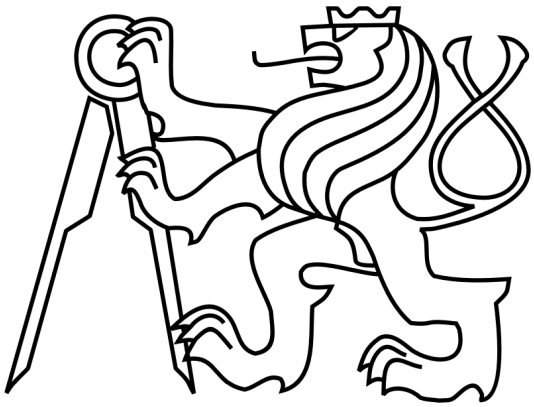
České vysoké učení technické v Praze
Fakulta elektrotechnická



BAKALÁŘSKÁ PRÁCE

Návrh analyzátoru sběrnice CAN pro nákladní automobily

Praha, 2007 **Autor: František Kořínek**

Prohlášení

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

V Praze dne _____ _____

podpis

Poděkování

Děkuji vedoucímu diplomové práce Ing. Pavlu Burgetovi, za jeho otevřený přístup a trpělivost. Dále chci poděkovat Ing. Radku Jarošovi a Ing. Luboši Jelínkovi za cenné rady a připomínky při vývoji. V neposlední řadě také mé rodině za jejich pochopení a podporu při studiu.

Abstrakt

Tato bakalářská práce popisuje návrh analyzátoru sběrnice CAN pro nákladní automobily. Cílem této práce bylo vytvořit software pro monitorování zpráv na sběrnici CAN pro protokol SAE J1939. Tento protokol je dnes hlavním standardem v komunikaci řídících jednotek nákladních automobilů, autobusů a zemědělských strojů. Jako hardware je použita DEMO deska s procesorem Freescale ColdFire MCF 5213. Softwarové řešení je takové, že lze pomocí PC nastavovat filtrování jednotlivých zpráv ze sběrnice. Toto nastavování probíhá pomocí konzolové aplikace.

Abstract

This bachelor work describes project of CAN bus analyzer for motor-trucks. The main aim of this study is to create software for monitoring massages on CAN bus for protocol SAE 1939. This protokol is in these days the main standard in communication of control units in motor-trucks, buses and agricultural machines. There was used DEMO board with pocessor Freescale ColdFire MCF 5213 as hardware. The software solution is to enable to set filtering of massages from bus with PC help. This setting is running with help of consol application.

vložit originální zadáníííííííííííííííí !!!!!

Obsah

Seznam obrázků

Kapitola 1

Úvod

Tato práce vznikla na základě mé spolupráce s firmou DevCom s.r.o., která se zabývá autodiagnostikou a vývojem speciální elektroniky. Firma DevCom rozšiřuje svůj autodiagnostický systém i na nákladní automobily. Pro toto rozšiřování je potřeba nastudovat a implementovat standardy v této oblasti implementované. Zde je to sběrnice CAN (*Control Area Network*) a komunikační standard SAE J1939. Sběrnice CAN je velmi rozšířená sběrnice, používaná v průmyslu. Díky svým vlastnostem a podpoře výrobců si získala výsadní postavení hlavně v automobilovém průmyslu. Protokol J1939 je protokol přímo navržený pro sběrnici CAN. Tento protokol obsahuje dokumenty odpovídající struktuře ISO/OSI. Zde jsem používal tři hlavní části: vrstvu linkovou, síťovou a aplikační (diagnostickou). V této práci jsem se nezaměřoval na vývoj hardware, pro tuto část práce jsem použil hardware používaný ve firmě DevCom. Je to DEMO deska PCB 058, na které je osazen mikroprocesor MCF 5213 s budičem CAN 82C250T. Softwarové řešení je rozděleno na čtyři hlavní části. V první jsem řešil komunikaci mikroprocesoru s PC přes sériové rozhraní, v druhé komunikaci po sběrnici CAN, třetí část řeší ovládání programu pomocí menu a čtvrtá spojuje předchozí tři části v systému přerušení.

Kapitola 2

Sběrnice CAN

V této kapitole bych chtěl přiblížit základní vlastnosti sběrnice CAN, která je jedním ze základů této bakalářské práce. Popíši zde části problematiky sběrnice, s nimiž jsem se setkal při řešení.

2.1 Úvod CAN

Sběrnice CAN byla vyvinuta firmou Bosch ve spolupráci s firmou Intel v polovině osmdesátých let a postupem času si získala dominantní postavení mezi sběrnicemi určenými pro automobilový průmysl. Zároveň začala být používána i jako průmyslová komunikační sběrnice na nižší systémové úrovni a na úrovni snímačů a akčních členů.

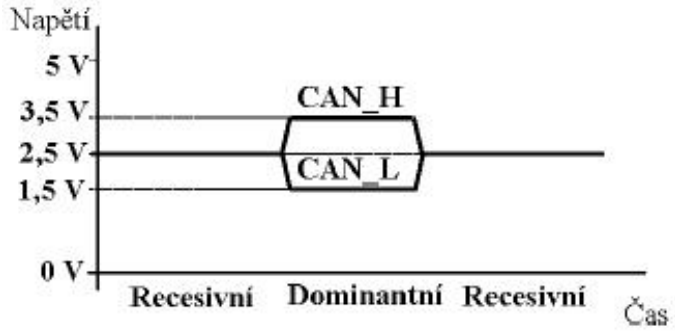
2.2 Základní vlastnosti

Sběrnice je navržena pro přenos s vysokým stupněm zabezpečení proti chybám s rychlostí přenosu do 1Mbit/s. Pomocí ní lze uskutečnit distribuované řízení systémů v reálném čase. Sběrnice je typu *multi-master*, to znamená, že každý uzel v síti může řídit ostatní uzly. Tento typ řízení umožňuje jednodušší řízení a zvyšuje spolehlivost. Porucha jednoho uzlu v této síti se neprojeví na funkčnosti celé sítě. Sběrnice je postavena na náhodném prioritním přidělování. Komunikace na síti je zprostředkována pomocí zpráv (datová zpráva a žádost o data). Řízení komunikace (*Network managemet*) zajišťují dvě speciální zprávy (chybové zprávy a zprávy o přetížení), tyto zprávy určují signalizaci

chyb a zastavení komunikace. Ve zprávě vyslané na sběrnici nejsou informace o cílovém uzlu, proto ji mohou přijmout všechny uzly připojené na sběrnici. Tyto uzly poznají podle identifikátoru, který uvozuje zprávu, její význam a prioritu. Identifikátorem lze zabezpečit, že uzel může pomocí tzv. *Acceptance Filteringu* přijímat pouze určité zprávy. Této vlastnosti budu využívat ve své práci pro nastavování filtrů pro zprávy protokolu J1939.

