České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídicí techniky

DIPLOMOVÁ PRÁCE

Mobilní CAN Analyzátor

Praha, 2005 Pavel Pičman

Prohlášení

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb. , o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne _____ _____

Pavel Pičman

Poděkování

Děkuji především vedoucímu diplomové práce Ing. Pavlu Růžičkovi za cenné rady, podněty a připomínky.

Dále můj vděk patří všem, kteří mě při práci podporovali i jinak než odbornými radami.

Abstrakt

Tato diplomová práce popisuje návrh a realizaci mobilního analyzátoru CAN sběrnice. Hlavním záměrem bylo vytvořit skutečně mobilní zařízení, které bude mít malé rozměry a dlouhou provozní dobu bez nutnosti externího napájení. Mobilní analyzátor se skládá ze dvou hlavních částí. První část je uživatelské rozhraní pro ovládání analyzátoru a pro zobrazování výstupů z CAN sběrnice. Tato část je řešena pomocí PDA. Druhou částí je elektronické měřící zařízení realizující fyzické připojení ke sběrnici CAN.

Abstract

This diploma thesis describes design and realization of mobile CAN bus analyzer. The most important idea was to create really mobile device, with small dimensions and long operation time without external power source. Mobile analyzer consists of two parts. The first one is user interface for control and display output from CAN bus. I chose PDA as the most suitable device for this part. The second one is electronic measurement device which realizes connection with CAN bus.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní době je sběrnice CAN velmi rozšířena a její obliba díky dobrým vlastnostem a podpory ze strany výrobců nadále roste. Analyzátorů CAN sběrnice je celá řada, od velkých externích zařízení připojených k osobním počítačům přes PCI karty až k PCMCIA kartám do Notebooků. Všechny tyto varianty mají jednu nevýhodu, kterou je velikost jejich výsledného řešení. Problém je zde také s nutností externího napájení, samozřejmě kromě řešení s přenosným počítačem, i když i zde je pracovní doba omezená kapacitou baterií.

Ve své diplomové práci jsem se zaměřil na vývoj zařízení, které by bylo mobilní, to znamená malé rozměry a dlouhou pracovní dobu bez nutnosti napájení. V první části jsem vybíral vhodné zařízení, které by sloužilo jako uživatelské rozhraní. Nakonec jsem vybral PDA s operačním systémem PalmOS. Tato zařízení se vyznačují dlouhou provozní dobou, dostatečným grafickým a výpočetním výkonem.

Druhou částí byl návrh elektronického zařízení, které by komunikovalo s CAN sběrnicí a data posílalo do PDA. Na tuto část byly kladeny nemalé nároky. Především malá spotřeba, malé rozměry a podpora obou standardů CAN protokolu CAN 2.0A a CAN2.0B. Pro komunikaci mezi PDA a měřícím zařízením bylo nutné navrhnout vhodný komunikační protokol pro přenos dat a řídících zpráv z CAN sběrnice.

Kapitola 2

CAN sběrnice

V této kapitole bych chtěl přiblížit funkci protokolu CAN, který je základem mé diplomové práce. Je zde popsáno, kde všude je možné se s tímto protokolem setkat. Dále je zde popsána komunikace jednotlivých zařízení, podoba datových rámců, systém priorit a řešení kolizí.

2.1 Úvod

CAN(Controller Area Network)[7] je sériová sběrnice, která byla původně vyvinuta pro automobilový průmysl na začátku 90. let. CAN protokol byl standardizován v roce 1993 jako ISO 11898-1, je zde popsána linková vrstva z referenčního ISO/OSI[1] modelu. Jedná se o CAN 2.0A, později byl ještě specifikován standard CAN 2.0B, která zavádí dva nové pojmy - standardní a rozšířený formát zprávy. Tyto standardy definují pouze fyzickou a linkovou vrstvu z ISO/OSI modelu, aplikační vrstvu definují vzájemně nekompatibilní standardy např. CANopen a DeviceNet. CAN má dnes přes 40 různých výrobců integrovaných řadičů, díky tomu se také šíří mimo své původní určení.

2.2 Základní vlastnosti

CAN protokol byl navržen tak, aby umožňoval distribuované řízení při přenosové rychlosti sběrnice do 1Mbitu. Velkým kladem této sběrnice je její zabezpečení proti chybám. Jedná se o protokol typu multi-master[8], kde každý uzel sběrnice může být master a řídit tak chování jiných uzlů. Není tedy nutné řídit celou síť z jednoho

[1] ISO (International Standard Organization) definuje 7 vrstev pro síťové protokoly a distribuované aplikace

nadřazeného uzlu. Tato vlastnost přináší zjednodušení řízení a zvyšuje spolehlivost (při poruše jednoho uzlu může zbytek sítě pracovat dál). Pro řízení přístupu k médiu je použita sběrnice s náhodným přístupem, která řeší kolize na základě prioritního rozhodování. Výhodou tohoto přístupu je, že při kolizi nedojde k zastavení veškeré činnosti na sběrnici, ale přenese se jen zpráva s nejvyšší prioritou. Po sběrnici probíhá komunikace mezi dvěma uzly pomocí zpráv:

- datová zpráva (data frame transmission),
- žádost o data (remote transmission request, RTR),
- síťová zpráva (signalizace chyb, pozastavení komunikace), zajištěná pomocí dvou dalších speciálních zpráv (chybové zprávy a zprávy o přetížení).

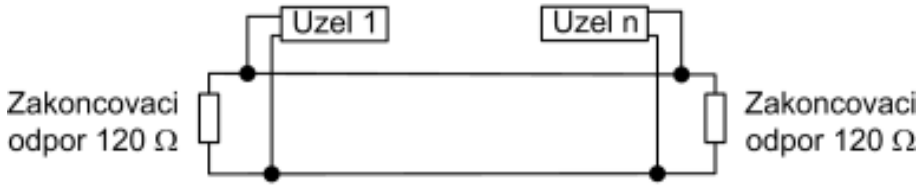
Zprávy, které se pohybují po sběrnici CAN neobsahují žádné údaje komu jsou určeny a jsou přijímány všemi uzly na sběrnici. Každá zpráva je uvozena identifikátorem, který udává význam přenášené zprávy a její prioritu viz kap. 2.6 a 2.7.

2.3 Fyzická vrstva

Protokol CAN definuje vlastní rozhraní k fyzickému přenosovému médiu a v tomto směru se odlišuje od modelu ISO/OSI. Vlastnosti fyzické vrstvy jsou velkou předností protokolu CAN. Základním požadavkem na fyzické přenosové médium protokolu CAN je, aby realizovalo funkci logického součinu. Za účelem zvýšení rychlosti a odolnosti proti rušení je účelné, aby spoj byl symetrický. Standard protokolu CAN definuje dvě vzájemné komplementární hodnoty bitu na sběrnici (*dominant* a *recessive*). Jedná se o obdobu klasických logických úrovní, jejichž hodnoty nejsou určeny a skutečná reprezentace záleží na konkrétní realizaci fyzické vrstvy.

Pravidla pro stav na sběrnici jsou jednoduchá a jednoznačná. Vysílají-li všechny uzly sběrnice *recessive* bit, pak na sběrnici je úroveň recessive. Vysílá-li alespoň jeden uzel *dominant* bit, je na sběrnici úroveň *dominant*. Příkladem může být optické vlákno, kde stavu *dominant* bude odpovídat stav svítí a *recessive* stav nesvítí.

Pro realizaci fyzického přenosového média se nejčastěji používá diferenciální sběrnice definovaná podle normy ISO 11898. Tato norma definuje jednak elektrické vlastnosti vysílacího budiče a přijímače, tak zároveň principy časování, synchronizaci a kódování jednotlivých bitu. Sběrnici tvoří dva vodiče označované CAN_H a CAN_L. Hodnota na sběrnici je definována rozdílem napětí těchto dvou vodičů. Pro eliminaci odrazu na vedení je sběrnice na obou koncích přizpůsobena zakončova-



Obrázek 2.1: Fyzické uspořádání sítě CAN podle ISO 11898

cími odpory o velikosti 120Ω. Jednotlivá zařízení jsou na sběrnici připojena pomocí konektoru, nejčastěji jsou používány konektory D-SUB.

Na sběrnici může být teoreticky připojeno neomezené množství uzlů. S ohledem na zatížení sběrnice, pro zajištění správných statických i dynamických parametrů, uvádí norma jako maximum připojených 30 uzlů. Norma uvádí pro přenosovou rychlost 1Mbit/s maximální délku sběrnice 40m. Pro jiné přenosové rychlosti délku sběrnice norma neudává. Maximální délky sběrnice pro jinou přenosovou rychlost než 1Mbit/s uvedené v tabulce 2.1 jsou pouze informativní a závisí na mnoha parametrech (např. typu použitého kabelu).

Přenosová rychlost [kbit]	**Délka sběrnice [m]**
1000	40
500	112
300	200
100	640

Tabulka 2.1: Maximální délka sběrnice pro různé přenosové rychlosti

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.

