežádoucímu jevu je uveden ve specifikaci CAN 2.0 [3].

Na následujícím obrázku (Obr. 4.5) je vyobrazena komunikace na sběrnici, kdy v jednom časovém okamžiku současně požádaly tři uzly o přístup na sběrnici. V tomto případě nastane proces porovnávání identifikátoru (arbitráž), který je obsažen v každé zprávě v poli identifikátoru. V tomto případě se porovnávají jednotlivé bity na jednotlivých pozicích od každého uzlu, které zahájily vysílaní ve stejný časový okamžik, tzn. v prvním kroku se porovnává bit ID10 u všech tří zpráv. Všechny bity mají úroveň recessive, to znamená, že na sběrnici není žádný tranzistor v sepnutém stavu a tudíž je na sběrnici taky stav recessive.

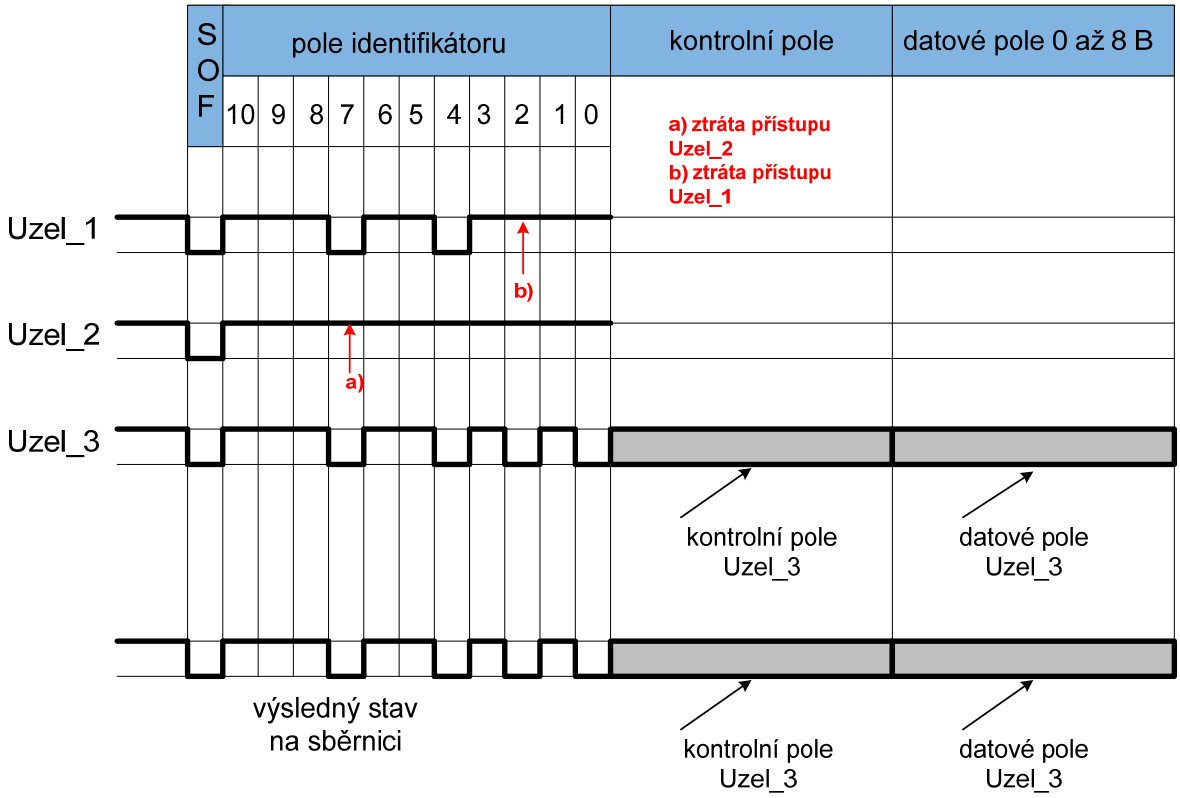
V dalším a následujícím kroku jsou všechny hodnoty identifikátoru (ID9 a ID8) totožné, tudíž i výsledný stav na sběrnici je stejný jako u prvního bitu (recessive).

Hodnota bitu ID7 v poli identifikátoru je pro Uzel_1 – dominant, Uzel_2 – recessive, Uzel_3 – dominant. V tomto případě je jeden z tranzistorů v sepnutém stavu (Uzel_1 a Uzel_2), a tak je na sběrnici stav dominant. Takto Uzel_2 prohrává arbitráž, protože obsahuje menší prioritu v poli identifikátoru [1]. Tento uzel však neztrácí možnost odeslat svou zprávu, ale o vysílání se může pokusit až po ukončení arbitráže a odeslání zprávy s vyšší prioritou. Na pozicích bitů pole identifikátoru (ID6 – ID3) se postupuje analogicky jako v předchozích případech [1].

V bodě *b)* vyznačeném na Obr. 4.5 ztrácí možnost přístupu na sběrnici Uzel_1. V tomto případě, kdy požádaly všechny tři uzly ve stejný časový okamžik přístup na sběrnici „vítězí“ Uzel_3, protože má nejvyšší prioritu identifikátoru. Tento uzel odešle zbylé části zprávy – kontrolní pole, datové pole apod., viz Kap. 4.4. Po odeslání této kompletní zprávy může opětovně požádat o přístup na sběrnici Uzel_1 a Uzel_2. V tomto případě Uzel_2 ztrácí přístup před Uzlem_1, takto je zpráva Uzlu_1 přenesena přednostně.

Výsledné pořadí přenesených zpráv na sběrnici:

1. Uzel_3
2. Uzel_1
3. Uzel_2



Obr. 4.5: Arbitráž tří uzlů o přístup na sběrnici ve stejný časový okamžik

4.4 Přenos zprávy na sběrnici CAN

Před vysláním zprávy na sběrnici si uzel, který chce začít vysílat, musí ověřit, zda na sběrnici nedochází ke komunikaci (sběrnice musí být volná – tzv. *bus free*). Výjimku tvoří chybový rámec (Error Frame), který může vysílání zahájit okamžitě, více Kap. 4.4.3. Uzel, kterému je umožněn přístup na sběrnici může vyslat data jednomu, nebo více uzlům současně. Součástí každé zprávy je identifikátor, pomocí něhož uzly připojené v síti rozpoznají, pro který uzel je daná zpráva určena. V případě vysílání zpráv od více uzlů v jednom časovém okamžiku nastává proces arbitráže, který je popsán v Kap. 4.3.2. Výhodou tohoto řízení přístupu na sběrnici je, že nedochází ke ztrátě žádné zprávy, jen zpráva s nižší prioritou je poslána později[3]. V případě jednoduchých aplikací, které mají uzly ovládat (např. vypni/zapni) nemusí být součásti zprávy datové pole. Tyto jednoduché aplikace lze ovládat pomocí identifikátoru, který je obsažen v každé zprávě. Tímto mechanismem můžeme snížit velikost (zprávy bez

[3] V případě, že je sběrnice volná (*bus free*).

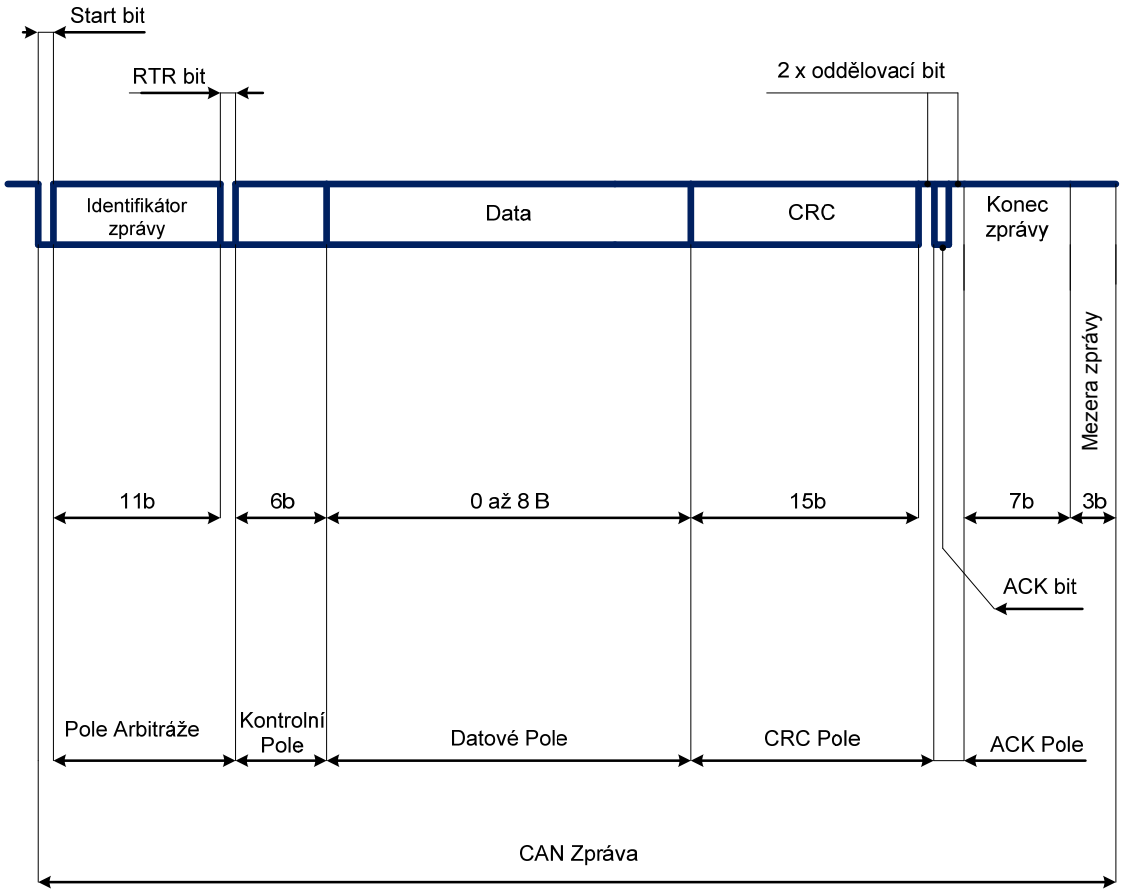
datového pole) přenášených zpráv na sběrnici a tak zvýšit propustnost sběrnice. Na sběrnici jsou definovány tyto rámce:

- Data Frame (datový rámec),
- Remote Frame (žádost o data),
- Error Frame (chybový rámec),
- Overload Frame (zpráva o přetížení).

4.4.1 Datový rámec (Data Frame)

Formát datového rámce přenášeného na sběrnici CAN je uveden na Obr. 4.6. Datový rámec se skládá z následujících části [1], [2]:

- Zpráva je zahájena jedním bitem, který je označen jako SOF (Start of Frame), který odpovídá úrovni dominant. Tímto bitem jsou zahájeny datové rámce, nebo rámce, které žádají o data.
- Po bitu SOF následuje pole identifikátoru (Message Identifier), jehož délka je ve standardním rámci 11b a v rozšířeném rámci, který je uveden ve specifikaci CAN 2.0B 29b. Specifikace CAN 2.0B je zpětně kompatibilní s verzí CAN 2.0A. Toto umožní na sběrnici provozovat současně, jak uzly podporující specifikaci CAN 2.0A, tak uzly podporující specifikaci CAN 2.0B.
- Po identifikátoru zprávy následuje RTR bit (Remote Transmission Request).
- Dále následuje kontrolní pole (velikost 6b).
- Datové pole (velikost 0 až 8B).
- Zabezpečení zprávy CRC (velikost 15b).
- Potvrzovací pole ACK (Acknowledgement Field).
- Konec zprávy EOF (End of Frame), délka 7 bitů ve stavu recessive.
- Mezera mezi jednotlivými zprávami (Intermission Field) - délka 3b ve stavu recessive.



Obr. 4.6: CAN zpráva (Data Frame) podle specifikace CAN 2.0A

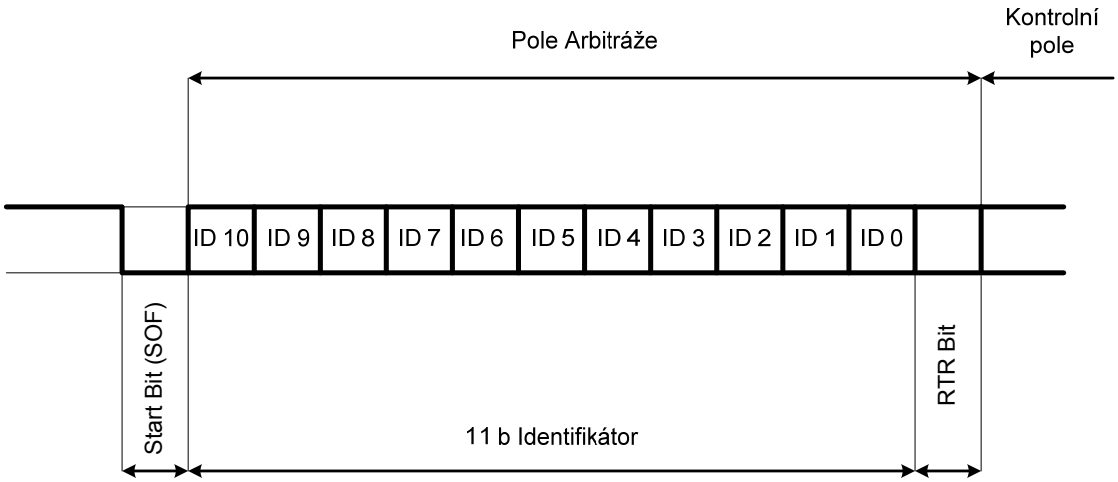
4.4.1.1 Pole Arbitráže

Toto pole se skládá z identifikátoru zprávy a RTR bitu. Priorita zprávy se určuje podle identifikátoru zprávy, který má délku 11b a jednotlivé bity nesou označení ID10 – ID0 pro standardní rámec[4]. Rozšířený rámec používá identifikátor, který má délku 29b (2^{29} adresovatelných zpráv). Tyto bity jsou označeny ID28 – ID0 (specifikováno v CAN 2.0B). Funkce identifikátoru je dále popsána v Kapitolách 4.3.2, 4.4 a 4.4.1.

RTR bit (Remote Transmission Request) může nabývat dvou stavů:

- Dominant (log „0“) – v tomto případě bit RTR signalizuje, že se jedná o datovou zprávu (Data Frame),
- Recessive (log „1)“ - v tomto případě bit RTR signalizuje, že se jedná o žádost o data.

[4] 2^{11} adresovatelných zpráv

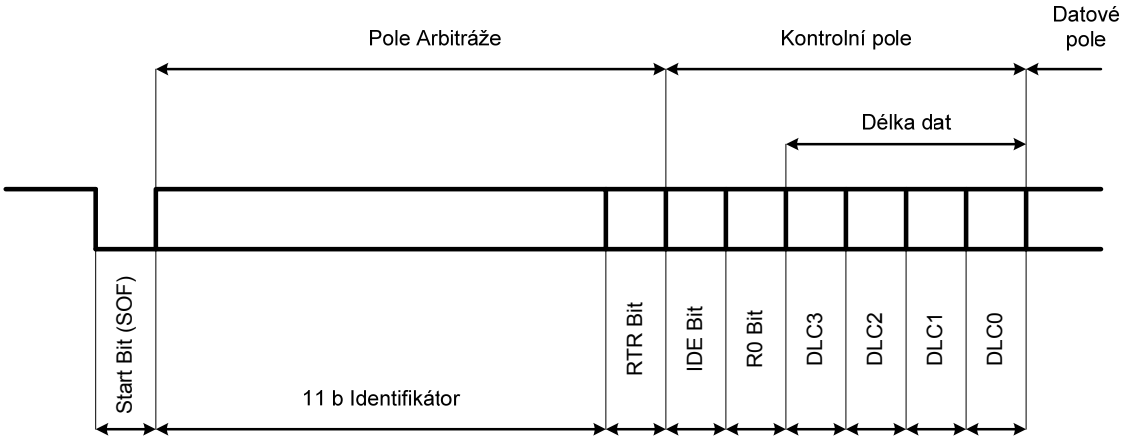


Obr. 4.7: Pole Arbitráže

4.4.1.2 Kontrolní pole

Kontrolní pole obsahuje bit IDE a R0. R0 bit není využit a je rezervován pro budoucí použití (hodnota tohoto bitu je standardně dominant). V bitu IDE je obsažena informace zda se jedná o standardní rámec (bit IDE bude obsahovat stav dominant), nebo rozšířený rámec (bit IDE bude obsahovat stav recessive). Po těchto dvou bitech následuje čtveřice bitů DLC3 až DLC0 (kombinací těchto bitů se vyjádří délka dat obsažená v datovém poli).

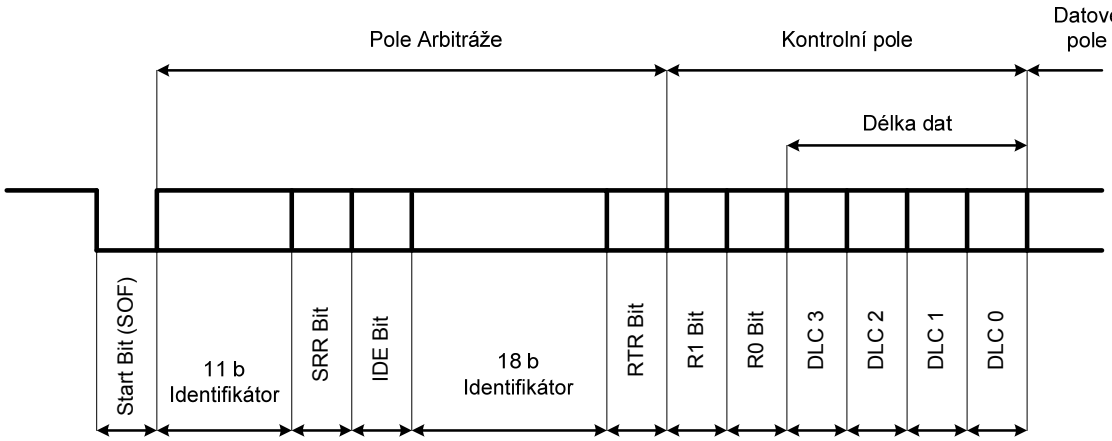
Standardní rámec včetně kontrolního pole podle specifikace CAN 2.0A:



Obr. 4.8: Standardní rámec včetně kontrolního pole podle specifikace CAN 2.0A

Rozšířený rámec včetně kontrolního pole podle specifikace CAN 2.0B:

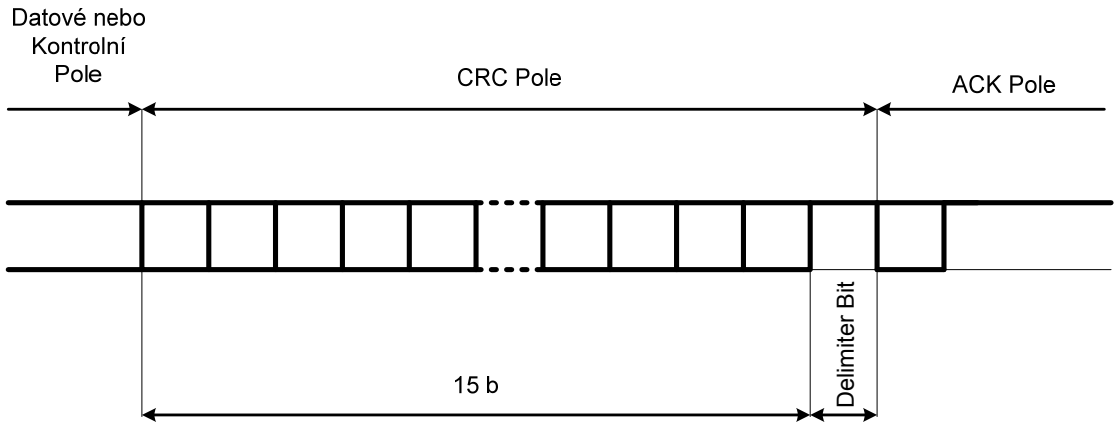
Pokud se bude jednat o rozšířený rámec, tak pole identifikátoru bude rozděleno na dvě části. První část bude mít velikost 11b, po těchto bitech následuje bit SRR (Substitute Remote Request), který má hodnotu recessive. Hodnota bitu IDE v rozšířeném rámci bude recessive. Po tomto bitu bude následovat dalších 18 bitů identifikátoru zprávy, které jsou zakončeny bitem RTR a bity R1 a R0 pro budoucí využití.



