České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra Řídicí Techniky

DIPLOMOVÁ PRÁCE

Analyzátor sběrnice CAN pro nákladní automobily

Praha, 2010 Author: František Kořínek

Autor: Bc. František Kořínek
Konzultant: Ing. Pavel Burget.
Katedra řídicí techniky,
Fakulta elektrotechnická,
České vysoké učení technické v Praze
rok: 2010

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 své rodině za pochopení a podporu při studiu.

Prohlášení

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

V Praze dne 11. 5. 2010

podpis

Abstrakt

Tato diplomová práce popisuje návrh analyzátoru sběrnice CAN pro nákladní automobily. Cílem této práce bylo vytvořit prostředek pro diagnostiku zpráv na sběrnici CAN pro protokol SAE J1939. Tento protokol je dnes hlavním standardem v komunikaci řídících jednotek nákladních automobilů, autobusů a zemědělských strojů. Hardwarová část je navržena s procesorem Freescale ColdFire MCF 52259, na kterém je implementován firmware běžící v real-time systému MQX. Hardware filtruje a posílá zprávy ze sběrnice CAN přes ethernet do diagnostické aplikace na PC. Tato aplikace zobrazuje pomocí normy J1939 hodnoty parametrů a stavů na monitoru počítače.

Abstract

This bachelor work describes project of CAN bus analyzer for motor-trucks. The main aim of this study is to create mean for monitoring messages on CAN bus for protocol SAE 1939. This protocol is the main standard in communication of control units in motor-trucks, buses and agricultural machines in these days. The hardware part is designed with Freescale ColdFire MCF 52259 processor, where the fimware is impemented, which runs in reaĺ-time system MQX. The hardware filters and sends messages from CAN bus through ethernet into the diagnostic aplication on PC. This aplication displays by the help of standard J1939 the values of parameters and states in PC display.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. František Kořínek**

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

Název tématu: **Analyzátor sběrnice CAN pro nákladní automobily**

Pokyny pro vypracování:

1. Navrhněte a realizujte desku s procesorem Freescale ColdFire MCF 52239, který obsahuje periferie FlexCAN, USB, Ethernet, UART a je členem rodiny V2 core. Při návrhu počítejte s co největší rychlostí a kapacitou dat přenášených do aplikace na PC (analyzátor dat).
2. Navrhněte strukturu aplikace na PC, která bude pracovat jako analyzátor dat podle protokolu SAE J1939. Požadavkem je schopnost ukládat a zpracovávat data z desky realizované v bodě 1, která se sbírají ze sběrnice CAN v automobilu. Požadavkem je, aby analyzátor byl schopen zobrazovat aktuální hodnoty parametrů a stavů řídicích jednotek automobilu.
3. Realizujte analyzátor na PC v jazyce C#.
4. Proveďte testování v reálném provozu a zhodnocení výsledků. Na základě provedených testů navrhněte možné úpravy a vylepšení.

Práce navazuje na bakalářskou práci Návrh analyzátoru sběrnice CAN a bude řešena u firmy DEVCOM, s.r.o.

Seznam odborné literatury:

MCF52239 ColdFire® Integrated Microcontroller Reference Manual

Vedoucí: Ing. Pavel Burget, Ph.D.

Platnost zadání: do konce zimního semestru 2010/11

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 11. 6. 2009

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

CAN	*Control Area Network*
RTOS	*Real-Time Operating System*
RTCS	*Real-Time TCP/IP Communication Suite*
SAE	*Society of Automotive Engineers*
PC	*Personal Computer*
FEC	*Fast Ethernet Controller*
UART	*Universal Asynchro-nous Receiver Transmitter*
CAN	*Control Area Network*
ECU	*Electronic Control Unit*
PGN	*Parameter Group Number*
MAC	*Medium Access Control*
MAC	*Logical Link Control*
DMA	*Direct memory access*
SMD	*surface mount device*
SMT	*surface mount technology*
API	*Application Programming Interface*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
IP	*Internet Protocol*
I/O	*Input/output*
BSD	*Berkeley Software Distribution*
ABS/ASR	*Anti-lock Brake System/Anti-Slip Regulation*
GUI	*Graphical User Interface*

Kapitola 1

Úvod

1.1 Motivace

Tato diplomová práce vznikla na základě mé spolupráce s firmou DevCom s.r.o., která se zabývá autodiagnostikou a vývojem speciální elektroniky. Navazuji v ní na svou bakalářskou práci, v níž jsem řešil komunikaci po sběrnici CAN, filtrování zpráv a komunikaci s PC přes sériovou linku. V diplomové práci jsem chtěl na vytvořený firmware navázat a využít ho. Nakonec jsem se však rozhodl pro jiné řešení. Tím mám na mysli použití real-time operačního systému MQX RTOS pro navrhovaný firmware.

V diplomové práci již nebylo cílem komunikaci na sběrnici CAN pouze skenovat a ukládat, ale vytvořit komplexní prostředek pro monitoring komunikace ECU v nákladních automobilech a zemědělské a stavební technice. To znamenalo vytvořit hardware schopný komunikovat s ECU automobilu, k němu vytvořit firmware, který by hardware ovládal a nastavoval jeho parametry podle potřeb uživatele, a nakonec vytvořit samotné prostředí, jež by umožňovalo ovládání hardwaru a příjem z něj vyslaných dat.

Prvním bodem mé práce bylo vybrat a vytvořit hardware, na kterém by bylo možné implementovat MQX RTOS. Trend používání real-time systémů je v současné době velmi rozšířený, navíc s jejich využitím roste i efektivita práce. Proto jsem se rozhodl použít mikroprocesor Freescale ColdFire MCF 52259. Spolu s tímto procesorem lze zakoupit demo desku M52259DEMOMCU, která obsahuje procesorem podporované periferie. Tato demo deska je ideálním prostředkem pro seznámení s real-time systémem MQX, s vybraným procesorem i s vývojovým prostředím. Jako vývojové prostředí jsem zvolil Freescale CodeWarrior, podporující MXQ RTOS a dále například funkce pro debbug v reálném čase. Základním požadavkem na hardware je přijímání zpráv ze sběrnice CAN pomocí budiče

sběrnice a přeposílání vybraných zpráv na ethernet. Komunikace prostřednictvím ethernetu je dnes běžná ve všech odvětvích a pro potřeby komunikačního rozhraní mezi navrhovaným hardwarem a PC je jednou z nejlepších variant. Ethernet je dostatečně rychlý a má i dostatečnou kapacitu přenosu dat.

Hlavním požadavkem na PC aplikaci je, aby byla schopna monitorovat data posílaná mezi jednotlivými ECU na sběrnici CAN v reálném čase. Monitoring v reálném čase je ovlivněn reakčními dobami hardwaru analyzátoru, hardwaru PC, firmwaru analyzátoru a PC aplikace, tato omezení jsou však kompenzována vnímáním změny zobrazení u člověka. Pro monitorování dat je využita norma SAE J1939, podle níž se příchozí data přepočítávají a zobrazují na displeji PC podle příslušnosti k jednotlivým ECU. Aplikace je napsána v prostředí Microsoft Visual Studio v programovacím jazyce C#, který dnes patří k nejpoužívanějším programovacím jazykům.

Tato práce zahrnuje všechny činnosti, jimiž bych se v budoucnu rád zabýval, tedy činnosti od vývoje hardwaru přes moderní použití mikroprocesorů až k programování ve vyšších programovacích jazycích. Zvládnutí těchto oblastí je základem k vytváření komplexních elektronických zařízení.

1.2 Autodiagnostika

Autodiagnostika je v současné době neodmyslitelnou součástí automobilového průmyslu. Výrobci automobilů dnes používají různé standardy, fyzickou komunikační vrstvou počínaje a aplikační vrstvou konče. Výrobci osobních automobilů upřednostňují tvorbu vlastních protokolů, protože je to z ekonomického hlediska příliš nezatěžuje. Oproti tomu pro výrobce malých sérií výrobků, jako jsou právě nákladní vozy a další těžká technika, je výhodnější používat jednotný standard.

Pro osobní automobily je od roku 2000 v platnosti soubor norem EOBD(*European On Board Diagnostic*). Většina automobilek již na tyto normy začala přecházet. EOBD obsahuje například normy SAEJ1850PWM, SAEJ1850VPW, ISO9141-2, ISO14230, KWP2000, ISO15765CAN.

Pro nákladní automobily a zemědělskou a stavební techniku zastupuje funkci sjednotitelské normy SAE J1939. Nejedná se ovšem o plně autodiagnostickou normu. Norma popisuje pouze zprávy předávané mezi jednotlivými ECU automobilu. Tyto zprávy obsahují stavy a parametry těchto jednotek, nepodporuje však například čtení závad jednotlivých

ECU na vyžádání aplikace. Což je klasická vlatnost plně autodiagnostických norem. V normě jsou nicméně popsány aktuální stavy a parametry řídících jednotek, proto pro jednoduchost dále v textu uvádím termín autodiagnostika.

Touto normou se ve své práci budu zabývat podrobněji. První část normy popisuje fyzickou vrstvu modelu ISO/OSI. Jedná se o upravenou normu pro sběrnici CAN. Tato sběrnice je dnes jednou z nejrozšířenějších sběrnic v automobilovém průmyslu, proto ji věnuji pár řádků v dalším odstavci.

1.3 Sběrnice CAN

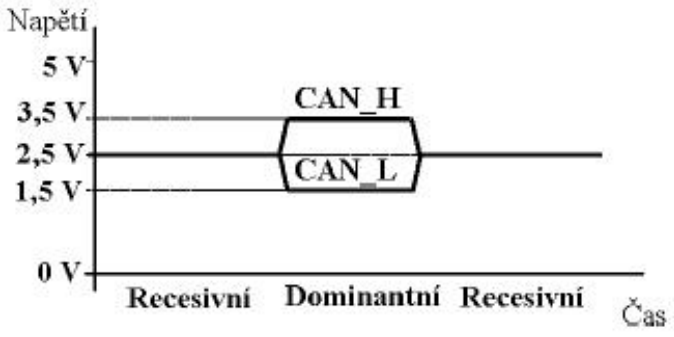
Sběrnice CAN byla vyvinuta firmou Bosch ve spolupráci s firmou Intel v polovině osmdesátých let a postupem času si získala dominantní postavení mezi sběrnicemi určenými pro automobilový průmysl. Zároveň začala být používána i jako průmyslová komunikační sběrnice na nižší systémové úrovni a na úrovni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

1.3.1 Základní vlastnosti

Sběrnice je navržena pro přenos s vysokým stupněm zabezpečení proti chybám s rychlostí přenosu do 1Mbit/s. S její pomocí lze uskutečnit distribuované řízení systémů v reálném čase. Sběrnice je typu *multi-master*, což znamená, že každý uzel v síti může řídit ostatní uzly. Tento typ řízení je jednodušší 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 význam a prioritu zprávy podle identifikátoru, který ji uvozuje. Identifikátorem lze zabezpečit, aby uzel pomocí tzv. *Acceptance Filteringu* přijímal pouze určité zprávy. Této vlastnosti budu ve své práci využívat pro nastavování filtrů pro zprávy protokolu J1939.

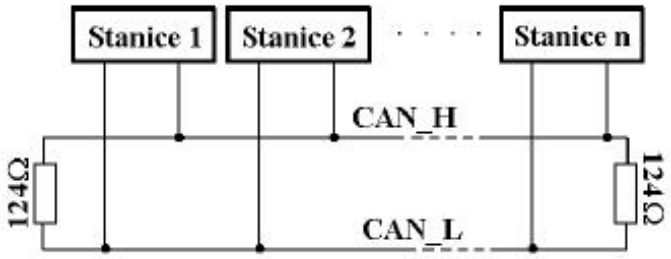
1.3.2 Fyzická vrstva

Fyzická vrstva realizuje elektrické spojení řídících jednotek. Sběrnici tvoří dva vodiče (označované CAN_H a CAN_L), kde je *dominant* či *recessive* úroveň na sběrnici 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 1.1: Logické úrovně sítě CAN.

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



Obrázek 1.2: Schéma sběrnice CAN.

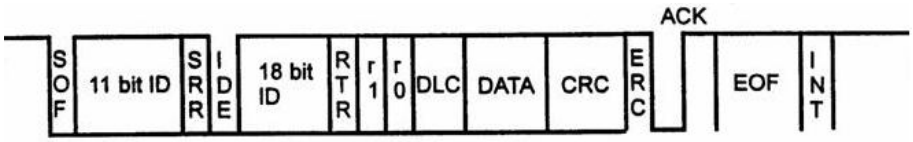
1.3.3 Linková vrstva

Tak jako v modelu ISO/OSI je i v protokolu CAN 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*).

1.3.4 Datová zpráva

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 později a který 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 1.3: 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* formátu
- DLC - délka datového pole (platné hodnoty 0...8)

- ACK - potvrzovací pole - Příjem každé zprávy musí být potvrzen alespoň jedním příjemcem.
- End of Frame - 7 bitů recesivní úrovně (porušení bit *stuffingu*)

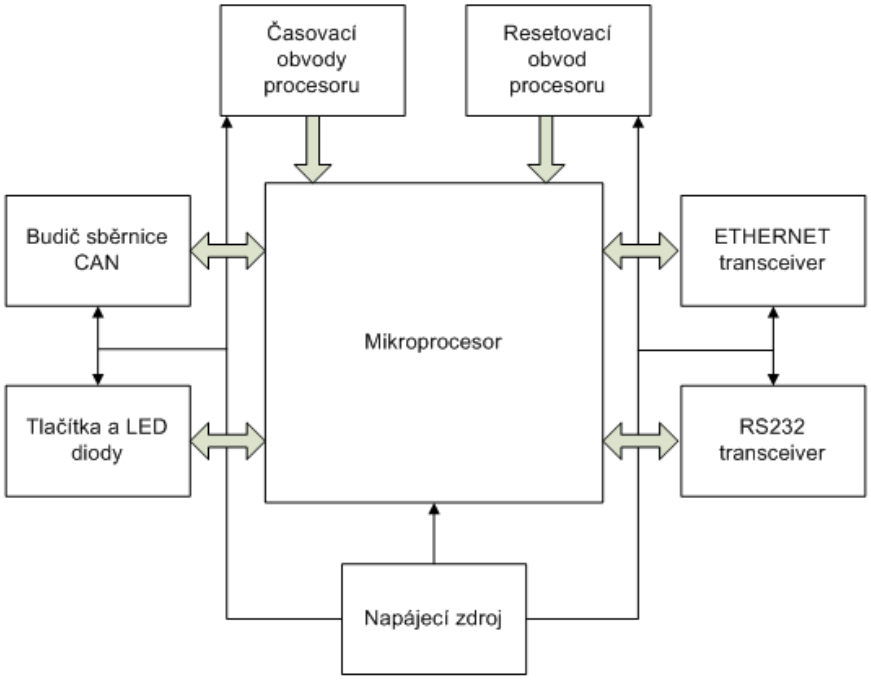
Kapitola 2

Hardware

Deska plošného spoje je navržena pro přenos komunikace ze sběrnice CAN do PC přes ethernet. Hardware je koncipován tak, aby jej bylo možné používat jako samostatné autodiagnostické zařízení napájené ze sítě automobilu bez použití externího napájecího zdroje. K návrhu hardwaru jsem použil návrhový systém OrCAD 16.0.