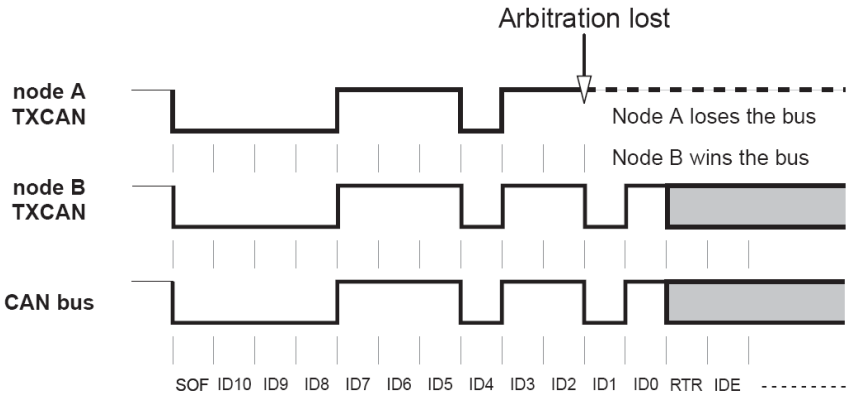
- LLC (Logical Link Control) je podvrstva řízení datového spoje, což zde znamená filtrování přijatých zpráv (Acceptance Filtering) a hlášení o přetíženích (Overload Notification).

2.4.1 Řízení přístupu ke sběrnici

Vzhledem k tomu, že se jedná o síť typu multi-master, každý z účastníků může zahájit vysílání, jakmile je připraven a síť je v klidovém stavu (bus free). Kdo přijde první, ten vysílá. Ostatní mohou vysílat až poté, co je zpráva odvysílána. Vyjímku tvoří chybové rámce, které se dají vysílat okamžitě po identifikaci chyby kterýmkoli účastníkem.

Zahájí-li vysílání současně několik uzlů, pak přístup na sběrnici získá ten, který přenáší zprávu s vyšší prioritou (zpráva s nižším identifikátorem). Identifikátor je uveden na začátku zprávy viz. 2.6. Každý vysílač porovnává hodnotu právě vysílaného bitu s hodnotou na sběrnici a zjistí-li, že na sběrnici je jiná hodnota než kterou vysílá (jedinou možností je, že vysílač vysílá *recessive* bit a na sběrnici je úroveň *dominant*), okamžitě přeruší další vysílání viz obrázek 2.2. Tím je zajištěno, že zpráva s vyšší prioritou bude odeslána přednostně a že nedojde k jejímu poškození. To by mělo za následek opakování vysílání zprávy a zbytečné prodloužení doby potřebné k přenosu zprávy. Uzel, který nezískal při kolizi přístup na sběrnici musí vyčkat až bude sběrnice opět v klidovém stavu a pak zprávu vyslat znovu.

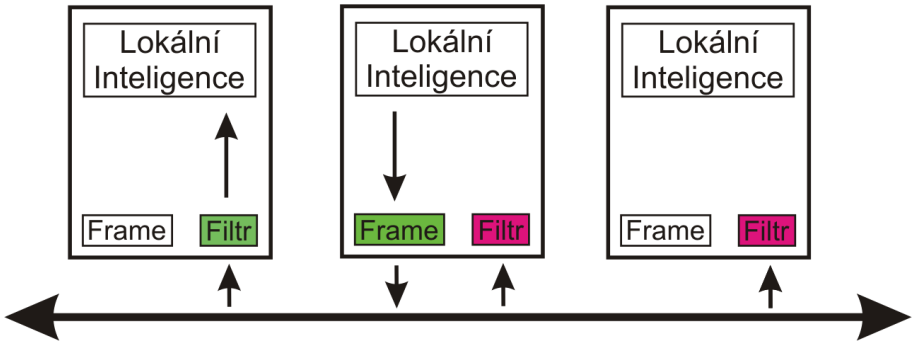
Komunikace je vidět na obrázku 2.3, jeden uzel vysílá zprávu a zbylé uzly poslouchají zda na sběrnici není zpráva, která je obsažená v jejich vstupním filtru. Pokud ano, zpráva je přijata a předána dále k vyhodnocení.



Obrázek 2.2: Řešení priorit přístupu ke sběrnici CAN

2.4.2 Dekódování bitů

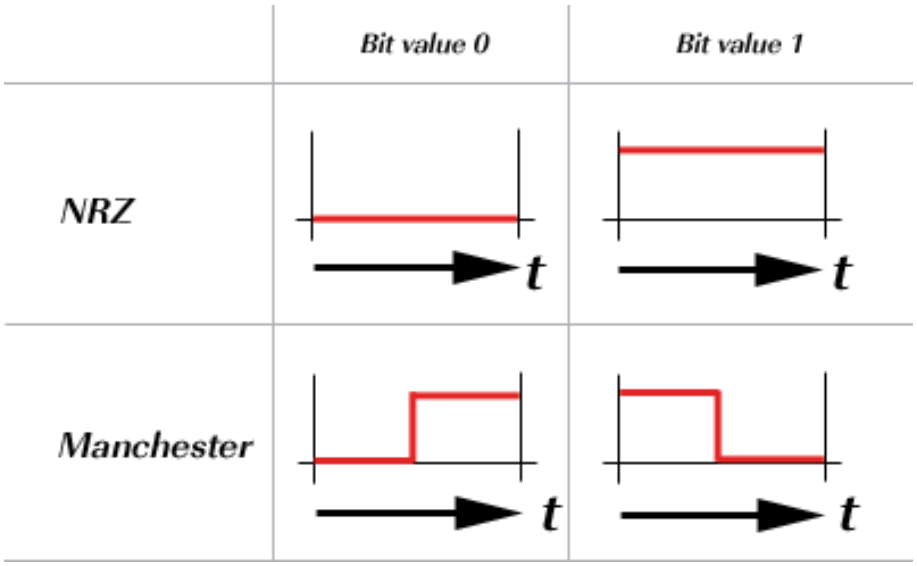
Při použití Non-Return-to-Zero (NRZ) bitového kódování signálu, zůstává hodnota bitu po celý jeho čas stejná. Jednotlivé bity jsou pak rozlišovány podle časových slotů a je nutná časová synchronizace. Jiné metody dekódování bitů jsou např. Manchester. Při přenosu zpráv, může zůstat sběrnice dlouhou dobu na jedné hodnotě



Obrázek 2.3: Komunikace na sběrnici CAN

bitu. Proto se zavádí opatření, které musí zabezpečit maximální dovolený interval mezi dvěma hranami signálu. Toto je důležité pro účel synchronizace.

Bit stuffing je metoda, která vloží bit s komplementární hodnotou za každých 5 stejných po sobě jdoucích bitů. Samozřejmě při přijímání dat se tento bit zase vymaže, aby se neztratila hodnota posílaných dat.



Obrázek 2.4: Porovnání NRZ s Manchester kódováním bitu

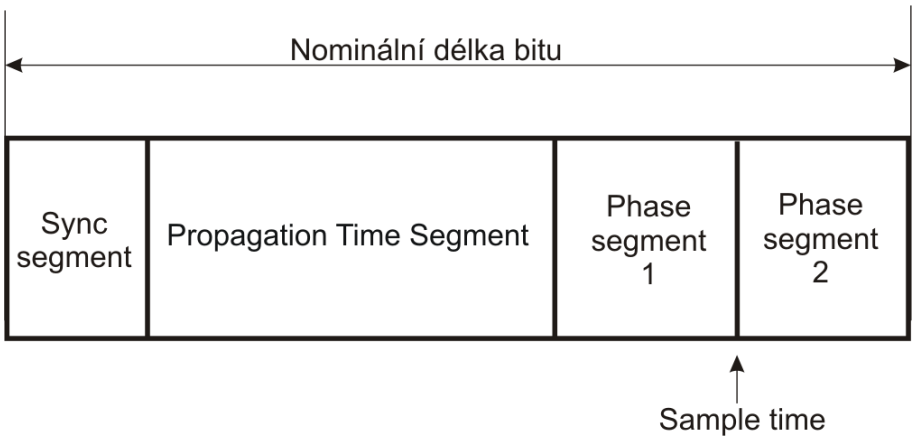
2.4.3 Časování a synchronizace bitů

CAN sběrnice používá synchronní posílání bitů zprávy. Tato vlastnost sebou přináší nutnost používat kvalitní metody synchronizace. Při asynchronním přenosu se znakovým principem posílání dat, dochází k synchronizaci na začátku každé zprávy pomocí start bitu. Pokud se používá synchronní režim je prováděna synchronizace pouze na začátku vysílání prvních dat. Proto pro bezchybné čtení dat je nutné provádět resynchronizaci během přenosu každé zprávy.

CAN protokol řeší synchronizaci přenosu na začátku zprávy pomocí sestupné hrany *SOF bitu*(start of frame) a následně na každé změně dat z *recesive* na *dominant* hodnotu. Konstrukce bitu CAN zprávy je vidět na obrázku 2.5. Délku bitu tvoří čtyři nepřekrývající části (segmenty), každý segment je tvořen násobkem nejmenšího časového úseku (Time Quantum), který používá CAN zařízení.