Obr. 4.9: Rozšířený rámec včetně pole podle specifikace CAN 2.0B

4.4.1.3 CRC Pole

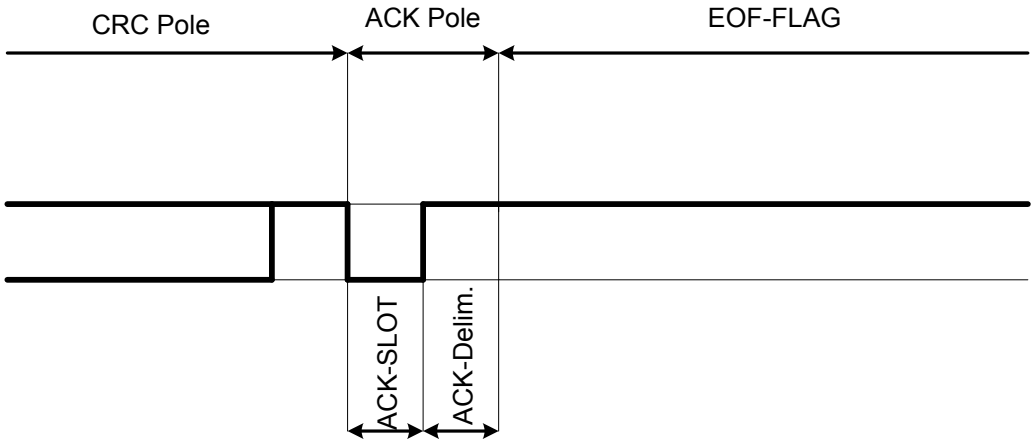
CRC pole má velikost 15b a jeden oddělovací bit (Delimiter Bit), který je ve stavu recessive. Každá zpráva je zakončena CRC kódem (Cyclic Redundancy Check) o délce 15b, který je vytvořen ze všech předcházejících bitů dané zprávy, které jsou obsaženy v části: Start Of Frame (SOF), Poli Arbitráže, Kontrolním Poli a Datovém Poli. Tento kód je vytvořen generujícím polynomem $x^{15}+ x^{14}+ x^{10}+ x^{8}+ x^{7}+ x^{4}+ x^{3}+ 1$. Pokud je detekována chyba jakýmkoliv uzlem na sběrnici, tak je vygenerována chyba CRC.



Obr. 4.10: CRC Pole

4.4.1.4 ACK Pole

ACK pole obsahuje dva bity a to bit ACK-slot a oddělovací bit ACK-Delimiter. Pokud je zpráva přijata úspěšně, tak hodnota bitu ACK-slot změní hodnotu z recessive na dominant. Potom je tato potvrzovací zpráva odeslána ostatním uzlům. Potvrzování přijetí zprávy je prováděno všemi uzly na sběrnici a to i uzly, které mají vypnuto filtrování zpráv (Acceptance Filtering). V případě detekování chyby na sběrnici je vyslána zpráva o chybě Error Frame viz Kap. 4.4.3. EOF (End of Frame Flag) – konec zprávy 7 recessivních bitů.



Obr. 4.11: ACK Pole

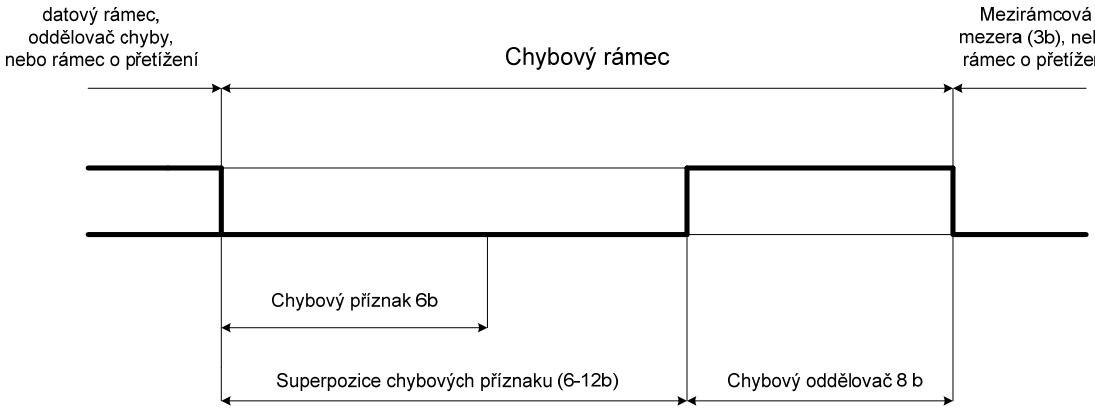
4.4.2 Žádost o data (Remote Frame)

Žádost o data má stejnou strukturu, jak je uvedeno v Kap. 4.4.1 (Datový rámec). Zda se jedná o *Datový rámec*, nebo *Žádost o data* určuje RTR bit:

- Dominant (log „0“) – v tomto případě bit RTR signalizuje, že se jedná o datovou zprávu (Data Frame),
- Recessive (log „1)“ - v tomto případě bit RTR signalizuje, že se jedná žádost o data [5], [6].

4.4.3 Chybový rámec (Error Frame)

Tato zpráva se generuje, jestliže došlo k nějaké chybové události v přenášené zprávě, např. chyba CRC, chyba bitu, chyba vkládání bitu, chyba rámce. Jestliže některá z dříve uvedených událostí nastane, tak je okamžitě generován chybový rámec (Error Frame). Podle stavu, ve kterém se daný uzel nachází se generuje šest bitů recessive, nebo šest bitů dominant příznaku chyby. Pokud je generován aktivní příznak chyby (přenášená zpráva je poškozena), tak i další uzly na sběrnici musí začít vysílat chybový rámec. Potom je hlášení chyb signalizováno superpozicí všech chybových příznaků, které uzly vysílají na sběrnici. Délka toho příznaku může být 6 až 12 bitů [1].

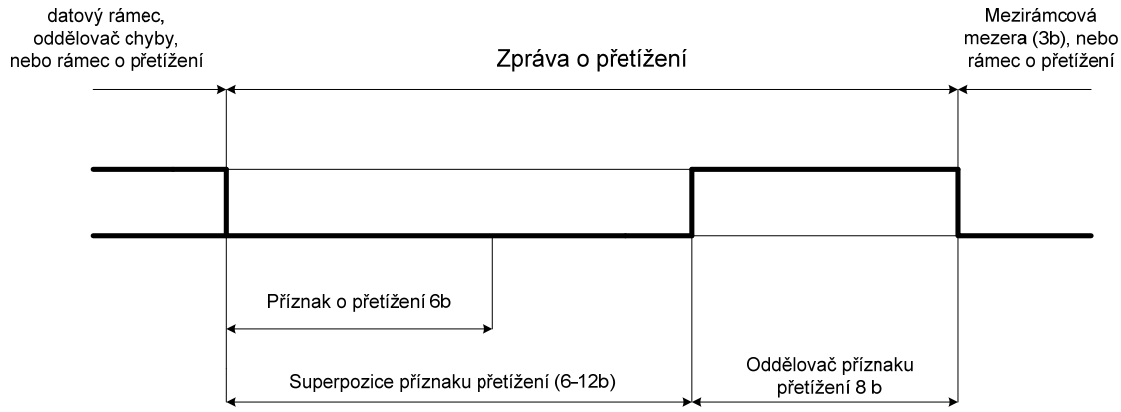


Obr. 4.12: Chybový rámec

4.4.4 Zpráva o přetížení (Overload Frame)

Zpráva o přetížení se skládá z příznaku přetížení a z mezery zprávy o přetížení. Tato zpráva se používá k prodloužení odezvy a může být zahájena po konci datové zprávy (End of Frame), po oddělovači chyby (CRC Delimiter), nebo předcházejícího

přetížení o zprávě (Overload Frame). Za pomocí tohoto mechanismu můžeme prodloužit čas odvysílání datové zprávy (Data Frame), nebo zprávy žádosti o data (Remote Frame) po dobu, kdy uzly nestačí zpracovávat zprávy [2].



Obr. 4.13: Zpráva o přetížení

5 Architektura MailBox a FIFO, filtrování zpráv (acceptance filtering) a post-processing

V současné době se v jednočipových mikropočítačích, které obsahují CAN rozhraní, používají dvě architektury organizace paměti. Je to architektura MailBox a architektura FIFO (First In First Out). Každá z těchto architekur přináší své výhody a nevýhody. V další části této diplomové práce se budu těmito architekturami podrobněji zabývat. Nejčastěji se lze setkat u výrobců, kteří vyrábějí mikrokontroléry podporující CAN sběrnici, s architekturou FIFO [9], [19], [20]. Architektury FIFO a MailBox nám určují, jakým způsobem budou zprávy, které jsou určeny pro daný uzel uloženy v bufferu (paměti). Aby uzel připojený na sběrnici rozpoznal, že zpráva, která se na sběrnici nachází je pro něj určena, musí porovnat identifikátor. Tento identifikátor se porovnává s identifikátorem, který má přijmout. To se děje pomocí akceptačních filtrů (acceptance filters), jejichž funkce je popsána v Kap. 5.1. Po průchodu zprávy akceptačními filtry je zpráva uložena v bufferu. Konkrétní princip akceptačních filtrů pro jednotlivé architektury bude uveden v následující části diplomové práce, kde se zaměřím i na jejich konfiguraci.

5.1 Filtrování zpráv (Acceptance Filtering)

Zprávy na sběrnici jsou vysílány všesměrově (broadcast), tímto způsobem jsou všechny zprávy dostupné uzlům připojených na sběrnici. Pokud by uzly připojené na sběrnici měly zpracovávat všechny zprávy, tak by docházelo k velkým nárokům, které by vedly ke zpoždění, zahlcení apod. Proto se používá mechanismus filtrování zpráv, který slouží k tomu, aby k danému uzlu přišly jen ty zprávy, které jsou pro něj určeny[5]. Zpráva, která je akceptována akceptačním filtrem je pak následně uložena v paměti (bufferu) a mikropočítač je o této skutečnosti informován. Způsob organizace uložení dat v paměti je uveden v Kap. 5.2.

[5] To se děje pomocí identifikátoru zprávy.

5.1.1 Funkce Akceptačních filtrů

Zpráva, která je určena konkrétnímu uzlu musí projít přes jeho akceptační filtry. Akceptačními filtry je filtrován pouze identifikátor zprávy, který porovnává hodnotu Identifikátorového pole s rozsahem hodnot, které má nastaveny v registrech Acceptance Code Register a Acceptance Mask Register [1], [2].

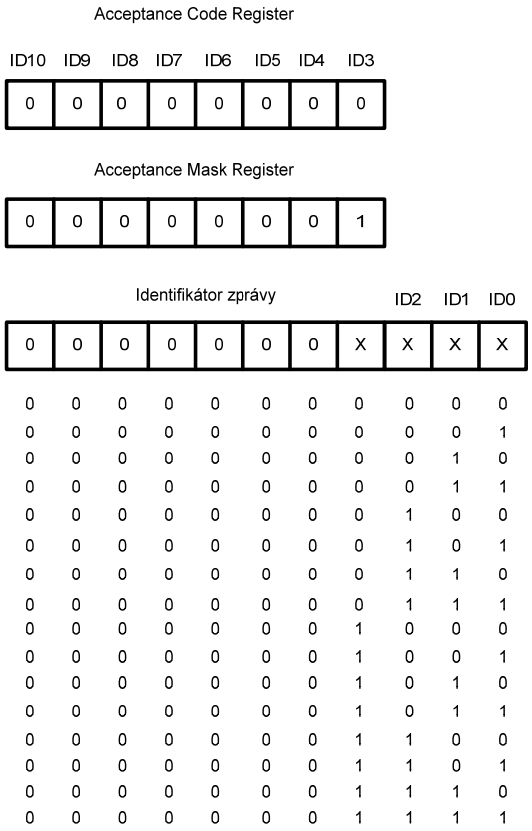
V případě příjmu jedné konkrétní zprávy bude použito Acceptance Code Register, se kterým se musí přesně shodovat identifikátor zprávy, jinak tato zpráva nebude přijata. Pokud bude uzlem přijímáno více než jedna zpráva, bude použito Acceptance Code Register spolu s maskou (Acceptance Mask Register). Tato maska definuje, které bity z Acceptance Code Register mohou v identifikátoru zprávy nabývat jakékoliv hodnoty (don´t care) viz Obr. 5.1.

Z uvedeného příkladu plyne (Obr. 5.1), že hodnoty na pozicích ID2 až ID0 nejsou zahrnuty v akceptačním registru a tudíž na těchto pozicích mohou být libovolné kombinace „0“ a „1“. Při této konfiguraci může být CAN řadičem přijato celkem 8 zpráv za podmínky, že na pozici bitu ID3 bude konkrétní hodnota „0“, nebo „1“ v identifikátoru zprávy.

Celkový počet, které by mohl CAN řadič uložit do své paměti je 16 (2^4 kombinací), pokud na pozicích ID10 až ID3 bude pouze nastavena hodnota „0“ (v Acceptance Code Register) viz Obr. 5.1. Na pozici bitu ID3 (v Acceptance Mask Register) je nastavena hodnota „1“ (označována jako don´t care) => touto pozicí může projít „1“, nebo „0“ a zbylé hodnoty, které neprochází akceptačním filtrem (ID2 - ID0), mohou nabývat různých kombinací [2].

Pokud by byla nastavena hodnota Acceptance Code Register ve všech pozicích na „0“ a hodnota v Acceptance Mask Register na všech pozicích na „1“, tak bychom mohli pro tento uzel směrovat 2048 různých zpráv[6]. Tento způsob je vhodné požít tam, kde je potřeba zpracovávat všechny zprávy na sběrnici.

[6] 2^{11} kombinací - pokud využijeme základního rámce, ve kterém je definován identifikátor o délce 11b.



Obr. 5.1: Filtrování zpráv

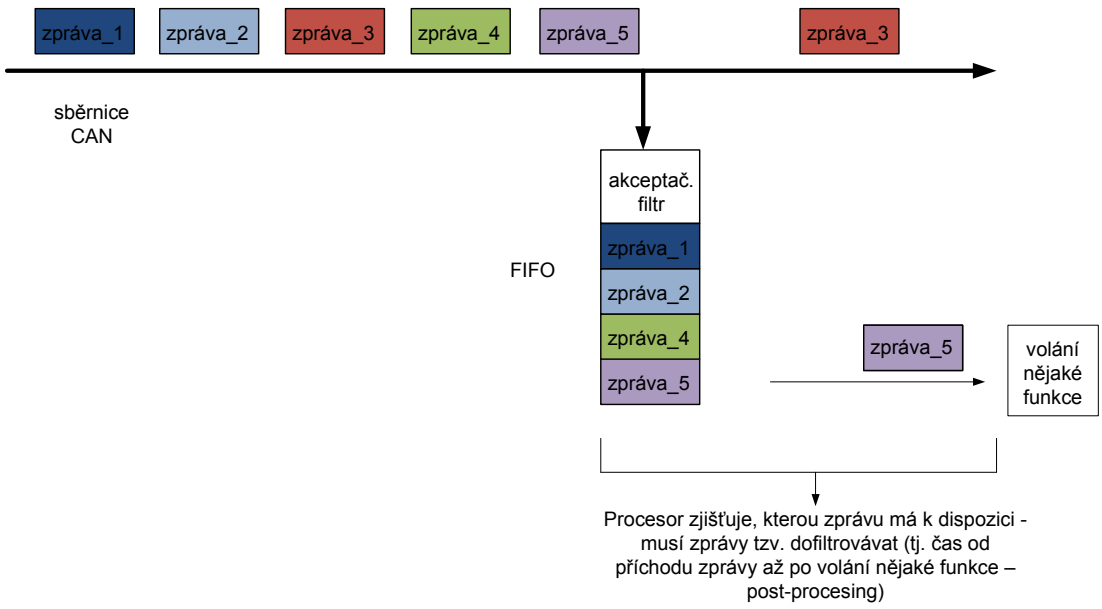
5.2 Architektura FIFO a MailBox

V této části diplomové práce se budu zabývat architekturou FIFO a architekturou MailBox. Jedná se o způsob řazení paměti příchozích a odchozích zpráv. Architekturu FIFO používá například firma Freescale u MSCAN řadiče, který je součásti mikrokontroléru MC9S12XDP512 [9]. Architekturu MailBox používá pro příjem zpráv mikrokontrolér od firmy ST typ 72561 [10].

V automobilech, jak už bylo řečeno, sběrnice CAN nachází uplatnění v různých oblastech bezpečnostních, komfortních apod. Na sběrnici mohou probíhat různé situace, protože každé zařízení použité v automobilu vyžaduje jinou četnost zpráv a tudíž i klade různé nároky na vytížení sběrnice. S tímto úzce souvisejí nároky, které jsou kladeny na procesor. Výběr architektury je důležitý tam, kde předpokládáme, že budeme provozovat na sběrnici velké množství zpráv a budeme klást velké nároky na mikroprocesor. To znamená, že budeme vyžadovat, aby nároky kladené na post-processing jádrem procesoru byly co nejmenší.

Při použití architektury FIFO ve spojení s CAN řadičem je předpoklad, že procesor musí zjistit jakou zprávu má k dispozici v FIFO bufferu. V tomto případě musí procesor tyto zprávy tzv. dofiltrovávat softwarově, aby zjistil, která zpráva mu dorazila. Na základě konkrétní zprávy je volána nějaká funkce. Z tohoto vyplývá, že doba od příchodu zprávy až po volání funkce vyžaduje čas spotřebovaný ze strany procesoru, tj. čas nezbytně nutný k tomu, aby byla nová zpráva přijata – tomuto se právě říká post-processing. Při použití architektury FIFO v jednočipových mikroprocesorech obsahující CAN řadič se předpokládá, že tuto architekturu je vhodné použít tam, kde potřebujeme zachytit větší množství zpráv po dobu, kdy procesor vykonává jinou činnost – např. ovládá nějakou aplikaci. Praktická měření týkající se architektury FIFO jsou uvedeny v Kap. 7.

Na následujícím obrázku Obr. 5.2 je blokově znázorněn provoz na sběrnici CAN v případě, kdy CAN řadič používá architekturu FIFO. Na sběrnici jsou zprávy_1 až 5. Jak je uvedeno na Obr. 5.2, tak přes akceptační filtr prošly zprávy 5, 4, 2 a 1. Procesor zjišťuje jakou zprávu má právě k dispozici, v tomto případě je to zpráva_5, která volá nějakou funkci.