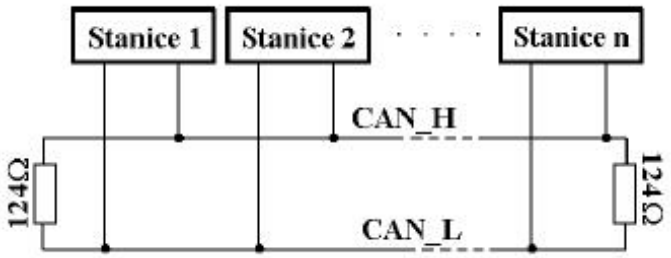
2.3 Fyzická vrstva

Fyzická vrstva realizuje elektrické spojení řídících jednotek. Sběrnici tvoří dva vodiče (označované CAN_H a CAN_L), kde *dominant* či *recessive* úroveň na sběrnici je definována rozdílovým napětím těchto dvou vodičů. Úroveň *recessive* odpovídá velikosti rozdílového napětí $V_{diff} = 0V$ a úroveň dominant $V_{diff} = 2V$. Na koncích vedení jsou připojeny odpory o velikosti 120 ohmů, které slouží k potlačení odrazů na vedení. Jestliže všechny uzly vysílají na sběrnici *recessive* bit, pak je na sběrnici úroveň *recessive*. Jestliže alespoň jeden z uzlů vysílá *dominant* bit, je na sběrnici stav *dominant*. Na sběrnici je vlastně realizována funkce logického součinu.



Obrázek 2.1: Logické úrovně sítě CAN.

Fyzická vrstva popisuje dále platnosti vysílacího budiče a přijímače, principy časování, synchronizaci a kódování bitů. Prakticky je možno na sběrnici připojit až 30 jednotek, rychlost sběrnice se zmenšuje s délkou sběrnice.

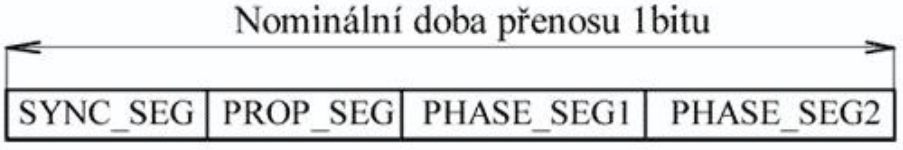


Obrázek 2.2: Schéma sběrnice CAN.

2.3.1 Časování bitů (*Bit timing*)

Tuto teoretickou část jsem řešil při nastavování periferie FlexCAN procesoru, zde je úvod do problematiky.

Pro udržení synchronizace mezi uzly CAN během přenosu zpráv se používají změny úrovně signálu na sběrnici. Doba trvání jednoho informačního bitu se dělí na čtyři časové segmenty. Každý segment se dělí na časová kvanta. Během SYNC_SEG se očekává hrana signálu. PROP_SEG slouží ke kompenzaci doby šíření signálu po sběrnici. PHASE_SEG1 a PHASE_SEG2, mezi kterými se nachází vzorkovací bod stavu sběrnice, se využívají ke kompenzaci fázových chyb na sběrnici. Je-li očekávána hrana signálu mimo SYNC_SEG, mění se jejich délka o programovatelný počet časových kvant. Aby se tento způsob kompenzace mohl realizovat bez vlivu na obsah přenášených zpráv, je použita metoda doplnění bitů opačné polarity. Obsahuje-li zpráva 5 bitů se stejnou polaritou, zařadí se automaticky do řetězce bitů bit s opačnou polaritou, který se na přijímací straně opět vyřadí.



Obrázek 2.3: *Bit timing.*

2.4 Linková vrstva

Tak jako v modelu ISO/OSI i v protokolu CAN je linková vrstva rozdělena na podvrstvu LLC a MAC:

- MAC (*Medium Access Control*) reprezentuje jádro protokolu CAN. Úkolem je provádět kódování dat, vkládat doplňkové bity do komunikace (*Stuffing/Destuffing*), řídit přístup všech uzlů k médiu s rozlišením priorit zpráv, detekce chyb a jejich hlášení a potvrzování správně přijatých zpráv.
- LLC (*Logical Link Control*) je podvrstva řízení datového spoje, což zde znamená filtrování přijatých zpráv (*Acceptance Filtering*) a hlášení o přetíženích (*Overload Notification*).

2.4.1 Řízení přístupu k médiu a řešení kolizí

Sběrnice je typu *multimaster*, kde všechny uzly mohou vysílat, když jsou připraveny a sběrnice je v klidovém stavu (*bus free*). Na sběrnici vysílají jednotlivé uzly v pořadí, v jakém o vysílání požádají. Ostatní vysílají až po úspěšném odeslání zprávy. Chybové rámce se dají vysílat okamžitě, když jsou identifikovány jakýmkoliv uzlem.

Při současném zahájení vysílání několika uzlů se přidělí vysílání uzlu, který vysílá s největší prioritou (nejnižším identifikátorem). Identifikátor je uveden na začátku zprávy. Každý vysílač porovnává hodnotu právě vysílaného bitu s hodnotou na sběrnici a zjistí-li, že na sběrnici je jiná hodnota než vysílá (jedinou možností je, že vysílač vysílá *recessive* bit a na sběrnici je úroveň *dominant*), okamžitě přeruší další vysílání. Tím se zajišťuje, že zpráva s vyšší prioritou je odeslána přednostně a že nedojde k jejímu poškození, což by mělo za následek opakování zprávy a zbytečné prodloužení doby potřebné k přenosu zprávy. Jestliže uzel nezíská přístup na sběrnici, vyčká, než bude sběrnice opět ve stavu *bus free*, a pak zkusí vysílat znovu.

2.4.2 Zabezpečení přenášených dat

Protokol CAN se vyznačuje silným mechanismem zabezpečení přenášených dat. Používá tyto mechanismy:

2.4.2.1 Potvrzení přijetí zprávy (*acknowledge*)

Každé zařízení, připojené ke sběrnici, musí správně přijatou zprávu potvrdit. Činí tak změnou bitu v poli ACK (1 bit) z úrovně *recessive* vysílané vysílačem na úroveň *dominant*. To platí i pro ta zařízení, která mají zapnuto filtrování a tedy zprávu nepřijímají.

2.4.2.2 Vkládání bitu (*bit stuffing*)

Při vyslání pěti bitů stejné úrovně na sběrnici je do zprávy vložen bit opačné úrovně. Tímto způsobem lze detekovat chyby a sesynchronizování přijímacích uzlů.

2.4.2.3 Monitoring

Vysílač porovnává vysílanou hodnotu bitu s úrovní na sběrnici. Jsou-li obě hodnoty stejné, vysílač pokračuje ve vysílání. Pokud je na sběrnici detekována jiná úroveň než odpovídá vysílanému bitu a probíhá-li právě řízení přístupu na sběrnici, vysílá se (*Arbitration Field*), přeruší se vysílání a přístup k médiu získá uzel vysílající zprávu s vyšší prioritou. Pokud je rozdílnost vysílané a detekované úrovně zjištěna jinde než v *Arbitration Field* a v potvrzení přijetí zprávy (*ACK Slot*), je vygenerována chyba bitu.

2.4.2.4 CRC kód (*Cyclic Redundancy Check*)

CRC je kód o délce 15 bitů a je to poslední pole vysílané zprávy. Je možno generovat ho ze všech odvysílaných bitů zprávy. Tento kód je generován podle polynomu: $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$. Když nějaký uzel detekuje chybu, vygeneruje chybu CRC.