- ***Synchronization segment*** – jedná se o první časový úsek použitý pro synchronizaci různých CAN uzlů.
- ***Propagation Time Segment*** – tento časový úsek je použit ke kompenzaci signálových zpoždění na sběrnici. Zpoždění je dáno jednak šířením signálu a také zpožděním jednotlivých převodníků u CAN uzlů.
- ***Phase Segment 1*** – tento úsek slouží ke kompenzování fázových poruch hran. Během resynchronizace se může prodlužovat.
- ***Sample Time*** – okamžik, ve kterém se přečte hodnota ze sběrnice a vezme se jako platný údaj.
- ***Phase Segment 2*** – tento časový úsek slouží ke kompenzování fázových poruch hran. Během resynchronizace se může naopak o proti prvnímu zkracovat.

Výsledkem resynchronizace může být zkrácení první části (Phase Segment 1) a prodloužení druhé části (Phase Segment 2). Tento případ nastane pokud například vysílací uzel má pomalejší oscilátor než přijímací uzel.



Obrázek 2.5: Stuktura jednoho bitu na CAN sběrnici

2.4.4 Zabezpečení přenosu zpráv

Protokol CAN se vyznačuje velkou spolehlivostí a bezpečností. V následujících bodech jsou popsány kontrolní mechanismy, které se používají:

- ***Kontrola odesílaných dat*** – každý vysílač porovnává bity, které vysílá na sběrnici a zpětným čtení porovnává hodnoty. Pokud zjistí rozdíl, následují dvě možnosti.
  V první případě nastal rozdíl v době vysílání *Arbitration Fields*(doba vysílání identifikátoru zprávy), viz 2.6, nebo *ACK Slot* (potvrzovací zpráva). Znamená to, že zároveň na sběrnici vysílá také jiné zařízení zprávu s větší prioritou. Uzel následně vysílání přeruší a bude čekat na uvolnění sběrnice.
  Ve druhém případě došlo k chybě dat a vygeneruje se *bit error*.

- ***Kontrolní CRC kód (Cyclic Redundancy Check)*** – je umístěn na konci každé zprávy a vypočítává se ze všech předcházejících bitů, které už byly odvysílány. Tento kód se u přijaté zprávy znovu spočítá a porovnají se hodnoty. Pokud jsou rozdílné, vysílá kterýkoliv uzel na sběrnici signál *CRC error*.

- ***Vkládání bitu (Bit stuffing)*** – zvyšuje bezpečnost a pokud přijímací zařízení zjistí chybu přenosu, je vygenerována chyba vkládání bitu viz. 2.4.2.

- ***Kontrola správnosti zprávy (Frame Check)*** - každá přijatá zpráva se kontroluje podle specifikace, jestli je daná hodnota na tomto místě možná či nikoliv. Pokud je detekována chybná hodnota je generována chyba rámce (formátu zprávy).

- ***Potvrzení přijetí zprávy (Acknowledge - ACK)*** - každý uzel, který se nachází na sběrnici musí korektně přijatou zprávu potvrdit. Tuto aktivitu dělají i uzly, které zprávu nemají ve svém vstupním filtru. Potvrzení se dělá změnou bitu ACK z *recessive*(vysílaná hodnota) na *dominat* bit.

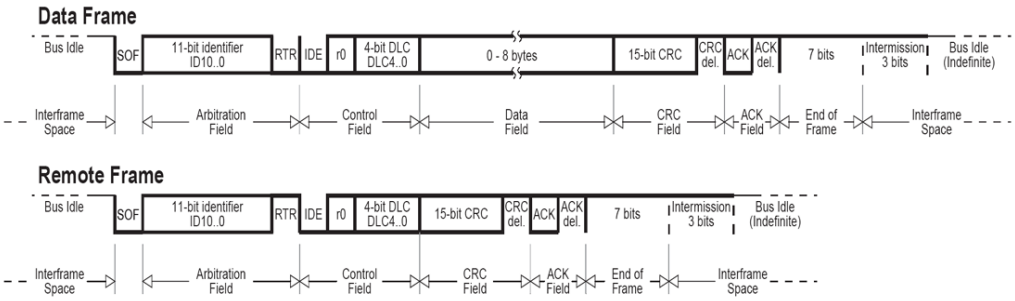
2.4.5 Struktura CAN zpráv

Jsou definovány čtyři základní typy zpráv. Jedná se o datovou zprávu(Data Frame), která je základní komunikační zprávou. Touto zprávou se posílají data po sběrnici. Množství dat, které je možné poslat je 0 - 8 bytu, počet je zapsán do zprávy. Další zprávou je žádost o data (Remote Frame), kterou vysílá uzel žádající data. Na pozici, kde je v datové zprávě uváděn počet posílaných dat, je uveden počet dat o které nyní uzel žádá. Další zprávy slouží k managmentu komunikace po sběrnici. Jedná se o signalizaci chyb, eliminaci chybových zpráv a vyžádání prodlevy.

2.4.5.1 Standartní zpráva

Jedná se o zprávu podle specifikace 2.0A. Tento typ je označovaný jako standardní formát zprávy (Standard Frame). Uspořádání bitů v této zprávě je na obrázku 2.6. Význam jednotlivých bitů (bloků bitů) je následující:

- SOF(Standard Of Frame) – 1 bit dominant, signalizuje začátek zprávy.
- Arbitration field – pole, které zahrnuje následující bity:
  - Identifikátor zprávy – 11 bitů, označuje zprávu, velikost určuje prioritu zprávy. Největší prioritu má nejmenší hodnota.
  - RTR (Remote Transmission Request) - 1 bit, rozlišuje datovou zprávu od žádosti o data.
- Control field – pole, které zahrnuje následující bity:
  - IDE (IDentifier Extension) – 1 bit, rozlišuje standardní a rozšířený formát zprávy.
  - R_0 – 1 bit, rezervovaný bit.
  - DLC (Data Length Code) – 4 bity, udává délku dat zprávy (počet bytu)
- Data field – pole dat obsahuje 0 - 8 bytu dat.
- CRC field – pole, které zahrnuje následující bity:
  - CRC – 15 bitů, CRC kód
  - ERC – 1 bit *recesive*, oddělovač CRC
- ACK field – pole které zahrnuje následující bity:
  - ACK – 1 bit, potvrzení zprávy, bit je vysílán *recessive* a je přepsán na *dominant* přijímacími uzly, kteří přijali data v pořádku.
  - ACD – 1 bit *recesive*, oddělovač ACK.
- End Of Frame – 7 bitů, ukončuje zprávu
- Interframe Space – 3 bity, 3 bity *recessive*, odděluje dvě zprávy.

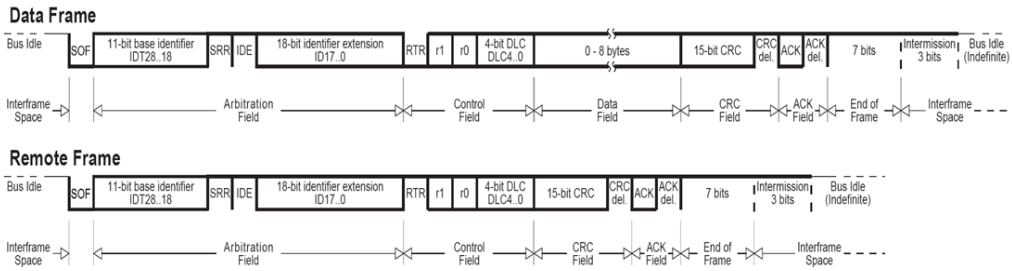


Obrázek 2.6: Datový rámec na sběrnici CAN podle standardu CAN 2.0A

2.4.5.2 Rozšířená zpráva

Rozšířený rámec (Extended Frame) používá celkem 29 bitový identifikátor zprávy. Ten je rozdělen do dvou částí o délkách 11 (stejný identifikátor je použit ve standardním formátu) a 18 bitů viz. 2.7. Bit RTR (Remote Request) je zde nahrazen bitem SRR (Substitute Remote Request), který má v rozšířeném formátu vždy hodnotu *recessive*. To zajišťuje, aby při vzájemné kolizi standardního a rozšířeného formátu zprávy na jedné sběrnici se stejným 11-ti bitovým identifikátorem, získal přednost standardní rámec. Bit IDE (Identifier Extended) má vždy *recessive* hodnotu. Bit (RTR) udávající, zda se jedná o datovou zprávu nebo žádost o data je přesunut za konec druhé části identifikátoru. Pro řízení přístupu k médiu jsou použity ID (11 bit), SRR, IDE, ID (18 bit), RTR. V tomto pořadí je určena priorita datové zprávy.

Rozšířený formát s sebou přináší některá úskalí. Zprávy v tomto rámci potřebují větší šířku pásma (okolo 20 %) a detekce chyb je menší, protože vybraný polynom pro 15 bitový CRC je optimalizován na rámec délky do 112 bitů.