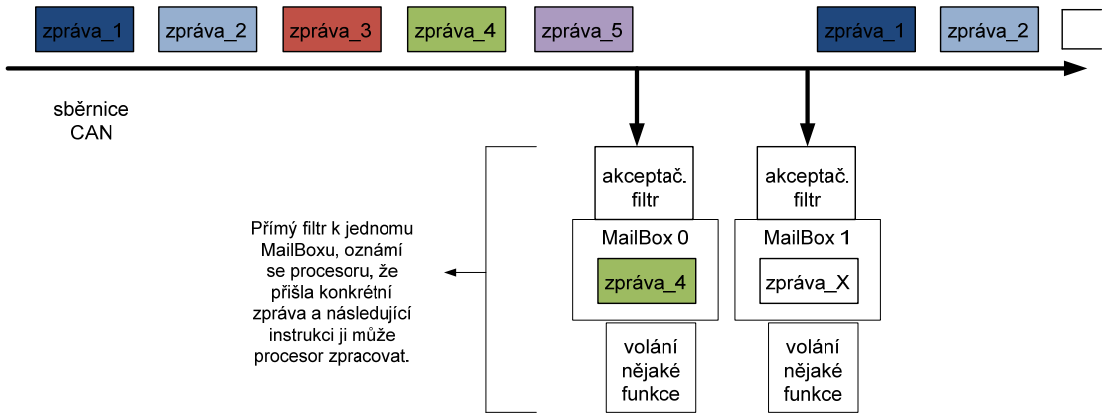


Obr. 5.2: Blokové schéma provozu na sběrnici CAN s využitím architektury FIFO

U architektury MailBox je vyhrazen pro každou příchozí zprávu paměťový prostor[7]. Při použití této architektury do paměti přichází konkrétní zpráva, jejíž identifikátor je nastaven v akceptačním filtru. Tento výběr se provádí hardwarově. Z tohoto plyne, že procesor není zatěžován výběrem z příchozích zpráv, má konkrétní zprávu ihned k dispozici. U architektury FIFO musí procesor příchozí zprávy tzv. dofiltrovávat, naopak u architektury MailBox toto není potřeba – téměř nulový post-processing. V praxi to probíhá tak, že do paměti je postoupena konkrétní zpráva, nebo lze nakonfigurovat, že do paměti přijde skupina zpráv. Po příchodu konkrétní zprávy se spustí přerušení (interrupt) a tím je procesor informován, že má k dispozici konkrétní zprávu od určitého MailBoxu. Takto může hned při následující instrukci volat danou funkci, kterou zpráva obsahuje. Při použití architektury MailBox se předpokládá, že tuto architekturu je vhodné použít tam, kde se na sběrnici vyskytují zprávy s malou četností. Takto je malá pravděpodobnost, že hned přijde další zpráva a přepíše předchozí zprávu. Praktická měření týkající se architektury MailBox jsou uvedeny v Kap. 7. V případě, kdy by se na sběrnici vyskytovalo více zpráv pro konkrétní uzel, tak jeden MailBox by byl nedostačující. Proto se tyto MailBoxy zdvojují, ztrojují apod., aby nedocházelo ke ztrátám zpráv. Toto sebou samozřejmě přináší zvýšení ceny koncového uzlu, protože jsou kladeny vyšší paměťové nároky, než je tomu u architektury FIFO.

Na Obr. 5.3 je uveden obdobný příklad, který je uveden na Obr. 5.2 s tím rozdílem, že zde je využito architektury MailBox. Filtrem zpráv projde zpráva_4, která je uložena v MailBoxu, který je označen MailBox 0. Takto má procesor ihned k dispozici zprávu a nemusí se zaobírat dofiltováním zpráv, jak je tomu u architektury FIFO.

[7] Každé zprávě je vyhrazen konkrétní MailBox.



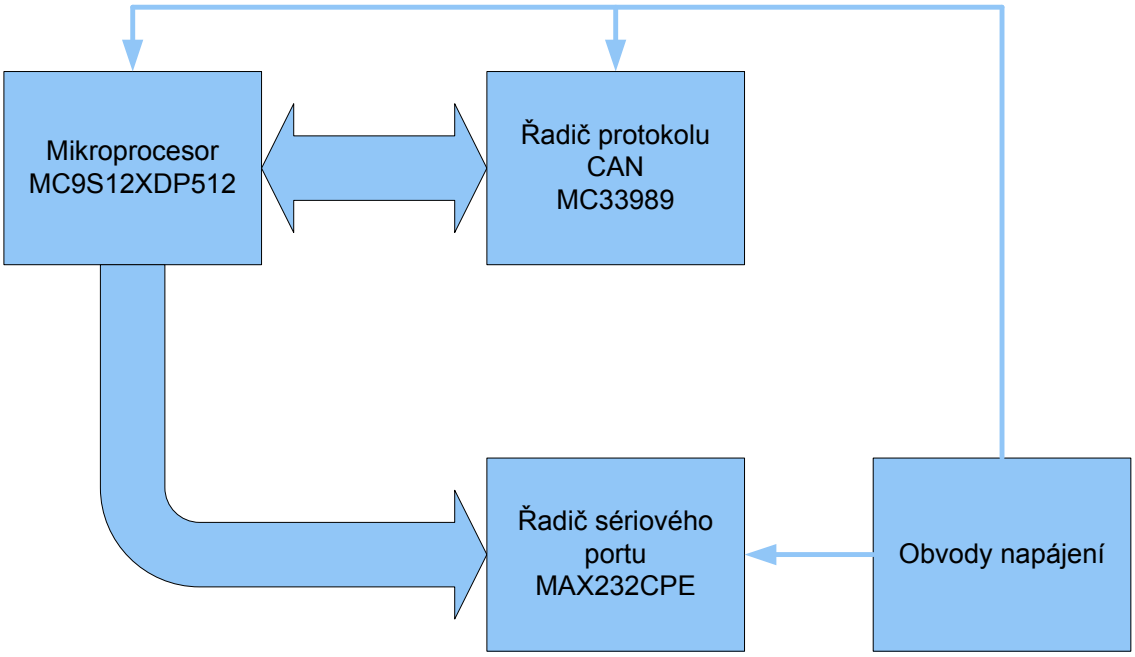
Obr. 5.3: Blokové schéma provozu na sběrnici CAN s využitím architektury MailBox

6 Návrh a realizace vývojových desek pro komunikační protokol CAN 2.0

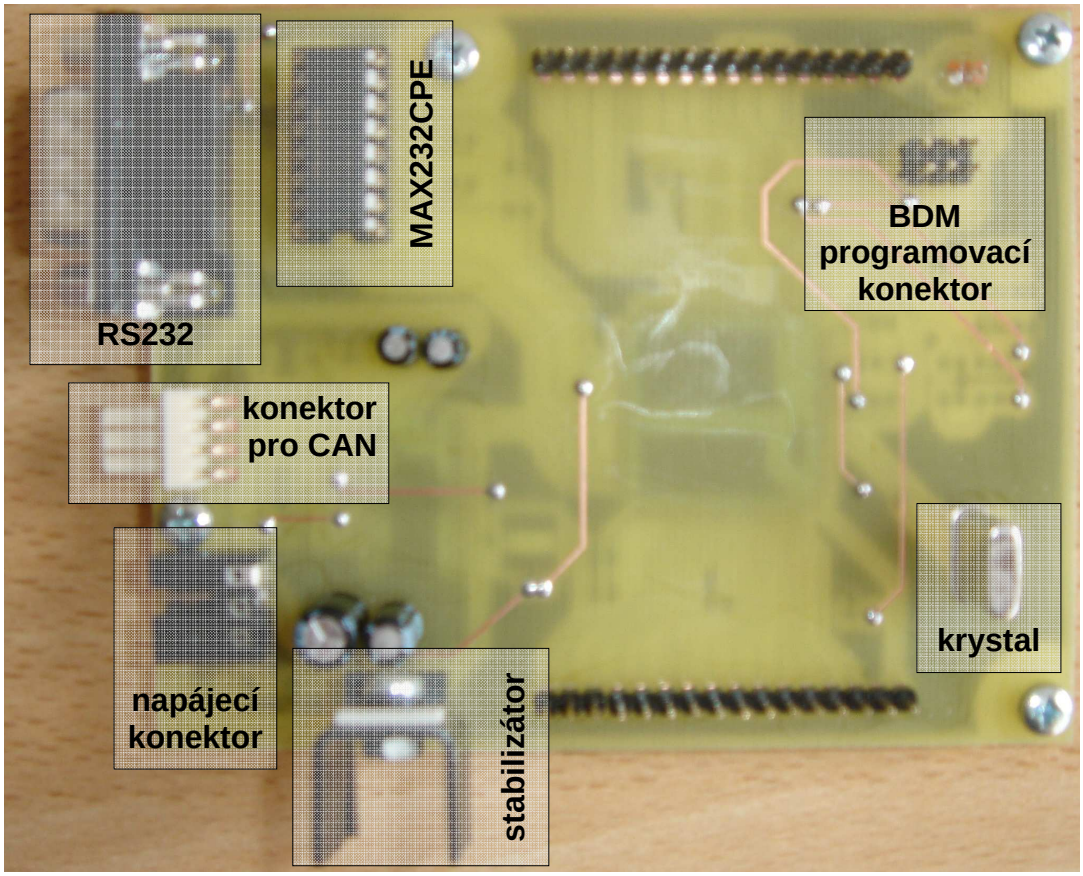
Nedílnou součásti diplomové práce bylo navrhnout vývojové desky pro praktické ověření výše zmíněné architektury FIFO a architektury MailBox. Na každé z těchto vývojových desek je implementován 16-bitový mikroprocesor od Firmy Freescale MC9S12XDP512 [9] a budič sběrnice CAN 2.0 MC33989 [18] zmíněné firmy. Vývojové desky jsou koncipovány tak, že každá z nich má své napájecí obvody a komponenty potřebné pro vytvoření sběrnice CAN 2.0. Tato sběrnice je realizována pomocí propojovacího kabelu, který je tvořen čtyřmi vodiči. Z toho jsou dva vodiče pro samotnou sběrnici a to vodič CAN_H a CAN_L, další dva vodiče jsou použity pro napájení +12V a vodič pro společnou zem GND. Na vývojové desce byl použit konektor RS232, jehož činnost zajišťuje integrovaný obvod MAX232CPE [22]. Volba použití samostatných vývojových desek byla zvolena z důvodu možné simulace výpadku spojení a také budoucího připojení do jakékoliv sítě podporující protokol CAN 2.0. Rozmístění výše uvedených součástek na vývojové desce je uvedeno na Obr. 6.2 a Obr. 6.3.

Na Obr. 6.1 je uvedeno blokové schéma zapojení vývojové desky pro sběrnici CAN 2.0. Podrobnější informace o návrhu jsou uvedeny v příloze diplomové práce, kde je dále uvedeno schéma zapojení, seznam použitých součástek a také návrh desky, který byl použit při výrobě. Celá část týkající se návrhu a výroby vývojových desek byla realizována v demo verzi programu Eaglu.

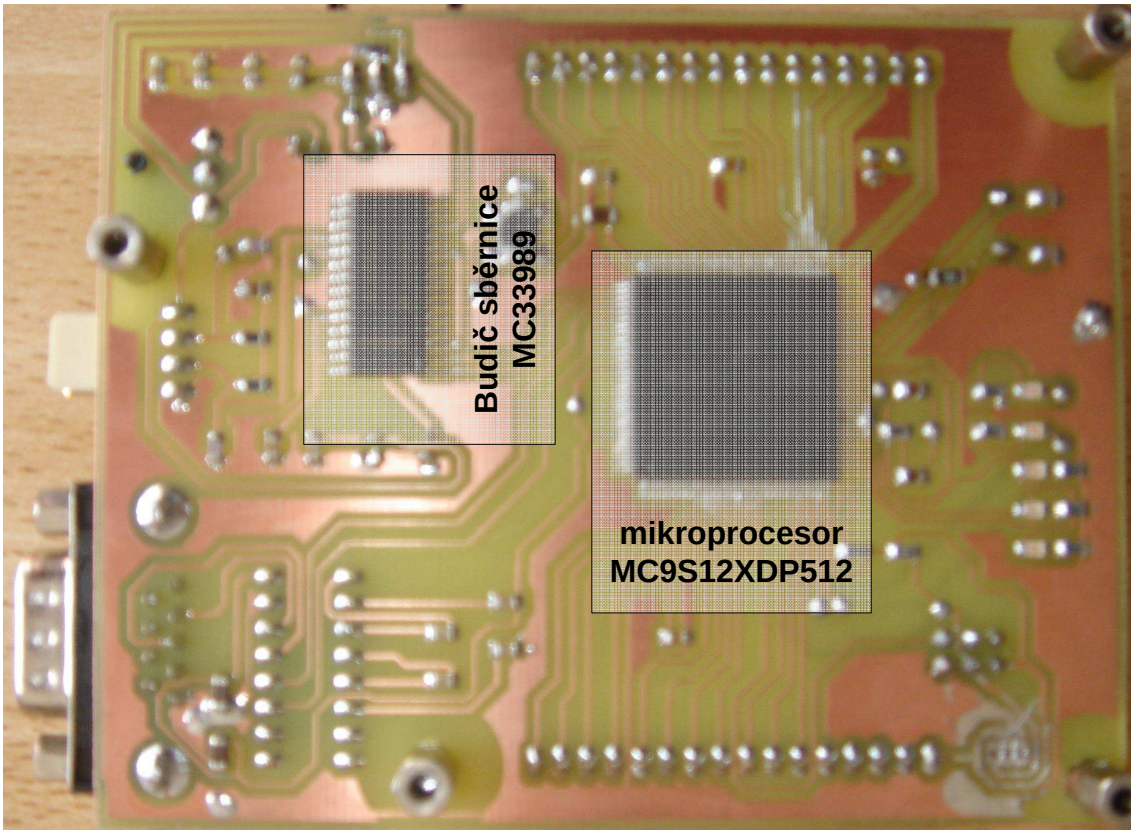
V následujících kapitolách 6.1 a 6.2 jsou stručně popsány základní vlastnosti hlavních obvodů, které byly použity při návrhu.



Obr. 6.1: Blokové schéma zapojení vývojové desky pro testování architektury FIFO a MailBox



Obr. 6.2: Pohled na vývojovou desku z vrchní strany



Obr. 6.3: Pohled na vývojovou desku ze spodní strany

6.1 Mikroprocesor MC9S12XDP512

Jedná se o 16-bitový mikroprocesor, který je použit při návrhu vývojových desek s pouzdrem LQFP (112 pinů). Řada mikrokontroléru S12XD je založena na efektivnějším jádru, než je tomu u mikrokontroléru HCS12 [9]. Mikroprocesory řady S12XD mají 2-5x vyšší výkon oproti řadě HCS12.

MC9S12XDP512- základní parametry:

- jedná se o 16-bitový mikroprocesor, který se vyrábí v několika pouzdrech a to 80-pin QFP, 112-pin LQFP a 144pin LQFP,
- zpětně kompatibilní s instrukční sadou HCS12,
- koprocesorová jednotka XGATE, používá rozšířený DMA modul, který zajišťuje vysokorychlostní přenosy mezi periferními jednotkami jako RAM, FLASH EEPROM a I/O porty,
- napájecí napětí (V_{CC}) 5 V a 3,3 V,
- kmitočet interní sběrnice 40 MHz,

- 512KB FLASH EEPROM,
- 32KB paměti RAM,
- 4KB paměti EEPROM,
- 5xMSCAN (Scalable Controller Area Network)
- 6xSCI (Serial Communacitions Interface),
- 3xSPI modul (Serial Peripheral Interface),
- jeden osmikanálový a jeden šestnáctikanálový 10 bitový A/D převodník (ADC),
- osmikanálový PWM modulátor,
- 59 vstupně/výstupních vývodů (80-pin QFP), 91 vstupně/výstupních vývodů (112-pin LQFP), 119 vstupně/výstupních vývodů (144-pin LQFP) [9].

XGATE:

XGATE modul umožňuje přímý přístup do DMA (Direkt Memory Access), standardní moduly umožňují jen částečný přístup. Tyto standardní moduly jsou určeny pro přenos dat mezi registry, periferii a paměťovým prostorem. V současných aplikacích je potřeba zpracovávat velké množství úloh v krátkých časových okamžicích. Tento požadavek právě splňuje modul XGATE, který je integrován v mikroprocesorech řady S12XD [9].

Hlavní výhody:

- zpracování komplexních úloh prostřednictvím HW bez přerušení od CPU,
- vlastní instrukční soubor, vysokorychlostní přenosy mezi periferními jednotkami (RAM, EEPROM, FLASH),
- programování XGATE modulu pomocí jazyka C.

Výhoda tohoto modulu se uplatňuje například při realizaci komplexního ovladače (FlexCAN, CANOpen, LIN). Tyto ovladače pak fungují nezávisle na chodu CPU. Přenos dat je pak realizován pomocí sdílené paměti RAM mezi CPU a XGATE modulem.

6.1.1 MSCAN modul (Scalable Controller Area Network) mikroprocesoru MC9S12XDP512

Mikroprocesor MC9S12XDP512 má implementován pět MSCAN (hardwarová implementace protokolu CAN od firmy Freescale) modulů. Tento procesor komunikuje s budičem rozhraní CAN (transciever) pomocí vývodu na procesoru RXCAN (přijímací vstup) a TXCAN (vysílací výstup).

MSCAN modul zajišťuje spolehlivý přenos dat, který nemůže být přerušen od uzlu, který chce vysílat zprávy s nižší prioritou. Tímto způsobem je zajištěno, že zprávy s vyšší prioritou jsou přeneseny přednostně [9].

Základní vlastnosti MSCAN modulu [9]:

- 5x CAN 2.0 A/B,
- maximální nastavitelná přenosová rychlost do 1Mb/s,
- podpora standardního (délka 11b) a rozšířeného identifikátoru (délka 29b),
- velikost přenášených dat - od 0B až 8B,
- 5x přijímací FIFO buffer,
- 3x vysílací FIFO buffer,
- konfigurovatelné akceptační filtry v těchto kombinacích:
  - dva dvatřicetibitové filtry
  - čtyři šestnáctibitové filtry
  - osm osmibitových filtů
- tři nízkopříkonové módy: Sleep mode, enable MSCAN a Power Down,

6.1.1.1 Organizace paměti FIFO u MSCAN modulu

6.1.1.1.1 Vysílací buffery

Modul MSCAN u mikroprocesoru MC9S12XDP512 obsahuje tři vysílací buffery, které jsou nazvány Tx0 až Tx2 viz Obr. 6.4. Každý z těchto bufferů vysílacích, ale i přijímacích disponuje velikostí 13B (pro datové struktury). Celková velikost tohoto bufferu je 16B viz Tab. 6.1. V osmibitovém registru TBPR (Transit Buffer Priority Register) se určuje prioritní řazení při požadavku na odeslání dat. Hodnota priority je obsažena v jednotlivých polích označených PRIO, která jsou součásti dříve zmíněného

registru. Hodnota priority se určuje podle velikosti binárního čísla obsaženého v registru TBPR (nejnižší binární hodnota obsažena v tomto registru definuje nejvyšší prioritu) [9].

Pokud nastane situace, kdy všechny buffery obsahují neodeslaná data a je požadavek na jejich odeslání, tak modul MSCAN určí, která data budou odeslána. Tato funkce se děje právě na základě obsahu dříve zmíněného prioritního pole registru TBPR. Pro přenos zprávy musí mít daný buffer, který bude vysílat zprávu nastaven TXEx flag (tento flag - vlajka má označení TXE0 až TXE2 pro jednotlivé buffery), který je obsažen v CANTFLG (MSCAN Transmitter Flag Register) registru. Jestliže je buffer k dispozici, procesor musí nastavit ukazatel na tento buffer a zapsat do registru CANTBSEL (MSCAN Transmit Buffer Selection Register). Takto je pak příslušný buffer dostupný uvnitř CANTXFG adresovacího prostoru. Tento algoritmus sdružený s CANTBSEL registrem zjednodušuje výběr vysílacího bufferu. Tato struktura navíc umožňuje jednoduchý softwarový ovladač, protože jen jedna adresová oblast je použitá pro vysílací proces a tak je i adresový prostor paměti minimalizován [9]. Procesor pak ukládá identifikátor, kontrolní bity a datová pole do jednoho vysílacího bufferu. Pokud je daný buffer připraven pro přenos, tak je tato událost signalizována pomocí vymazání příslušného TXE flagu.