2.1 Blokové schéma

Základ hardwaru tvoří mikroprocesor Freescale Coldfire MCF 52259. Jako budič sběrnice CAN jsem použil PCA82C250 od společnosti Philips. Jedná se standardně používaný budič plně podporující standard sběrnice CAN ISO 11898. Pro komunikaci po ethernetu jsem se rozhodl použít transceiver KSZ8041NL od společnosti Micrel. Hardware obsahuje také sériové rozhraní RS232 s obvodem MAX3232CSE od společnosti Maxim. Posledním velkým blokem hardwaru je napájecí zdroj. Zdroj je postaven na obvodu LM2575S společnosti National Semiconductor. Všechny tyto hlavní součástky jsou zapojeny podle doporučení výrobce do funkčních celků a vzájemně propojeny na desce plošného spoje v jeden celek. Následující blokové schéma zjednodušeně ukazuje bloky obsažené na desce plošného spoje.



Obrázek 2.1: Blokové schéma hardwaru.

2.2 Návrh schématu

V návrhu schématu jsem vycházel z doporučení vydaných výrobci jednotlivých použitých součástek. Schematický návrh jsem provedl v programu Capture, který je součástí systému OrCAD. Tento program poskytuje velké množství součástek, které je možné použít. Pro součástky, pro něž neexistuje schematická značka, lze tyto značky jednoduše přikreslit. Mnou navržené schéma obsahuje jednotlivé listy odpovídající výše uvedenému blokovému schématu. Tyto listy jsou přiloženy v příloze diplomové práce. Pro přiblížení funkce nejdůležitějších součástek zde uvádím jejich stručné charakteristiky.

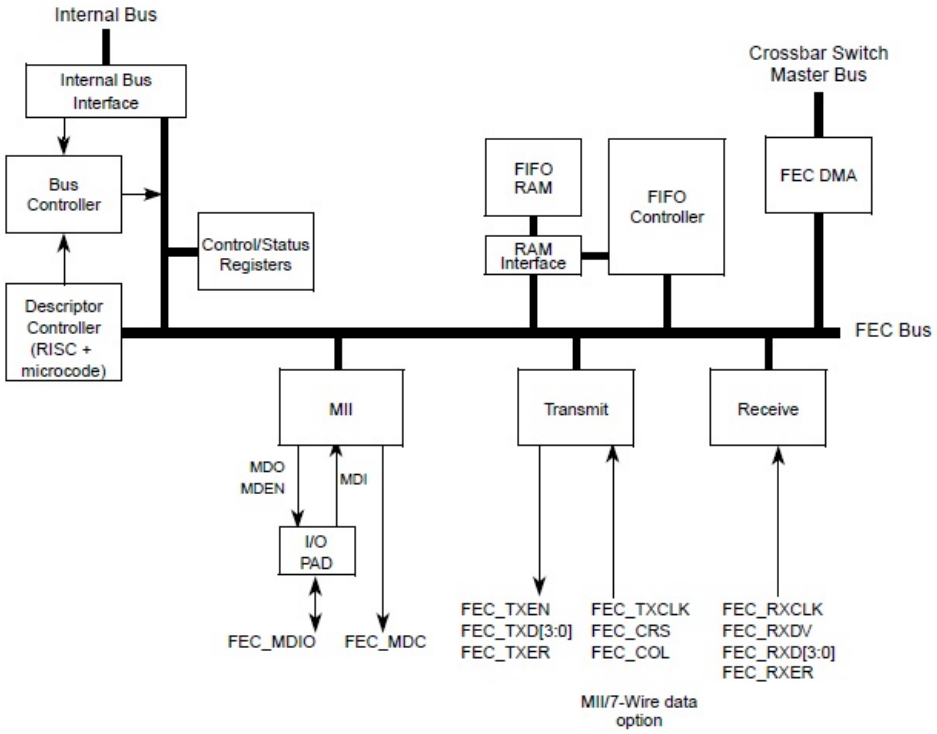
2.2.1 Mikroprocesor MCF 52259

Tento procesor patří do skupiny 32bitových *embedded* (vestavěný, zabudovaný) procesorů. Je to procesor s redukovanou instrukční sadou (RISC). Jeho taktovací frekvence je 80 MHZ. Je vybaven 64KB statické paměti RAM a 512KB paměti Flash. Má v sobě zabudovaný systém výjimek a přerušení, který je nástupcem systému používaného v pro-

cesorech Motorola 68000. Nejdůležitější periferie procesoru pro tuto práci jsou FEC (*Fast Ethernet Controller*), komunikační kontrolér FlexCAN a UART (*Universal Asynchronous Receiver Transmitter*). 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.

2.2.1.1 FEC

Fast Ethernet Controller je nezávislá periferie mikroprocesoru, která zprostředkovává komunikaci přes ethernet. Tento kontrolér podporuje jak 100Mbps, tak i 10Mbps sítě. Plně podporuje duplexní přenos dat s použitím odděleného vedení pro příjem a vysílání dat. MAC vrstva implementovaná v hardwaru obstarává automatický příjem zpráv a zpracovává a kontroluje příchozí pakety (pomocí kontrolního součtu). Prostřednictvím DMA řadiče ukládá data do výstupních bufferů a následně je odesílá na fyzickou vrstvu. Dále uvedu několik základních vlastností této periferie.



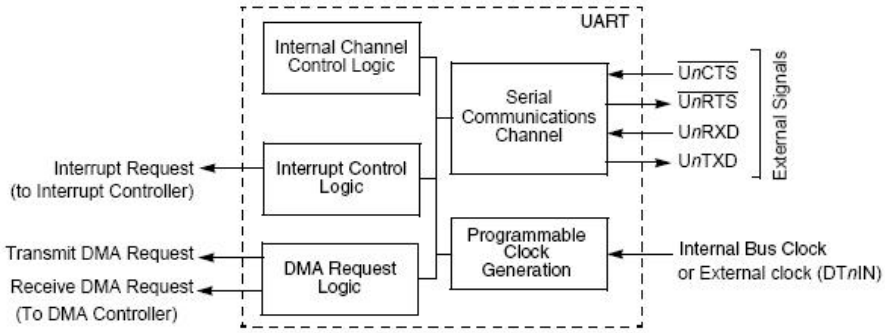
Obrázek 2.2: Blokové schéma periferie FEC.

Základní vlastnosti FEC:

- MAC je navržen pro podporu 10 a 100 Mbps ethernetové sítě (IEEE 802.3).
- Plně duplexní řízení přenosu.
- Podpora plně duplexního přenosu (propustnost dat až 200 Mbps).
- Opakované odesílání paketů z paměti FIFO bez potřeby použití sběrnice procesoru.
- Rozpoznávání adres.
- Vyhrazený DMA řadič, který umožňuje obousměrný přenos dat bez zapojení procesoru.
  - Rámce s plošnou broadcast adresou mohou být buď vždy přijaty, nebo vždy odmítnuty.
  - Přesné porovnání jednotlivých 48bitových individuálních adres (unicast adresy).
  - Kontrola 64bitových hash individuálních adres.
  - Kontrola 64bitové hash skupiny adres (multicast adresy).
  - Sledovací mód umožňuje příjem všech paketů v síti.

2.2.1.2 UART

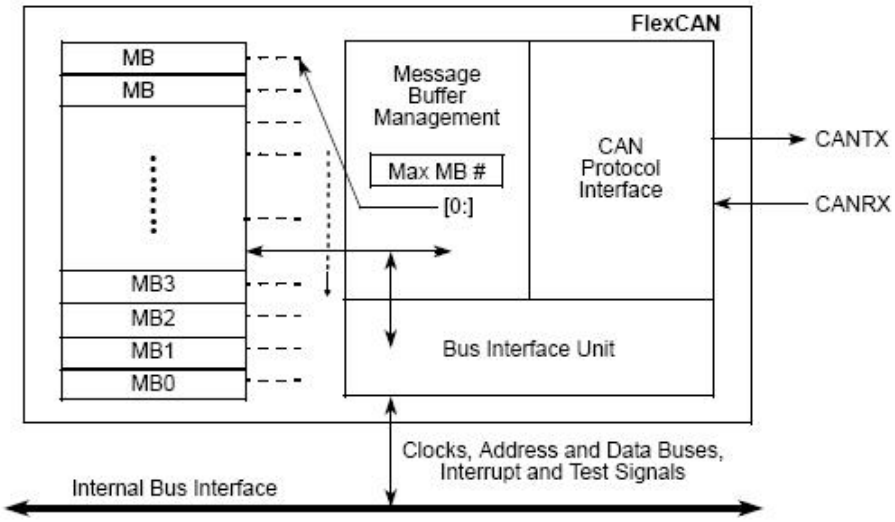
Sériový komunikační kanál zajišťuje full-duplexní asynchronní nebo synchronní přijímač a vysílač s operační frekvencí interní hodinové sběrnice nebo externího zdroje. Vysílač převádí paralelní data z CPU (*Central Processing Unit*) na sériová. Do těchto dat vkládá start bity, stop bity a paritní bity. 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*).



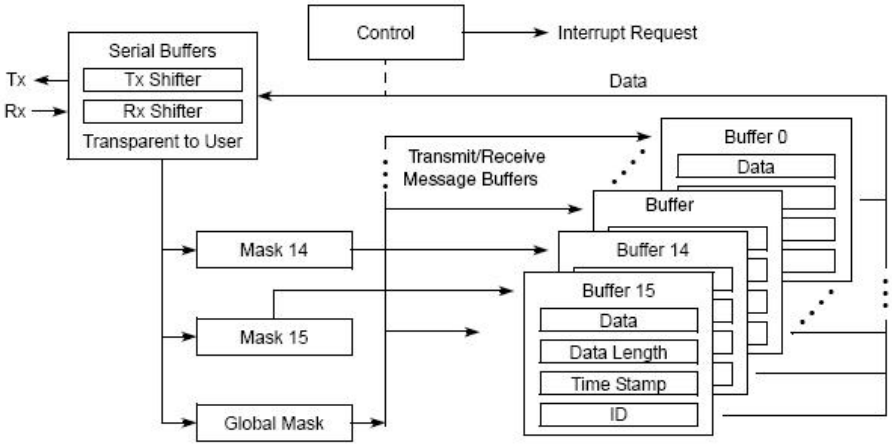
Obrázek 2.3: Blokové schéma periferie UART.

2.2.1.3 FlexCAN

Kontrolér FlexCAN je sériová asynchronní komunikační periferie procesoru, která zprostředkovává vysílání a přijímání zpráv ze sběrnice CAN. Na obrázku 2.4 je blokové schéma celé periferie. Obrázek 2.3 ukazuje strukturu bufferů zpráv (*Massage Buffer*). Struktura jednotlivých bufferů odpovídá položkám zprávy CAN. Navíc je zde vstupní *Serial buffer*. Je to záchytný buffer, který je důležitý při *acceptance filteringu*. Dále struktura obsahuje registry masek a registry přerušení. Podrobný popis funkce těchto registrů lze nalézt v literatuře [1].



Obrázek 2.4: Blokové schéma periferie FlexCAN.



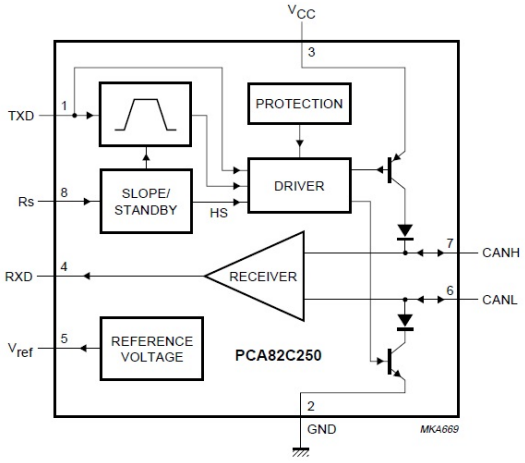
Obrázek 2.5: Blokové schéma architektury bufferů zpráv.

2.2.2 Transceiver KSZ8041NL

Transceiver KSZ8041NL fyzické vrstvy podporuje starší sítě 10BASET i nové sítě 100BASE-TX. Tento obvod je napájen napětím 1.8V a jeho spotřeba je díky technologii CMOS okolo 150mW.

2.2.3 Budič CAN PCA82C250

Budič PCA82C250 je dnes standardně používaný budič sběrnice CAN, plně podporující ISO 11898. Jeho maximální přenosová rychlost je 1 Mbaud. Z blokového schématu níže je patrná celková obvodová funkce.



Obrázek 2.6: Blokové schéma budiče PCA82C250.

2.3 Návrh desky pološného spoje

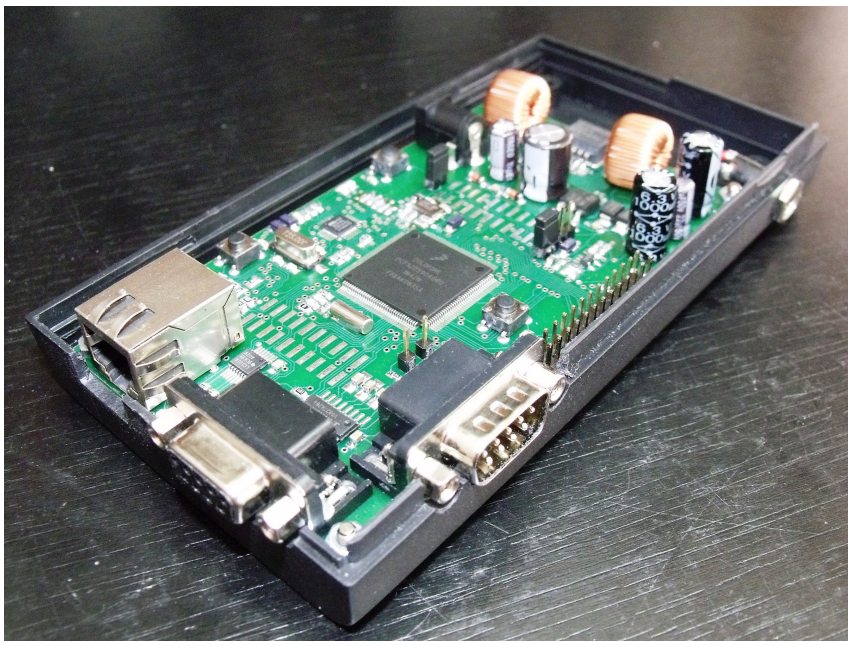
Desku plošného spoje jsem navrhoval v programu Layout Plus systému OrCAD 16.0. Je navržena jako dvouvrstvá deska s použitím klasických vývodových součástek i součástek technologie SMD. Pro rozmísťování součástek technologie SMD jsem použil třídu přesnosti 5. Důvodem použití této vyšší třídy bylo 144vývodové pouzdro procesoru, které je navrženo právě v této třídě. Pro SMD součástky jsem zvolil jednotnou velikost 0805. Volil jsem ji jako kompromis mezi náročností ručního osazování a úsporou místa na desce plošného spoje.

Pro návrh bylo zapotřebí úpravy některých součástek obsažených v knihovnách pouzder systému OrCAD. Při rozmísťování součástek jsem počítal s vlivy rušení a přeslechů a snažil jsem se desku navrhnout co nejoptimálněji. V neposlední řadě jsem počítal i s umístěním desky do krabičky o přijatelné velikosti. Po rozmístění součástek, spojů a prokovů následovalo vygenerování technologických dat a jejich úprava. Takto upravená data jsem poslal do společnosti PragoBoard s.r.o. ke zhotovení desky.