Obrázek 2.7: Datový rámec na sběrnici CAN podle standardu CAN 2.0B

2.4.5.3 Žádost o data

Formát žádosti o data je podobný jako formát datové zprávy. Pouze je zde RTR bit (pole řízení přístupu na sběrnici) nastaven do úrovně *recessive* a chybí datová oblast. Pokud nějaký uzel žádá o zaslání dat, nastaví takový identifikátor zprávy, jako má datová zpráva, jejíž zaslání požaduje. Tím je zajištěno, že pokud ve stejném okamžiku jeden uzel žádá o zaslání dat a jiný data se stejným identifikátorem vysílá, přednost v přístupu na sběrnici získá uzel vysílající datovou zprávu. Úroveň RTR bitu datové zprávy je *dominant* a tudíž má tato zpráva vyšší prioritu.

2.4.5.4 Chybová zpráva

Chybová zpráva slouží k signalizaci chyb na sběrnici CAN. Jakmile libovolný uzel na sběrnici detekuje v přenášené zprávě chybu (chyba bitu, chyba CRC, chyba vkládání bitů, chyba rámce), vygeneruje ihned na sběrnici chybový rámec. Podle toho, v jakém stavu pro hlášení chyb se uzel, který zjistil chybu, právě nachází, generuje na sběrnici buď aktivní (šest bitů *dominant*) nebo pasivní (šest bitů *e*) příznak chyby. Při generování aktivního příznaku chyby je přenášená zpráva poškozena (vzhledem k porušení pravidla na vkládání bitů), a tedy i ostatní uzly začnou vysílat chybové zprávy. Hlášení chyb je pak indikováno superpozicí všech chybových příznaků, které vysílají jednotlivé uzly. Délka tohoto úseku může být minimálně 6 a maximálně 12 bitů.

2.4.5.5 Zpráva o přetížení

Zpráva o přetížení slouží k oddálení vysílání další datové zprávy nebo žádosti o data. Zpravidla tento způsob využívají zařízení, která nejsou schopna kvůli svému vytížení přijímat a zpracovávat další zprávy. Struktura zprávy je podobná zprávě o chybě, ale její vysílání může být zahájeno po konci zprávy (End of Frame), oddělovače chyb nebo předcházejícího oddělovače zpráv přetížení.

Kapitola 3

Návrh elektroniky

V této kapitole je popsán výběr jednotlivých komponent analyzátoru. Výběr uživatelského rozhraní, realizace elektroniky zajišťující komunikaci mezi všemi částmi analyzátoru, od sběrnice CAN po uživatelské rozhraní.

3.1 Úvod

Analyzátor je složen ze dvou hlavních částí. První část tvoří PDA, které se stará o ovládání analyzátoru a pro zobrazování získaných dat. Druhou částí je samotná elektronika zajišťující komunikaci se sběrnicí CAN, zpracování získaných dat a odeslání dat do PDA.

Při návrhu byl kladen důraz na nízkou spotřebu energie a s tím spojenou dlouhou provozní dobu. Nemalá pozornost byla také věnována velikosti provedení, bylo nutné aby se zařízení vešlo to standardní ,,krabičky" určené pro externí zařízení. Proto byla zvolena pro realizaci technologie SMD, která oba požadavky splňuje. Tato volba sebou přináší také drobné potíže s výběrem konkrétních součástek, protože ne vše co lze najít v katalozích na webových stránkách lze u nás koupit a navíc v množství několika kusů. Pokud není možné součástku koupit, je tu možnost objednání vzorků pro vývoj zařízení zdarma.

3.2 Uživatelské rozhraní

Pro uživatelské rozhraní jsem měl na výběr mezi dvěma hlavními platformami PDA:

- PocketPC - PDA s operačním systémem Microsoft PocketPC 2002 a starší verze systému.

- Palm - PDA s operačním systémem Palm OS.

Rozhodl jsem se pro platformu Palm, speciálně pro model Tungsten T s operačním systémem Palm OS verze 5.0. Pro moji práci jsem vybral starší model na platformě Palm OS, který patřil před časem ke špičce. Důvod pro tuto volbu byla skutečnost, že model je v praxi vyzkoušený, provoz je bezproblémový. Chyby které se vyskytovali při uvedení na trh byly už opraveny.

Není jednoduché říci, který z operačních systému je lepší. Windows CE a nástupce Pocket PC jsou plně multitaskové[1] operační systémy. Mají rozdělenou vnitřní paměť na operační paměť a systém souborů. Uspořádání je obdobné jako u stolních počítačů s verzemi Microsoft Windows. Jedná se v podstatě o zmenšenou verzi stolního počítače a s tím jsou také spojené nemalé nároky na výkon procesoru a velikosti paměť. Na proti tomu Palm OS se snaží o jednoduché aplikace se snadným ovládáním. Nároky na paměť a výkon procesoru jsou znatelně menší.

3.2.1 PALM Tungsten T

Je PDA založené na procesoru Texas Instruments OMAP1510 a je s instalovaným operačním systémem Palm OS verze 5.0. Operační systém v PDA je nahrán v paměti Flash ROM.[2] Základní hardwarové údaje jsou:

- Procesor Texas Instruments OMAP1510.
- Paměť 4MB Flashable ROM a 16 MB RAM.
- Bluetooth (verze 1.1)
- RS232 a USB (verze 1.1) rozhraní.
- Slot pro přídavnou paměťovou kartu typu SD (Secure Digital) nebo MMC (MultiMedia).

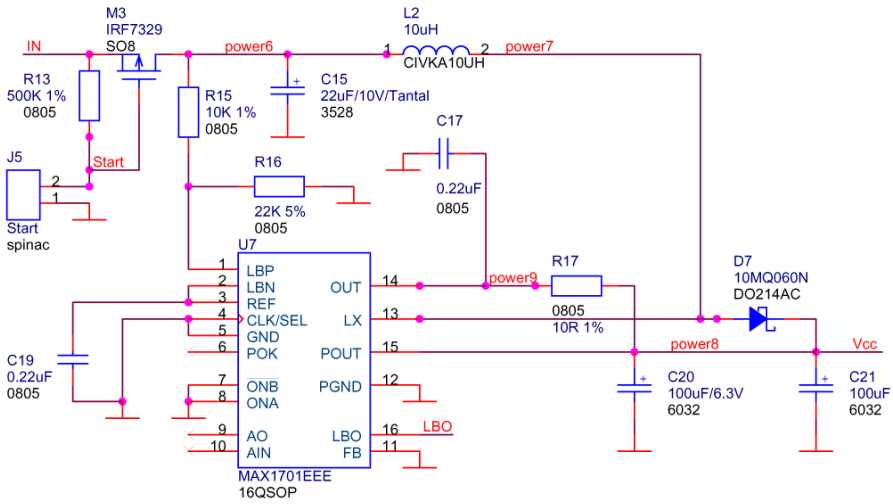
3.3 Napájení

3.3.1 DC-DC měnič

Pro napájení byl zvolen samostatný napájecí zdroj (2x akumulátor typu AAA), i když byla možnost napájení přímo z akumulátorů obsažených v PDA. Tím by se

[1] Umožňuje běh několika aplikací a jejich vzájemné přepínání bez přerušení činnosti.
[2] Paměť typu ROM, která je umožňuje přeprogramování.

ale razantně snížila provozní doba. Pro dosažení požadovaného napájecího napětí (stabilní hodnota 3,3V) je zde použit DC-DC měnič, který napětí zvýší a stabilizuje na požadované hodnotě. Využívá se blokujícího zapojení impulsního zdroje napětí, kde se v prvním kroku energie akumuluje v cívce L2 3.1 a v druhém se sečte s hodnotou v akumulátorech. Díky tomu je možné z akumulátorů získat téměř všechnu energii. Zvolil jsem obvod od firmy Maxim MAX1701EEE, který pracuje s účinností 96%. Pro tento měnič viz. obr. 3.1 stačí pro požadovaný výstup 3.3V jen 1.1V na akumulátorech.