Tab. 6.1: Organizace vysílacího a přijímacího bufferu [9]

Adresa	Registr
0x00X0	Identifier Register 0
0x00X1	Identifier Register 1
0x00X2	Identifier Register 2
0x00X3	Identifier Register 3
0x00X4	Data Segment Register 0
0x00X5	Data Segment Register 1
0x00X6	Data Segment Register 2
0x00X7	Data Segment Register 3
0x00X8	Data Segment Register 4
0x00X9	Data Segment Register 5
0x00XA	Data Segment Register 6
0x00XB	Data Segment Register 7
0x00XC	Data Lenght Register
0x00XD	Transmit Buffer Priority Register
0x00XE	Time Stap Register (High byte)
0x00XF	Time Stap Register (Low byte)

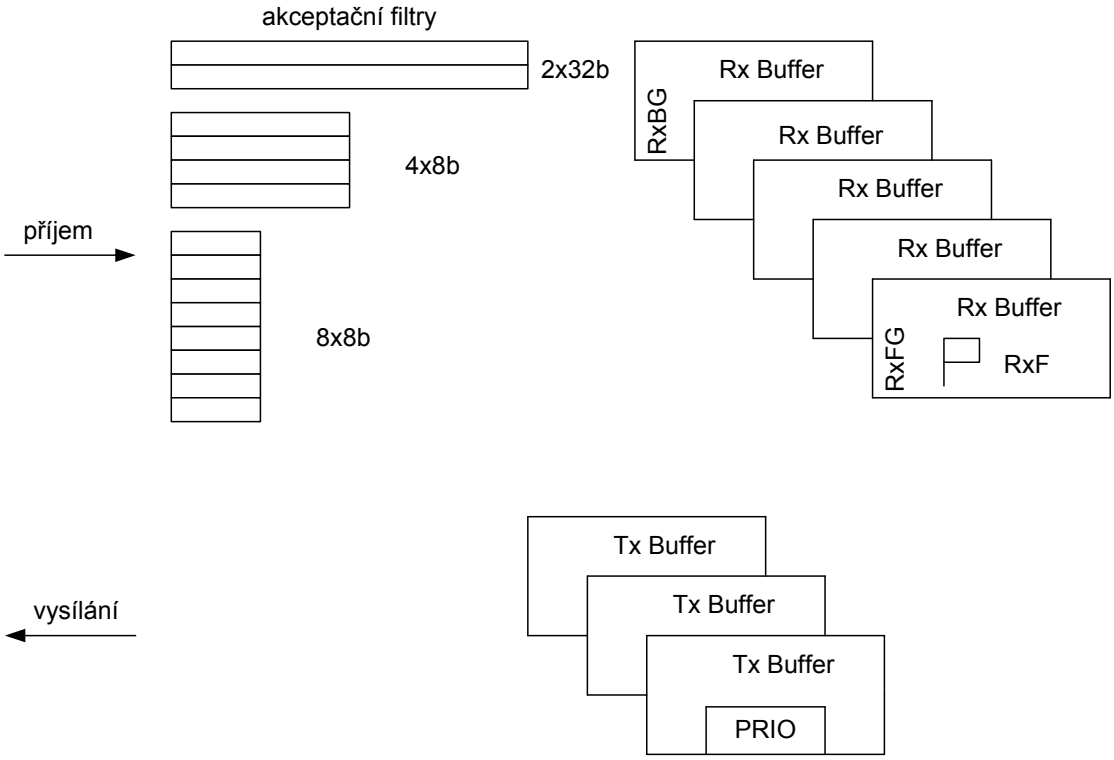
MSCAN pak signalizuje úspěšný přenos zprávy nastavením příslušného TXE flagu. Přerušení je generováno, když příslušný TXE flag je nastaven. Pokud nastane situace, kdy má být odeslána zpráva s vysokou prioritou a právě probíhá přenos zprávy s nižší prioritou, tak nemůže být přenos neúspěšný, ale musí být nastaven příslušný bit v ABTRQ (Abort Request Register) registru [9].

6.1.1.1.2 Přijímací buffery

Modul MSCAN u mikroprocesoru MC9S12XDP512 obsahuje pět přijímacích bufferů, které jsou pojmenovány RxBG až RxFG viz Obr. 6.4. Každý z těchto bufferů disponuje velikostí 13B (pro datové struktury). Zadní přijímací buffer (RxBG) je výlučně použit pro modul MSCAN, přední RxFG buffer je adresovatelný od CPU. Tato struktura navíc umožňuje jednoduchý softwarový ovladač, protože jedna adresová oblast je použitá pro přijímací proces. Každý z přijímacích bufferů má celkovou velikost 16B (kontrolní bity, identifikátor – standardní, nebo rozšířený), data a časovou známku pokud je povolena. Nastavená RXF flag signalizuje stav předního přijímacího bufferu. Pokud buffer obsahuje správně přijatou zprávu s příslušným identifikátorem,

tak dojde k nastavení dříve zmíněného RXF flagu. Při přijímaní zprávy je každá zpráva kontrolována jestli souhlasí její hodnota s akceptačním filtrem a současně je zapsána do aktivního RxBG. Po úspěšném přijetí platné zprávy, MSCAN posune obsah z RxBG do přijímací FIFO (pokud RXF není nastaven), nastaví RXF flag a generuje přerušení příjmu do CPU. Ovladač tak musí číst přijatou zprávu z RxFG a pak vymazat RXF flag pro povolení přerušení k uvolnění předního bufferu. Další nová zpráva je přijata do dalšího dostupného RxBG. Pokud MSCAN modul přijme špatnou zprávu (chybný identifikátor, chyba přenosu, apod.), tak aktuální buffer bude přepsán další zprávou [9].

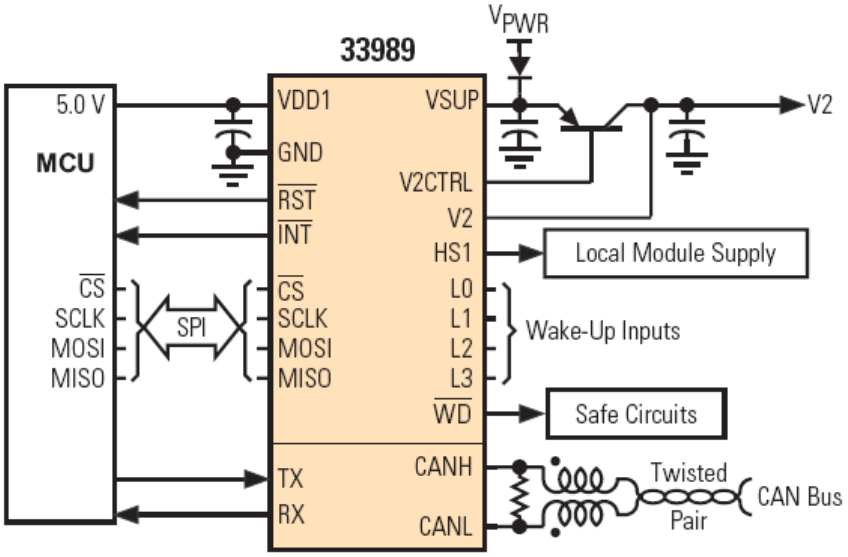
Pokud je MSCAN modul ve vysílacím režimu, tak přijímá jeho vlastní zprávu do posledního přijímacího bufferu RxBG, ale tato zpráva se nepřesune do FIFO a generuje přerušení příjmu. Pokud nastane situace, kdy zprávy zaplní přijímací FIFO buffer a je přijata další platná zpráva, tak další platná zpráva v pořadí je odmítnuta. Vygeneruje se chybové přerušení s overrun signalizací (pokud je povoleno) [9].



Obr. 6.4: Organizace vstupního a výstupního bufferu MSCAN modulu

6.2 Budič sběrnice CAN 2.0 Freescale MC33989

Při návrhu vývojové desky jsem použil budič (transciever) sběrnice CAN 2.0 MC33989. Tento transciever je integrován do pouzdra 28SOICW s 28 vývody. K mikroprocesoru MC9S12XDP512 je připojen pomocí pinů Rx a Tx na modul MSCAN3 (pin 97 a 98). Tento transciever v sobě kromě fyzického rozhraní sběrnice dále obsahuje integrovaný stabilizátor napájení (SBC), který může být použit pro napájení mikroprocesoru. Dále obsahuje SPI (Serial Peripheral Interface) rozhraní, kterým je připojen k mikroprocesoru. Pomocí tohoto rozhraní se například nastavují jednotlivé režimy zařízení (nízkopříkonový apod.). Při návrhu vývojové desky jsem integrovaný stabilizátor napájení nevyužil a to důvodu, že výstup tohoto stabilizátoru může být zatížen proudem maximálně 200mA [18]. Vzhledem ke skutečnosti, že jsou na vývojové desce připojeny LED diody a vyvedeny další piny mikroprocesoru pomocí nichž mohou být připojeny další obvody je předpoklad, že výstupní proud vnitřního stabilizátoru (max. 200mA) by nedostačoval. Z tohoto důvodu jsem volil oddělené napájení pomocí stabilizátoru (LM2940CT). Více o použitých součástkách a návrzích je uvedeno v příloze.



Obr. 6.5: Řadič protokolu CAN 2.0 MC33989

Základní vlastnosti budiče sběrnice MC33989:

- 5V regulátor, maximální odběr 200mA,
- rozhraní sběrnice CAN 2.0, přenosová rychlost do 1Mb/s,

- SPI sériové periferní rozhraní,
- nízká spotřeba v pohotovostním a klidovém režimu,
- čtyři operační módy – tyto módy jsou ovládány pomocí SPI rozhraní [18]:
    - pohotovostní režim,
    - provozní režim,
    - stop režim,
    - klidový režim.

6.3 Softwarová implementace architektury MailBox u mikroprocesoru MCS12XDP512

V této části diplomové práce bude popsán zdrojový kód pro implementaci architektury MailBox u mikroprocesoru MCS12XDP512. Budou zde popsány hlavní části tohoto programu pro vytvoření výše zmíněné architektury.

Klíčové vlastnosti MSCAN CAN driveru:

- řídí přenos zpráv na sběrnici,
- řídí ukládání zpráv do paměti.

Soubor xgCAN_init.h [21]

V tomto souboru jsou definovány všechny důležité hodnoty jako:

- definice MSCAN modulu,
- časování, definice velkosti jednotlivých zásobníků (MailBoxů),
- ID jednotlivých MailBoxů

Všechny výše uvedené vlastnosti je možné poměrně rychle přednastavit na požadované hodnoty, které potřebujeme pro vlastní aplikaci. Jak už bylo zmíněno výše, tak mikroprocesor MC9S12XDP512 obsahuje pět MSCAN modulů. Na návrhu vývojové desky (viz příloha) jsou využity Rx a Tx piny MSCAN modulu číslo 3. Postup nastavení bude aplikován na modulu MSCAN3. Obdobným způsobem konfigurace se postupuje i v případě použití jiných MSCAN modulů daného mikroprocesoru.

```
/*****************************************************************/
/*          DEFINICE PARAMETRŮ PRO MSCAN MODUL                   */
/*                        CAN3                                   */
*****************************************************************/
#define CAN_RECEIVE_NOTIFICATION    //povolení přerušení příjmu, CPU je informován o nově přijatých datech

#define USE_CAN3                    //aktivování rozhraní CAN3 (musí být definováno)

                    /* Hardwarová inicializace */
#define CANCTL0_CAN3  0x00          //MSCAN timer zakázán, wake-up mód zakázán

#define CANCTL1_CAN3  CANE          //povolení MSCAN modulu, hodinový signál z oscilátoru clock = OSC_CLK, listen only zakázán, wake-up filtr zakázán

                /* Definice parametrů časování MSCAN modulu */
#define CANBTR0_CAN3  0x40          //SJW = 1,  prescaler = 1
#define CANBTR1_CAN3  0x14          //SAMP = 0 (vzorkování – 1 vzorek na bit), TSEG2 = 2, TSEG1 = 5, 8 Tq per bit; nastavení přenosové rychlosti viz 6.3.1

#define CANRIER_CAN3  RXFIE         //přijímací buffer plný (úspěšný příjem zpráv) událost způsobuje žádost o přerušení příjmu

#define CANTIER_CAN3  0             //vysílací buffer prázdný, negeneruje přerušení
```

```
/* Akceptační filtry */
#define CANIDAC_CAN3  IDAM0              //nastavení akceptačního filtru - 4 x 16-bitů
#define CANIDAR0_CAN3 0                  //nastavení hodnot v Acceptance Code Register (horních 8 bitů)
#define CANIDAR1_CAN3 0                  //nastavení hodnot v Acceptance Code Register (dolních 8 bitů)
#define CANIDMR0_CAN3 0xFFFFFFFF         //nastavení hodnot v Acceptance Mask Register – všechny bity
#define CANIDMR1_CAN3 0xFFFFFFFF         //nastavení hodnot v Acceptance Mask Register – všechny bity
/*Funkce akceptačních filtrů je popsána v Kap. 5.1*/

/* Konfigurace MailBoxů */
#define RXBOXSIZE_CAN3  16               //nastavení počtu přijímacích Mailboxů; maximum 16
#define TXBOXSIZE_CAN3  4                //nastavení počtu vysílacích Mailboxů; maximum 16

/* definice jednotlivých MailBoxů a jejich identifikátor ID
(RXBOXSIZE_CAN3 - 1) - počet MailBoxů mínus jeden z důvodu, že MailBox 0 je speciální (jsou zachytávány všechny zprávy, které jsou akceptovány akceptačním filtrem) a které nemají shodný identifikátor, který je nastaven v MailBoxu 1 až RXBOXSIZE_CANx - 1 */

#define CAN3ID1          0x700
#define CAN3ID2          0x555
#define CAN3ID3          0x200
#define CAN3ID4          0x110
#define CAN3ID5          0x120
#define CAN3ID6          0x130
#define CAN3ID7          0x140
```

```
#define CAN3ID8         0x150
#define CAN3ID9         0x160
#define CAN3ID10        0x170
#define CAN3ID11        0x180
#define CAN3ID12        0x190
#define CAN3ID13        0x1a0
#define CAN3ID14        0x1b0
#define CAN3ID15        0x1c0
        /* definice jednotlivých MailBoxů pro vysílání a jejich identifikátory*/
#define CAN3ID16        0x2d0
#define CAN3ID17        0x1e0
#define CAN3ID18        0x555
#define CAN3ID19        0x550
```

6.3.1 Výpočet nastavené přenosové rychlosti

Přenosová rychlost 1Mbps:

$$f_{TQ} = \frac{f_{CANclk}}{(hodnota\_prescaleru)} = \underline{\underline{8MHz}}$$

$$přenosová\_rychlost = \frac{f_{TQ}}{number\_Of\_Time\_Quanta} = \underline{\underline{1Mbps}}$$

=> nastavená hodnota registrů CANBTR0 = 40_H, CANBTR1 = 14_H

Přenosová rychlost 50kbps:

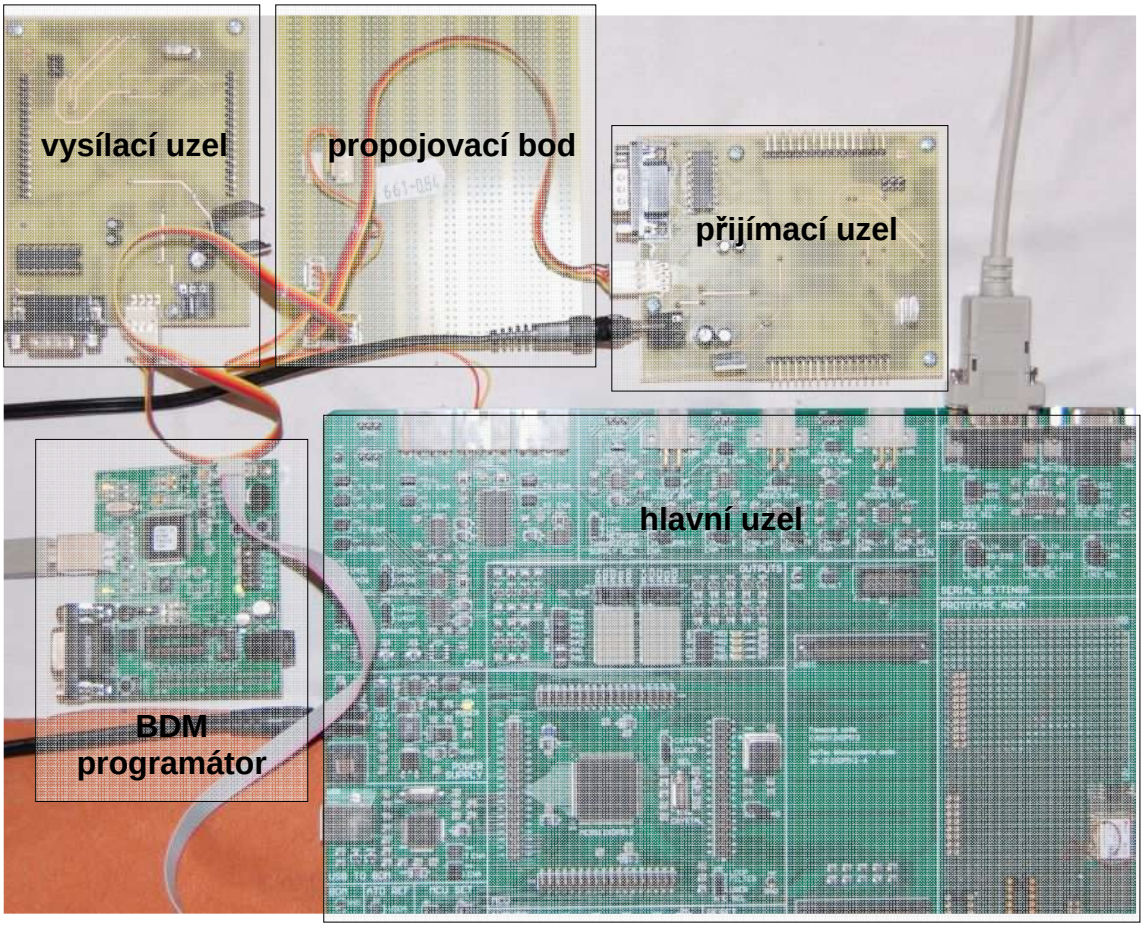
=> nastavená hodnota registrů CANBTR0 = 0x40 | 19, CANBTR1 = 14_H

7 Výsledky měření pro architektury FIFO a MailBox

Pro stanovení výsledků byly použity celkem tři uzly na sběrnici, které byly označeny jako vysílací, přijímací a hlavní. Na Obr. 7.1 je uvedena fotografie zapojení výše uvedených vývojových desek do sběrnicové architektury. Na tomto obrázku je také vyobrazen BDM programátor, který sloužil pro nahrávání softwaru. Pro koncepci sítě jsem využil dvě vývojové desky, které jsem navrhl (vysílací a přijímací uzel) a jednu originální vývojovou desku od firmy Freescale.

Dalším úkolem této diplomové práce bylo porovnat architekturu FIFO a MailBox v typických aplikacích. Byly nakonfigurovány různé režimy komunikace na sběrnici (které odpovídají typickým aplikacím):

- nízká přenosová rychlost, velké množství zpráv na sběrnici,
- vysoká přenosová rychlost, malý počet zpráv na sběrnici,
- vysoká přenosová rychlost, velký počet zpráv na sběrnici.



Obr. 7.1: Fotografie propojení jednotlivých uzlů pomocí protokolu CAN 2.0

Podrobné parametry jednotlivých konfigurací na sběrnici, které byly nastaveny pro jednotlivá měření jsou uvedeny v části výsledků měření viz 7.1, 7.2, 7.3 a 7.4. Mezi aplikace, které vyžadují zpracování malého počtu zpráv současně s vysokou přenosovou rychlostí v automobilech patří například jednotky řízení vstřikování, jednotky řízení brzd a podobně. Aplikace, které zpracovávají velké množství zpráv společně s nízkou přenosovou rychlostí patří v automobilu např. gateway. Poslední variantou, kdy je potřeba velké množství zpráv na sběrnici společně s vysokou přenosovou rychlostí jsou vysokorychlostní brány.

Komunikace byla nastavena tak, že vysílací uzel posílal zprávy na přijímací uzel viz Obr. 7.2. Na tomto přijímacím uzlu byla nakonfigurována architektura FIFO, nebo architektura MailBox.