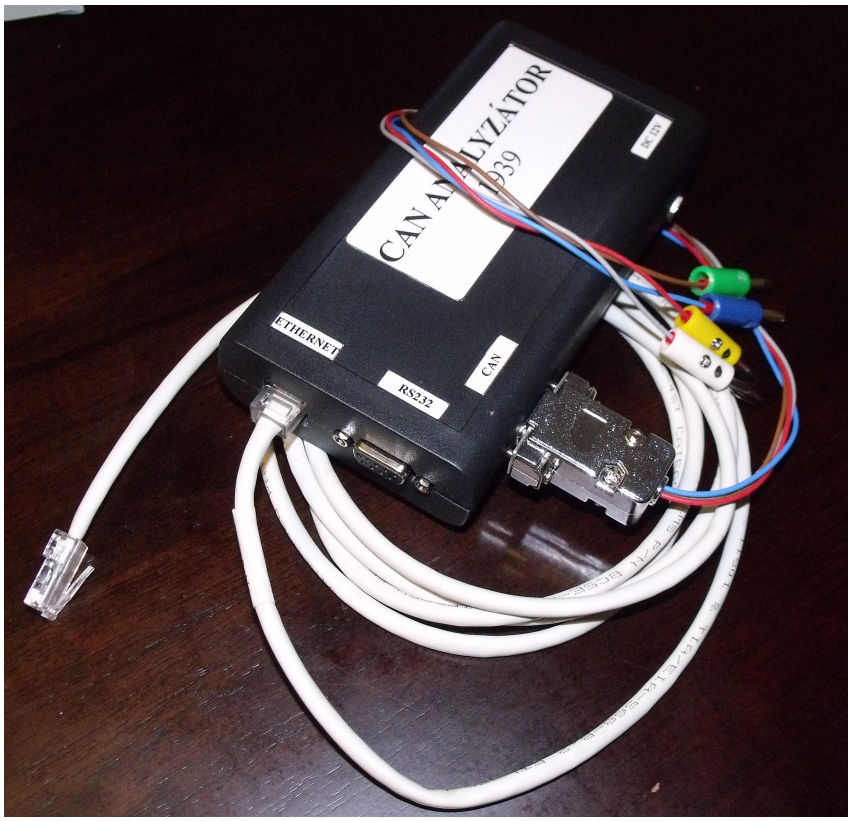
2.4 Osazení a oživení hardwaru

Při osazování desky jsem nalezl několik chyb, které však nebyly natolik závažné, aby nešly opravit. První z těchto chyb bylo otočení diod, které způsobovaly nefunkčnost zdroje napětí. Druhá chyba se vztahovala k BDM konektoru, kde jsem neošetřil signál TMS připojení na +3,3V přes odpor 10 kiloohmů. Dalším větším problémem byla momentální nedostupnost transformátorového oddělovacího konektoru RJ45 pro ethernet. Tento problém je vyřešen použitím jiného konektoru, u kterého jsou propojeny odpovídající vývody pomocí drátků.

Po odstranění těchto chyb v návrhu bylo možné přistoupit k nahrání firmwaru přes BDM konektor vyvedený na desce. Firmware jsem nahrával pomocí vývojového prostředí Freescale Codewarrior 5.9.0, které má v sobě implementovaný ovladač k programátoru PEMICRO MULTILINK. Po nahrání firmwaru do procesoru a otestování funkčnosti všech periferií jsem umístil desku plošného spoje do krabičky. Finální vzhled hardwaru a jeho umístění do krabičky je vidět na následujícím obrázku.



Obrázek 2.7: Osazená deska plošného spoje



Obrázek 2.8: CAN analyzátor J1939 .

Kapitola 3

Firmware

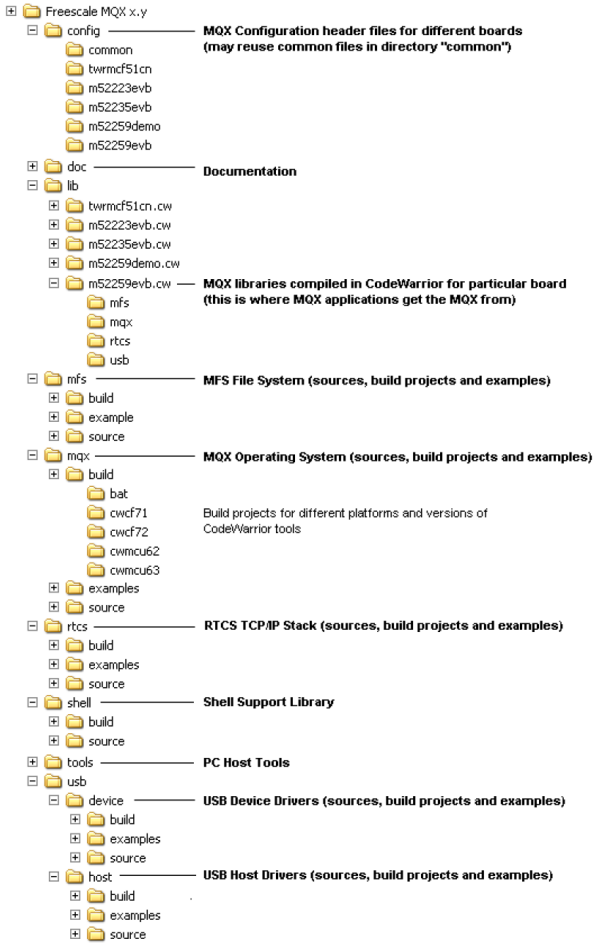
V této kapitole popíšu řešení softwaru implementovaného do výše popsaného hardwaru. Pokusím se teoreticky přiblížit vzniklé problémy a jejich řešení.

3.1 Zvolený způsob řešení

Na počátku práce jsem zvažoval, jakou cestou se vydat. Nejprve jsem chtěl navázat na svou bakalářskou práci, v níž jsem programoval firmware klasicky pomocí hlavní metody main bez pomoci jakéhokoliv operačního systému. Tento postup byl však kvůli nutnosti použití složitější periferie FEC náročný. V této fázi rozhodování jsem se přiklonil k realizaci firmwaru pomocí real-time embedded systému MQX RTOS (*Real-Time Operating System*). Tento systém poskytuje společnost Freescale pro své mikroprocesory s podporou programovacího prostředí Codewarrior. Základní evaluation licence je během prvního měsíce používání neomezená, poté je omezena kompilace na 64K bajtů kódu. Požádal jsem společnost Freescale o poskytnutí plné licence na dobu 6 měsíců, této žádosti bylo bez problémů vyhověno. Jelikož jsem se potřeboval seznámit jak s prostředím Codewarrior, tak se systémem MQX RTOS, zakoupil jsem vývojovou desku M52259DEMOMCU osazenou stejným procesorem, s jakým jsem vyvíjel. Návrh hardwaru a firmwaru jsem prováděl souběžně a směřoval je tak, aby byl vyvíjený firmware co nejlépe přenositelný z DEMO desky na můj hardware.

3.2 MQX RTOS

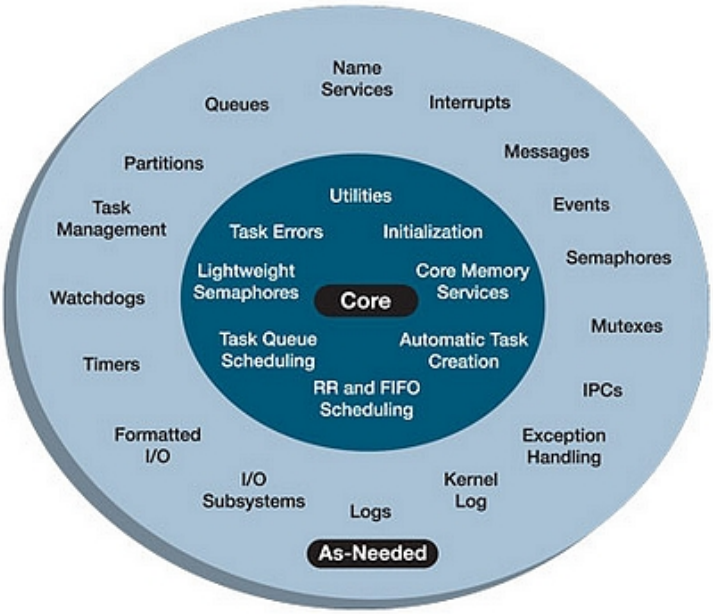
MQX RTOS je jeden z mnoha real-timových systémů pro embedded aplikace. Je navržen tak, aby poskytoval možnost optimální konfigurace pro procesory 8, 16 i 32bitové. Umožňuje nastavení a vyvážení výkonu mikroprocesoru s ohledem na velikost kódu programu. Poskytuje API prostředí, které usnadňuje práci s tímto systémem. MQX podporuje poslední verze mikroprocesorů Freescale ColdFire, pro které poskytuje běžně používané ovladače. Rychlý návrh programu umožňuje prostředí Codewarrior, s jehož pomocí lze snadno používat ovladače k jednotlivým druhům mikroprocesorů. Dále lze v tomto prostředí také pohodlně debuggovat. Obrázek 3.1 zobrazuje strukturu MQX RTOS. Mnou používané části této struktury jsou popsány v následujících kapitolách.



Obrázek 3.1: Struktura MQX RTOS.

3.2.1 Jádro MQX

Jádro MQX RTOS je založeno na moderní architektuře component-based microkernel, která dovoluje uživateli definovat velikost a rychlost vybraných komponent. To je u embedded systémů velmi důležitá vlastnost. Jednotlivé komponenty jsou vidět na následujícím obrázku.



Obrázek 3.2: Komponenty MQX RTOS.

- Small code density - Freescale MQX RTOS je navržen rychlostně a velikostně tak, aby pracoval v embedded systémech co možná nejefektivněji. RTOS zaručuje opravdový real-timový výkon s přepínáním kontextu rutin programu a nízkoúrovňových přerušení. Může být nakonfigurován jako malý s 12KB ROM a 2.5KB RAM na procesoru ColdFire s jádrem V2 s tím, že obsahuje jádro MQX, dva aplikační tasky, jeden lightweight semaphore, přerušovací stack, fronty a paměťový manažer.

- Component-based architecture - Poskytuje plně funkční RTOS jádro s přídavnými volitelnými službami. Obsahuje 25 komponent, z toho 8 komponent je jádrových a 17 volitelných. Volitelné komponenty jsou přidávány pouze podle potřeby použití. Tímto způsobem docílíme minimalizace paměťových požadavků.

- Full and lightweight components - Klíčové komponenty jsou obsaženy ve dvou verzích. V plné a tzv. lightweight verzi. Lightweight komponenty šetří paměť a lze

jimi řídit výkon aplikace. Máme na výběr lightweight semaphores, events, timers, logs a paměťové komponenty.

- Real-time, Priority-based preemptive, multithreading - V RTOS se vlákna vykonávají podle své priority. Jestliže je vlákno s vysokou prioritou připraveno, může během krátkého časového intervalu převzít procesorový čas od vláken s nižší prioritou a být vykonáno. Navíc vlákno s vysokou prioritou může běžet bez přerušení až do svého dokončení. Tento přístup se nazývá priority-based preemptive scheduling.

- Optimized for Freescale architecture - Optimalizovaný kód urychluje klíčové části real-timového systému, jako je přepínání kontextu.

- Scheduling - Freescale MQX TROS šetří čas vývojářům, protože je zbavuje nutnosti obsluhy přerušení a vytváření vlastního plánovacího systému. To je také velmi důležité, požadujeme-li užití komunikačních protokolů jako jsou USB a TCP/IP.

- Code Reuse - Freescale MQX RTOS poskytuje rámec s jednoduchým API rozhraním pro vytváření a organizaci vlastností napříč širokým portfoliem embedded procesorů společnosti Freescale.

- Intuitive API - Díky dostupné dokumentaci a kompletnímu API rozhraní je psaní kódu pro Freescale MQX RTOS velmi rychlé a přímočaré.

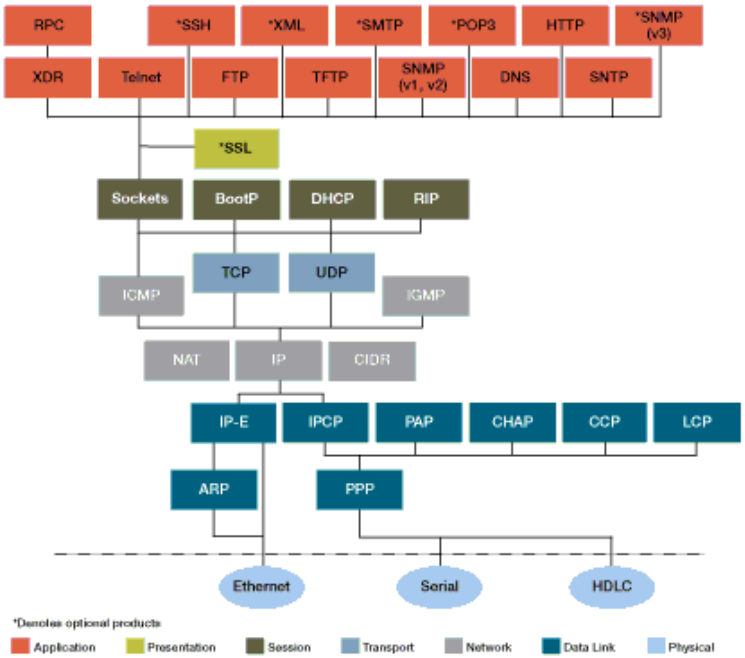
- Fast boot sequence - Rychlá startovací sekvence zajišťuje velmi rychlé rozběhnutí aplikace po restartu hardwaru.

3.2.2 TCP/IP Stack

TCP/IP (*Transmission Control Protocol / Internet Protocol*)je komunikační protokol pro komunikaci po internetu. První část TCP je odvozena od transportní vrstvy modelu ISO/OSI, druhá část IP je síťovou vrstvou tohoto protokolu. Tento TCP/IP stack v sobě však zastřešuje i spoustu dalších komunikačních protokolů odpovídajících dalším vrstvám modelu ISO/OSI. Další podporované protokoly jsou na obrázku níže.

TCP/IP stack je navržen pro použití v multi-taskových systémech jako je MQX RTOS. V těchto systémech ho lze pomocí uživatelského kódu používat ve dvou režimech. První režim je tzv. single-stack (superloop) a druhý multi-stack. Dále jsou uvedeny hlavní vlastnosti TCP/IP stacku.

- Podpora TCP, IP, UDP, ARP, ICMP, CIDR, IGMP, a PPP.
- Podpora protokolů aplikační vrstvy jako jsou DNS, RPC/XDR, BootP, DHCP, HTTP, FTP, TFTP, Telnet, SNMPv1 a SNMPv2c.
- Podpora nízkoúrovňových protokolů jako jsou ethernet (IEEE 802.3) a PPP (obsahující CHAP, LCP, PAP, CCP, a IPCP).
- Je kompatibilní s RFC 1122 (Požadavky pro IPv4 Hosts).
- Je kompatibilní s RFC 1812 (Požadavky pro IPv4 Routers).
- Poskytuje Berkeley Socket (BSD) API a podporuje steam a datagram sockety.
- Podporuje high-performance, re-entrant operace.
- Podpora multihomingu (připojení na několik různých IP sítí) a podpora více zařízení.
- Obsahuje podporu pro konfiguraci síťových parametrů.
- Pracuje s ethernetovými drivery obsaženými v MQX RTOS.
- Podpora dynamicky nastavitelných bran(gateways).
- Poskytuje síťové informace a diagnostiku.
- Je 100% naprogramován v jazyce ANSI C.