Obrázek 3.1: DC-DC měnič

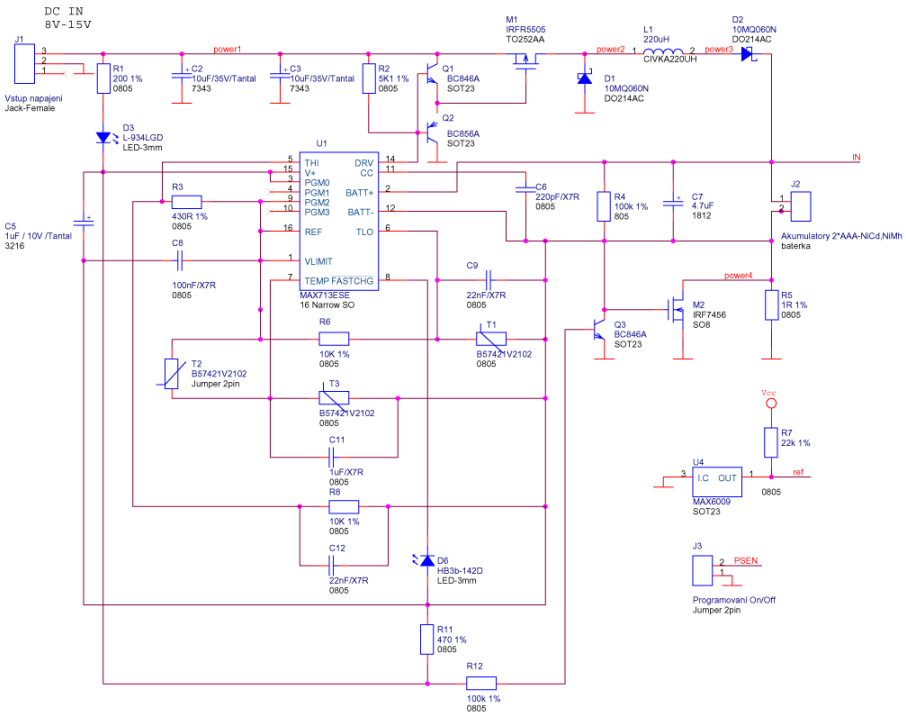
Tato hodnota odpovídá téměř vybitým akumulátorům. Pro správnou funkci je třeba rychlá shottky dioda D7 s velkým závěrným napětím. Důležitá je rovněž cívka L2, která musí mít nízký stejnosměrný odpor, max. 0,5Ω.

3.3.2 Nabíjení

Pro vyloučení dalšího zařízení nutného k funkci analyzátoru je v systému integrována nabíječka akumulátorů. Akumulátory je díky tomu možné dobíjet při provozu a zároveň pracovat s analyzátorem. Pro nabíječku jsem zvolil obvod od firmy Maxim MAX713SE, který splňoval požadavky nejlépe. Základní parametry obvodu jsou:

- Velký rozsah napájecího napětí.
- Lineární nebo pulsní řízení nabíjecího proudu.

- Akumulátory typu NiCd a NiMH.
- Podpora pro rychlé nabíjení se sledováním teploty akumulátorů.
- Ve stavu vypnutého nabíjení je odběr obvodu jen 5 μA.



Obrázek 3.2: Zapojení nabíjecího s obvodem MAX713SE

Pro nabíjení jsem zvolil pulsní řízení. Při spojitém řízení dochází na řídícím členu (většinou bipolární tranzistor) k tepelným ztrátám vlivem současné přítomnosti napětí a proudu. Při pulzním řízení se jako řídící člen požívá MOSFET tranzistor. Ten je stejně jako u DC-DC měniče buď zapnut, nebo vypnut. V tomto režimu vznikají tepelné ztráty jen na přechodech mezi napěťovými úrovněmi. Spínací frekvence tranzistoru se pohybuje kolem 20kHz.

Pro nastavení omezení proudu v režimu Fast charge (rychlého nabíjení) slouží odpor R6 viz. schéma 3.2. Tento odpor je zapojen sériově s akumulátory. Ve stavu, kdy akumulátory slouží jako zdroj energie vznikají na tomto odporu ztráty v závislosti na velikosti odebíraného proudu. Proto se pomocí tranzistoru M2 vyřadí pokud neprobíhá nabíjení. Tranzistor M2 je rozpínán při připojení externího napájení.

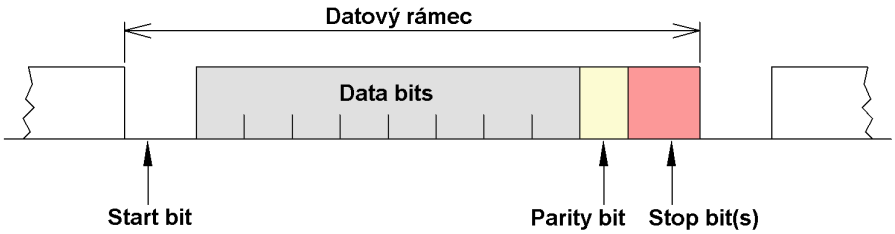
3.4 Komunikace RS232

Jedná se o sériovou komunikaci standardizovanou Electronic Industry Association EIA. Tato komunikace se hodí pro spojení dvou zařízení. Modulační rychlost přenosu je nutné před každou komunikací mezi dvěma zařízeními nastavit na stejnou hodnotu, neboť určuje dobu existence jednoho bitu na lince. Přenosová jednotka je Bd (Baud rate), která je úměrná počtu změn signálových úrovní na lince za sekundu.

Často používané označení RS-232C definuje napěťové úrovně. Log. 1 (vysoká úroveň) odpovídá napěťové úrovni −3 až −15 V, log. 0 (nízká úroveň) úrovni +3 až +15 V. Obvody rozhraní jsou nesymetrické, proto se uvedené úrovně vztahují vůči potenciálu nulového signálového vodiče.

Součástí každého zařízení komunikujícího po sériové lince standardu RS-232 je vysílač (Transmiter - signál TxD), přijímač (Receiver- signál RxD) a řídicí signály. Jedná se o plně duplexní komunikaci umožnující současné vysílání a přijímání dat. Počet řídicích signálu není přesně definován a záleží na implementaci. Nejčastěji se používají RTS (Request To Send), CTS (Clear To Send), DSR (Data Set Read) a DTR (Data Terminal Ready).

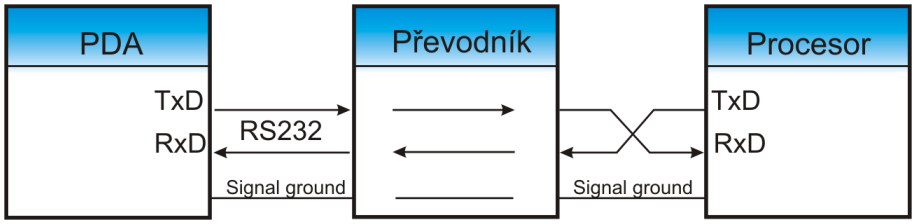
Přenos dat po médiu se děje na úrovni znaku (1 byte), což je zároveň pevný přenosový rámec, který se používá po celou dobu. Kromě datové informace je s každým znakem vysílána i informace synchronizační.



Obrázek 3.3: Datový rámec na sběrnici RS232

Přenosový rámec začíná START bitem, který má nízkou logickou úroveň, za ním pak následují datové bity, paritní bit a STOP bity s vysokou logickou úrovní viz obrázek 3.3. Počet datových bitu (5 až 8, často 7 resp. 8), přítomnost paritního bitu, typ parity a počet STOP bitu (1 nebo 2) lze nakonfigurovat. Jedná se o asynchronní přenos využívající pro svou synchronizaci pouze START a STOP bity, proto jsou jejich logické úrovně odlišné, aby přijímač rozpoznal, zda se jedná o začátek rámce (každý rámec oznámí svůj začátek přijímači nízkou úrovní signálu TxD).

Pro komunikaci mezi procesorem 3.6 a PDA jsem zvolil tzv. NULL Modem. Jedná se pouze o 3 vodiče TxD, RxD a Signálová zem. PDA podporuje i další řídící signály, ale ze strany procesoru podporovány nejsou. Pro úspěšnou komunikaci je třeba ještě upravit logické úrovně, k tomu slouží speciální obvody od mnoha výrobců.



Obrázek 3.4: Komunikace mezi PDA a procesorem

Zvolil jsem obvod MAX3223, který má nízkou spotřebu a velmi užitečnou funkci Autoshutdown. Tato funkce vypne obvod pokud na protější straně sběrnice RS232 není připraveno žádné zařízení komunikovat po dobu 30 μs. V tomto stavu je spotřeba 1 μA. Do aktivního stavu se obvod dostane do 100 μs. Tato doba je relativně dlouhá, ale pro toto použití dostačující.

Pokud je zařízení připojeno k PDA a neběží obslužný software nebo neprobíhá komunikace, provede se vypnuti obvodu a úspoře energie. Pokud nastane požadavek na od komunikaci uživatele obvod se zapne. Stejně tak v případě požadavku na komunikaci ze strany procesoru.

3.5 Paměť

Paměť byla přidána z důvodu malé interní RAM v procesoru. Jelikož data na sběrnici CAN jsou přenášena rychlostí až 1Mbit a rychlost mezi procesorem a PDA je jen 57600, je nutné data dočasně ukládat. Pro tento účel jsem vybral paměť SRAM[3] pamětí. o velikosti 1Mbit. Organizace paměti je 128K×8bitů. První hodnota udává počet adresovatelných míst v paměti a druhé šířku slova v bitech, které je tam možné uložit. Paměť je vybrána podle možností procesoru jež schopen adresovat paměť o velikosti 64K×8bitů. Použitá paměť je 2× větší než je adresný prostor procesoru. Poslední bit z adresy je nastavován zvlášť podle pozice v paměti, více v softwarové části v kapitole 4.4.3.1.