2.4.2.5 Kontrola zprávy (*message frame check*)

Zpráva se kontroluje podle formátu udaného ve specifikaci a pokud je na nějaké pozici bitu zprávy detekována nepovolená hodnota, je vygenerována chyba rámce (formátu zprávy).

2.4.3 Signalizace chyb

Každý uzel má zabudována dvě interní čítače chyb udávající počet chyb při příjmu a při vysílání. Podle obsahů čítačů může uzel přecházet, co se týká hlášení chyb a jeho aktivity na sběrnici, mezi třemi stavy (aktivní, pasivní, odpojený). Pokud uzel generuje

příliš velké množství chyb, je automaticky odpojen (přepnut do stavu *Bus-off*) Z hlediska hlášení chyb tedy rozdělujeme uzly do následujících tří skupin:

2.4.3.1 Aktivní (*Error Active*)

Tyto uzly se mohou aktivně podílet na komunikaci po sběrnici a v případě, že detekují libovolnou chybu v právě přenášené zprávě (chyba bitu, chyba CRC, chyba vkládání bitů, chyba rámce), vysílají na sběrnici aktivní příznak chyby (*Active Error Flag*). Aktivní příznak chyby je tvořen šesti po sobě jdoucími bity *dominant*, čímž dojde k poškození přenášené zprávy (poruší se pravidlo vkládání bitů).

2.4.3.2 Pasivní (*Error Passive*)

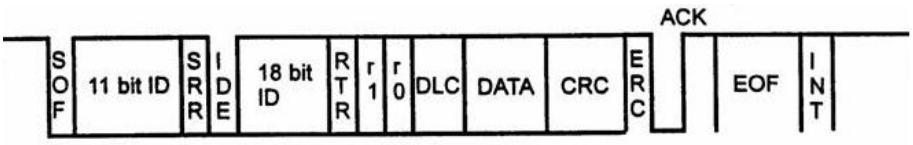
Tyto uzly se také podílejí na komunikaci po sběrnici, ale z hlediska hlášení chyb vysílají pouze pasivní příznak chyby (*Passive Error Flag*). Ten je tvořen šesti po sobě jdoucími bity *recessive*, čímž nedojde k destrukci právě vysílané zprávy.

2.4.3.3 Odpojené (*Bus-off*)

Tyto uzly nemají žádný vliv na sběrnici, jejich výstupní budiče jsou vypnuty.

2.4.4 Datová zpráva (*Data Frame*)

V protokolu CAN se vyskytují dva druhy datových zpráv, tyto zprávy se liší pouze délkou identifikátoru. První zpráva má identifikátor o délce 11 bitů a nazývá se Standard Frame. Druhý typ zprávy, který budu využívat níže a je specifikován v protokolu J1939, je tzv. *Extended Frame*. Tento formát zprávy obsahuje navíc 18 bitů identifikátoru, jinak se od předchozího nijak neliší. Tento formát je definován jako CAN 2.0B.



Obrázek 2.4: Přenosový rámec CAN 2.0B.

- SOF - začátek rámce
- Arbitration Field - Identifikuje zprávu a rozhoduje o prioritním přístupu ke sběrnici na základě *dominantních* úrovní signálů. Čím nižší identifikátor, tím vyšší priorita.
- RTR - 0 - *remote frame* bit
- SRR,IDE - identifikátor *extended* formatu
- DLC - délka datového pole (platné hodnoty 0...8)
- ACK - potvrzovací pole - Příjem každé zprávy musí být potvrzen alespoň jedním příjemcem.
- End of Frame - 7bitů recesivní úrovně (porušení bit *stuffingu*)

2.4.5 Žádost o data (*Remote Frame*)

Žádost o rámec má obdobný formát jako datový rámec. Neobsahuje však datové pole a bit RTR je *recessive* (v datovém rámci je *dominant*). Uzel takto žádá některý jiný uzel na síti o vysílání datového rámce se shodným identifikátorem, jaký je v žádosti.

2.4.6 Zpráva o chybě (*Error Frame)*

Chybový rámec sestává z polí *error flag* a *error delimiter*. Uzel, který zjistí chybu v řetězci přijímaných bitů, začne vysílat 6 *dominant* bitů, čímž poruší strukturu rámce. Ostatní uzly začnou též vysílat 6 *dominant* bitů. Celková délka *error flag* tak může být 6 až 12 bitů. Za nimi následuje pole *error delimiter* s 8 *recessive* bity.

2.4.7 Zpráva o přetížení (*Overload Frame*)

Rámec přeplnění má obdobnou strukturu jako chybový rámec. Uzel vyšle tento rámec především tehdy, když potřebuje určitý čas na zpracování předchozí zprávy.

Kapitola 3

Komunikační Protokol J1939

Zde popíši vlastnosti protokolu J1939, jehož nastudování je základem této bakalářské práce.

3.1 Úvod J1939

V oblasti autodiagnostiky osobních automobilů dnes neexistuje žádný veřejný publikovaný a přístupný standard. Každý výrobce si vytváří své vlastní systémy kódování zpráv. V této oblati je vývoj atodiagnostiky dosti komplikovaný, ale to neplatí o oblasti nákladních atomobilů, autobusů a zemědělské techniky. Pro tuto oblast je vytvořen komunikační protokol, který komunikaci mezi jednotlivými komponenty přesně specifikuje. Zatím je to jen doporučení, ale výrobci této techniky se ho s menšími odchylkami drží. Vyvinula ho Americká asociace automobilových inženýrů SAE (*Society of Automotive Engineers*), přesněji skupina elektrizace a elektronizace nákladních vozidel a autobusů. Toto doporučení je navrženo na standard sběrnice CAN (ISO 11898) popsaný výše. Dokument má 9 průběžně doplňovaných a aktualizovaných částí.

Obrázek 3.1: Model ISO/OSI

- Část J1939/21 - V Data Link Layer jsou definovány obecné vlatnosti sběrnice CAN pro komunikaci jednotek v rámci hnacího řetězce vozidla.
- Část J1939/31 - Network Layer předepisuje vytváření svazků CAN-ových sítí a jejich vzájemné propojení v rámci této sítě.
- Část J1939/71 - Vehicle Application Layer je nejrozsáhlejší ze všech, obsahuje definice jednotlivých zpráv a jejich vlastnosti.
  - Identifikátor zprávy
  - Frekvenci nebo podmínky pro vysílání
  - Uspořádání a kódování datové části

Mým hlavním úkolem bylo prostudovat tyto tři části. Další části dokumentu bylo mutno prostudovat také, ale ty odpovídají s různými omezeními standardu ISO 11898.

3.2 Obecné vlastnosti

- Pevně stanovená přenosová rychlost na 250 000 bitů/s.

- Maximální délka sběrnice 40 m.
- Maximální počet uzlů je 30.
- Dvě varianty přenosového média.
  - Stíněný kroucený pár se zemí.
  - Kroucený čtyřdrát s aktivním zakončením, nevyžaduje stínění.
- Lze přenést 1850 zpráv za sekundu (zátěž sběrnice 100%).
  - Používá se periodický přenos (od 10 ms do 1 s).
- Datová část zprávy má právě 8 bajtů.
- Používá se výhradně 29 bitový identifikátor (specifikace CAN 2.0B) s jinou interpretací.