Obrázek 3.3: Struktura RTCS protokolů.

3.2.3 CAN API funkce

V MQX RTOS jsou integrovány ovladače k běžným periferiím podporovaných mikroprocesorů. Dále je zde také podpora I/O streamového přístupu. API funkce umožňují urychlení práce s periferiemi procesoru. Lze si je upravovat podle potřeby a aktuálního nasazení v aplikaci. Mnou použité API funkce pro periferii CAN popíšu dále.

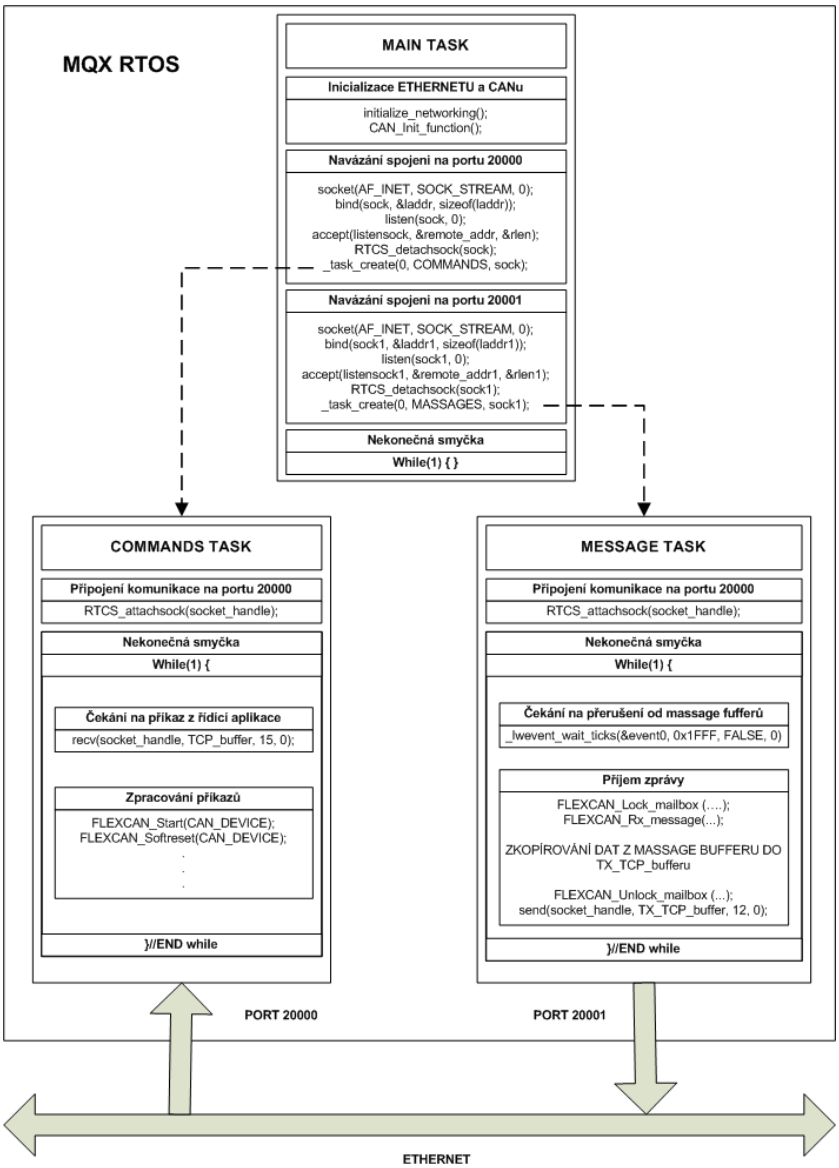
3.3 Prostředky použité pro vývoj firmwaru

Pro vývoj firmwaru jsem použil výše zmíněnou DEMO desku M52259DEMOMCU. Tato deska má osazenu periferii FlexCAN, UART a FEC. Jako vývojové prostředí jsem používal CodeWarrior pro ColdFire verzi 7.1.2 Professional suite. MQX RTOS jsem použil ve verzi 3.5. Tento software byl nainstalovaný na mém laptopu DELL studio XPS 1540 s 32bitovým operačním systémem Windows 7 Professional.

Jako další podpůrný software jsem použil konzoli Hercules pro ladění ethernetového spojení a pro pomocné debugg výpisy. Pro simulaci spojení po sběrnici CAN jsem použil hardwarový převodník USB2CAN a k němu dodávaný software PP2CAN.

3.4 Blokové řešení programu

Na blokovém schématu je znázorněna struktura firmwaru, každý z jednotlivých bloků znázorňuje samostatně běžící task (úkol) programu.



Obrázek 3.4: Blokové schéma firmwaru.

3.5 Popis bloků programu

3.5.1 Inicializace (MAIN TASK)

Po resetu mikroprocesoru a inicializaci MQX RTOS je jako první uživatelský task spuštěn MAIN TASK, který má za úkol připravit hardware pro následující tasky, tyto tasky vytvořit a sám skočit. Jelikož je tento task první (rodič) a vytváří se při spuštění MQX RTOS, je mu přiřazena nejnižší priorita, aby v budoucnu zbytečně nepřerušoval ostatní tasky.

3.5.1.1 Inicializace RTCS (*Real-Time TCP/IP Communication Suite*)

Deklarace funkce ve zdrojovém kódu:

uint_32 initialize_networking(void)

V této funkci jsou nastaveny globální proměnné nutné pro specifikaci funkcí RTCS. Dále je zde vytvořen RTCS task. Jsou zde nastaveny parametry spojení, jako je IP adresa, maska sítě a brána sítě. Následně je získána MAC adresa a inicializován hardware (v tomto případě transceiver KSZ8041NL).

3.5.1.2 Inicializace FlexCAN

Deklarace funkce ve zdrojovém kódu:

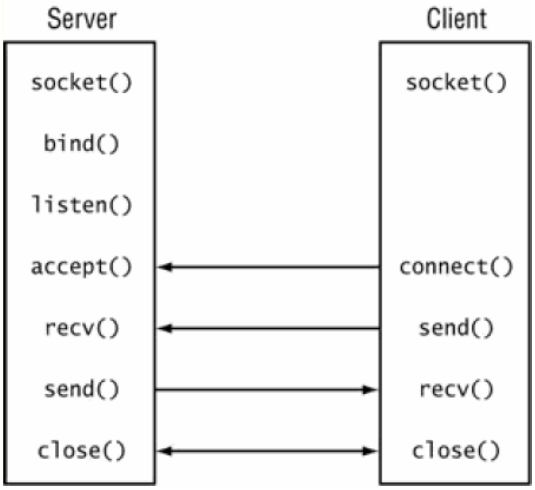
void CAN_Init_function(void)

Inicializací periferie FlexCAN nastavíme počáteční vlastnosti kontroléru. V této funkci se dále volají funkce, kterými nastavíme rychlost sběrnice, počet přijímacích bufferů a masku bufferů, od kterých se má zpracovávat přerušení. Pomocí funkce **_lwevent_create(&event0, LWEVENT_AUTO_CLEAR)** zde vytvoříme strukturální proměnnou, jejímž prostřednictvím MQX RTOS předává informaci, že bylo vyvoláno přerušení.

3.5.1.3 Navázání TCP/IP spojení

Po inicializačních funkcích následuje implementace serverového spojení pomocí Berkeley Socket (BSD) API. TCP/IP spojení je streamového charakteru a s tímto parametrem

musíme také nastavovat funkce. Na následujícím obrázku je zobrazena sekvence volání BSD funkcí.



Obrázek 3.5: Sekvence volání BSD funkcí.

Po navázání spojení, v tomto případě na portu 20000, je volána funkce **RTCS_detachsock(sock)**, která oddělí právě získané spojení od tasku main a která parametry spojení uloží do parametru, s nímž je volána. Následuje volání funkce **_task_create (0, COMMANDS_TASK, sock)**. Tato funkce vytvoří nový task, jemuž předá parametry spojení odebrané tasku main.

Po vykonání této sekvence se táž sekvence vykoná znovu s tím, že server místo čekání na portu 20000 čeká na portu 20001 a pomocí funkce **_task_create()** vytvoří task MESSAGE_TASK s příslušnými parametry portu 20001.

V této chvíli je navázání spojení dokončeno. Na konci aplikačního kódu tasku main je nekonečná smyčka while, která je prázdná, a tento task v průběhu programu dále nic nevykonává. Task main nelze ukončit, protože v něm bylo navázáno spojení a protože jej MQX RTOS potřebuje ke svému běhu.

3.5.2 Nastavení (COMMANDS TASK)

Tento task slouží k přijímání ethernetových příkazů z řídící PC aplikace. Těmito příkazy se nastavuje periferie FlexCAN a ukončení ethernetového spojení mezi hardwarem a PC aplikací. Tomuto tasku je přiřazena nejvyšší priorita mezi uživatelskými tasky, aby bylo možné kdykoli přistoupit k ovládání hardwaru. Komunikace je definována jednoduchým příkazovým protokolem. Tento protokol je uveden v tabulce 3.1.

Po vytvoření tasku je volána funkce **RTCS_attachsock(socket_handle)**, která převezme parametry spojení od tasku main. Po tomto převzetí je možno z COMMANDS_TASK odesílat a přijímat ethernetové zprávy na portu 20000.

Následně program vstoupí do nekonečné smyčky while. První funkcí v této smyčce je BDS funkce **int_32 recv(uint_32 socket, char _PTR_ buffer, uint_32 buflen, uint_32 flags)**, tato funkce je tzv. čekací. Program je na této funkci zastaven, dokud nepřijde nějaká zpráva. První parametr funkce určuje socket, na němž funkce naslouchá, druhý je ukazatel na pole, kam funkce uloží příchozí data, třetí určuje délku přijímaných dat a poslední je pro streamový příjem ignorován hodnotou 0.

Po přijetí zprávy z daného socketu se rozdekódují data podle tabulky 3.1 a ve struktuře switch se rozhodne, která funkce se má vykonat. Po vykonání a odeslání echo zprávy do PC se program opět dostane na čekací funkci **recv(...)**.

Command	Specific	Data	Popis funkce	Funkce
01	-	-	Spuštění FlexCAN v Normal módu.	FLEXCAN_Start (...)
02	-	-	Provedení soft resetu (zachová nastavení některých registrů).	FLEXCAN_Softreset (...)
03	01	-	Nastavení příjmu všech zpráv.	FLEXCAN_Set_global_extmask (...)
03	02	-	Nastavení příjmu pouze filtrovaných zpráv.	FLEXCAN_Set_global_extmask (...)
03	03	4 bajty	Nastavení příjmu zpráv podle rozsahu.	FLEXCAN_Set_global_extmask (...)
04	0-12	4 bajty	Inicializuje ID buffer s číslem Specific na hodnotu Data. Aktivace inicializovaného bufferu.	FLEXCAN_Initialize_mailbox (...) FLEXCAN_Activate_mailbox (...)
05	-	12 bajtů	Vyslání zprávy na sběrnici CAN (4 bajty ID, 8 bajtů DATA)	FLEXCAN_Tx_message (...)
06	-	-	Nastavení Normal módu	FLEXCAN_Select_mode (...)
07	-	-	Nastavení Listen-only módu	FLEXCAN_Select_mode (...)
08	-	-	Nastavení Timer Synchronization módu	FLEXCAN_Select_mode (...)
09	-	-	Nastavení Loop Back módu	FLEXCAN_Select_mode (...)
0A	-	-	Nastavení Bus Off Recovery módu	FLEXCAN_Select_mode (...)
0B	-	-	Nastavení Freeze módu	FLEXCAN_Select_mode (...)
0C	-	-	Nastavení Disable módu	FLEXCAN_Select_mode (...)
0D	-	-	Ukončení komunikace na portu 20000.	shutdown(...)
0E	-	-	Ukončení komunikace na portu 20001.	shutdown(...)
0F	-	-	Ukončení všech tasků a restart MXQ RTOS.	_mqx_exit(...)

Tabulka 3.1: Komunikační protokol

3.5.3 Příjem zpráv ze sběrnice CAN (MESSAGE TASK)

Poslední aplikační task slouží k přijímání zpráv ze sběrnice CAN a jejich okamžitému přeposílání přes ethernet do aplikace na PC. Tento task má střední prioritu, to znamená, že v případě potřeby přenastavení hardwaru může být přerušen.

Po vytvoření tasku je stejně jako u tasku COMMANDS volána funkce **RTCS_attachsock(socket_handle)**, která převezme parametry spojení od tasku main. Po tomto převzetí je možno z MESSAGE_TASK odesílat a přijímat ethernetové zprávy na portu 20001.

Po tomto přiřazení komunikace program opět vstoupí do nekonečné smyčky while. Na začátku této smyčky je umístěna funkce **_mqx_uint _lwevent_wait_ticks (LWEVENT_STRUCT_PTR lwevent_group_ptr, _mqx_uint bit_mask, boolean all, uint_32 tick_timeout)**, která čeká na přerušení. Toto přerušení je nastavováno do strukturální proměnné funkcí **_lwevent_set (LWEVENT_STRUCT_PTR lwevent_group _ptr, _mqx_uint flags)**. Pomocí této funkce je tato proměnná nastavena v rychlé rutině **void FLEXCAN_ISR (pointer)**, kterou jsme při inicializaci CANu přiřadili vektorům přerušení jednotlivých bufferů periferie FlexCAN.

Po vyvolání přerušení od bufferu periferie FlexCAN a nastavení strukturální proměnné se vyvolá čekací funkce **_lwevent_wait_ticks** a povolí další běh programu podle globální proměnné EvenMailbox, v níž je uložen obsah registru IFLAG určující, od kterého bufferu je přerušení vyvoláno. Zavoláme funkci zamykající příslušný přijímací buffer a zavoláme funkci **uint_32 FLEXCAN_Rx_message(uint_8, uint_32, uint_32_ptr, uint_32, uint_32_ptr, pointer, uint_32)**, která přijme data. Tato data jsou dále upravena do jednoho pole bajtů. V této fázi odemkneme zamčený buffer pro další zprávy. Posledním krokem je odeslání upraveného pole přijatých dat po ethernetu do PC aplikace. To se provede funkcí **int_32 send(uint_32 socket, char _PTR_ buffer, uint_32 buflen, uint_32 flags)**, které mimo samotného pole dat musíme předat ještě délku a parametry spojení.

Kapitola 4

PC software

V poslední části diplomové práce jsem naprogramoval PC aplikaci komunikující s navrženým hardwarem. Tato aplikace je schopná nastavovat hardwarové parametry a přijímat zprávy ze sběrnice CAN přes ethernet. Tyto zprávy zobrazuje podle normy SAE J1939 jako parametry a stavy řídících jednotek nákladních automobilů.

4.1 Zvolený způsob řešení

Vstupním parametrem pro mě byla volba programovacího jazyka. Rozhodl jsem se pro programovací jazyk C# a programovací prostředí Microsoft Visual Studio 2008 s použitím NET Framework 3.5.