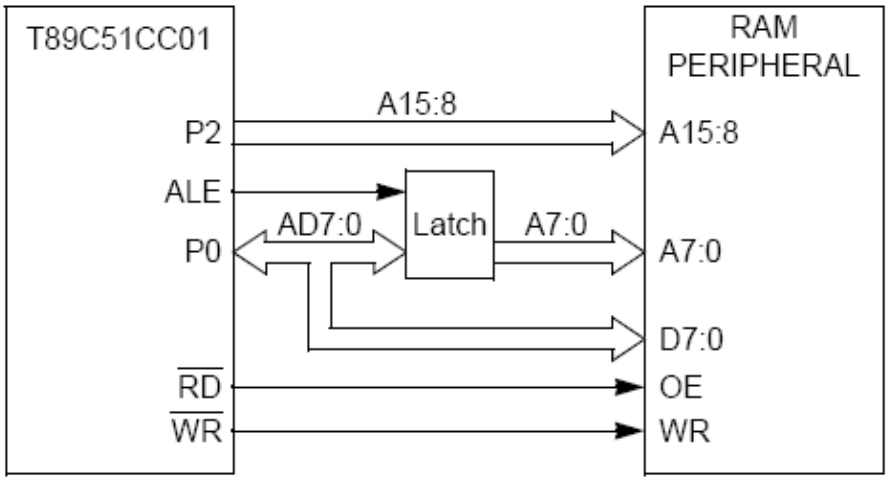
Zapojení převodníku je naznačeno na obrázku 3.5 a detailně potom na schématu 3.6.

Proces čtení a zápisu dat do externí paměti se děje následovně. Při požadavku

[3]Statická paměť RAM

o čtení dat se na portu P2 objeví horních 8 bitů adresy a na portu P0 se objeví spodních 8 bitů adresy. Následně se signálem ALE zapíše spodních 8 bitů z portu P0 do registru D-latch a ten je okamžitě promítne na výstup. Nyní se port P0 přepne do stavu čtení a čeká jaká data se objeví na výstupu paměti. V tuto chvíli je na adresových vstupech paměti platná adresa a signálem RD se na výstup paměti zapíše hodnota dat na této adrese, kde si je po uplynutí signálu RD přečte procesor.

Při požadavku o zápis dat je proces podobný. Na portu P2 objeví horních 8 bitů adresy a na portu P0 se objeví spodních 8 bitů adresy. Následně se signálem ALE zapíše spodních 8 bitů z portu P0 do registru D-latch a ten je okamžitě promítne na výstup. Nyní se na port P0 zapíší data, která chceme zapsat do paměti a aktivuje se signál WR který data zapíše do paměti.



Obrázek 3.5: Blokové schéma zapojení paměti a procesoru

Název	**Funkce**
$A_0 \sim A_{16}$	Adresové vstupy 0 - 16 bit
$\overline{WE}$	Povolení zápisu dat do adresované buňky v paměti
$\overline{CS}_1$,CS_2	Vstupy pro aktivování paměti.
$\overline{OE}$	Zápis dat z paměťové buňky na výstupy.
$I/O_0 \sim I/O_7$	Vstupy, výstupy dat.
V_{cc}	Napájení paměti 3,3V.
V_{ss}	GND 0V.

Tabulka 3.1: Popis jednotlivých pinů paměti.

3.6 Procesor

Poslední a nejdůležitější částí je procesor. Jeho činnost se dá shrnout v následujících bodech:

- Komunikace s PDA.
- Vysílání a čtení dat z CAN sběrnice.
- Sledování vnějších událostí.
- Ukládání a čtení dat z externí paměti RAM.

Bylo mnoho možností jaký obvod zvolit, téměř každý výrobce procesorů má zástupce i v této problematice. Hlavní kritéria pro výběr procesoru byla:

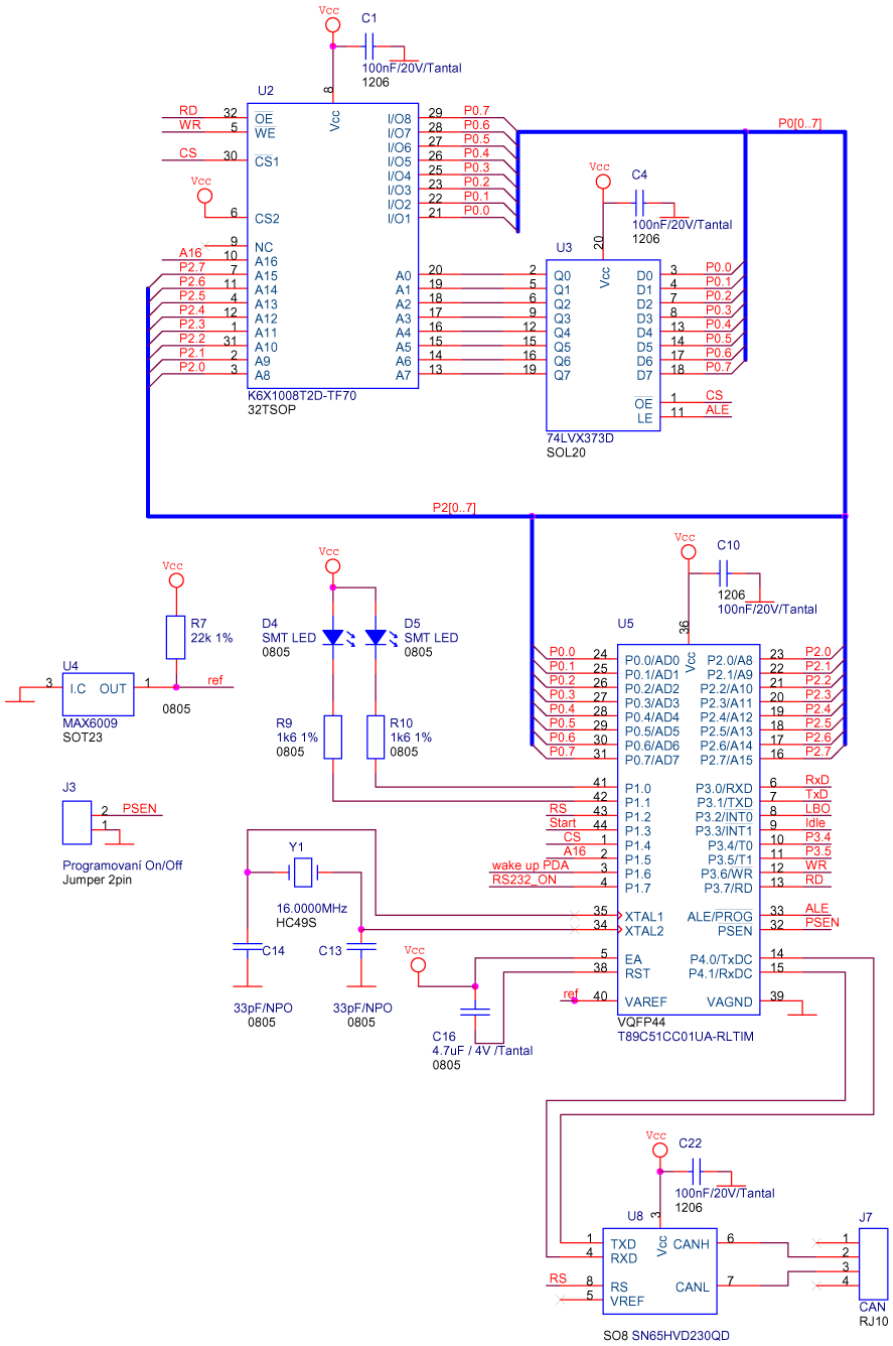
- Nízká spotřeba.
- Provedení v SMD technologii.
- Možnost programování In-System[4].
- Podpora pro standard CAN 2.0B do rychlosti 1Mbit.
- Počet nezávislých objektů[5] pro CAN zprávy.
- Podpora pro externí paměť RAM.
- Sériové rozhraní pro komunikaci s PDA.

Rozhodl jsem se pro procesor T89C51CC01UA-RLTIM od firmy Atmel [6]. Tento procesor má podle mého názoru nejlepší poměr nabízených funkcí a spotřeby. Porovnání jednotlivých parametrů je vidět v tabulce 3.2 a spotřeba jednotlivých procesorů v tabulce 3.3. Všechny procesory zde uvedené mají podporu pro CAN 2.0B.

Základní charakteristika procesoru:

- Architektura 80C51.
- 256 Bytů interní paměti RAM.
- 32 KBytů Flash Memory[9] paměti programu.
- 10-bitový A/D[6] převodník s 8 přepínanými vstupy.