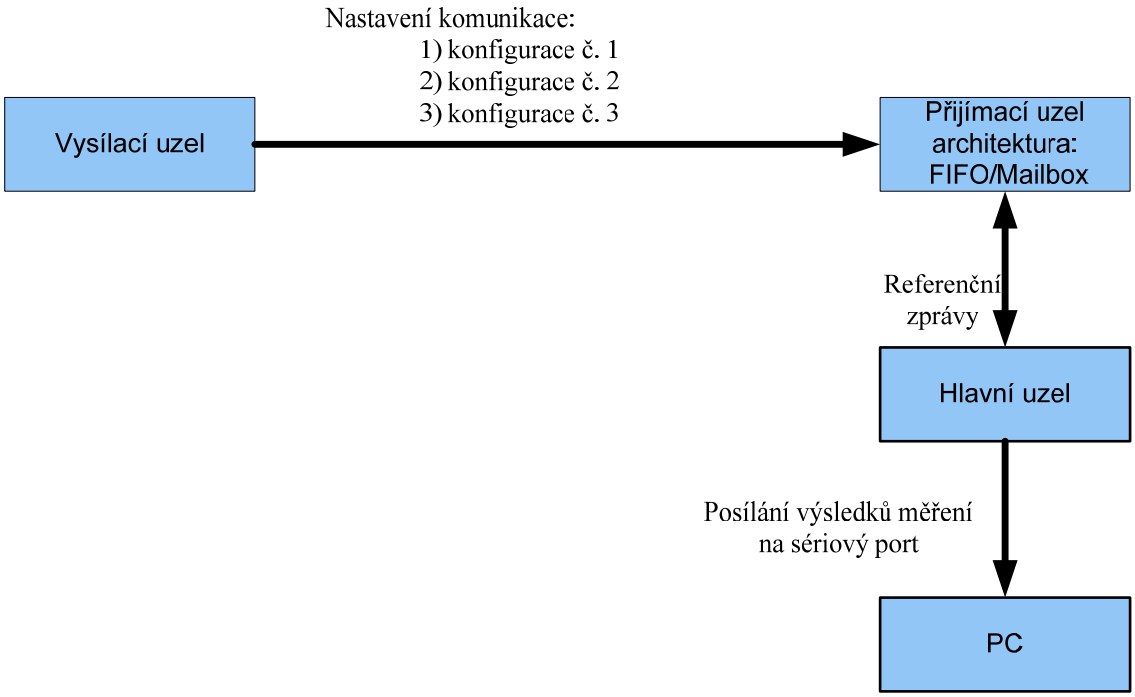
Nastavení komunikace mezi vysílacím a přijímacím uzlem (níže uvedené konfigurace byly nastaveny, jak pro architekturu FIFO, tak pro architekturu MailBox):

1) konfigurace č. 1:
   - přenosová rychlost sběrnice 50kbps,
   - 12 zpráv na sběrnici, každá zpráva jiné ID nastavené v identifikátorovém poli,
   - časová prodleva mezi každou zprávou 50ms,
   - časové zpoždění na každou zprávu bylo nakonfigurováno na 1ms a 100ms.

2) konfigurace č. 2:
   - přenosová rychlost sběrnice 1Mbps,
   - 2 zprávy na sběrnici, každá zpráva jiné ID nastavené v identifikátorovém poli,
   - časová prodleva mezi každou zprávou 50ms,
   - časové zpoždění na každou zprávu bylo nakonfigurováno na 1ms a 100ms.

3) konfigurace č.3:
   - přenosová rychlost sběrnice 1Mbps,
   - 12 zpráv na sběrnici, každá zpráva jiné ID nastavené v identifikátorovém poli,

- časová prodleva mezi každou zprávou 50ms,
- časové zpoždění na každou zprávu bylo nakonfigurováno na 1ms a 100ms.



Obr. 7.2: Blokové schéma nastavení komunikace mezi jednotlivými uzly

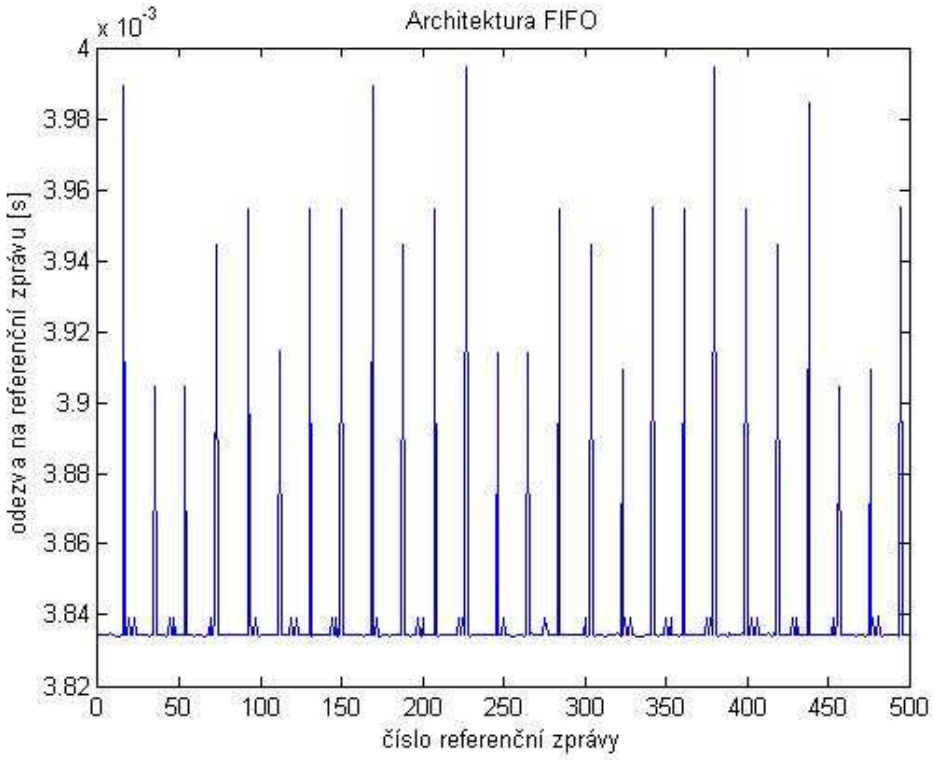
Měření pro jednotlivé výše uvedené konfigurace probíhalo tak, že hlavní uzel posílal na přijímací uzel dotaz, tzv. referenční zprávu, která měla vysokou prioritu zpracování. Tento přijímací uzel odpověděl zprávou zpět na hlavní uzel. A právě odezva mezi odesláním a příjmem zprávy byla měřena. V tomto zpoždění je zahrnut také čas nutný k sestavení zprávy, její přenos apod. Měření probíhalo časovačem, který pracoval na frekvenci sběrnice 20MHz a pomocí něj se počítalo přetečení. Hodnoty, které byly takto naměřeny, se posílaly na terminál v textové podobě přes sériový port počítače. Hodnoty z terminálu byly exportovány do programu Matlab, kde docházelo k přepočítání skutečných časů vzhledem k nastavené frekvenci na sběrnici. Měření bylo realizováno pro vyslání 500 referenčních zpráv z hlavního uzlu na přijímací uzel.

Takto byly postupně nastaveny tři různé konfigurace provozu na sběrnici (uvedeny výše), které byly implementovány mezi vysílací a přijímací uzel. Tyto měření probíhaly jak pro architekturu FIFO, tak pro architekturu MailBox, aby mohlo dojít ke

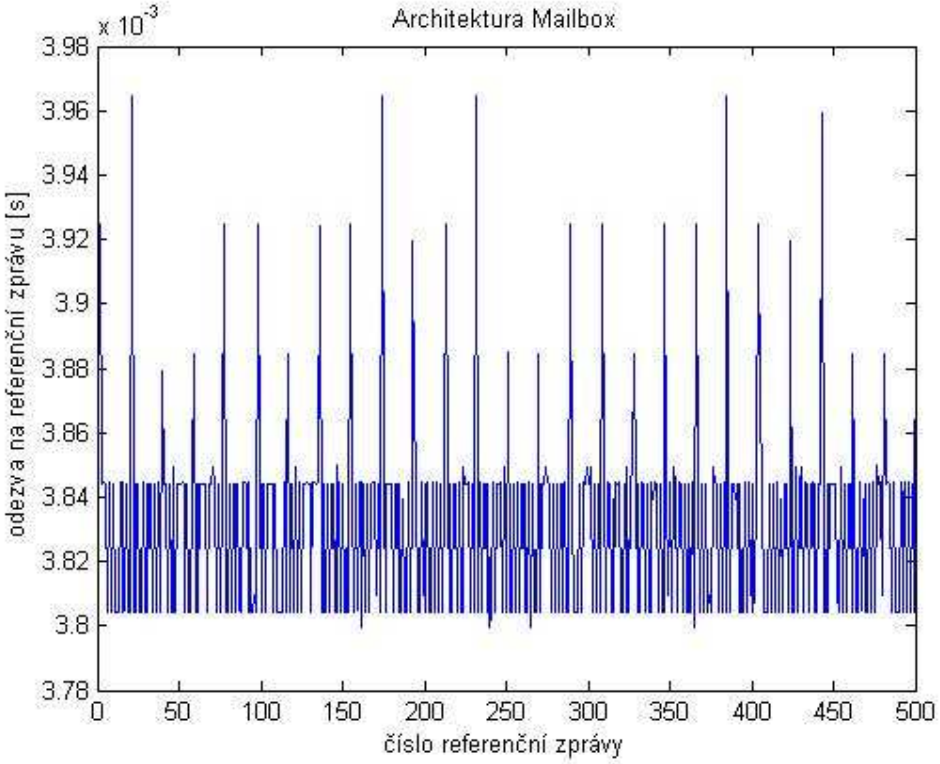
srovnání, které je uvedeno v kap. 7.4. Přehledné srovnání jednotlivých měření je dále uvedeno v Tab. 7.1.

7.1 Výsledky měření pro nízkou přenosovou rychlost a velký počet zpráv na sběrnici

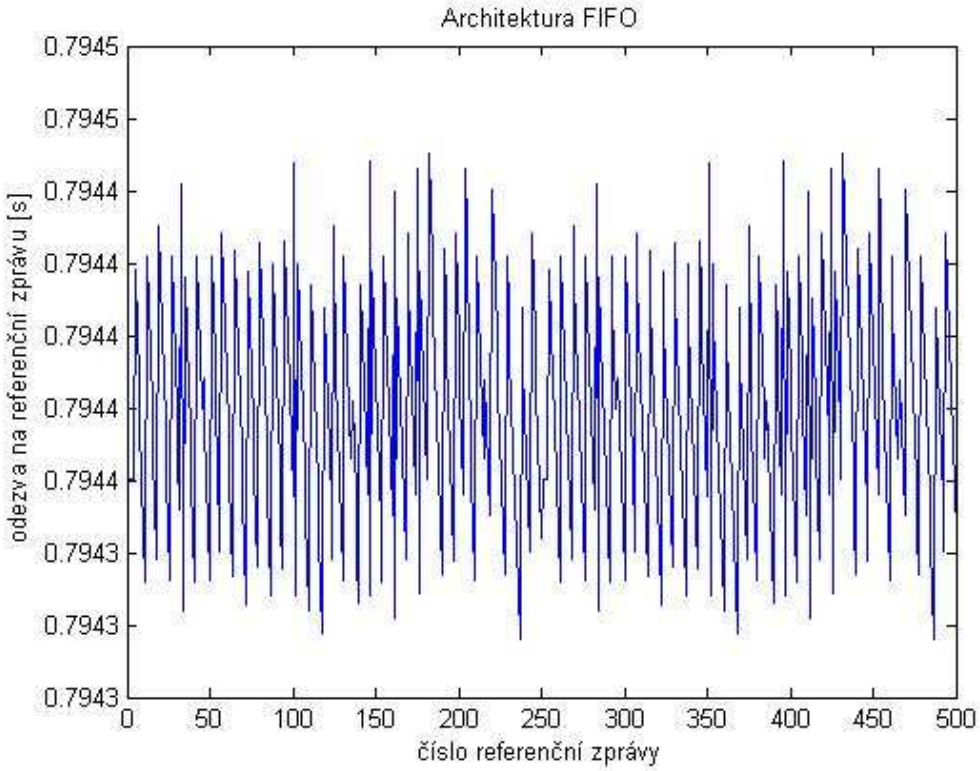
V tomto měření byla nastavena přenosová rychlost na 50kbps. Na sběrnici bylo nakonfigurováno dvanáct zpráv s různou hodnotou identifikátoru. Tyto zprávy se vysílají s časovým rozestupem 50ms. Reakce na každou z těchto zpráv byla v prvním měření nastavena na 1ms a ve druhém měření na 100ms. Měření probíhalo pro architekturu FIFO a MailBox. V grafech, které jsou uvedeny níže je uvedena doba odezvy na každou z 500 vyslaných referenčních zpráv do přijímacího uzlu.



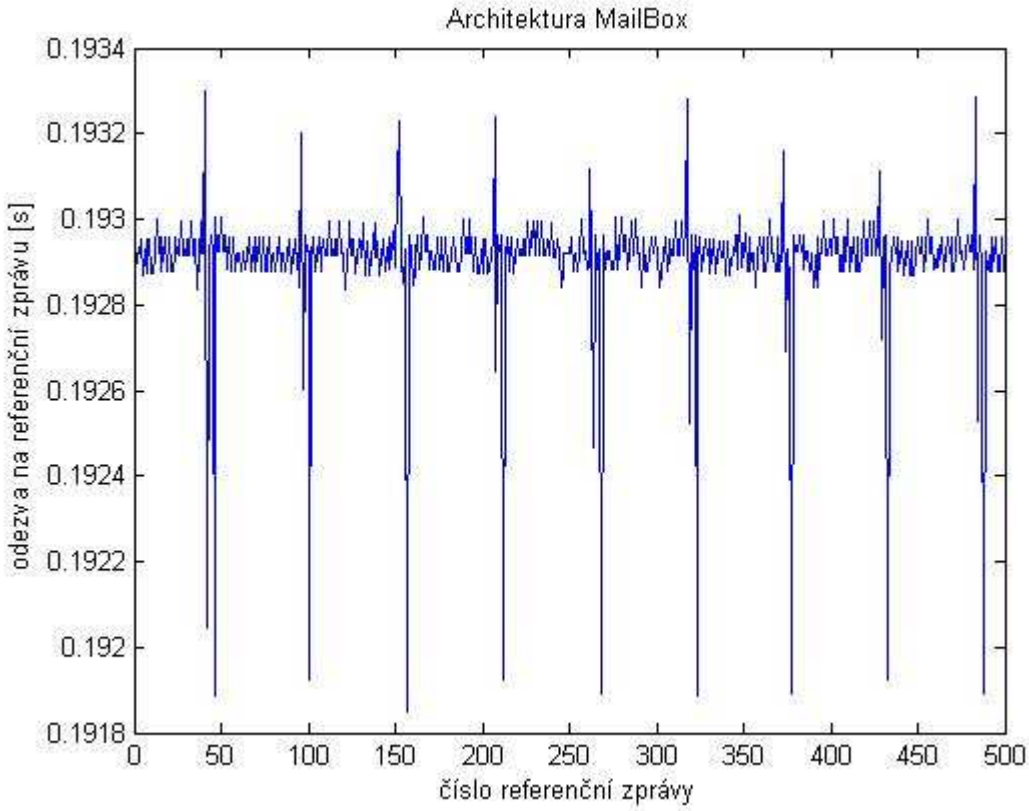
Obr. 7.3: Odezva na 500 referenčních zpráv u architektury FIFO, přenosová rychlost 50kbps, 12 zpráv na sběrnici, zpoždění na každou zprávu 1ms



Obr. 7.4: Odezva na 500 referenčních zpráv u architektury MailBox, přenosová rychlost 50kbps, 12 zpráv na sběrnici, zpoždění na každou zprávu 1ms



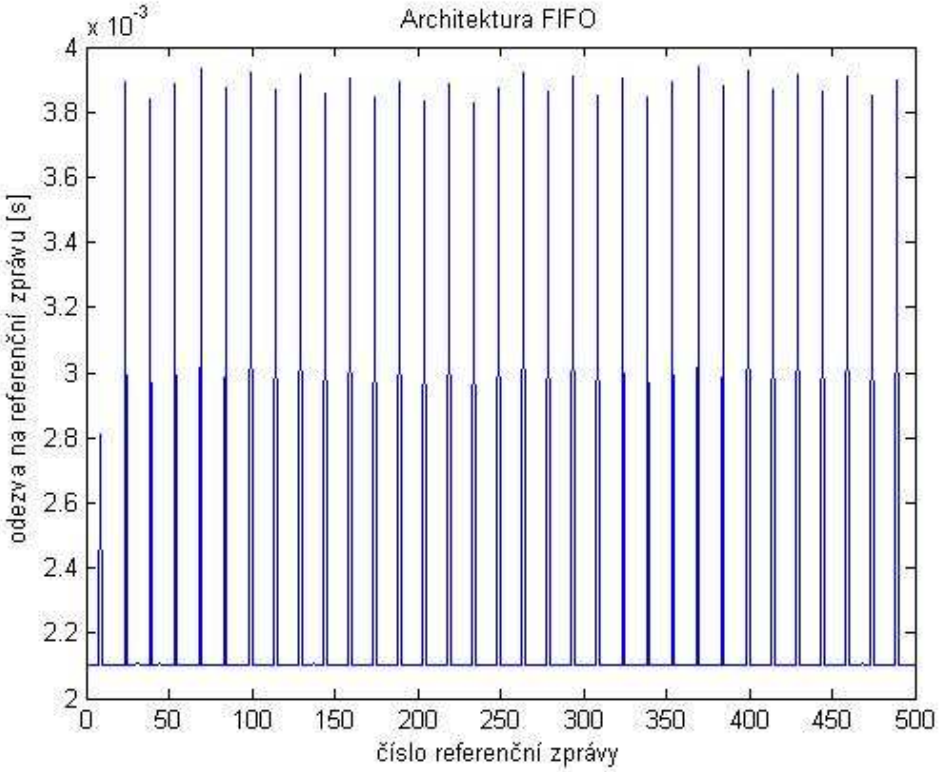
Obr. 7.5: Odezva na 500 referenčních zpráv u architektury FIFO, přenosová rychlost 50kbps, 12 zpráv na sběrnici, zpoždění na každou zprávu 100ms



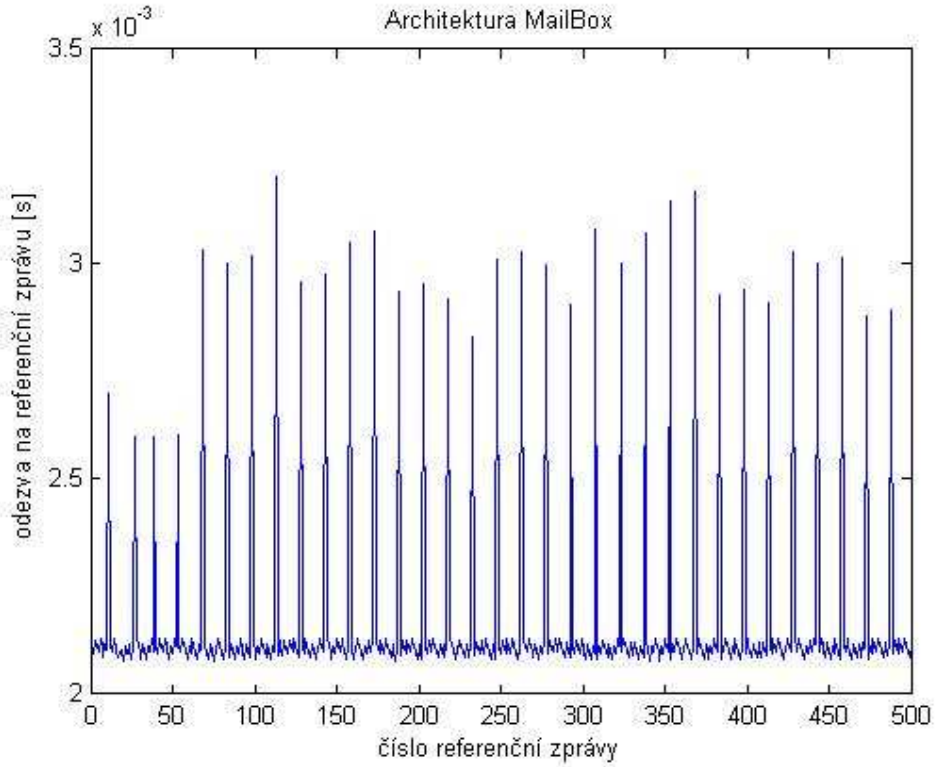
Obr. 7.6: Odezva na 500 referenčních zpráv u architektury MailBox, přenosová rychlost 50kbps, 12 zpráv na sběrnici, zpoždění na každou zprávu 100ms