Aplikaci jsem vyvíjel po částech, nejprve jsem tedy řešil komunikaci po ethernetu. Jelikož firmware je navržen jako server, PC aplikace je navržena jako klient. Dalším krokem bylo vymyslet algoritmus přepočtů, předávání dat a zobrazování na monitor. Nakonec jsem se rozhodl, že pro co nejvyšší jednoduchost a rozšířitelnost programu o nová specifika přepočtových parametrů a stavů z normy SAE J1939 použiji mnou definované inicializační textové soubory pro uložení údajů potřebných pro přepočty a zobrazení. Tyto parametry jsou v průběhu diagnostiky čteny z těchto souborů a dále zpracovávány.

Aplikace je postavena na třech základních funkcích. První je tzv. ON-LINE diagnostika, tato funkce přepočítává a zobrazuje data okamžitě po jejich příchodu ze sběrnice CAN. Druhou funkcí je OFF-LINE diagnostika, tato funkce je naopak schopna přepočítávat a zobrazovat data uložená v souboru. Poslední funkce slouží k ukládání dat do souboru pro následnou analýzu.

4.2 SAE J1939

SAE (*Society of Automotive Engineers*) J1939 obsahuje 9 průběžně doplňovaných a aktualizovaných částí uvedených na obrázku 4.1.

Obrázek 4.1: Model ISO/OSI

- Část J1939/21 - V Data Link Layer jsou definovány obecné vlastnosti 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
  - Frekvence nebo podmínky pro vysílání
  - Uspořádání a kódování datové části

4.2.1 Obecné vlastnosti

- Přenosová rychlost pevně stanovená 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.

4.2.2 Linková vrstva

V této vrstvě jsou definovány obecné komunikační vlastnosti sběrnice CAN. Je zde popsána struktura datových rámců identifikace, transportní protokol pro přenos vícebajtových 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 (*Parameter Group*), které jsou označeny číslem PGN (*Parameter Group Number*).

4.2.3 Skupiny parametrů

Skupiny parametrů v sobě sdružují podobné signály. V normě SAE J1939-71 (*Vehicle Application Layer*) jsou uvedeny skupiny parametrů a v nich obsažené signály. Někteří výrobci si přidávají potřebné specifické parametry. Každá skupina parametrů je definována jedinečným číslem PGN (*Parameter Group Number*). Toto číslo je v identifikátoru zprávy složeno ze dvou 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 (*Electronic Control Unit*). Tento typ vysílání se nazývá *broadcast*. Toto PGM 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*), 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.

4.2.3.1 Struktura identifikátoru

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

Priority 3 Bit	Reserved 1 Bit	Data page 1 Bit	PDU format 8 Bit	PDU specific 8 Bit	Source address 8 Bit

Obrázek 4.2: 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.

4.2.4 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 imobilizéru, teplotu povrchu pneumatik a vozovky nebo laserové navádění tahače na přívěs. Pro lepší využití přenosové kapacity jsou některé parametry sdružovány do skupin. 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 pouze 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.

4.2.4.1 Network management

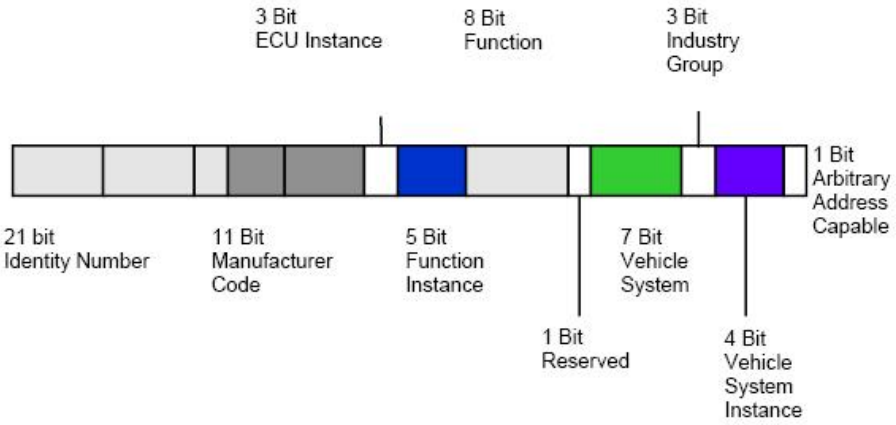
V J1939 má každá síťová jednotka svou unikátní adresu. Každá zpráva, která je jednotkou poslána, obsahuje tuto zdrojovou adresu. Existuje 255 možných adres.

- 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.

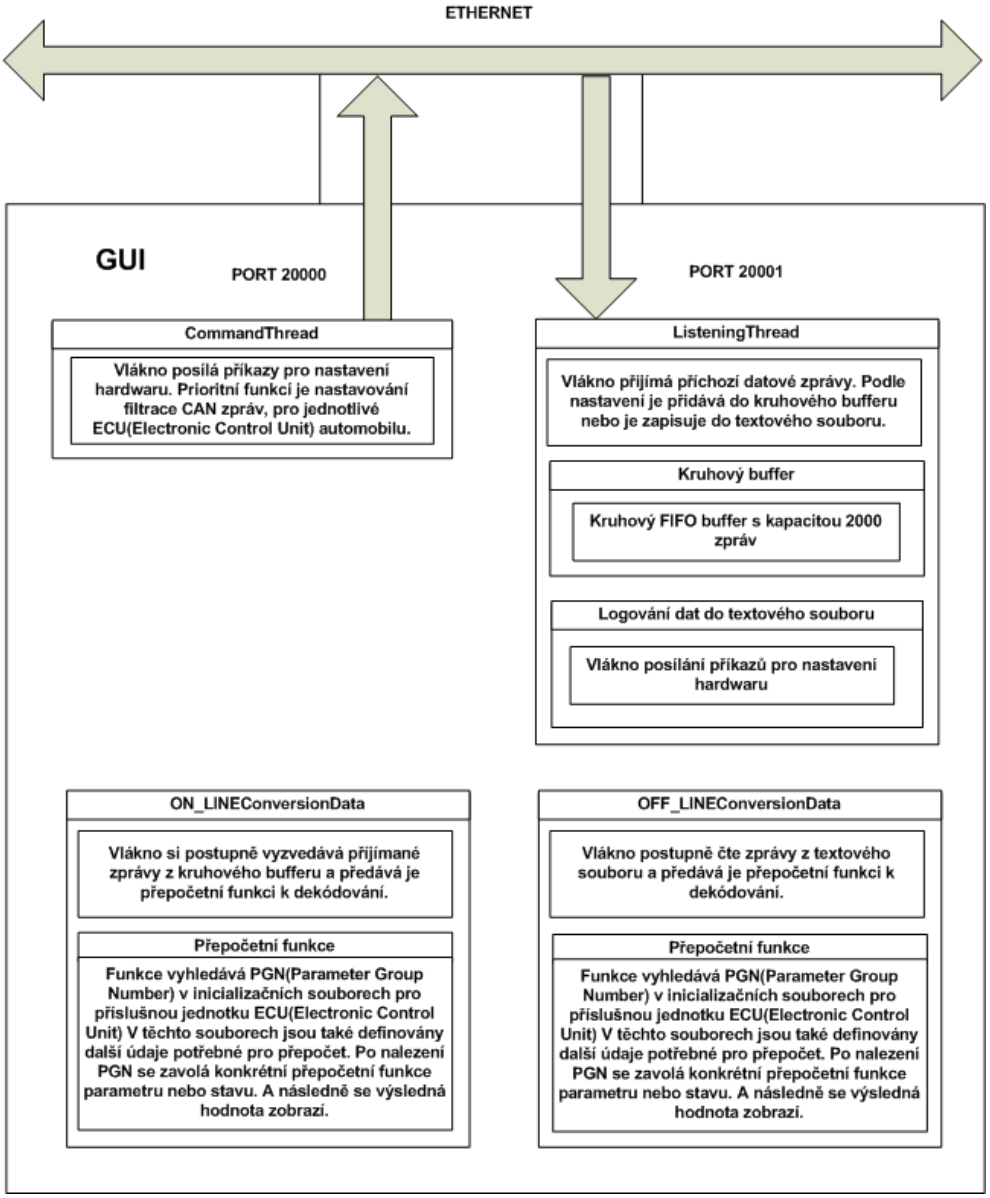
Každý typ zařízení má preferovanou adresu. Dříve než zařízení může adresu použít, musí se zaregistrovat na sběrnici. Tento proces se nazývá *address claiming*. Proto zařízení posílá takzvanou *AddressClaim* skupinu parametrů s požadovanou zdrojovou adresou. Tato skupina parametrů obsahuje 64bitové jméno zařízení. Pokud je adresa již používána jiným zařízením, získá tuto adresu to zařízení, jehož jméno má vyšší prioritu. Jméno zařízení obsahuje informace o zařízení a popisuje jeho funkci.



Obrázek 4.3: Struktura jména zařízení.

4.3 Blokové řešení programu

Na blokovém schématu je znázorněna struktura hlavních částí programu. Jsou zde zobrazena samostatně běžící vlákna, která pracují pod GUI *Graphical User Interface*.



Obrázek 4.4: Blokové schéma PC softwaru.

4.4 Popis a řešení funkcí diagnostiky

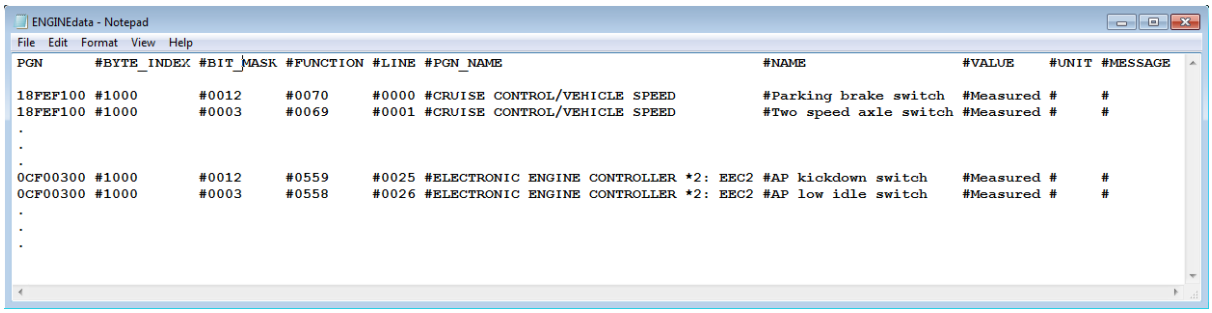
V jednotlivých podkapitolách je vysvětlena funkčnost jednotlivých částí programu a popsáno její softwarové řešení.

4.4.1 Komunikace s hardwarem

Jak jsem již zmínil v kapitole Zvolený způsob řešení, je PC software navržen jako klient. Firmware mikroprocesoru v sobě má implementovaný server, který čeká na připojení klienta na dvou portech, jak je popsáno v kapitole 3.5.1.3 Navázání TCP/IP spojení. Po stisknutí tlačítkové komponenty v prostředí GUI vytvoříme pomocí třídy TcpClient dvě vlákna, která naváží spojení s hardwarem na portech 20000 a 20001. IP adresa hardwaru je pevně dána hodnotou (169.254.3.3). Navázání spojení probíhá ve stejné sekvenci, jakou očekává hardware. Jako první se tedy naváže spojení na portu 20000 a poté na portu 20001. V tuto chvíli je aplikace schopna komunikovat s hardwarem pomocí protokolu podle tabulky 3.1 uvedené v kapitole 3.5.2 Nastavení (COMMANDS TASK).

4.4.2 Předávání a zpracování dat

Program pracuje s velkým množstvím dat, která jsou dána aplikační částí normy SAE J1939/71. Pro lepší práci s těmito daty jsou vytvořeny skupiny podle adresy řídící jednotky, to znamená například motor, převodovka atd. V těchto souborech jsou zapsány parametry pro přepočty a zobrazení jednotlivých parametrů náležících této konkrétní jednotce. V každém souboru se může nacházet několik PGN se svými parametry a stavy. Na následujícím obrázku je příklad inicializačního souboru.



Obrázek 4.5: Příklad inicializačního souboru.

Popis parametrů inicializačního souboru:

- PGN - Porovnává se s PGN přicházejícím ze sběrnice CAN nebo čteným ze souboru. Funkce porovnání čte řádek po řádku a pokud najde stejné PGN, pokračuje se v přepočtu a zobrazení aktuálního parametru. Pokud ne, vezme se další příchozí nebo čtené PGN a proces se opakuje.
- BYTE_INDEX - Určuje pro každý parametr, které bajty datové příchozí zprávy ze sběrnice CAN nebo zprávy čtené ze souboru jsou potřeba pro přepočet aktuálního parametru. Maximální rozsah parametru jsou 4 bajty, jednotlivé znaky toho parametru určují index bajtu příchozích dat (1-8).
- BIT_MASK - Určuje, na kterých bitech bajtu určeného hodnotou BYTE_INDEX je přenášena informace o stavu (hodnota parametru BYTE_INDEX musí pro tzv. bitové parametry indexovat pouze jeden bajt příchozích hodnot).
- FUNCTION - Je to indexem konkrétní přepočetní a zobrazovací funkce. Každé číslo v této položce je jedinečné a určuje ho norma J1939 jako Suspect Parameter Number u každého přepočtu parametru.
- LINE - Je to index pro určení výpisu parametru na správné místo komponenty ListView používané k zobrazování parametrů.
- PGN_NAME - Zde je zapsán název PGN každého parametru. Tato hodnota se do příslušného sloupce komponenty ListView načte pouze jednou.
- NAME - Zde je zapsáno jméno parametru nebo stavu skupiny PGN. Tato hodnota se do příslušného sloupce komponenty ListView načte taktéž pouze jednou.
- VALUE - Hodnota VALUE je pouze pro počáteční vypsání do příslušného sloupce komponenty ListView.
- UNIT - V hodnotě UNIT jsou zapsány fyzikální jednotky příslušného parametru. Tato hodnota se do příslušného sloupce komponenty ListView načte pouze jednou.
- MESSAGE - Hodnota MESSAGE je pouze pro počáteční vypsání do příslušného sloupce komponenty ListView.

4.4.3 ON-LINE Diagnostika

ON-LINE Diagnostika je první funkcí programu, umožňuje čtení dat ze sběrnice CAN a jejich přepočet a zobrazení. Načítání dat probíhá v přijímacím vlákně na portu 20001. Pomocí globální proměnné se v tomto vlákně nastaví ukládání příchozích zpráv do kruhového bufferu, z něhož si pak přepočetní funkce postupně odebírá přijaté zprávy. Tento buffer má nastavenu kapacitu na 2000 příchozích zpráv a po jeho naplnění se nejstarší zprávy přepisují.

Pro omezení přijímaných zpráv je nutné nastavit hardware. Nastavování identifikátorů, lépe řečeno PGN pro periferii mikroprocesoru FlexCAN, probíhá odesláním sekvence zpráv podle protokolu definovaného v tab. 3.1. Pro příjem zpráv je používáno 13 prvních bufferů FlexCAN, z tohoto důvodu lze přijímat pouze 13 skupin parametrů. Tato nastavovací sekvence se vyšle vždy, když přepneme do záložky vybrané ECU. Spolu s vysláním nastavovací sekvence pro ID bufferu se načtou hodnoty z výše popsaných položek inicializačního souboru. V následující tabulce jsou uvedeny řídící jednotky, které diagnostická aplikace podporuje spolu s podporovanými skupinami parametrů.