3.3 Linková vrstva

V této vrstvě jsou definovány obecné komunikační vlatnosti sběrnice CAN. Je zde popsána struktura datových rámců identifikace, transportní protokol pro přenos více bajtových zpráv a kódování skupin parametrů. Tato vrstva byla z hlediska mé práce ta nejdůležitější. Podle hlaviček zpráv, které jsou specifikovány právě v této vrstvě, jsem řešil filtraci dat. Vrstva definuje rozdělení parametrů do skupin PG (*Parametr Group*), které jsou označeny číslem PGN (*Parametr Group Number*).

3.3.1 Skupiny parametrů

Skupiny parametrů v sobě sdružují podobné signály. V přepisu SAE J1939-71 (*Vehicle Application Layer*) jsou skupiny parametrů a signály, které obsahují. Někteří výrobci si přidávají potřebné specifické parametry. Každá skupina parametů je definována jedinečným číslem PGN (*Parametr Group Number*). Toto číslo je v identifikátoru zprávy složeno ze dovu částí. První je PDU *format*, druhá PDU *specific*. Existují dva typy skupin parametrů (PGN):

- *Global* PGN pro skupiny parametrů, které jsou vysílány všem jednotkám ECU (*Elektronic Control Unit*). Tento typ vysílání se nazývá *broadcast*. Toto PGN používá všech 16 bitů, hodnota horních 8 bitů (PDU *format*) musí být větší než 239.

- *Specific* PGN pro skupiny parametrů, které jsou posílány jednotlivým ECU. Tato komunikace se nazývá (*peer-to-peer*). Toto PGN používá pouze 8 vyšších bitů (PDU *format*) a jejich hodnota musí být menší než 240. Dolních 8 bitů (PDU *specific*) musí být vždy 0.

PGM může definovat $(240 + (16 * 256)) = 8672$ různých skupin parametrů.

3.3.2 Struktura identifikátoru

CAN identifikátor zprávy v protokolu J1939 obsahuje PGN, zdroujovou adresu, prioritu, *data page bit* a cílovou adresu (pouze pro *peer-to-peer* PG).

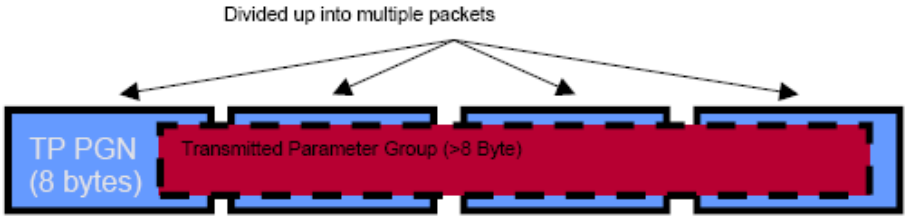
Obrázek 3.2: Struktura idnetifikátoru zprávy.

Když je PDU *format* < 240 (*peer-to-peer*), PDU *specific* obsahuje cílovou adresu. *Global* (255) může být také použita jako cílová adresa, tato skupina parametrů je určena pro všechny ECU. Pro tento případ je PGN tvořeno jen PDU formátem. Když je PDU

format >= 240 (*broadcast*), PDU *format* spolu s PDU *specific* utváří PGN vysílané skupiny parametrů.

3.3.3 Transportní protokol

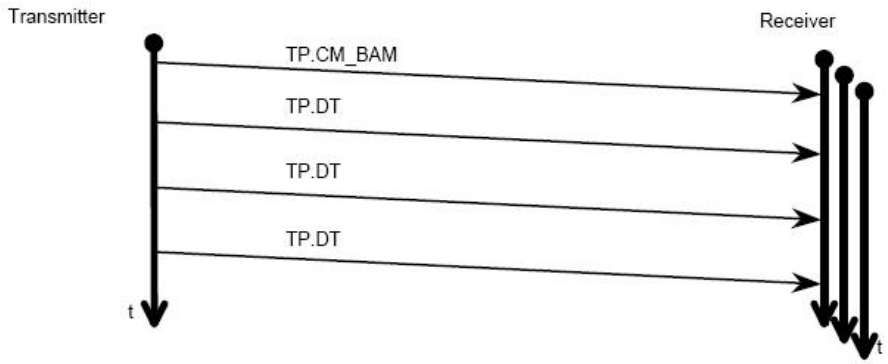
Skupiny parametrů, které obsahují více než 8 datových bajtů, jsou vysílány pomocí transportního protokolu.



Obrázek 3.3: Struktura transportu zpráv.

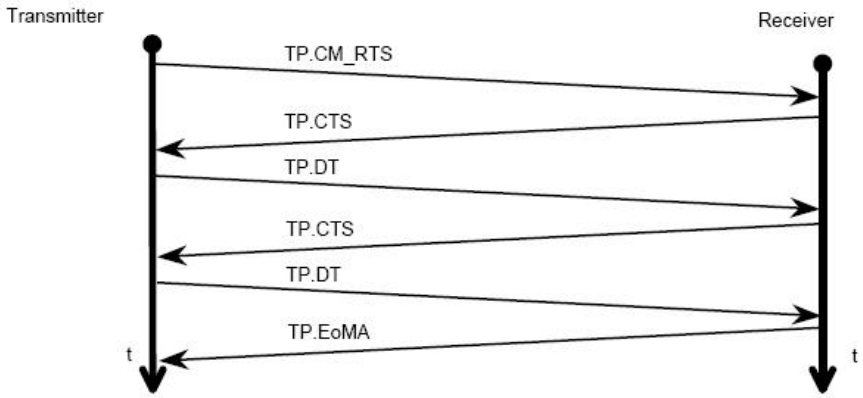
Pro *peer-to-peer* a *broadcast* vysílání existují dva různé protokoly. Pro tento přenos má protokol dvě speciální skupiny parametrů, které používá pro *Connection Management* (TP.CM) a *Transmission Data* (TP.DT).

Pro *broadcast transmission* je použit BAM protokol. Po BAM-PG (*Broadcast Announce Message*) vysílač pošle všechna data skupiny parametrů v minimálním intervalu 50ms.



Obrázek 3.4: Komunikace *broadcast*.

Při komunikaci *peer-to-peer* vysílač naváže spojení zprávou request to send. Přijímač vyšle odpověď *clear to send* a *end of message acknowledge*.



Obrázek 3.5: Komunikace *peer-to-peer*.

3.4 Aplikační vrstva

Aplikační vrstva obsahuje definice parametrů jednotlivých zpráv. Celkem definuje předpis SAE J1939 (verze z roku 1999) 145 zpráv, které specifikují přenos i takových informací jako blokování immobilizéru, teplotu povrchu pneumatik a vozovky nebo laserové navádění tahače na přívěs. Pro lepší využití přenosové kapacity jsou některé parametry sdružovány do skupim. Pro přenášené veličiny jsou definovány atributy:

- Délka dat - Kolik bajtů dat obsahuje jednotlivý parametr.
- Typ veličiny - Říká, jestli data jsou typu stavová nebo měřená.
- Rozsah platnosti příchozích dat.
- Fyzické rozlišení - Rozlišení fyzikální veličiny.
- Diagnostické údaje - Tyto údaje se vysílají na vyžádání.