7.2 Výsledky měření pro vysokou přenosovou rychlost a malý počet zpráv na sběrnici

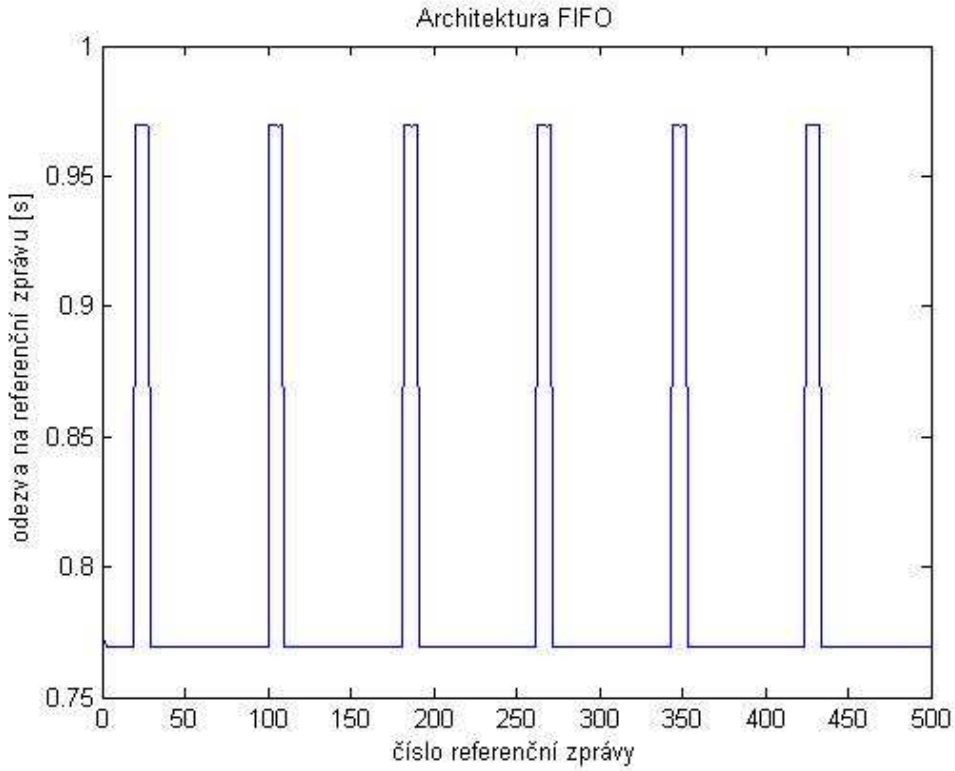
V tomto měření byla nastavena přenosová rychlost na 1Mbps. Na sběrnici byly nakonfigurovány dvě zprávy s různou hodnotou identifikátoru. Tyto zprávy se vysílají s časovým rozestupem 50ms. Reakce na každou z těchto zpráv byla v prvním měření nastavena na 1ms a ve druhém měření na 100ms. Měření probíhalo pro architekturu FIFO a MailBox. V grafech, které jsou uvedeny níže je uvedena doba odezvy na každou z 500 vyslaných referenčních zpráv do přijímacího uzlu.



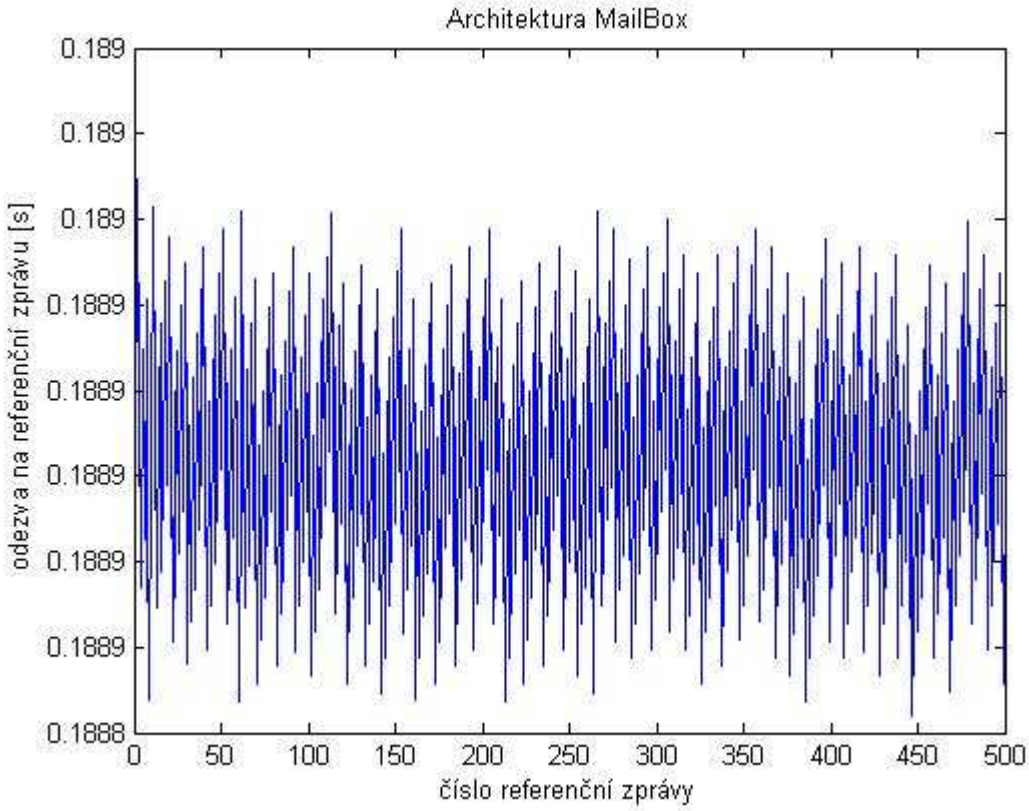
Obr. 7.7: Odezva na 500 referenčních zpráv u architektury FIFO, přenosová rychlost 1Mbps, 2 zprávy na sběrnici, zpoždění na každou zprávu 1ms



Obr. 7.8: Odezva na 500 referenčních zpráv u architektury MailBox, přenosová rychlost 1Mbps, 2 zprávy na sběrnici, zpoždění na každou zprávu 1ms



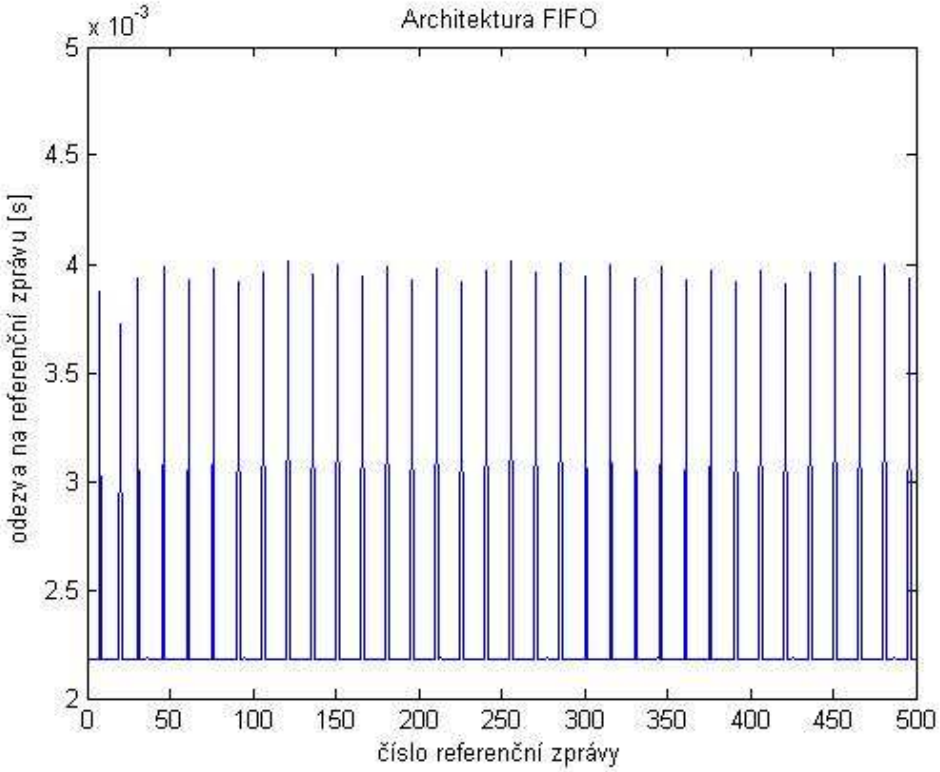
Obr. 7.9: Odezva na 500 referenčních zpráv u architektury FIFO, přenosová rychlost 1Mbps, 2 zprávy na sběrnici, zpoždění na každou zprávu 100ms



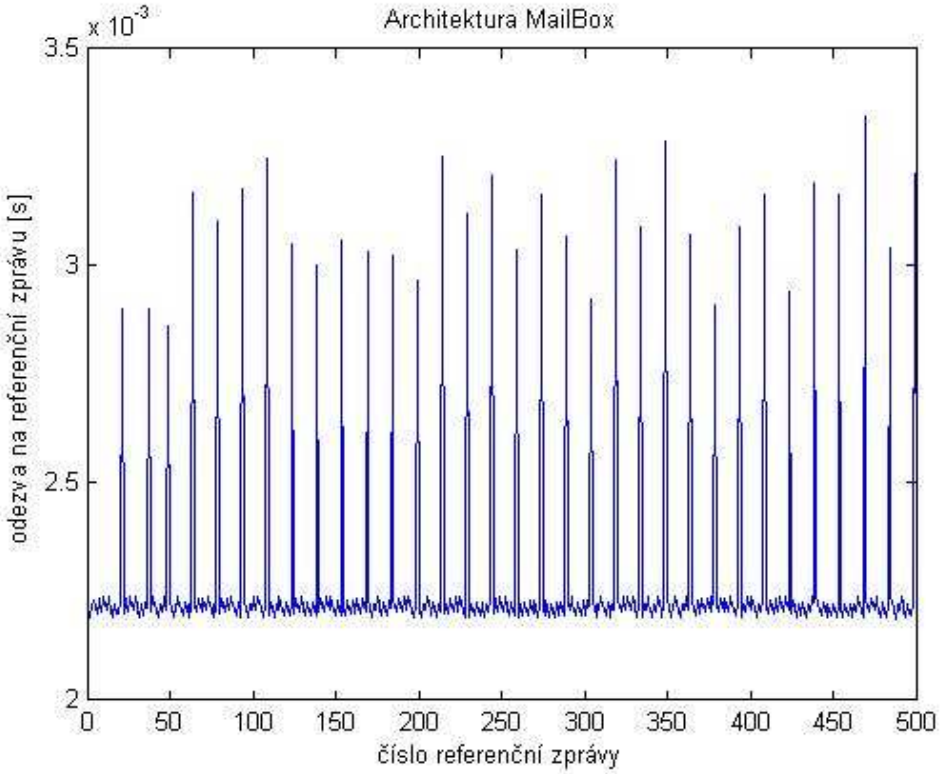
Obr. 7.10: Odezva na 500 referenčních zpráv u architektury MailBox, přenosová rychlost 1Mbps, 2 zprávy na sběrnici, zpoždění na každou zprávu 100ms

7.3 Výsledky měření pro vysokou přenosovou rychlost a velký počet zpráv na sběrnici

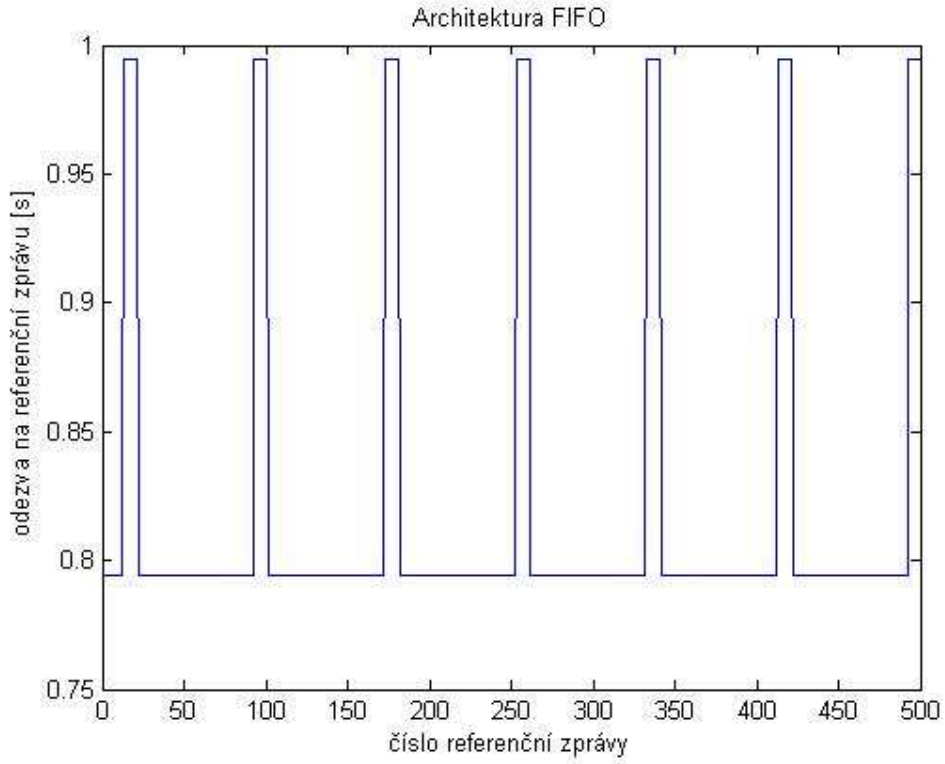
V tomto měření byla nastavena přenosová rychlost na 1Mbps. Na sběrnici bylo nakonfigurováno dvanáct zpráv s různou hodnotou identifikátoru. Tyto zprávy se vysílají s časovým rozestupem 50ms. Reakce na každou z těchto zpráv byla v prvním měření nastavena na 1ms a ve druhém měření na 100ms. Měření probíhalo pro architekturu FIFO a MailBox. V grafech, které jsou uvedeny níže je uvedena doba odezvy na každou z 500 vyslaných referenčních zpráv do přijímacího uzlu.



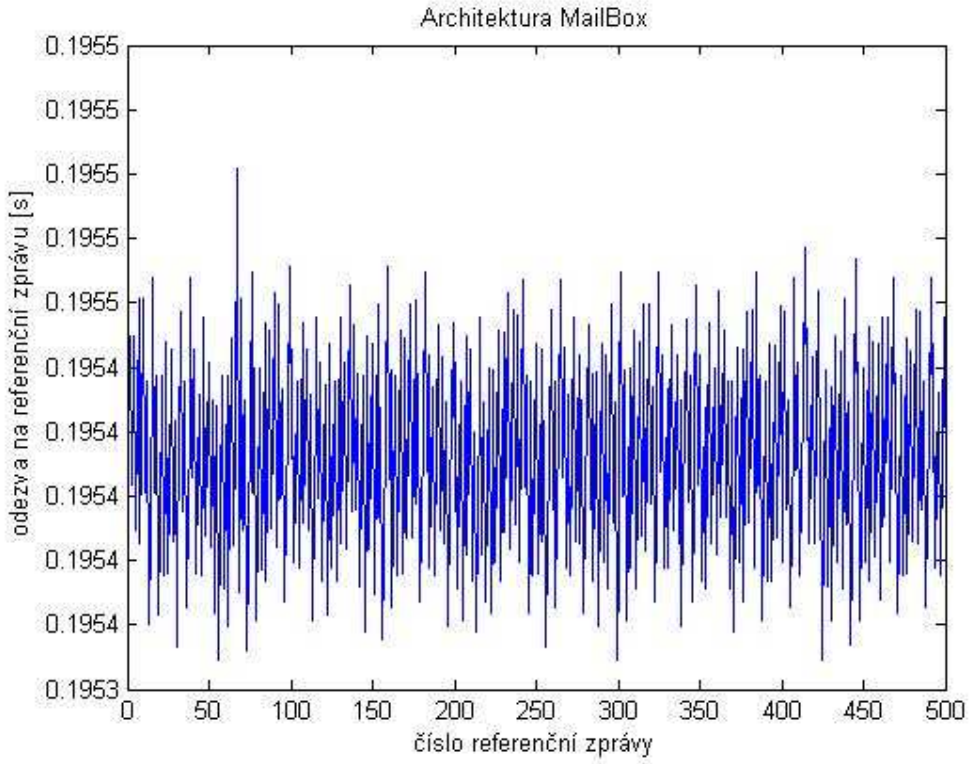
Obr. 7.11: Odezva na 500 referenčních zpráv u architektury FIFO, přenosová rychlost 1Mbps, 12 zpráv na sběrnici, zpoždění na každou zprávu 1ms



Obr. 7.12: Odezva na 500 referenčních zpráv u architektury MailBox, přenosová rychlost 1Mbps, 12 zpráv na sběrnici, zpoždění na každou zprávu 1ms



Obr. 7.13: Odezva na 500 referenčních zpráv u architektury FIFO, přenosová rychlost 1Mbps, 12 zpráv na sběrnici, zpoždění na každou zprávu 100ms



Obr. 7.14: Odezva na 500 referenčních zpráv u architektury MailBox, přenosová rychlost 1Mbps, 12 zpráv na sběrnici, zpoždění na každou zprávu 100ms

7.4 Shrnutí výsledků měření pro jednotlivé architektury

Výsledky a srovnání jednotlivých případů na sběrnici je uvedeno v Tab. 7.1. V této tabulce je uvedena průměrná hodnota odezvy referenční zprávy pro různé konfigurace na sběrnici. Hodnoty pro architekturu MailBox byly upraveny tak, že byla odečtena hodnota 30µs, která odpovídá době zpracování MailBoxu (zpoždění od přerušení do doby, než se vykoná funkce pro daný MailBox). Tento údaj byl naměřen osciloskopem.