Po spuštění diagnostického vlákna, přesněji řečeno po nastavení proměnné pro povolení ON-LINE diagnostiky, se v inicializačním souboru provádí hledání stejného PGN, jako je PGN aktuálně vyzvednuté zprávy z kruhového bufferu. Po nalezení tohoto PGN v inicializačním souboru se na stejném řádku vyčtou inicializační parametry popsané v kapitole Předávání a zpracování dat. Tyto parametry (BYTE_INDEX a BIT_MASK) jsou následně předány dekódovací funkci, která s jejich pomocí vybere a přetransformuje data vyzvednuté zprávy z kruhového bufferu. Tato vybraná data jsou poté předána podle inicializačního parametru FUNCTION konkrétní přepočtové funkci. Tato funkce vrací hodnotu, která se podle inicializačního parametru LINE vepíše do komponenty ListView. Po dokončení tohoto cyklu je PGN hledáno na dalším řádku souboru. Tento cyklus se opakuje, dokud není v inicializačním souboru nalezeno jiné PGN. V tomto případě se vyzvedne z kruhového bufferu další zpráva a celý cyklus pokračuje znovu.

Zobrazování dat při ON-LINE diagnostice neprobíhá v důsledku použití kruhového bufferu v reálném čase. Pro lidské oko je však rychlost přijímaných zpráv i tak příliš vysoká a z tohoto důvodu je zapotřebí, přepočty a zobrazování hodnot ještě zpomalit. Toto zpomalení pomůže také snížit procesorový čas přiřazený tomuto vláknu. Zpomalení je docíleno uspáním diagnostického vlákna na 30 milisekund po přepočtení a zobrazení jednoho parametru.

Adresa ECU	**Název ECU**	**Softwerem podporovaná PGN**
00	ENGINE	#CONTROL/VEHICLE SPEED #ENGINE CONTROLLER *1: EEC1 #ENGINE CONTROLLER *2: EEC2 #ENGINE TEMPERATURE #INLET/EXHAUST CONDITIONS #FUEL ECONOMY #ENGINE HOURS, REVOLUTIONS
03	TRANSMISSION	Data nejsou podporována.
0B	ABS_ASR	#WHEEL APPLICATION PRESSURE HIGH RANGE INFORMATION: EBC3 #WHEEL SPEED INFORMATION #ELECTRONIC BRAKE CONTROLLER 1: EBC1 Used for brake control information #COMBINATION VEHICLE WEIGHT
0F	ENGINE_RETARDER	Data nejsou podporována.
10	DRIVE_CHAIN_RETARDE	Data nejsou podporována.
11	CRUISE_CONTROL	Data nejsou podporována.
17	DASHBOARD	#VEHICLE WEIGHT
1D	IMMOBILIZER	Data nejsou podporována.
29	EXHAUST_RETARDER	#ELECTRONIC RETARDER CONTROLLER 1: ERC1
EE	TACHOGRAPH	#TACHOGRAPH: TCO1 #HIGH RESOLUTION VEHICLE DISTANCE
27 30 19	OTHER ECU	#AMBIENT CONDITIONS #DASH DISPLAY #ENGINE FLUID LEVEL/PRESSURE #SUPPLY PRESSURE #CAB MESSAGE 1: CM1

Tabulka 4.1: Softwarem podporované ECU a PGN

4.4.4 OFF-LINE Diagnostika

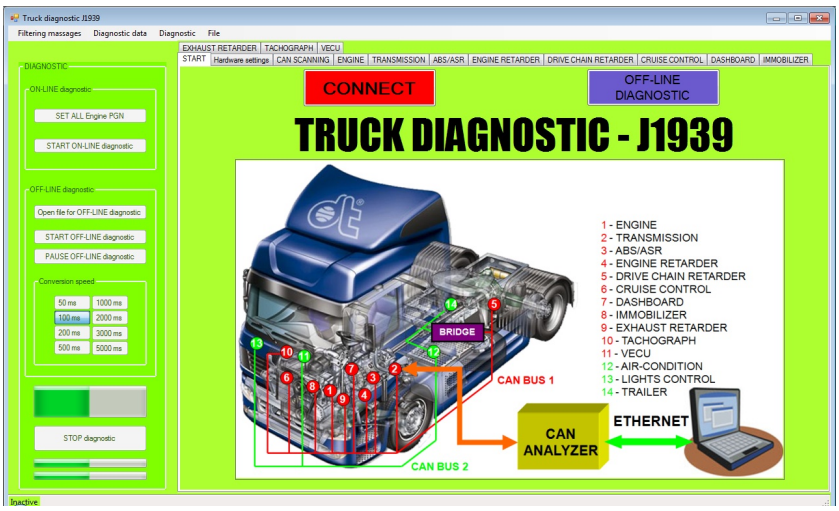
OFF-LINE diagnostika je postavena na stejném principu a s použitím stejných funkcí. Jediný rozdíl spočívá v tom, že se diagnostická data neodebírají z kruhového bufferu, jak je tomu u ON-LINE diagnostiky, ale čtou se z otevřeného textového souboru. U OFF-LINE diagnostiky lze nastavit rychlost přepočtu a zobracování dat. Tato funkce pouze nastavuje délku uspání vlákna. Toto nastavení lze provést v rozsahu 50 - 5000 milisekund. Textový soubor a v něm uložená data musí být ve stejném formátu jako soubory nalogované pomocí tohoto softwaru. Bližší popis následuje dále.

4.4.5 Logování dat do souboru

Logování dat je důležité při práci v terénu, kde nemáme možnost dohledat si všechny informace. Tato aplikace dovoluje ukládání dat čtených ze sběrnice CAN do textového souboru. Logování se provádí v přijímacím vlákně. Po vybrání nebo vytvoření souboru je nastavena proměnná povolující logování. To se provádí pomocí třídy StreamWriter. Přijatá data jsou ukládána v hexadecimálním formátu. Prvních 8 znaků vyjadřuje 4 bajty identifikátoru zprávy a zbylých 16 znaků odpovídá 8 datovým bajtům zprávy.

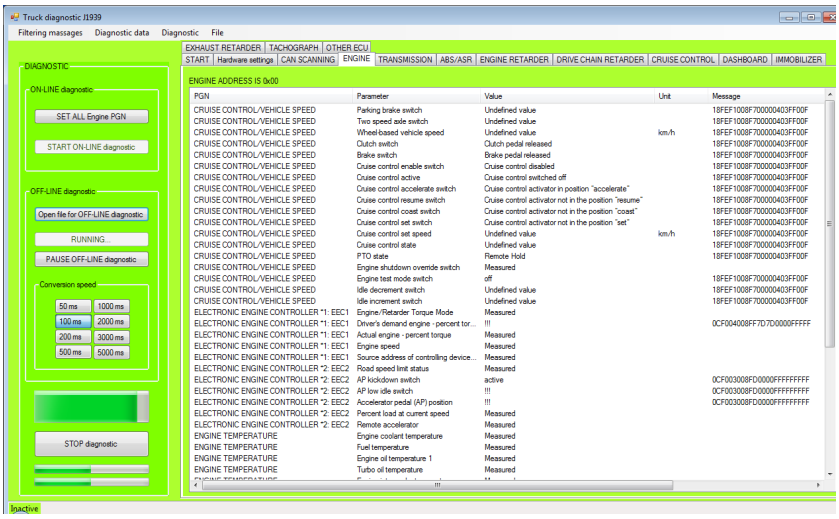
4.5 GUI a ovládání aplikace

Návrh GUI je založen na jednom hlavním okně aplikace a komponentě TabControl. Dále GUI obsahuje hlavní lištu menu a lištu status bar. Po spuštění aplikace se objeví startovací záložka. Zde je možnost výběru pro připojení k hardwaru a následnou práci v on-line režimu, nebo můžeme připojení vynechat a pracovat OFF-LINE diagnostikou.



Obrázek 4.6: PC software po spuštění.

- V případě připojení hardwaru a stisknutí tlačítka CONNECT se aplikace připojí k zařízení a zobrazí se okno diagnostiky motoru ENGINE. Pokud jsou přijímána data, okamžitě začíná přepočet a zobrazení popsané v kapitole 4.4.3 ON-LINE Diagnostika. Indikaci probíhající diagnostiky zajišťuje komponenta ProgressBar.



Obrázek 4.7: PC software - probíhající diagnostika.

- Pokud jsme se rozhodli pro OFF-LINE diagnostiku, můžeme si pomocí tlačítek na levé straně panelu otevřít soubor s diagnostickými daty a tuto diagnostiku provést. Lze pro ni také nastavit rychlost přepočtu a případně ji pozastavit nebo zcela zastavit.

- Pokud klikneme na záložku CAN SCANNING, najdeme v ní okno zobrazující příchozí zprávy. Lze zde také nastavit soubor, do něhož chceme tyto zprávy logovat. Vedle tohoto okna je umístěn jednoduchý textový editor, kde si můžeme zobrazit nalogovaná data nebo upravit inicializační soubor.

- V hlavní liště menu nalezneme kromě tlačítek vykonávajících stejné funkce jako tlačítka v záložkách i položky pro nastavení filtrů pro vybraná PGN. Tato volba slouží k nastavení příjmu pro logování a vizuální skenování.

Kapitola 5

Závěr

Výstupem diplomové práce je komplexní zařízení pro monitorování sběrnice CAN, schopné zobrazovat aktuální parametry a stavy řídících jednotek podporujících komunikaci pomocí normy SAE J1939. Jednotlivé části tohoto zařízení zhodnotím samostatně.

První částí byl hardware, jeho základní vlastností je umožnění komunikace po fyzické vrstvě sběrnice CAN a ethernetu. Jako podpůrné rozhraní obsahuje sériové rozhraní RS232. Hardware je možno napájet přímo z napájecí sítě automobilu pomocí diagnostického konektoru nebo pomocí zdířky pro napájení umístěné v pouzdře zařízení. Pro připojení na sběrnici CAN slouží konektor Cannon 9 a k němu odpovídající protějšek s vyvedenými kabely pro signály CAN Low, CAN High, +12V a GND s použitím banánkových svorek. Toto řešení je dočasné a v budoucnu bude upraveno pro redukce konkrétních automobilů. Při testování hardwaru jsem neobjevil žádné chyby, které by měly vliv na jeho stabilitu a funkčnost v normálních podmínkách. Hardware jsem osadil do plastové ochranné krabičky.

Firmware procesoru umožňuje komunikaci prostřednictvím ethernetového protokolu TCP/IP spolu s přijímáním zpráv ze sběrnice CAN. Ethernetové spojení je využito k nastavování parametrů hardwaru a posílání zpráv přijatých ze sběrnice CAN. V hardwaru je implementován jednoduchý protokol pro komunikaci, která umožňuje nastavovat specifickou funkčnost periferie mikroprocesoru FlexCAN. Například nastavení filtrace CAN zpráv, zastavení a povolení příjmu všech zpráv a uvedení FlexCAN do několika jeho módů. Odesílání zpráv na ethernet se provádí okamžitě po přijetí každé zprávy ze sběrnice CAN, bez použití vyrovnávacího bufferu. Toto umožňuje dostatečná rychlost a datová prostupnost ethernetu. Tato vlastnost je podstatná k tomu, aby nedocházelo ke ztrácení komunikačních dat, k čemuž docházelo při komunikaci přes rozhraní RS232, které jsem používal ve své bakalářské práci. Firmware plní tyto základní specifikace potřebné pro

monitoring dat v reálném čase. Jednou z věcí, které jsem nestihl ošetřit, je korektní ukončení vláken a restartování operačního systému MQX. Celkově však firmware plní funkci přeposílání a filtrace zpráv a je schopen reagovat na nastavovací příkazy.

PC aplikace umožňuje nastavovat vlastnosti hardwaru popsané v mnou navrženém jednoduchém komunikačním protokolu a je schopna přijímat data příchozí ze sběrnice CAN. Jak nastavování, tak příjem probíhá pomocí protokolu TCP/IP. Aplikace přepočítává a zobrazuje hodnoty příchozích zpráv podle normy J1939. Tyto parametry jsou zobrazovány ve skupinách podle řídících jednotek, které tyto parametry vysílají na sběrnici CAN. Zobrazení je prováděno v reálném čase s využitím kruhového bufferu v tzv. ON-LINE módu. Aplikace je dále schopna ukládat příchozí zprávy do textového souboru a následně je využívat pro druhý typ diagnostiky tzv. OFF-LINE mód. Ten umožňuje zobrazení těchto načtených zpráv později, bez připojení hardwaru.

Norma J1939 je dosti rozsáhlá a popisuje okolo 1500 parametrů a stavů členěných do skupin PGN. Do PC aplikace, která s touto normou pracuje, jsem se snažil implementovat co nejvíce PGN. Jako základní jsem si vybral PGN, které obsahovala komunikace získaná firmou DevCom s.r.o. z nákladních automobilů Scania a IVECO. Měl jsem k dispozici několik takových souborů z různých nákladních automobilů, ve kterých jsem dohledal 20 PGN. S těmito daty jsem také prováděl testování přepočtů a zobrazení hodnot. Bohužel jsem neměl zatím možnost testování na nějakém nákladním automobilu, proto jsem testování prováděl pomocí zařízení USB2CAN, které je schopné vysílat zprávy na sběrnici s definovanou periodou určenou normou J1939. K těmto testům jsem však využil reálná data firmy DevCom. Myslím si, že tato simulace je dostačující, nicméně v budoucnu provedu i testy v reálném provozu. Přepočty a zobrazování parametrů a stavů probíhaly v pořádku, nicméně musím konstatovat, že pro velké množství zpracovávaných dat mohlo dojít k chybám. Alespoň z mé dosavadní zkušenosti se zpracováváním diagnostických dat vyplývá, že se tyto chyby vyskytují a také opravují průběžně.

Zjednodušeně lze říci, že zařízení je schopné plnit funkci monitoringu zpráv na sběrnic CAN nákladních automobilů. Po připojení dokáže z nákladního automobilu vyčíst informace a tyto informace zobrazit nebo uložit, což bylo cílem diplomové práce.