[4]Programování procesoru přímo v zařízení přes sériové či jiné rozhraní.
[5]Místo v paměti procesoru, kde se uloží všechna data z přijaté CAN zprávy.
[6]Anologově - číslicový převodník



Obrázek 3.6: Schéma procesoru s pamětí a CAN převodníkem

Výrobce	Procesor	Rychlost [MHz]	I_{cc} [V]	CAN Objekty	RAM [kBytes]
Dalas	DS80C390	max 40	4,5-5,5	15 - in/out	4
Atmel	T89C51CC01	max 40	3-5,5	15 - in/out	1,5
Atmel	T89C51CC02	max 40	3-5,5	4 - in/out	0,5
Atmel	T89C51CC03	max 40	3-5,5	15 - in/out	2,5
Microsystems	ST10CT167	max 25	3-5,5	15 - in/out	4
Microsystems	ST10F167	max 25	3-5,5	15 - in/out	4
PHILIPS	P83C591	max 25	5 ±5%	13 - in/out	64
Microchip	PIC18F248 PIC18F258	max 40	2-5,5	2/3 - in/out	4
Microchip	PIC18F6585 PIC18F6680	max 40	2-5,5	16 - in/out	4
Motorola	MC68HC912D60A	max 16	2-5,5	4/4 - in/out	2

Tabulka 3.2: Přehled procesorů a jejich vybavení.

Výrobce	Procesor	In-System	I_{cc}[mA]
Dalas	DS80C390	NELZE	120
Atmel	T89C51CC01 T89C51CC02 T89C51CC03	RS232, CAN RS232, CAN RS232, CAN	0,7×MHz+3 0,7×MHz+3 0,7×MHz+3
Microsystems	ST10CT167 ST10CT167	NELZE NELZE	5×MHz+20 5×MHz+120
PHILIPS	P83C591	NELZE	45
Microchip	PIC18F248 PIC18F258 PIC18F448 PIC18F458	RS232 RS232 RS232 RS232	$F_{osc} = 25$; $V_{dd} = 5V$ 14 $F_{osc} = 16$; $V_{dd} = 3{,}3V$ 5,5
Motorola	MC68HC908	RS232	30

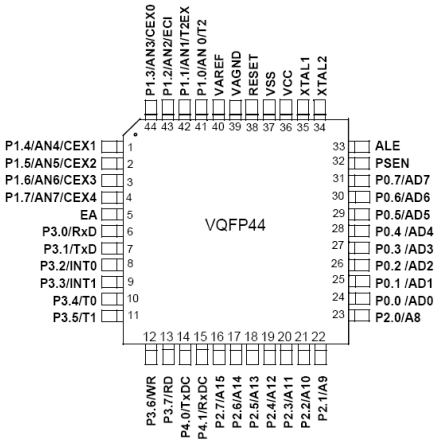
Tabulka 3.3: Porovnání spotřeby jednotlivých procesorů.

- 15 nezávislých objektů CAN
- 14 zdrojů přerušení organizovaných do 4 úrovní priorit.

- 3 16-bitové čítače/časovače pracujících ve 4 módech.
- Full Duplex UART kompatibilní s ostatními 80C51.

Procesor je řízen hodinovým signálem, který je generován pomocí krystalu. Zvolil jsem krystal o frekvenci 16MHz. Tato hodnota je nejnižší, při které procesor zvládne plně obsluhovat CAN při maximální rychlosti 1Mbit. Vyšší hodnota krystalu zvedá spotřebu procesoru podle vztahu uvedeného v tabulce 3.3, což při snaze o co nejmenší spotřebu je nežádoucí.

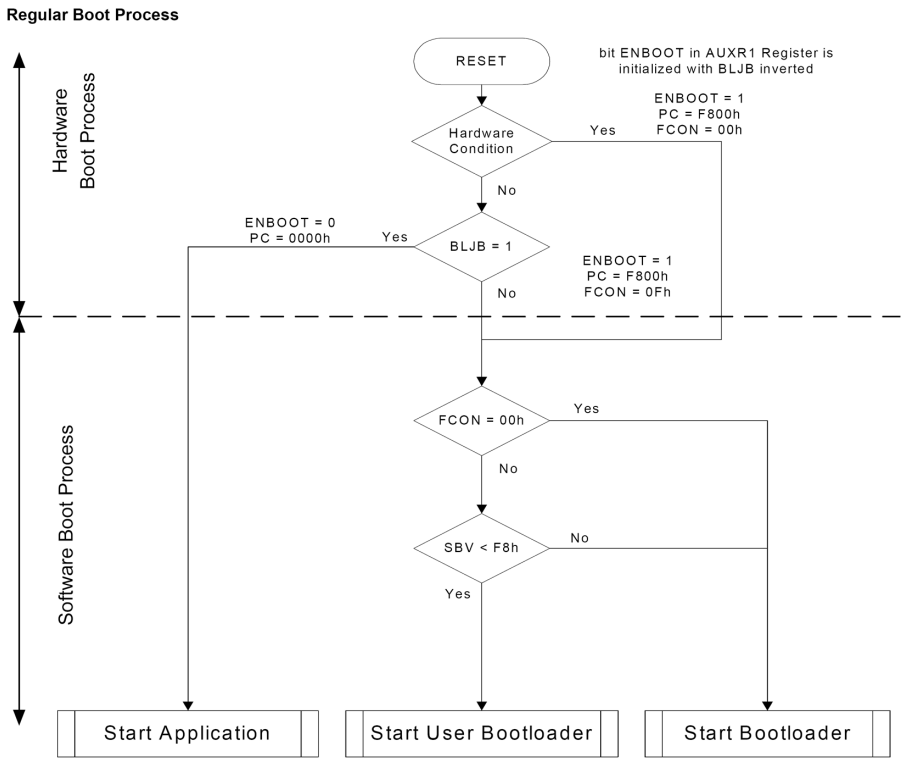
Procesor má integrován A/D převodník napětí. Převodník, který používám k měření napětí akumulátorů, potřebuje ke své činnosti zdroj referenčního napětí. Hodnota může být v rozsahu 2,4V – 3V. Zvolil jsem obvod od firmy Maxim MAX6009 odběr má velmi malý v řádu jednotek μA.



Obrázek 3.7: Procesor T89C51CC01UA-RLTIM v pouzdru VQFP44

3.6.1 Bootloader

Bootloader je program v procesoru od výrobce a slouží ke komunikaci se softwarem pro programování. Existují dvě verze UART a CAN, podle toho po jaké sběrnici chceme s procesorem komunikovat při programování. Tento program standardně probíhá po resetu procesoru, pokud již nebyly změněny některé hardwarové bity, které start ovlivňují. Proces rozhodování je vidět na obrázku 3.9. Při resetu dojde k testování, zda není pin procesoru PSEN připojen na 0V. Pokud není, rozhoduje nastavení bitu BLJB. Standardně je od výroby nastaven na 1, to znamená, že procesor automaticky spouští na bootloader. Pokud bit změníme(popsáno níže) na 0 pustí se program v paměti programů. Bootloader je míti vlastní dle potřeby, je na něj vyhrazeno speciální místo v paměti.

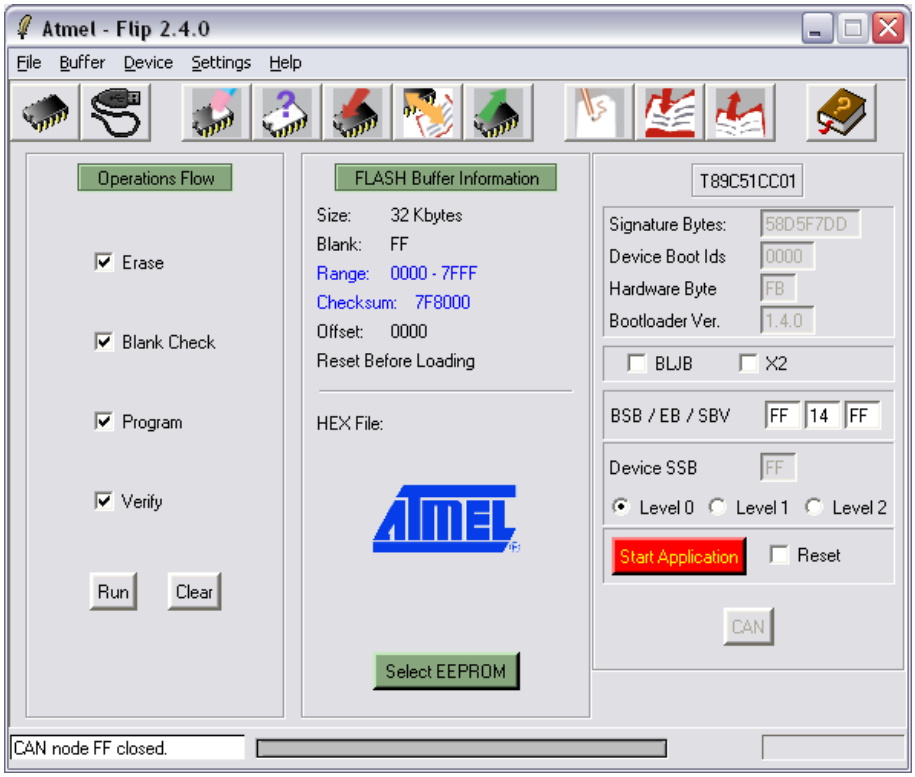


Obrázek 3.8: Startovací proces procesoru po resetu

Pro programování procesoru a nastavování tzv. harwarových bitů je nutné mít obslužný program FLIP (nejlépe verze 2.4 a vyšší)viz. obrázek 3.9. Tento program je zdarma ke stažení na stránkách výrobce. Podporuje operační systémy Microsoft Windows a také Linux. Tento program se spojí s bootloaderem v procesoru a umožní nahrát uživatelský program do procesoru a další operace. Program nabízí různé rychlosti připojení z důvodu závislosti rychlosti RS232 na frekvenci krystalu.