Tab. 7.1: Shrnutí naměřených hodnot pro architekturu FIFO a MailBox pro různé konfigurace na sběrnici

přenosová rychlost 50kbps, 12 zpráv na sběrnici - vysílání co 50ms, zpoždění na referenční zprávu 1ms

nastavené zpoždění na každou zprávu	**FIFO [ms]**	**MailBox [ms]**	rozdíl
1ms	3,840	3,819	9µs
100ms	794,379	192,873	602ms

přenosová rychlost 1Mbps, 2 zprávy na sběrnici - vysílání co 50ms, zpoždění na referenční zprávu 1ms

nastavené zpoždění na každou zprávu	**FIFO [ms]**	**MailBox [ms]**	rozdíl
1ms	2,187	2,161	26µs
100ms	790,976	182,903	608ms

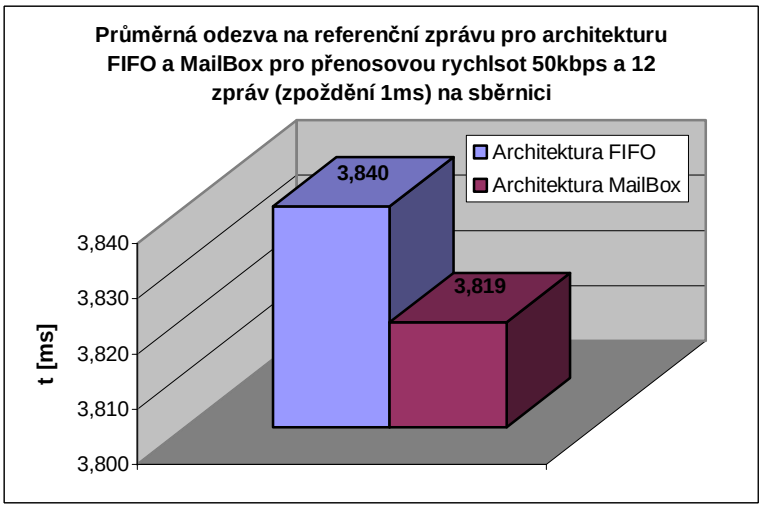
přenosová rychlost 1Mbps, 12 zprávy na sběrnici-vysílání co 50ms, zpoždění na referenční zprávu 1ms

nastavené zpoždění na každou zprávu	**FIFO [ms]**	**MailBox [ms]**	rozdíl
1ms	2,303	2,269	31µs
100ms	819,174	195,412	623ms

7.4.1 Shrnutí výsledků pro nízkou přenosovou rychlost a velký počet zpráv na sběrnici

Při tomto měření byla na sběrnici nastavena přenosová rychlost 50kbps. Vysílací uzel odesílal na přijímací uzel zprávy (12 zpráv) s časovou prodlevou 50ms. Reakce na

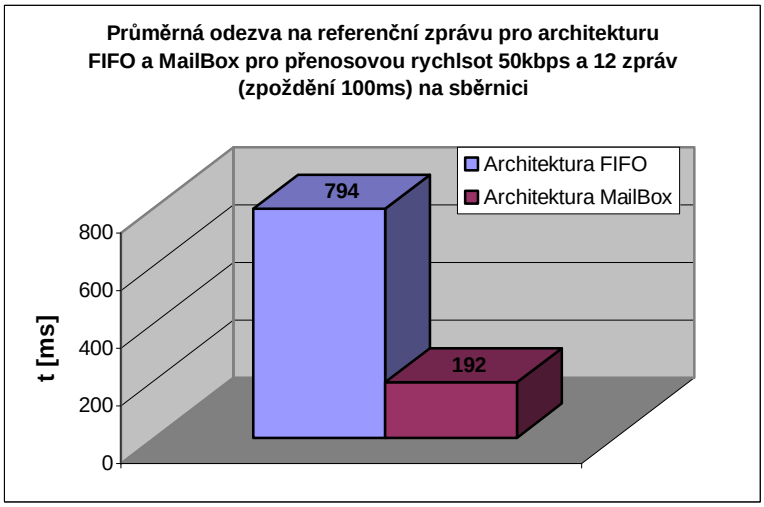
každou z těchto dvanácti vyslaných zpráv byla nastavena u prvního měření 1ms a u druhého měření 100ms. Při nastaveném zpoždění 1ms byla reakce obou architektur (FIFO a MailBox) velice podobná. Při tomto měření byla rychlejší odezva na referenční zprávu u architektury MailBox (~9µs). Tento malý rozdíl byl zřejmě způsoben nastavením časové prodlevy 50ms mezi jednotlivými zprávami. Toto nastavení mělo za následek to, že nedochází k zaplnění FIFO bufferu a to z důvodu, že prodleva mezi zprávami byla dostatečně dlouhá k tomu, aby nastavená reakce 1ms na každou zprávu jakýmkoliv způsobem více zatížila procesor. Toto zpoždění (1ms) se vykonávalo v době, kdy byla nastavená mezera mezi jednotlivými zprávami. Proto bylo zvoleno zpoždění 100ms, kde se výhody a nevýhody těchto rozdílných architektur dostatečně projevily.



Obr. 7.15: Graf srovnání architektury FIFO a MailBox při zpoždění 1ms na zprávu v přijímacím uzlu

Při nastaveném zpoždění 100ms na každou zprávu byla reakce obou architektur (FIFO a MailBox) velmi rozdílná. U architektury FIFO byla naměřena doba odezvy na referenční zprávu téměř 795ms. U architektury MailBox byla naměřena doba odezvy na referenční zprávu 193ms. Z tohoto měření vyplynulo, že architektura MailBox při této konfiguraci má lepší dobu odezvy o 602ms na referenční zprávu viz Tab. 7.1. Tento rozdíl odpovídá lepší reakci architektury MailBox pro danou konfiguraci téměř o 400% proti architektuře FIFO. U architektury FIFO se musí počkat, než se daná zpráva dostane na pořadí – dáno organizaci bufferu (musí se nejprve zpracovat zprávy, které přišly do daného uzlu před referenční zprávou). Po postoupení referenční zprávy do

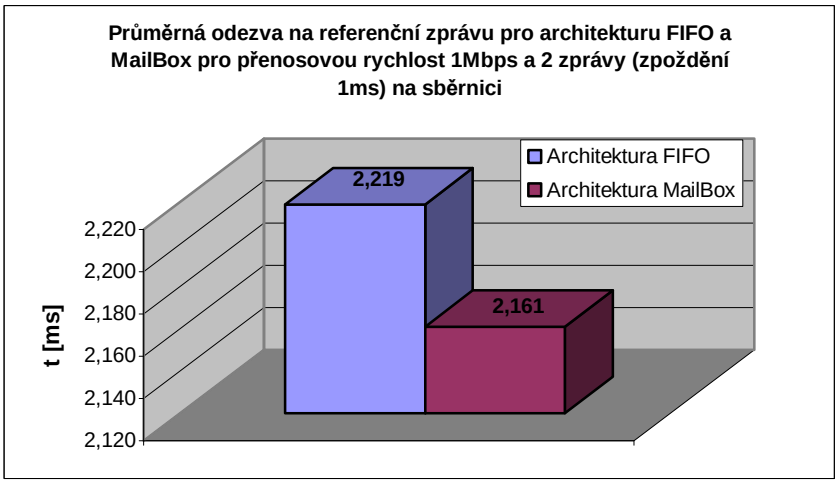
předního bufferu musí procesor tzv. dofiltrovávat – musí zjistit jakou zprávu má k dispozici na základě identifikátoru zprávy. U architektury MailBox je referenční zpráva téměř ihned k dispozici, dokončí se zpracování předešlé rozpracované zprávy. Poté je možno začít zpracovávat referenční zprávu, tudíž se nečeká jako u architektury FIFO, než referenční zpráva projde celým FIFO bufferem. Velkou výhodu této architektury (MailBox) bych právě viděl v tom, že odezva na prioritní zprávu je mnohonásobně nižší než je tomu u architektury FIFO.



Obr. 7.16: Graf srovnání architektury FIFO a MailBox při zpoždění 100ms na zprávu v přijímacím uzlu

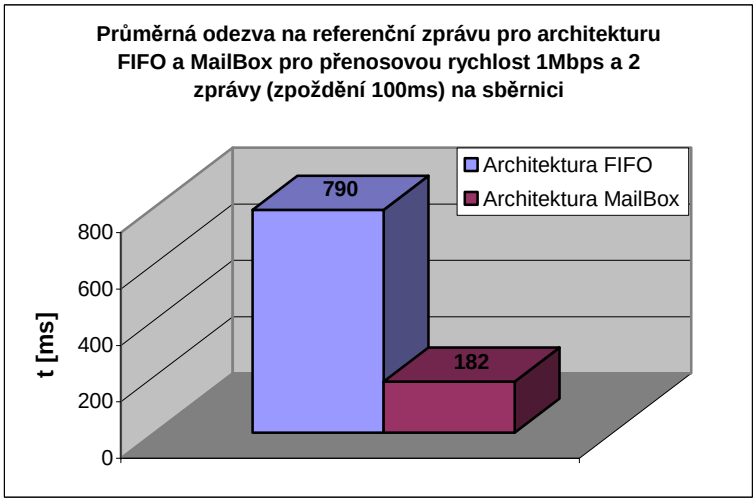
7.4.2 Shrnutí výsledků pro vysokou přenosovou rychlost a malý počet zpráv na sběrnici

Při tomto měření byla na sběrnici nastavena přenosová rychlost 1Mbps. Vysílací uzel odesílal na přijímací uzel dvě zprávy s časovou prodlevou 50ms. Reakce na každou z těchto vyslaných zpráv byla u prvního měření 1ms a u druhého měření 100ms. Při nastaveném zpoždění 1ms byla reakce obou architektur (FIFO a MailBox) obdobná. Při tomto měření měla rychlejší odezvu na referenční zprávu architektura MailBox (~26µs). Tyto rozdíly byly způsobeny vlastnostmi, které byly popsány v 7.4.1.



Obr. 7.17: Graf srovnání architektury FIFO a MailBox při zpoždění 1ms na zprávu v přijímacím uzlu

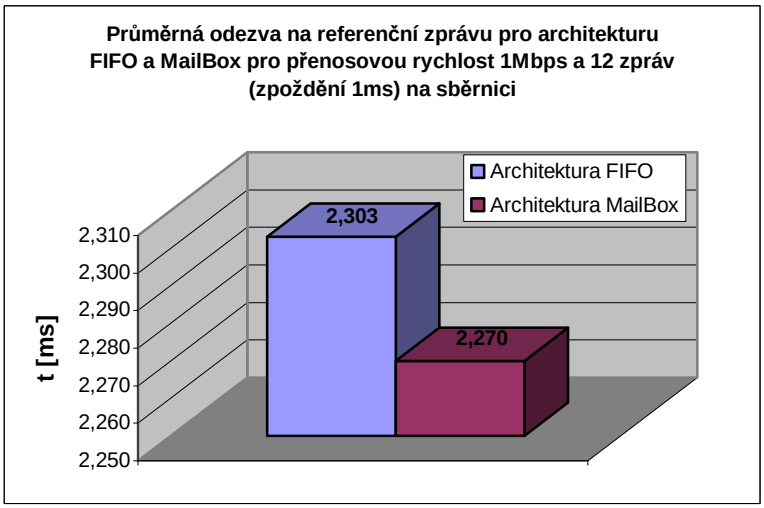
Při zpoždění 100ms na každou zprávu byla reakce obou architektur (FIFO a MailBox) velmi rozdílná. U architektury FIFO byla naměřena doba odezvy na referenční zprávu téměř 790ms. U architektury MailBox byla naměřena doba odezvy na referenční zprávu 182ms. Z tohoto měření vyplynulo, že architektura MailBox při dané konfiguraci má lepší dobu odezvy na referenční zprávu téměř o 608ms viz Tab. 7.1.



Obr. 7.18: Graf srovnání architektury FIFO a MailBox při zpoždění 100ms na zprávu v přijímacím uzlu

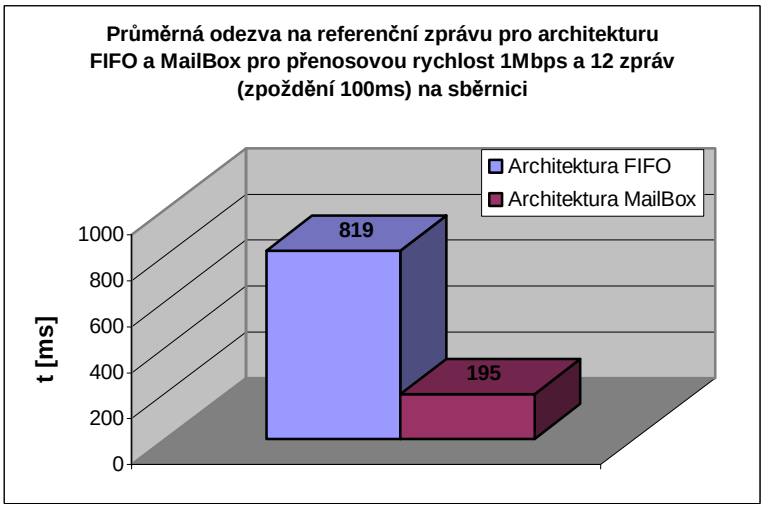
7.4.3 Shrnutí výsledků pro vysokou přenosovou rychlost a velký počet zpráv na sběrnici

Při tomto měření byla na sběrnici nastavena přenosová rychlost 1Mbps. Vysílací uzel odesílal na přijímací uzel zprávy (12 zpráv) s časovou prodlevou 50ms. Reakce na každou z těchto vyslaných zpráv byla u prvního měření 1ms a u druhého měření 100ms. Při nastaveném zpoždění 1ms byla reakce obou architektur (FIFO a MailBox) obdobná. Při tomto měření měla rychlejší odezvu na referenční zprávu architektura MailBox (~31µs). Tyto rozdíly byly způsobeny vlastnostmi, které byly popsány v 7.4.1.



Obr. 7.19: Graf srovnání architektury FIFO a MailBox při zpoždění 1ms na zprávu v přijímacím uzlu

Při nastaveném zpoždění 100ms na každou zprávu byla reakce obou architektur (FIFO a MailBox) velmi rozdílná. U architektury FIFO byla naměřena doba odezvy na referenční zprávu téměř 820ms. U architektury MailBox byla naměřena doba odezvy na referenční zprávu 196ms. Z tohoto měření vyplynulo, že architektura MailBox při této konfiguraci má lepší dobu odezvy na referenční zprávu téměř o 623ms viz Tab. 7.1.



Obr. 7.20: Graf srovnání architektury FIFO a MailBox při zpoždění 100ms na zprávu v přijímacím uzlu

8 Závěr:

V úvodu této diplomové práce jsem se zaměřil na použití sběrnic v automobilovém průmyslu. Byly uvedeny komunikační protokoly, které se v současné době používají v automobilové technice s důrazem na komunikační protokol CAN 2.0.

V další části této práce byly popsány konkrétní vlastnosti komunikačního protokolu CAN 2.0. U tohoto protokolu byla uvedena úplná struktura rámce, který se používá pro přenos dat na sběrnici a podrobně popsány jednotlivé části tohoto rámce. Dále je zde popsán princip přenosu dat po sběrnici CAN 2.0, základní principy komunikace, a také princip arbitráže.

Dalším úkolem bylo porovnat architekturu FIFO a architekturu MailBox, podrobné informace jsou uvedeny v Kap. 5. V této kapitole je dále popsána funkce akceptačních filtrů a také náročnost na post-processing jádrem procesoru.

Další část diplomové práce byla zaměřena na praktické porovnání architektury FIFO a architektury MailBox v typických aplikacích. Pro stanovení skutečných časů těchto dvou odlišných architektur byly navrženy vývojové desky, jejichž popis je uveden v Kap. 6 a v příloze této práce. Byly měřeny tři základní konfigurace na sběrnici a to nízká přenosová rychlost a velký počet zpráv na sběrnici, vysoká přenosová rychlost a malý počet zpráv na sběrnici a poslední případ vysoká přenosová rychlost a velký počet zpráv – tyto konfigurace odpovídají typickým aplikacím. Podrobné výsledky a přesné parametry nastavené na sběrnici jsou uvedeny v Kap. 7. Z měření vyplynulo, že velmi závisí na dané konfiguraci, která je nastavena na sběrnici. Pokud byly kladeny malé nároky na vytížení daného uzlu, tak zpoždění na referenční zprávu s vysokou prioritou zpracování byly velice obdobné jak pro architekturu FIFO tak i pro architekturu MailBox pro všechny konfigurace. Rozdíly zpracování těchto architektur se pohybovaly od 9µs do 31µs. Výsledky pro jednotlivá měření jsou uvedeny v Kap. 7 a následné přehledné porovnání v Tab. 7.1.

Pokud byly kladeny vysoké nároky na vytížení daného uzlu, tak výsledné časy odezvy na referenční zprávu byly velmi odlišné. U těchto měření vždy architektura MailBox vyšla mnohonásobně lépe než architektura FIFO v typických aplikacích. Toto je zřejmě způsobeno faktem, který byl popsán v 7.4. U architektury MailBox můžeme na danou prioritní zprávu reagovat mnohem rychleji a to z důvodu, že pro každou zprávu je vyhrazen samostatný paměťový prostor (MailBox) s příslušným akceptačním

filtrem. Po příchodu zprávy s vysokou prioritou je procesor informován přerušením. Takto může procesor po dokončení rozpracované zprávy na tuto prioritní zprávu přednostně reagovat. Proto bych architekturu MailBox doporučil pro aplikace, kde vyžadujeme přednostní zpracování na určité zprávy. Jako nevýhodu této architektury bych považoval fakt, že je potřeba mnohem větší paměťový prostor vyhrazený pro uchování zprávy než je tomu u architektury FIFO. Pro uzel, který používá architekturu FIFO můžeme nakonfigurovat libovolný počet zpráv a stačí nám relativně malý paměťový prostor (u mikroprocesoru MC9S12XDP512 to byl FIFO buffer o pěti pozicích). Při použití architektury MailBox, kde daný uzel přijímá velký počet zpráv, musí být zajištěno, že pro každou zprávu je vyhrazen příslušný MailBox. To s sebou samozřejmě přináší vysoké požadavky na paměť a s tím i způsobuje vyšší cenu koncového uzlu. Existují prostředky jak u architektury MailBox provozovat velké množství zpráv a to tak, že pro daný MailBox se v akceptačním filtru povolí průchod více zpráv. V tomto případě je vhodné nastavit pro vysoce prioritní zprávy samostatné MailBoxy a do zbylých neobsazených MailBoxů nastavit filtry pro průchod více zpráv. Takto získáme výhodu proti architektuře FIFO v tom, že dokážeme zajistit přednostní zpracování určitých námi preferovaných zpráv.

Vhodná volba architektury v daném uzlu je však vždy závislá na konkrétní konfiguraci sítě. Z provedených měření u typických aplikací vyplynulo, že vždy vychází architektura MailBox lépe oproti architektuře FIFO. Největší výhoda architektury MailBox u typických aplikací se projevila při velkém zatížení uzlů.

9 Seznam literatury:

[1] ETSCHBERGER, K. *Controller Area Network*, Germany: IXXAT Press, 2001. ISBN 3000073760.

[2] PFEIFFER, O. *Embedded Networking with CAN and CANopen*, California: RTC Books, 2003. ISBN 3000073760.

[3] Robert Bosch GmbH. *CAN Specification Version 2.0*. Dostupné z WWW: http://www.semiconductors.bosch.de/pdf/can2spec.pdf

[4] ZEZULKA, F. *Automatizační prostředky*, Brno: VUT FEKT, 1999. ISBN 8021414820.