Literatura

[1] **Manuál mikroprocesoru:**
MCF52259 ColdFire® Integrated Microcontroller Reference Manual

[2] **Manuál MQX:**
Freescale MQX™ Reference Manual, Rev. 3

[3] **Manuál MQX RTOS:**
Freescale MQX™ Real-Time Operating System User Guide, Rev. 1

[4] **Manuál RTCS:**
Freescale MQX™ RTCS™ User's Guide, Rev. 3

[5] **Dokumentace a projekty firmy Freescale:**
http://www.freescale.com

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

[7] *Kořínek, F.:* **Návrh analyzátoru sběrnice CAN pro nákladní automobily**
Bakalářská práce, Praha 2007

[8] **Hardwarový informační server:**
http://hw.cz

[9] **Informační server o programování:**
http://www.root.cz/

[10] **Informační server o programování:**
http://www.zive.cz/

[11] **Informační server o programování:**
http://www.codeproject.com/

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[30] **Datasheet:** MCF52259 ColdFire Microcontroller, Rev. 0

[31] **Datasheet:** KSZ8041NL MICREL ETH

[32] **Datasheet:** PCA82C250

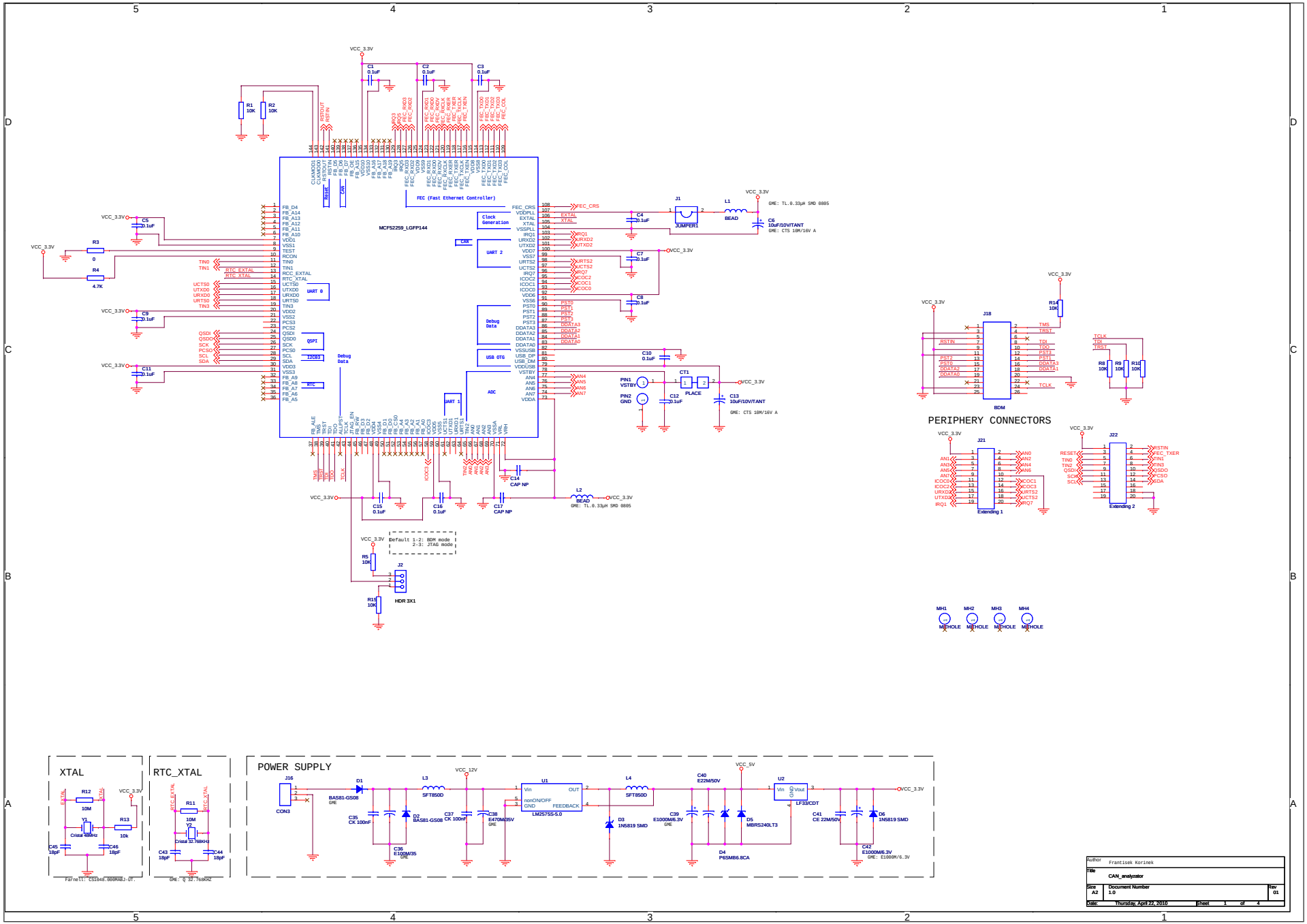
[33] **Datasheet:** KSZ8041NL_MICREL_ETH

Literatura

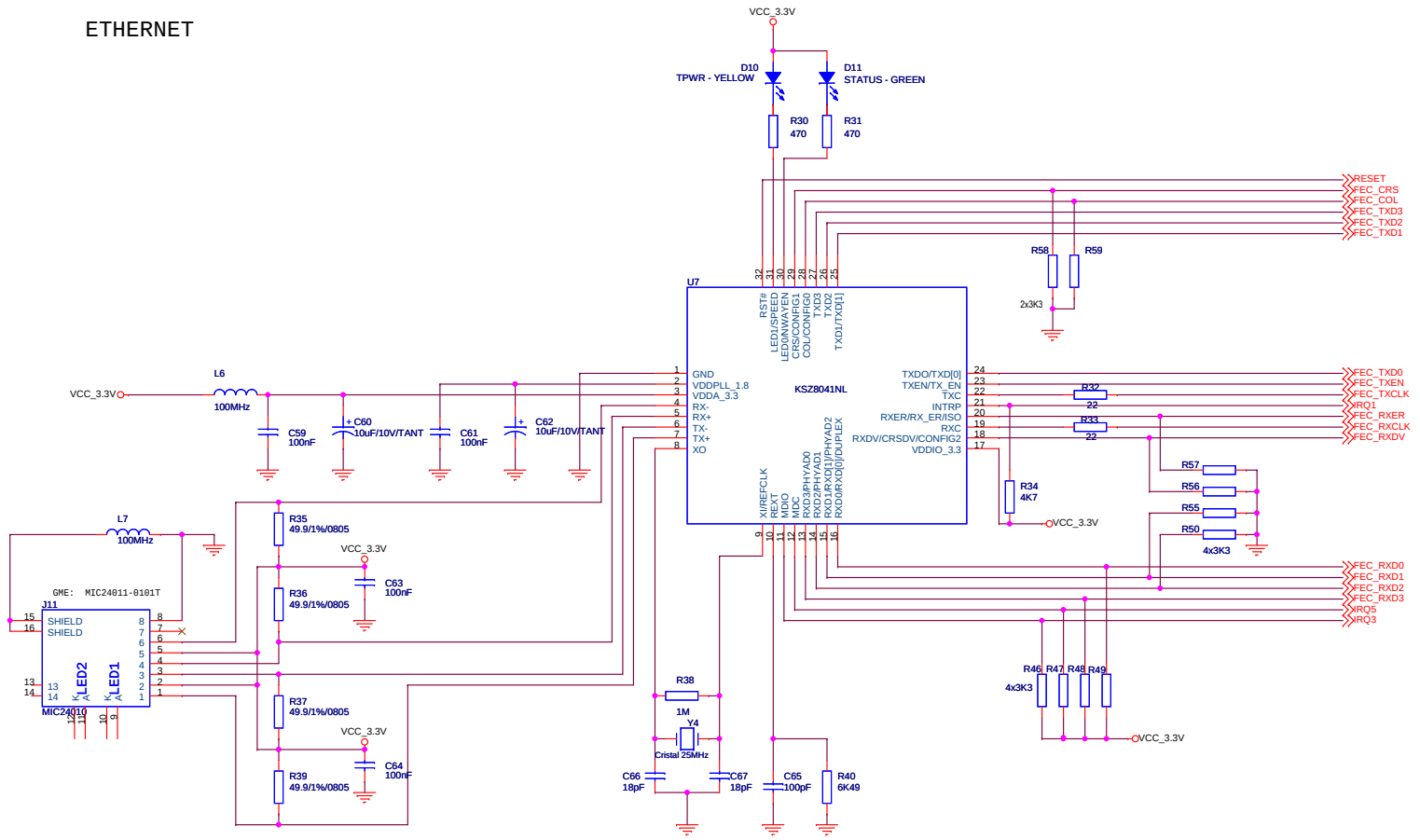
—¡¡¡ ——- ¿¿¿—

Příloha A

Schéma zapojení, osazovací výkres a vrstvy PCB

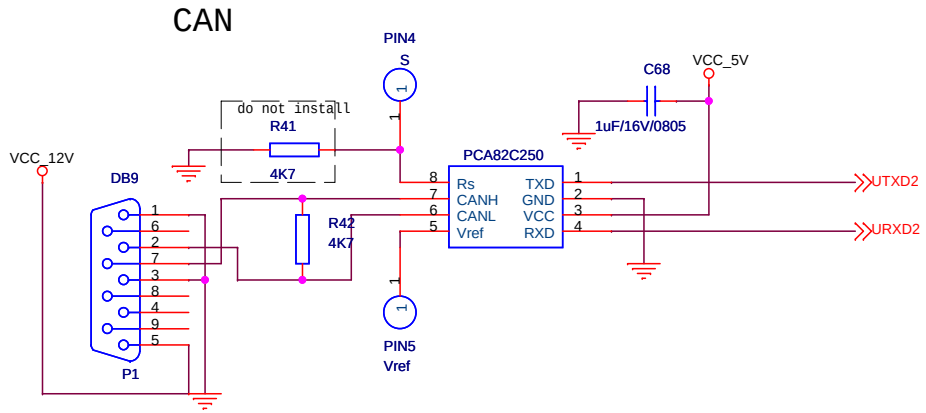

MCF52259_LGFP144
FEC (Fast Ethernet Controller)
Clock Generation
UART 2
UART 0
QSPI
I2C0
RTC
Debug Data
USB OTG
ADC
UART 1
CAN
Reset
CAN
J1
JUMPER1
L1
BEAD
GME: TL.0.33µH SMD 0805
C6
10uF/10V/TANT
GME: CTS 10M/16V A
C13
10uF/10V/TANT
GME: CTS 10M/16V A
CT1
PLACE
PIN1
VSTBY
PIN2
GND
L2
BEAD
GME: TL.0.33µH SMD 0805
Default 1-2: BDM mode
2-3: JTAG mode
J2
HDR 3X1
J18
BDM
PERIPHERY CONNECTORS
J21
Extending 1
J22
Extending 2
MH1
MH2
MH3
MH4
MTHOLE
XTAL
Farnell: CS10x8.000MAB-UT.
RTC_XTAL
GME: Q 32.768KHZ
POWER SUPPLY
J16
CON3
D1
BAS81-GS08
L3
SFT850D
U1
LM2575S-5.0
L4
SFT850D
D3
1N5819 SMD
U2
LF33/CDT
D5
MBRS240LT3
D4
P6SMB6.8CA
D6
1N5819 SMD
C42
E1000M/6.3V
GME: E1000M/6.3V
Author
Frantisek Korinek
Title
CAN_analyzator
Size
A2
Document Number
1.0
Rev
01
Date:
Thursday, April 22, 2010
Sheet
1
of
4

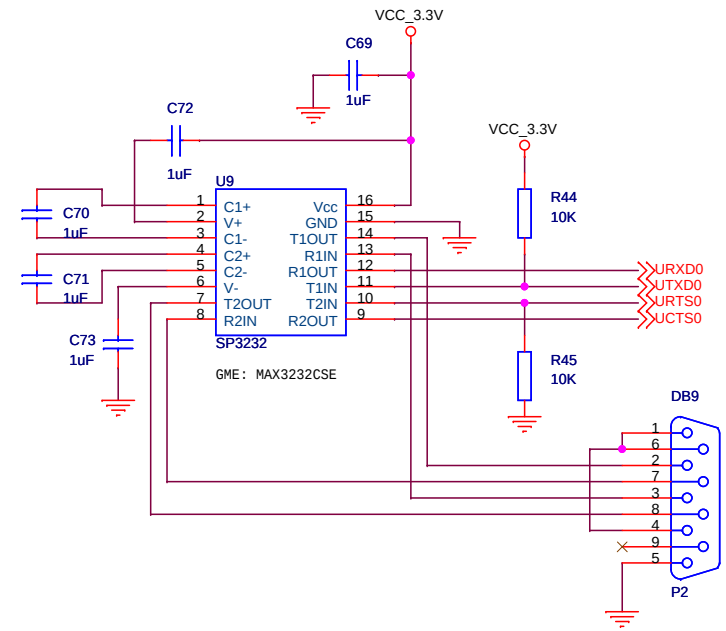
III



Author	Frantisek Korinek		
Title	CAN_analyzator		
Size A3	Document Number 1.0		Rev 01
Date: Thursday, April 22, 2010	Sheet 2 of 4		

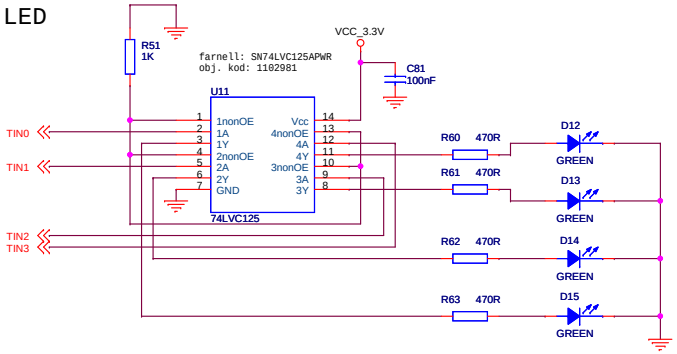
IV

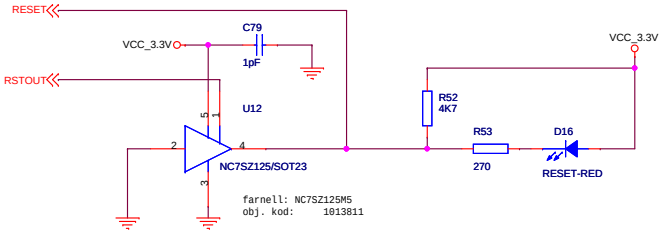




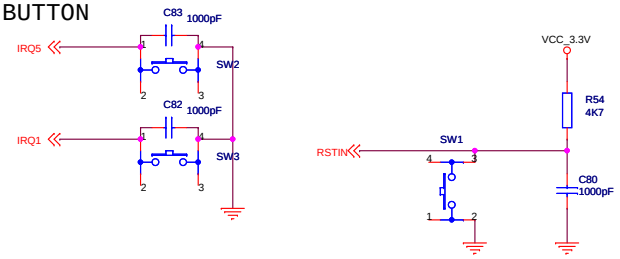
Title	CAN_analyzator	
Size A4	Document Number 1.0	Rev 01
Date: Thursday, April 22, 2010	Sheet 3 of 4	