Příklad skupiny parametrů:

Jméno skupiny:	Teplota motoru (ETEMP)
Perioda vysíláni:	1s
Délka dat:	bajtů
Data page:	0
PDU format:	254
PDU specific:	238
Priorita:	6
PG Number:	65,262 (FEEE16)
vysílá:	motor
identifikátor:	18FEEE00h

Popis dat:

Bajty:	1	teplota chladiva: -40° +210°C
	2	teplota paliva: -40° +210°C
	3,4	teplota oleje motoru: -273° +1735°C
	5,6	teplota oleje turbodmychadla: -273° +1735°C
	7	teplota mezichladiče motoru: -40° +210°C
	8	otevření termostatu mezichladiče: 0 - 100%

Časový interval vysílání zprávy je určován s ohledem na důležitost obsažených informací. Pro některé zprávy není perioda opakování určena, takové zprávy se vysílají jen na vyžádání (obvykle obsahují diagnostiku daného zařízení) nebo ve specifických případech (např. po zastavení motoru).

Datová část zprávy obsahuje aktuální hodnoty určených veličin. Zařízení, která zprávu vysílají, nemusí vyplnit všechny předpisem definované hodnoty, ale musí na jejich místě vysílat bajt, jehož všechny bity mají hodnotu rovnou 1. To zajišťuje kompatibilitu stávajících i budoucích verzí jednotek připojených na CAN. Data o rozsahu větším než 8 bajtů (např. informace o konfiguraci motoru) se vysílají v blocích po 8 bajtech s tím, že před zahájením takového přenosu je vysílána speciální informační zpráva.

3.5 Síťová vrstva

Množství zařízení připojených na síť není nekonečné, omezujícím faktorem je zejména přenosová kapacita. Proto se zařízení rozdělují do podsítí, které jsou navzájem spojeny mostem (*bridge*). Most propojuje jednotlivé podsítě a propouští jen některé informace, které jsou podstatné pro činnost druhé podsítě. Sítě se rozdělují podle příbuznosti a podobnosti, ale každý výrobce vytváří sítě odlišně. V moderních automobilech lze většinou nalézt 2 až 3 sítě. Příklad rozdělení sítí v automobilu:

- Síť hnacího řetězce: motor, převodovka, retardéry, tachograf, palubní deska, immobilizér, ABS/ASR, elektronicky řízené brzdy, stabilizační systém....
- Síť (sítě) doplňkových zařízení: ovládání světel, klimatizace, centrální zamykání, ovládání dveří a informační panely (autobusy), pérování, automatické stěrače, ovládání oken, polohování sedaček, GPS, navigace, ...

3.5.1 Network management

V J1939 má každá síťová jednotka svou unikátní adresu. Každá zpráva, která je poslána jednotkou, obsahuje tuto zdrojovou adresu. Existuje 255 možných adres.

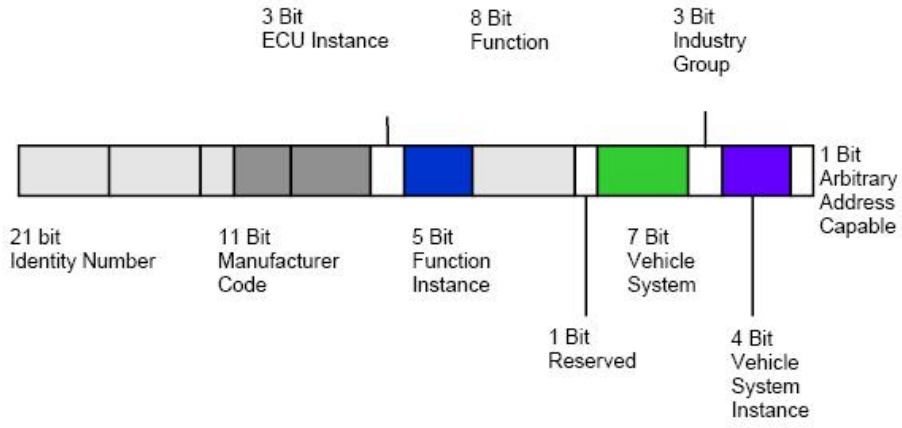
- 0-253 - Platné adresy ECU
- 254 - Nulová (žádná) adresa
- 255 - Globální adresa

Příklad adres zařízení v rámci hnacího řetězce vozidla:

00h ..... motor
03h ..... převodovka
0Bh ..... ABS / ASR.

Každý typ zařízení má preferovanou adresu. Předtím, než zařízení může použít adresu, musí se zaregistrovat na sběrnici. Tento proces se nazývá *address claiming*. Proto zařízení posílá takzvanou *AddressClaim* skupinu parametrů s požadovanou zdrojovou adresou. Tato skupina parametrů obsahuje 64 bitové jméno zařízení. Pokud je adresa již používána

jiným zařízením, pak zařízení, jehož jméno má vyšší prioritu, získá tuto adresu. Jméno zařízení obsahuje informace o zařízení a popisuje jeho funkci.



Obrázek 3.6: Struktura jména zařízení.

Kapitola 4

Softwarové řešení

V této kapitole jsou popsány vlastnosti programu a použitého hardwaru a jejich nastavování při programování i uživatelském použití.

4.1 Úvod

Tato práce vznikla s cílem využití ve firmě DevCom s.r.o. pro načítání komunikace nákladních automobilů a pozdější využití v autodiagnostice. Vývoj probíhal po konzultaci s vedením firmy tak, aby odpovídal stávajícím zvyklostem. Ve firmě DevCom mi byly poskytnuty veškeré podklady a prostředky pro vývoj a studium. Řešení jednotlivých částí práce je proto uzpůsobeno možnostem firmy a jejích prostředků. Výsledný program je na přiloženém CD v adresáři can_analyzátor.

4.2 Použitý hardware

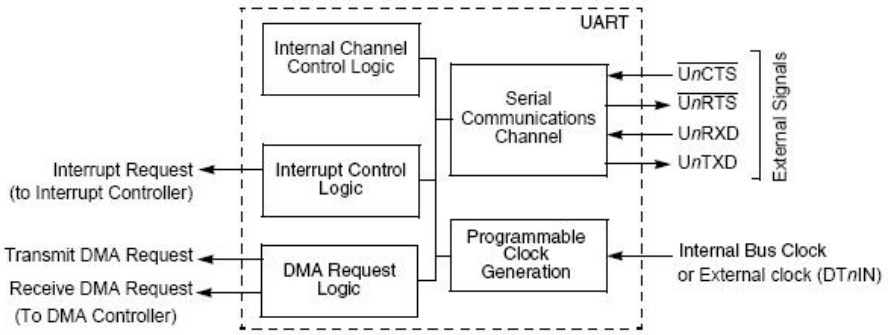
Cílem práce nebylo vyvíjet vlastní hardware, proto v tomto odstavci jen krátce popíši použitou DEMO destičku. Na této destičce je osazen mikroprocesor Freescale MCF 5213 ColdFire a budič sběrnice CAN PCA82C250T. Mimo tyto dvě hlavní součástky jsou osazeny i obvody pro jiné periferie, kterými se však nebudu zabývat. Nejdůležitější je použitý mikroprocesor. Firma DevCom používá mikroprocesory Freescale do většiny svých aplikací, kvůli široké možnosti jejich nasazení.