[5] CAN – Controller Area Network. Dostupné z WWW: http://fieldbus.feld.cvut.cz/can/

[6] VÁŇA, V. *Začínáme s mikrokontroléry HC08 NITRON*, 1. vyd. Praha: BEN, 2003. ISBN 80-7300-124-1

[7] POLÁK, K. Sběrnice CAN. *Internet Journal: Elektrorevue* (Dostupné z WWW: http://www.elektrorevue.cz/clanky/03021/index.html) [online]. 2003, No. 03021

[8] MACHÁLKA, L. *Nízkorychlostní přenosy dat v koncových zařízeních automobilů*, Bakalářská práce , Brno 2006

[9] Katalogový list. *MC9S12XDP512*. Dostupné z WWW: http://www.freescale.com/

[10] Katalogový list. *ST72561*. Dostupné z WWW: http://www.st.com/

[11] LIN Consortium. *LIN Specification Package Revision 1.3*. Dostupné z WWW: http://www.lin-subbus.org/

[12] LIN Consortium. *LIN Specification Package Revision 2.0*. Dostupné z WWW: http://www.lin-subbus.org/

[13] FlexRay Consortium. *FlexRay*. Dostupné z WWW: http://www.flexray.com/

[14] MIKÉSKA, Z. Specifikace rádiové části systému Bluetooth. *Internet Journal: Elektrorevue* (Dostupné z WWW: Obr. 3) [online]. 2003, No. 04003

[15] Bluetooth. Dostupné z WWW: http://www.bluetooth.com/

[16] Sdělovací technika. *Perspektivy automobilové techniky*. Dostupné z WWW: http://www.stech.cz/articles_print.asp?idk=97&ida=237

[17] CAN in Automation (CiA). Dostupné z WWW: http://www.can-cia.org/

[18] Katalogový list. *MC33989*. Dostupné z WWW: http://www.freescale.com/

[19] Katalogový list. *SAB 80C166*. Dostupné z WWW: http://www.infineon.com/

[20] Katalogový list. *SAE 81C90*. Dostupné z WWW: http://www.infineon.com/

[21] ROBB, S., MCASLAN, S. *XGATE Library: CAN Driver*. Dostupné z WWW: http://www.freescale.com/

[22] Katalogový list. *MAX220-MAX240*. Dostupné z WWW: http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf

PŘÍLOHY

Příloha č. 1: Schéma zapojení vývojové desky

Příloha č. 2: Obrazec desky plošných spojů vývojové desky (strana spojů bottom – modrá)

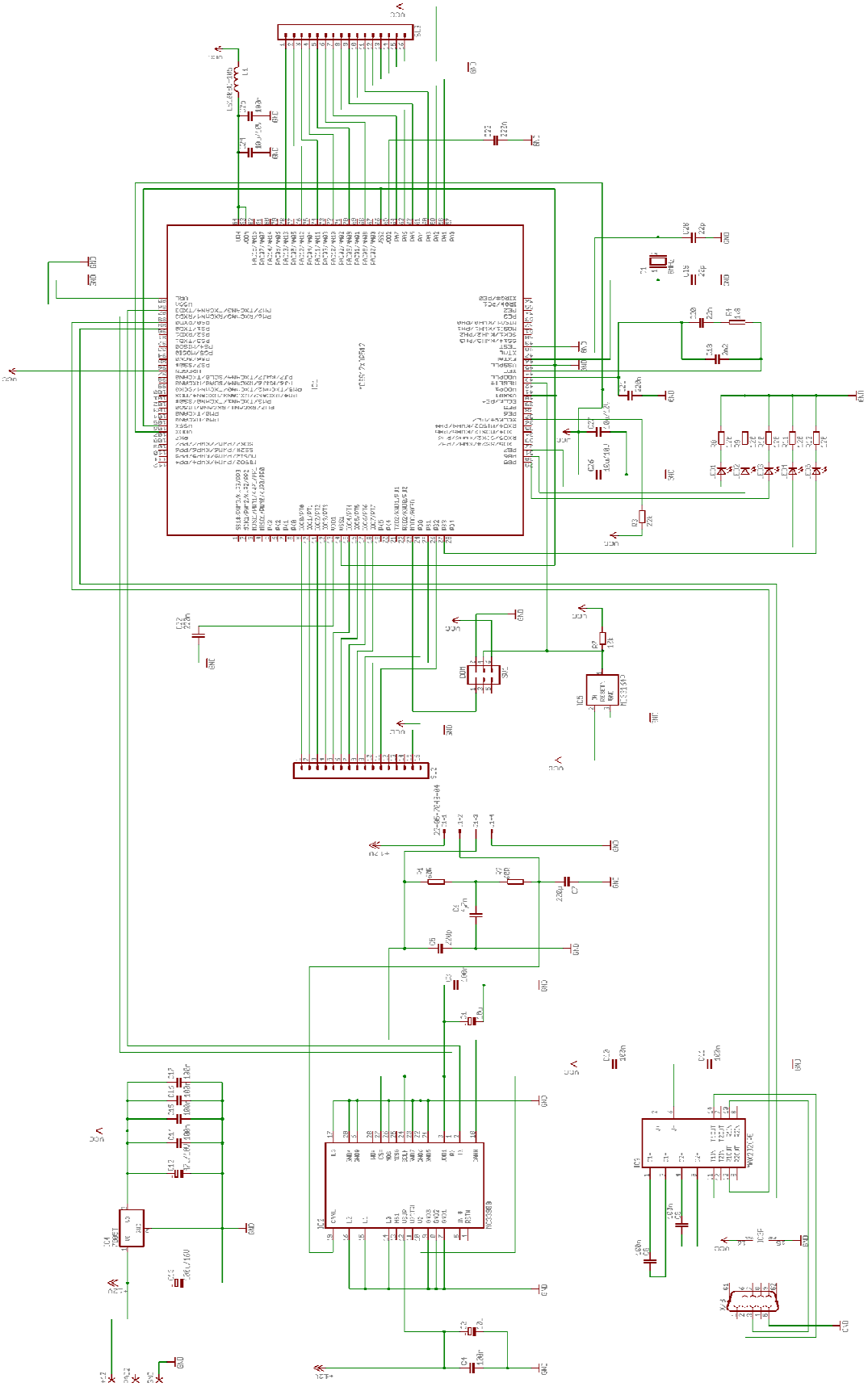
Příloha č. 3: Obrazec desky plošných spojů vývojové desky (strana součástek top – červená)

Příloha č. 4: Rozložení součástek na desce s plošnými spoji vývojové desky - pohled z vrchní strany (Top)

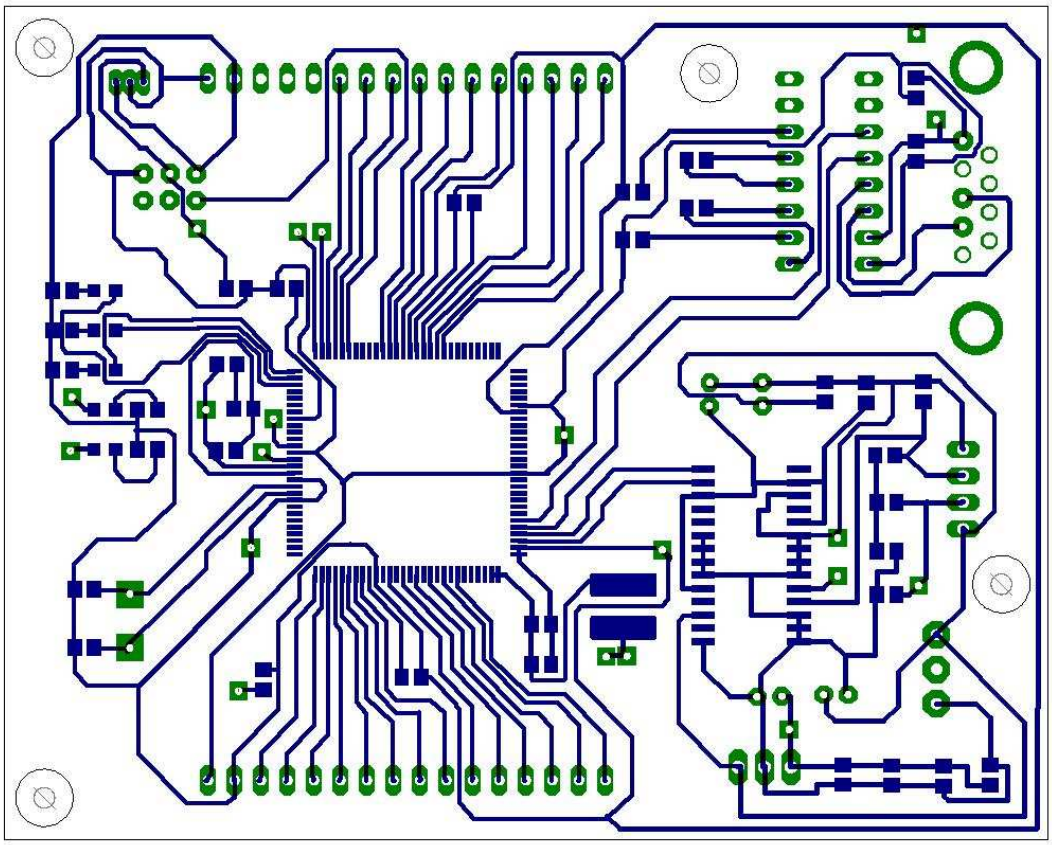
Příloha č. 5: Rozložení součástek na desce s plošnými spoji vývojové desky - pohled ze spodní strany (Bottom)

Příloha č. 6: Seznam součástek

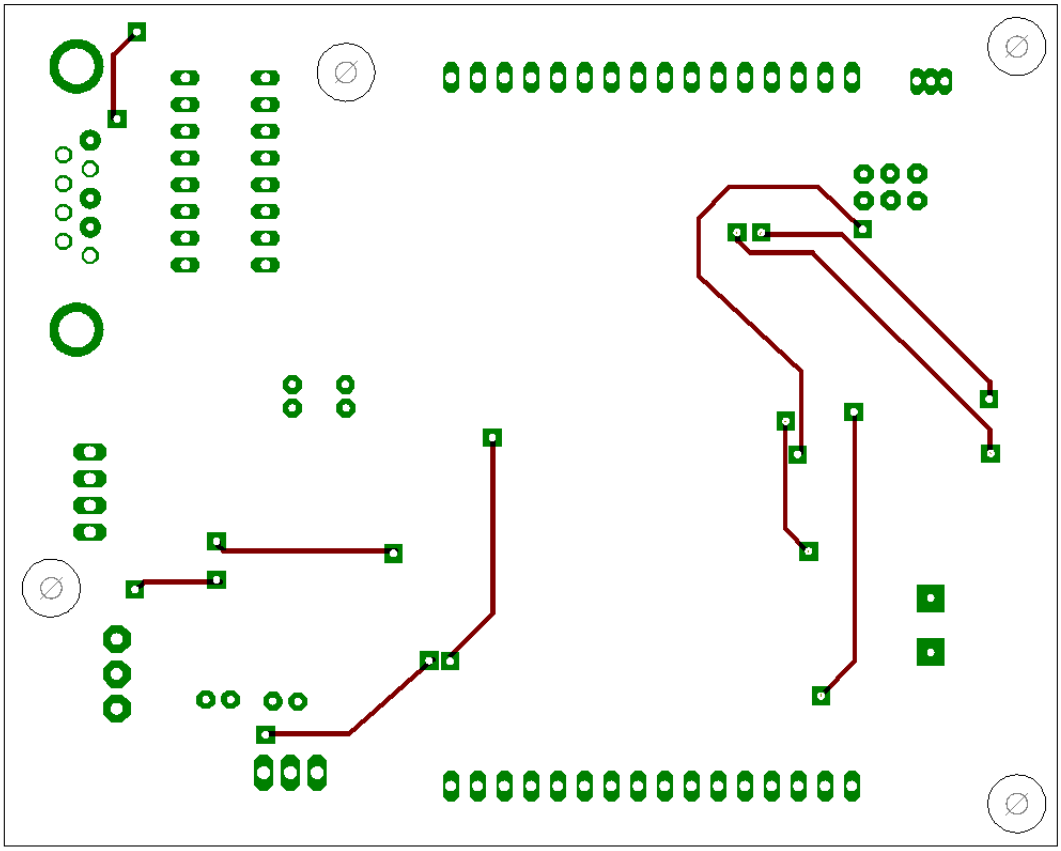
Příloha č. 1: Schéma zapojení vývojové desky



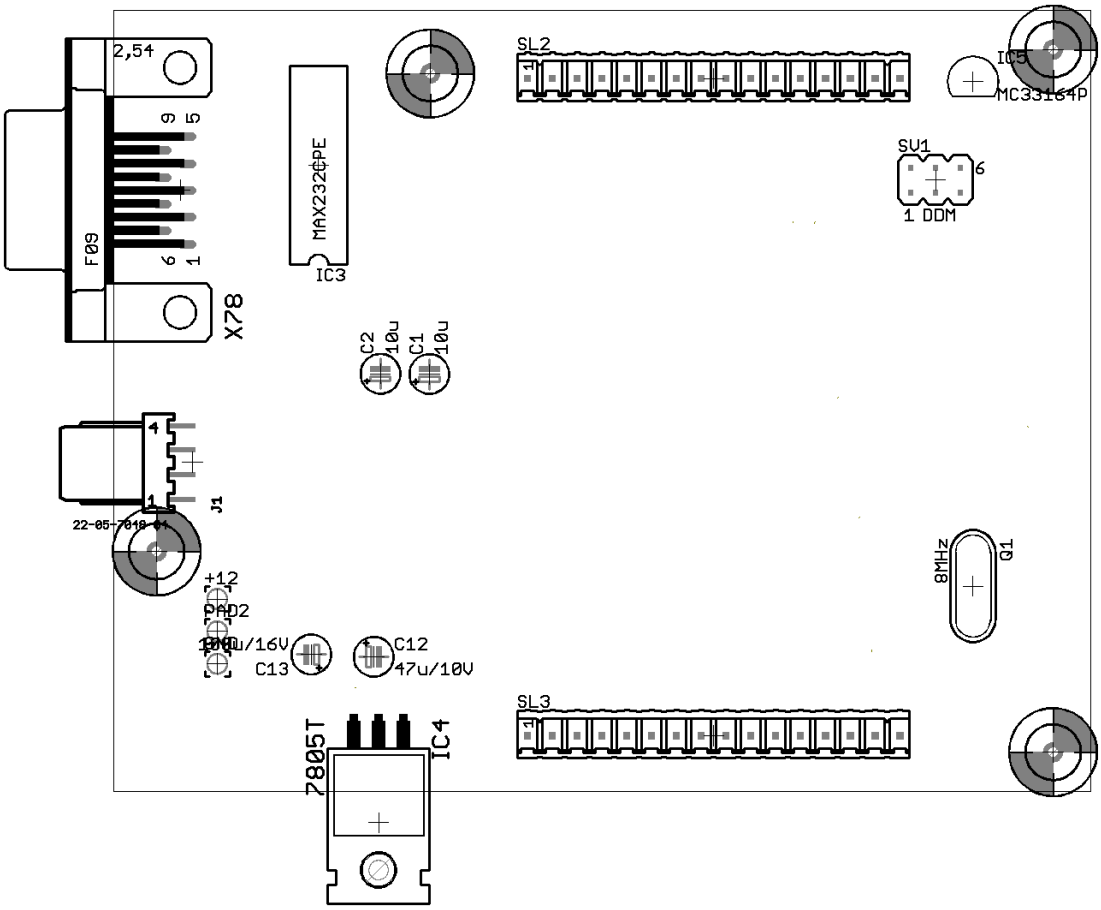
Příloha č. 2: Obrazec desky plošných spojů vývojové desky - strana spojů



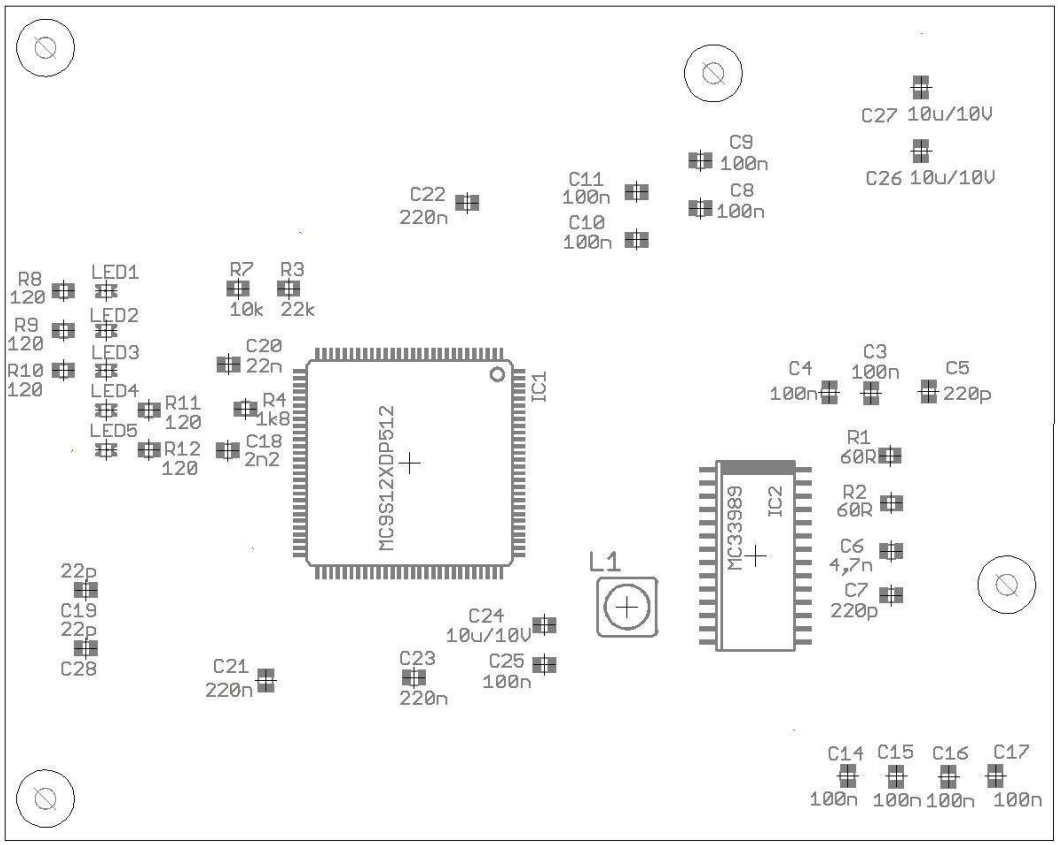
Příloha č. 3: Obrazec desky plošných spojů vývojové desky - strana součástek



Příloha č. 4: Rozložení součástek na desce s plošnými spoji vývojové desky - pohled z vrchní strany (Top)



Příloha č. 5: Rozložení součástek na desce s plošnými spoji vývojové desky - pohled ze spodní strany (Bottom)



Příloha č. 6: Seznam součástek

Seznam součástek:

Exported from DIPLOMKA.sch at 22.03.2008 22:41:53

EAGLE Version 4.16r2 Copyright (c) 1988-2006 CadSoft

Part	Value	Device	Package	Library
+12		2,54/1,0	2,54/1,0	wirepad
C1	10u	CPOL-EUE1.8-4	E1,8-4	rcl
C2	10u	CPOL-EUE1.8-4	E1,8-4	rcl
C3	100n	C-EUC0805	C0805	rcl
C4	100n	C-EUC0805	C0805	rcl
C5	220p	C-EUC0805	C0805	rcl
C6	4,7n	C-EUC0805	C0805	rcl
C7	220p	C-EUC0805	C0805	rcl
C8	100n	C-EUC0805	C0805	rcl
C9	100n	C-EUC0805	C0805	rcl
C10	100n	C-EUC0805	C0805	rcl
C11	100n	C-EUC0805	C0805	rcl
C12	47u/10V	CPOL-EUE1.8-4	E1,8-4	rcl
C13	100u/16V	CPOL-EUE1.8-4	E1,8-4	rcl
C14	100n	C-EUC0805	C0805	rcl
C15	100n	C-EUC0805	C0805	rcl
C16	100n	C-EUC0805	C0805	rcl
C17	100n	C-EUC0805	C0805	rcl
C18	2n2	C-EUC0805	C0805	rcl
C19	22p	C-EUC0805	C0805	rcl
C20	22n	C-EUC0805	C0805	rcl
C21	220n	C-EUC0805	C0805	rcl
C22	220n	C-EUC0805	C0805	rcl
C23	220n	C-EUC0805	C0805	rcl
C24	10u/10V	C-EUC0805	C0805	rcl
C25	100n	C-EUC0805	C0805	rcl
C26	10u/10V	C-EUC0805	C0805	rcl
C27	10u/10V	C-EUC0805	C0805	rcl
C28	22p	C-EUC0805	C0805	rcl
GND		2,54/1,0	2,54/1,0	wirepad
IC1	MC9S12XDP512	68HC912D60	TQFP112	
IC2	MC33989	T89C51CC02CA-TISIM	SO28W	
IC3	MAX232CPE	MAX232	DIL16	maxim
IC4	7805T	7805T	TO220H	linear
IC5	MC33164P	MC33164P	TO-226AA	linear
J1	22-05-7048-04	22-05-7048-04	7395-04	molex
L1	DS1608C-105	L-USWE-TPC	POWER-CHOKE_WE-TPC	rcl
LED1		LEDCHIPLED_0805	CHIPLED_0805	led
LED2		LEDCHIPLED_0805	CHIPLED_0805	led
LED3		LEDCHIPLED_0805	CHIPLED_0805	led
LED4		LEDCHIPLED_0805	CHIPLED_0805	led
LED5		LEDCHIPLED_0805	CHIPLED_0805	led
PAD2		2,54/1,0	2,54/1,0	wirepad
Q1	8MHz	XTAL/S	QS	special
R1	60R	R-EU_R0805	R0805	rcl
R2	60R	R-EU_R0805	R0805	rcl
R3	22k	R-EU_R0805	R0805	rcl
R4	1k8	R-EU_R0805	R0805	rcl
R5	1M	R-EU_M0805	M0805	rcl
R7	10k	R-EU_R0805	R0805	rcl

R8	120	R-EU_R0805	R0805	rcl
R9	120	R-EU_R0805	R0805	rcl
R10	120	R-EU_R0805	R0805	rcl
R11	120	R-EU_R0805	R0805	rcl
R12	120	R-EU_R0805	R0805	rcl
SL2		M16	16P	con-amp
SL3		M16	16P	con-amp
SV1	DDM	MA03-2	MA03-2	con-lstb
X78		F09HP	F09HP	con-subd