LED



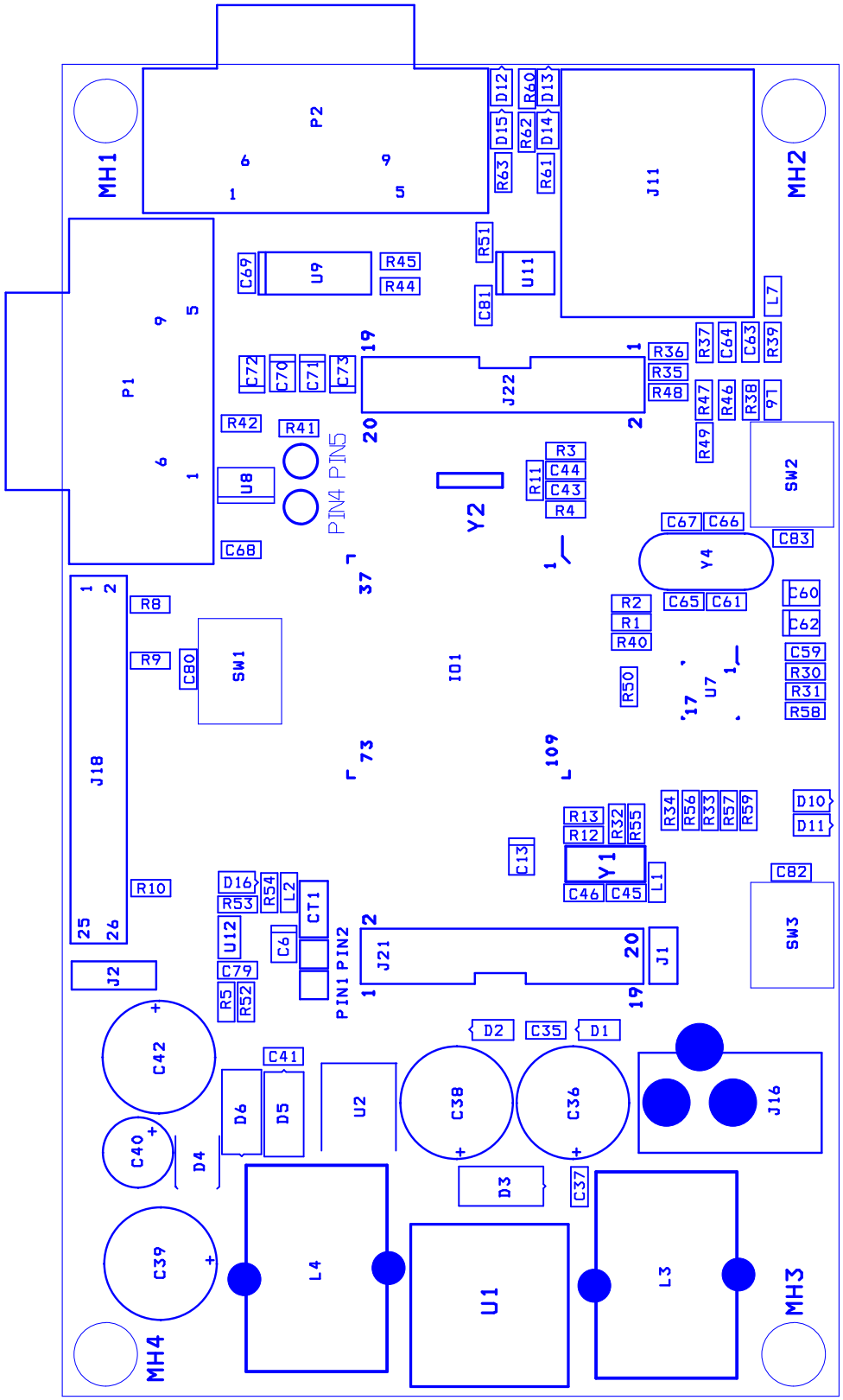


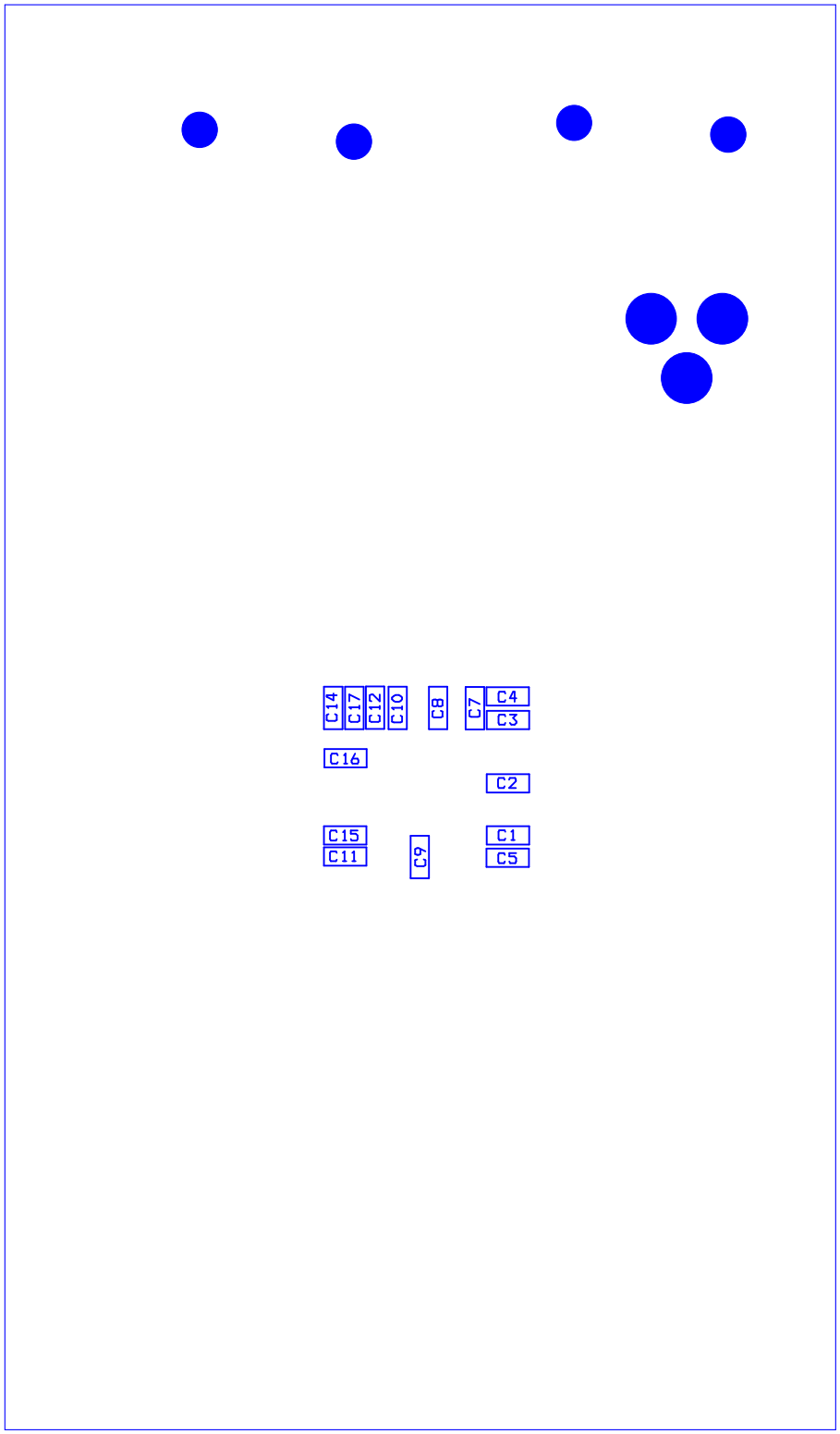
BUTTON

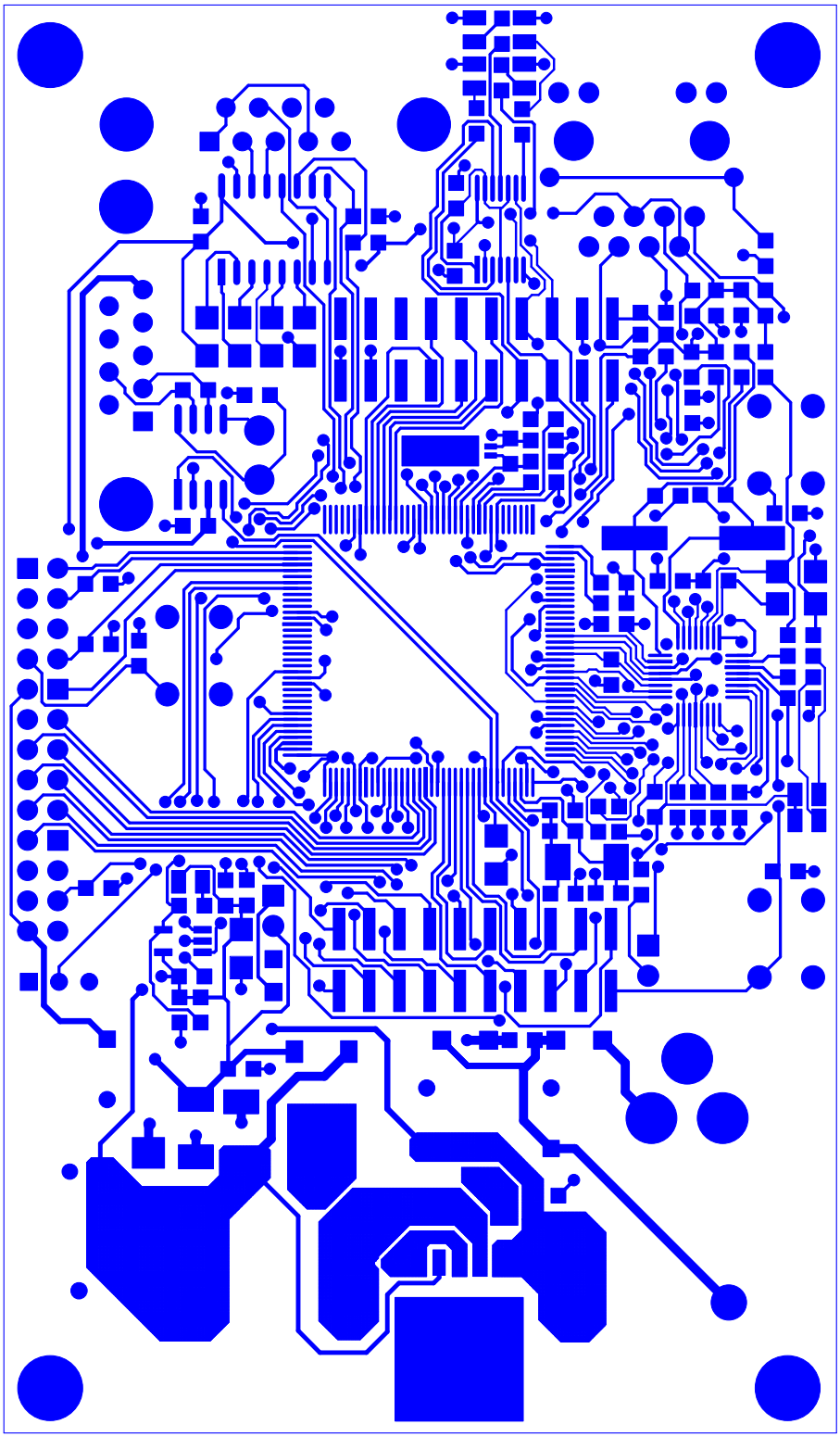


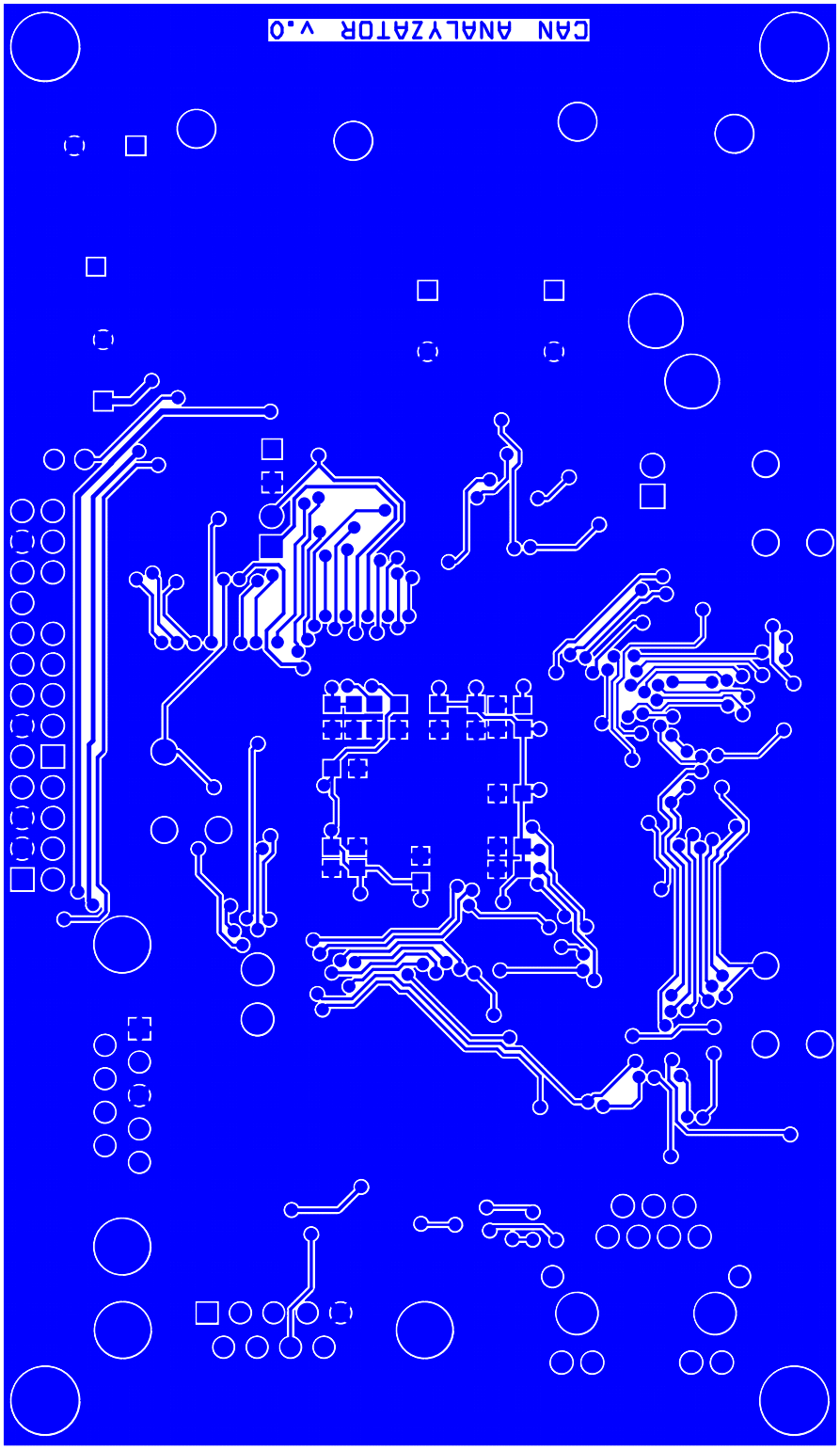
Author	Frantisek Korinek	
Title	CAN_analyzator	
Size A3	Document Number 1.0	Rev 01
Date: Friday, April 23, 2010	Sheet 4 of 4	

V1


MH1
MH2
MH3
MH4
P2
P1
J11
J22
J18
J21
J1
J2
J16
U1
U2
U7
U8
U9
U11
U12
U13
SW1
SW2
SW3
Y1
Y2
Y4
CT1
PIN1 PIN2
PIN4 PIN5
D1
D2
D3
D4
D5
D6
D10
D11
D12
D13
D14
D15
D16
C35
C36
C37
C38
C39
C40
C41
C42
C43
C44
C45
C46
C59
C60
C61
C62
C63
C64
C65
C66
C67
C68
C69
C71
C72
C73
C79
C80
C81
C82
C83
R1
R2
R3
R4
R5
R8
R9
R10
R11
R12
R13
R30
R31
R32
R33
R34
R35
R36
R37
R38
R39
R40
R41
R42
R44
R45
R46
R47
R48
R49
R50
R51
R52
R53
R54
R55
R56
R57
R58
R59
R60
R61
R62
R63
L1
L2
L3
L4
L7
L9
C6
C7d


C14
C17
C12
C10
C8
C7
C4
C3
C16
C2
C15
C11
C9
C1
C5




CAN ANALYZATOR v.0

Příloha B

Obsah přiloženého CD

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

- **Diplomová_práce.pdf**

- Adresář _HARDWARE : Projekt v prostředí OrCAD 16 a manuály k použitým součástkám.
- Adresář _FIRMWARE : Projekt v prostředí CodeWarrior s dokunentací společnosti Freescale.
- Adresář _PC_SOFTWARE : Projekt v prostředí MS Visual Studio 2008.
- Adresář _SUPPORT_SOFTWARE : Volně šiřitelný podpůrný software.