4.2.1 MCF5213 Coldfire

Tento procesor patří do skupiny 32-bitových *embedded* (vestavěný, zabudovaný) procesorů. Je to procesor s redukovanou instrukční sadou (RISC). Jeho taktovací frekvence je 66,80 MHZ. Je vybaven 32KB statické paměti RAM a 256KB paměti Flash. Má v sobě zabudovaný systém výjimek a přerušení, který je nástupcem systému používaného v procesorech Motorola 68000. Nejdůležitější periferie procesoru pro tuto práci jsou UART (*Universal Asynchronous Receiver Transmitter*) a komunikační kontrolér FlexCAN. Procesor obsahuje mnoho dalších periferií, které však v této práci nepoužívám. Více informací lze nalézt v [1]. Firma Freescale má velmi propracovaný systém podpory, na jejích internetových stránkách lze nalézt spoustu dokumentace a vzorových kódů pro jednotlivé procesory.

4.2.1.1 UART

Sériový komunikační kanál zajišťuje full-duplexní asynchronní nebo synchronní přijímač a vysílač, mající operační frekvenci interní hodinové sběrnice nebo externího zdroje. Vysílač převádí paralelní data z CPU (*Central Processing Unit*) na sériová. Do těchto dat vkládá start bity, stop bity a paritní bity. Přijímač tuto operaci provádí opačně, převádí sériová data ze vstupu (UnRXD) na paralelní, kontroluje start, stop a paritní bity. Příjem může být uskutečněn pomocí *pollingu* (odchytávání), přerušení nebo požadavku DMA (*Direct Memory Access*).

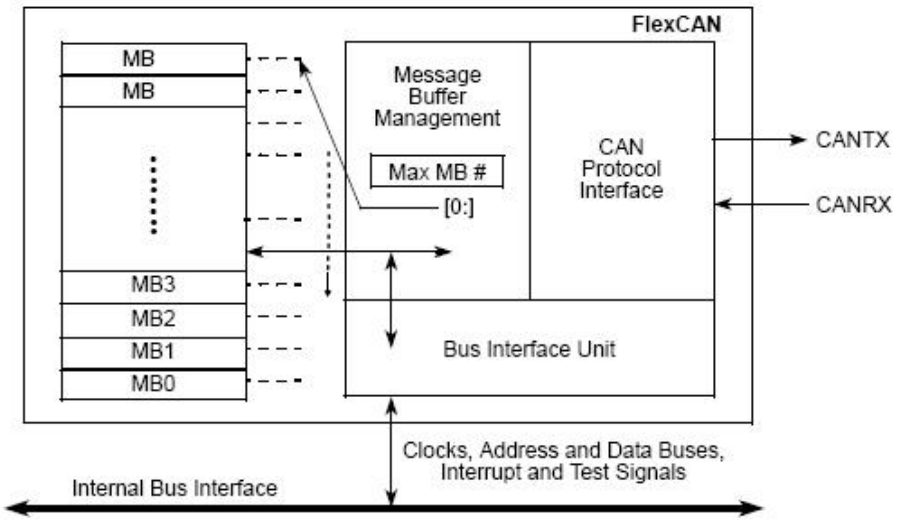


Obrázek 4.1: Blokové schéma periferie UART.

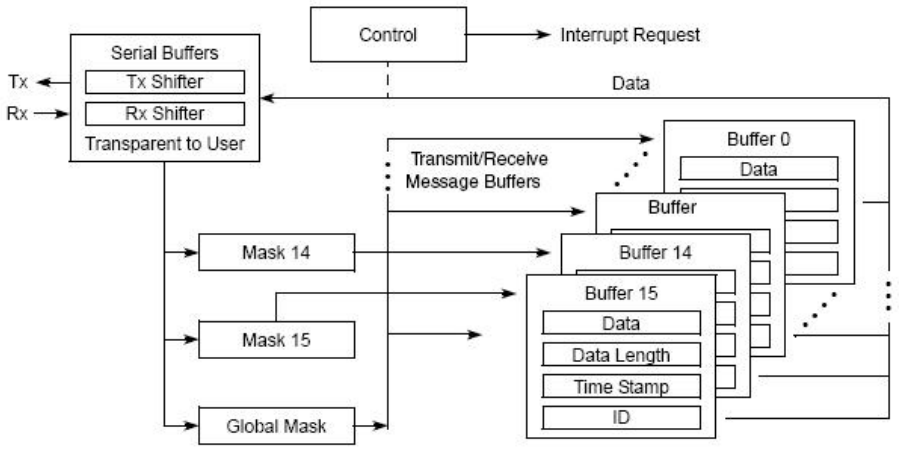
4.2.1.2 FlexCAN

Kontrolér FlexCAN je sériová asynchronní komunikační periferie procesoru, která zprostředkovává vysílání a přijímání zpráv ze sběrnice CAN. Na obrázku 4.2 je blokové

schéma celé periferie. Obrázek 4.3 ukazuje strukturu bufferů zpráv (*Massage Buffer*). Struktura jednotlivých bufferů odpovídá položkám zprávy CAN. Navíc je zde vstupní *Serial buffer*, je to záchytný vstupní buffer, který je důležitý při *acceptance filteringu* (viz. kapitola 2.2). Dále struktura obsahuje registry masek a registry přerušení. Podrobný popis funkce těchto registrů lze nalézt v literatuře [1].



Obrázek 4.2: Blokové schéma periferie FlexCAN.



Obrázek 4.3: Blokové schéma architektury bufferů zpráv.

4.3 Použitý software

Pro programování mikroprocesoru se dnes standardně používá programovací jazyk asembler nebo jazyk C. Pro pohodlnější programování a přehlednost kódu je lepší používat jazyk C, naproti tomu, když je potřeba naprogramovat rychlá rutina, používá se jazyk asembler. Já jsem používal výhradně jazyk C, inicializační rutiny procesoru jsem převzal z dokumentace firmy Freescale a DevCom. Jako programovací prostředí jsem použil Dev-C++ s GNU (*General Public License*) od firmy Bloodshet Software. Je to editor s podporou syntaktického zvýraznění a správou projektu. Pro kompilaci a slinkování projektu je použit kompilér a linker od firmy Freescale. Nahrávání programu probíhalo přes sériové rozhraní RS 232 pomocí aplikace Downloader vyvinuté ve firmě Devcom.

Pro odladění a sledování komunikace mezi PC a mikroprocesorem jsem dále používal konzolovou aplikaci Hercules a SerialWatcher. Pomocí konzole Hercules je také možno ovládat vytvořenou aplikaci. Další utilitou, která mi ulehčila práci při programování mikroprocesoru, je CFinit. Je to program od firmy MicroAPL, generující inicializační kód pro mikroprocesory Freescale. K testování filtrů a komunikace po sběrnici CAN jsem využil program PP2CAN s příslušným hardwarem USB2CAN. Všechny tyto programy jsou buď volně šiřitelné nebo byly zakoupeny firmou DevCom.

4.4 Ovládání analyzátoru

Program Analyzátor sběrnice CAN je implementován v procesoru a lze ho ovládat pomocí sériového rozhraní RS 232 z konzolové aplikace na PC. Ovládání je řešeno pomocí dialogových výpisů, které přiřazují každé operaci odpovídající ASCII znak.

Menu programu:

- **Příjem**
  - **Předdefinované hlavičky** - V tomto stupni menu lze natavit deset předdefinovaných zpráv protokolu J1939. Po nastavení lze tuto zprávu změnit, nastavit další v pořadí, vypsat všechny nastavené hlavičky nebo je smazat.
  - **Nastavení filtrů** - Zde lze zadat libovolnou hlavičku definovanou v protokolu J1939. Po nastavení lze tuto zprávu změnit, nastavit další v pořadí, vypsat všechny nastavené hlavičky nebo je smazat.

  - **Nastavení rozsahu příjmu (scaner)**
    - **Nastavení rozsahu přijímaných zpráv** - Ruční nastavení globální masky zpráv pro omezení rozsahu příjmu.
    - **Příjem všech zpráv ON/OFF** - Zapnutí a vypnutí scaneru.

- **Vysílání**
  - **Nastavení parametrů zprávy**.
    - **Nastavit zprávu**
      - **ID** - Identifikátor zprávy.
      - **LENGHT** - Délka zprávy (v protokolu J1939 musí být vždy 8 !!!)
      - **DATA** - Datové bajty vysílané zprávy.
    - **Vyslat nastavenou zprávu**
    - **Smazat nastavenou zprávu**
    - **Zobrazit nastavenou zprávu**

- **O programu**

4.5 Funkce UART

Popis funkcí periferie UART. Tyto funkce slouží k obsluze analyzátoru a výpisům na PC.

4.5.1 Inicializace UARTu

Deklarace funkce ve zdrojovém kódu:

int **uart_init**(int ch, unsigned long baudrate, unsigned long data_bits, unsigned long stop_bits, unsigned long parity, unsigned long ien)

Inicializace sériového rozhraní slouží k přípravě periferie UART na příjem a vysílání. Této funkci je nutné zadat číslo kanálu UART (MCF 5213 obsahuje tři kanály UART), přenosovou rychlost, počet datových bitů, počet stop bitů, typ parity. Je nutno také povolit obsluhu periferie pomocí přerušení. Ve funkci jsou ošetřeny rozsahy jednotlivých

vstupních hodnot a následné přiřazení vstupních hodnot do příslušných registrů. V literatuře [1] je uveden přesný postup inicializace periferie, popis jednotlivých registrů a jejich bitů.

4.5.2 Vysílací funkce UARTu

Deklarace funkce ve zdrojovém kódu:

void **uart_send**(char znak)

Tato funkce vyšle libovolný znak z procesoru do PC. Použil jsem tuto funkci pro řízení a zpřehlednění výpisů na sériové konzoli.

Nejvíce používaná funkce v projektu je **printf**. Slouží k výpisu znakových řetězců. Není to standardní funkce jazyka C, ale je to funkce ze zdrojového kódu společnosti Freescale a funguje obdobně.

4.5.3 Přijímací funkce UARTu

Deklarace funkcí ve zdrojovém kódu:

int **uart_IsChar**(void)

int **uart_recv**(void)

Funkce **uart_IsChar** vrátí 1, pokud UART detekuje příchozí znak, následující funkce **uart_recv** přečte příchozí znak z registru.

4.6 Funkce FlexCAN

Popis funkcí periferie FlexCAN. Tyto funkce slouží k nastavování vlastností analyzátoru, jako jsou filtry, rozsah příjmu (viz. kapitola 4.4 Ovládání analyzátoru).

4.6.1 Inicializace FlexCANu

Deklarace funkce ve zdrojovém kódu:

int **init_CanBus**(unsigned long baud_rate, unsigned int Rx_buff_num, int irq_Rx_Mask)

Inicializací periferie FlexCAN nastavíme počáteční vlastnosti kontroléru. Parametry pro tuto funkci jsou rychlost sběrnice, počet přijímacích bufferů a maska bufferů, od kterých se má zpracovávat přerušení. Je zde řešen *bit timing* sběrnice, jelikož rychlost sběrnice je pro protokol J1939 vždy 250Kbaud, použil jsem program CFInit zmíněný v kapitole 4.3. V literatuře [1] je uveden přesný postup inicializace periferie, popis jednotlivých registrů a jejich bitů.

4.6.2 Vysílací funkce FlexCANu

Deklarace funkce ve zdrojovém kódu:

int **CanBus_Trasmit**(tCAN_message Tx_mess)

Funkce vyšle rámec na sběrnici. Jako parametr je mu předána struktura Tx_mess, která obsahuje identifikátor zprávy, formát zprávy (pro protokol J1939 je vždy *extended*), časovou značku, počet bajtů dat (pro protokol J1939 je vždy 8) a jednotlivé bajty dat. Položky této struktury jsou překopírovány do odpovídajících registrových pozic vysílacího bufferu a následně zapsáním kódu pro vyslání poslány na sběrnici.

4.6.3 Přijímací funkce FlexCANu

Deklarace funkcí ve zdrojovém kódu:

int **CanBus_Receive_SetID**(int cnt,unsigned int Mess_ID,unsigned short IDE)

Funkce slouží k nastavování filtrů. Zapíše identifikátor zprávy Mess_ID do bufferu cnt. IDE udává formát správy (pro protokol J1939 je vždy extended). Kontrolér FlexCAN porovnává příchozí hlavičku zprávy s nastavenou a když se shodují, uloží celou příchozí zprávu do bufferu cnt.

int **CanBus_Receive_SetMASK**(int num_mask,unsigned int Mask,unsigned short IDE)

CanBus_Receive_SetMASK nastavuje globální masku pro všechny použité buffery zpráv. Touto funkcí lze nastavovat rozsah příjmu.

int **CanBus_Receive**(int num_buf,tCAN_message *Rx_mess)

Tato funkce je samotná přijímací funkce, po detekci přerušení od bufferu num_buf se do struktury Rx_mess překopírují data z bufferu, který vyvolal přerušení.

4.7 Stavový automat

Deklarace funkce ve zdrojovém kódu:

int **int PC_service**(void)

Stavový automat je funkce vytvářející konzolové menu programu. Funkce je plně průchozí. Je umístěna v hlavní programové smyčce, při každém průchodu smyčkou je v této funkci testován příchod znaku z PC. Když přijde znak odpovídající položce v menu programu, automat nastaví své příznakové proměnné na příslušné hodnoty odpovídající stavu vnoření do menu. Stavový automat volá výše popsané funkce a zajišťuje další operace s daty jako jsou mazání a změna dat, výpis nastavených dat a zadání dalších. V automatu je možné pohybovat se oběma směry ve struktuře menu.

4.8 Přerušení

Popisovaná kapitola slučuje všechny předchozí. Systém přerušení mikroprocesoru MCF 5213 má 57 zdrojů, které jsou schopny generovat přerušení. Těmto zdrojům lze přiřadit prioritu obsluhy 0 až 7, kde 7 má nejvyšší prioritu. V procesoru je nutno nastavit registry přerušení globálního systému a přiřadit vektorům přerušení obslužné rutiny, dále se musí nastavit registry určené pro přerušení u jednotlivých periférií. V programu analyzátoru

je použito přerušení od jednotlivých přijímacích bufferů periferie FlexCAN. Při detekci přerušení procesor vykoná obslužnou rutinu. Dále popíši funkce použité pro ukládání příchozích zpráv pod přerušením do kruhového bufferu.

Deklarace funkcí ve zdrojovém kódu:

void **CanBus_user_handle**(unsigned short source_int)

Tato funkce vykonává obsluhu přerušení od jednotlivých *massage bufferů*. Je jí předán zdroj přerušení a ona následně volá funkce CanBus_Receive a CanBus_Put_RxBuff popsanou dále.

void **CanBus_Put_RxBuff**(tCAN_message *Rx_mess)

Funkce uloží přijatou zprávu do kruhového bufferu a posune ukazatel bufferu. Kruhový buffer pro ukládání příchozích zpráv je použit z důvodu rozdílných rychlostí sběrnice CAN a sériové linky RS 232. Buffer má omezení v počtu přijatých zpráv, při přetečení dochází k smazání nejstarší zprávy.

int **CanBus_Get_RxBuff**(tCAN_message *Rx_mess)

Poslední z uvedených funkcí je opakem předchozí. Vyzvedává zprávy z kruhového bufferu pro výpis na sériovou linku.

Kapitola 5

Závěr

Cílem bakalářské práce bylo navrhnout softwarové řešení analyzátoru sběrnice CAN v nákladních automobilech, autobusech a zemědělské technice. V těchto odvětvích automobilového průmyslu je používán protokol SAE J1939. Podmětem pro tuto bakalářskou práci bylo mé zaměstnání ve firmě DevCom s.r.o., která vyvíjí a vyrábí autodiagnostický systém. V budoucnu chce tento systém rozšířit právě i na oblast nákladních automobilů. Realizace práce byla rozdělena na dvě hlavní části.

Jako první krok bylo nutné sehnat odpovídající studijní literaturu. V případě sběrnice CAN a mikroprocesoru MCF 5213 ColdFire to nebyl velký problém. O sběrnici CAN je napsáno mnoho dokumentů a podpora společnosti Freescale je na vysoké úrovni. Menší problém nastal při shánění materiálů k protokolu SAE J1939, nakonec byly části protokolu uvedené níže zakoupeny firmou DevCom.

Druhou částí práce bylo samotné programování mikroprocesoru. Podstatnou část času věnovaného programování zabralo studium mikroprocesoru a jeho periferií. Zpočátku jsem měl mylnou představu, že 32-bitový MCF 5213 je obdoba mikroprocesorů, s nimiž jsem se seznámil pří svém dosavadním studiu. Zprvu mi také dělalo problém zvyknout si na určité odlišnosti a omezení v používání jazyka C v mikrokontrolérech.

Rozvrhl jsem si postup práce takto. Jako první jsem začal oživovat mikroprocesor a periferii UART. Zde jsem měl největší problém s inicializací vstupů a výstupů. Tento problém jsem vyřešil pomocí programu CFInit, který vygeneruje inicializační kód. Druhá etapa byla oživení periferie FlexCAN. Tuto periferii jsem oživoval pomocí programu PP2CAN, kterým jsem simuloval řídící jednotku automobilu. V průběhu vývoje bakalářské práce jsem se rozhodl řešit ovládání pomocí konzolové aplikace. Pro toto ovládání jsem musel vytvořit stavový automat, který bude vypisovat texty na sériovou linku. V této části bylo nejtěžší odladit průchodnost automatu oběma směry z libovolného místa

v menu. Jako předposlední úkol jsem řešil implementaci hlaviček protokolu SAE J1939 do CAN-ových zpráv. Poslední a největší úkol bylo sjednocení všeho předchozího a zprovoznění analyzátoru pod přerušením. Toto mi zabralo nejvíce času, bylo zde nutné vracet se k odzkoušeným funkcím a upravovat je.

Všechny výše popsané body řešení jsem splnil a výsledný program je možné použít pro přijímání zpráv protokolu J1939 ze sběrnice CAN.

Literatura

[1] **Manuál mikroprocesoru:**
MCF5213 ColdFire® Integrated Microcontroller Reference Manual

[2] **Dokumentace Freescale:** http://www.freescale.com

[3] *Mann, B.:* **C pro mikrokontroléry**
BEN - technická literatura, Praha 2003

[4] **CAN specification:** www.semiconductors.bosch.de

[5] **Dokumentace CAN:** http://www.mcu.cz

[6] **Dokumentace CAN:** http://www.fieldbus.feld.cvut.cz

[7] **Dokumentace CAN:** http://www.elektrorevue.cz

[8] **Dokumentace CAN:** http://www.can-cia.org/

[9] **Dokumentace CAN:** http://hw.cz

[10] **Program a dokumentace PP2CAN:** http://pp2can.wz.cz

[11] **Dokumentace SAE J1939:** http://http://www.sae.org

[12] **SAE J1939:** Recommended Practice for a Serial Control and Communications Vehicle Network

[13] **SAE J1939-11:** Physical Layer-250K Bits/s, Shielded Twisted Pair

[14] **SAE J1939-21:** Data Link Layer

[15] **SAE J1939-31:** Network Layer

[16] **SAE J1939-71:** Vehicle Application Layer

[17] **Dokumentace SAE J1939:** http://www.elbas.cz

[18] **Dokumentace SAE J1939:** http://www.cse.dmu.ac.uk

[19] **Dokumentace SAE J1939:** http://www.vector-cantech.com

[20] **Dokumentace a projekty firmy DevCOM:** http://www.devcom.cz/

[21] **Vývojové prostředí DevC++:** www.bloodshed.net

[22] **Inicializační program CFInit:** http://www.microapl.co.uk

[23] **Dokumentace LaTeX:** http://www.dce.felk.cvut.cz

Literatura

—¡¡¡ ———- ¿¿¿—

Příloha A

Obsah přiloženého CD

K této práci je přiloženo CD, na kterém jsou uloženy zdrojové kódy a použitý volně dostupný software se studijními materiály.

- **Bakalářská_práce.pdf**
- MCF5213RM-1.pdf - Manuál k použitému procesoru.
- PCB058_schema.pdf - Schéma DEMO desky.
- devcpp-4.9.9.2_setup.exe - Vývojové prostředí DevC++
- HerculesSetup.exe - Program pro monitorování sériové linky.

- Adresář **can_analyzator**: Projekt v prostředí DevC++
- Adresář cfinit: Inicializační program CFInit.
- Adresář pp2can_demo: Demo programu k rozhraní USB2CAN.
- Adresář serialwatcher: Program monitorování sériové linky.