3.7 D-Latch

Vybral jsem obvod 74LVX373 od firmy Fairchild Semiconductor. Jedná se opět o obvod s nízkým odběrem. Tento obvod se často používá právě k pamatování části adresy jako v mém případě. Obvod má 8 vstupů a 8 výstupů a 2 řídící signály viz. tabulka 3.4. Data se přivedou na vstupy a signálem LE zapíší do vnitřní paměti (Jedná se o 8 D-klopných obvodů zapojených paralelně). Signálem $\overline{OE}$ se aktivují výstupy na kterých se objeví uložená data. Tyto data jsou nezávislá na aktuálním stavu vstupů. Pokud jsou výstupy neaktivní, nacházejí se ve stavu vysoké impedance. Tento stav se dá přirovnat k fyzickému odpojení obvodu. Této vlastnosti se využívá při sběrnicovém zapojení výstupů, kde pokud by dva propojené výstupy



Obrázek 3.9: Program FLIP

měly různé hodnoty došlo by k poškození obvodů.

Název	Funkce
$I_0 \sim I_7$	Datové vstupy 0 - 7 bit
LE	Signál pro zápis do vnitřní paměti. Aktivní úroveň Log 1
$\overline{OE}$	Aktivace výstupů. Aktivní úroveň Log 0
$O_0 \sim O_7$	Vstupy, výstupy dat
GND	Zem
V_{cc}	Napájecí vstup

Tabulka 3.4: Popis vstupů a výstupů.

3.8 CAN převodník

Obvod zajišťuje převod dat z CAN sběrnice pro procesor. Na sběrnici CAN je logická úroveň reprezentována rozdílem mezi vodiči CANH a CANL. Procesor vztahuje úroveň napětí k zemnící svorce. Zároveň tvoří galvanické oddělení elektroniky od sběrnice.

Pro výběr obvodu byly kladeny následující podmínky:

- Nízká spotřeba.
- SMD provedení pouzdra obvodu.
- Úsporný režim v případě nulové komunikace.

Podrobné srovnání jednotlivý obvodů je v tabulce 3.5.

Výrobce	Typ	Rychlost [Mbs]	ESD[7] [kV]	Napájení [V]	Spotřeba [mA]	Dočasně [V]
Bosch	CF150B	0.5	2	5	≤80	± 200
Texas Instruments	SN65HVD230 SN65HVD231	1	4	3,3	10	± 25
Mietec	MTC3054	1	2	5	≤110	± 200
Philips	82C250	1	2	5	≤70	± 100
Atmel	ATA6660	1	8	5	≤110	± 200
SGS-Thomson	L9615	0.5	2	5	≤80	± 200
Temic Siliconix	ATA6660	1	2	5	≤70	± 60

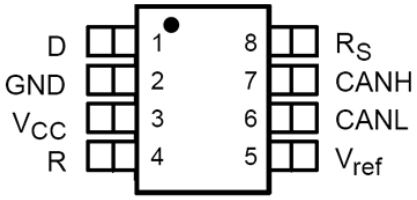
Tabulka 3.5: Převodníky mezi procesorem a CAN sběrnicí

Obvodů celá řada, ale pouze obvod SN65HVD230QD od firmy Texas Instruments vyhovoval stanoveným požadavkům. Jako jediný má možnost napájení 3,3V a nabízí různé režimy provozu. Jedná se o režimy *high speed*, *slope control* a *low power*. V režimu *high speed* je přechod mezi úrovněmi na výstupech CANH a CANL uskutečněn tak rychle jak dovolí vlastnosti hardwaru. Obvod umožňuje přijímání a odesílání zpráv. Druhý režim *slope control* je stejný jako první režim s tím rozdílem, že je kontrolována rychlost nástupné resp. sestupné hrany na výstupech CANH a CANL. Hodnoty jsou dány odporem připojeným mezi vstup R_s a GND. Pro odpor 10kΩ je hodnota 15V/μs a pro 100kΩ 2V/μs. Poslední režim *low power* vypne vysílací kanál a sníží spotřebu na jednotky μA. V tomto režimu lze data jen přijímat.

[7] ESD - odolnost zařízení vůči elektrostatickým pulzům

Název	Funkce
R	Výstup přijatých dat z CAN sběrnice pro procesor
D	Vstup dat od procesoru k odeslání na CAN sběrnici
CANH	Výstup na CAN sběrnici.
CANL	Výstup na CAN sběrnici.
R_s	Řízení módu běhu obvodu.
V_{ref}	Referenční napětí. Odpovídá polovině V_{cc}
GND	Zem
V_{cc}	Napájecí vstup

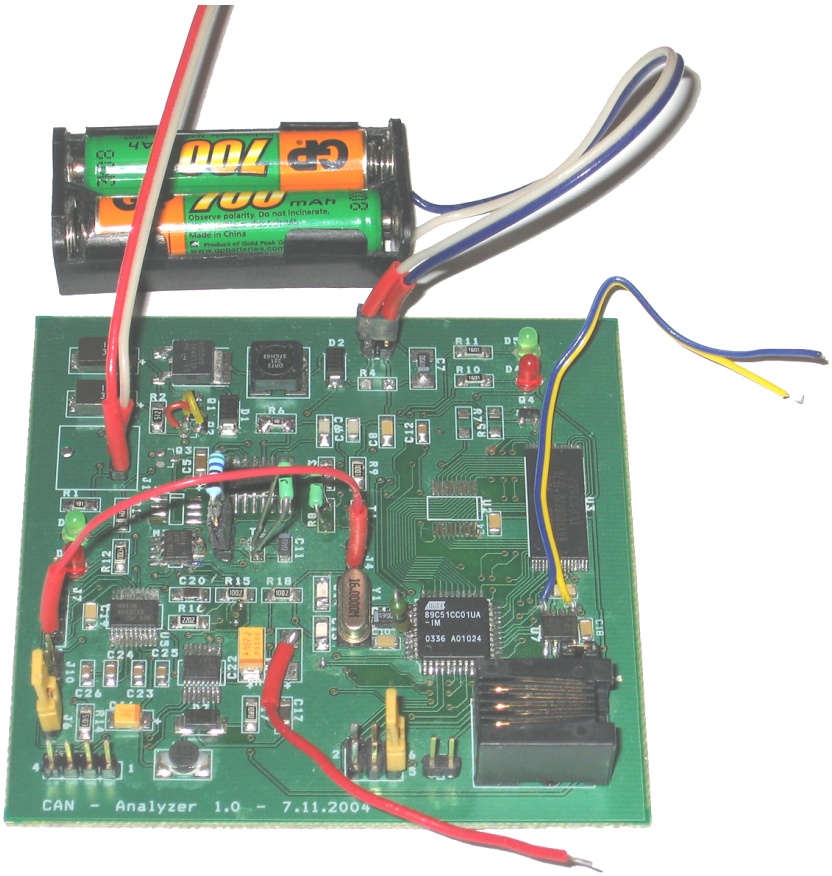
Tabulka 3.6: Popis vstupů a výstupů.



Obrázek 3.10: Převodník CAN SN65HVD230QD v pouzdru SO8

3.9 Realizace elektroniky hardwaru

Při realizaci analyzátoru, byla nejdříve navržena a vyrobena vývojová desku s vyvedenými ovládacími piny. Na této desce jsem odladil zapojení a následně vyrobil finální provedení s požadovanými rozměry. Jelikož jsem zvolil SMD provedení, nebylo možné testovat například na nepájivém poli.



Obrázek 3.11: Vývojová verze analyzátoru

Obrázek 3.12: Finální podoba analyzátoru

Kapitola 4

Softwarové řešení

V této kapitole je popsán vývoj programů pro měřící zařízení (hardware) a pro uživatelské rozhraní (PDA). Vzájemná komunikace, nastavování parametrů a řešení případných poruch, jak na straně hardware tak na straně PDA.

4.1 Úvod

Pro vývoj aplikací bylo nutné vybrat programovací prostředí a jazyk, ve kterém se bude programovat procesor a uživatelské rozhraní v PDA. Programů je celá řada jak z řad placených tak z řad volně šiřitelných. Výběr jsem směřoval právě k volně šiřitelným programům.

4.2 Výběr softwarového řešení

4.2.1 Firmware microprocesoru

Pro programování procesoru jsem jsem zvolil programovací prostředí KEIL[10], které je zdarma a pro menší velikosti do 2KB kódu nemá žádné omezení. Do toho programu jsem si z firemních stránek firmy Atmel nahrál definiční soubor k použitému procesoru. Tento soubor obsahuje definice všech registrů obsažených v procesoru a jejich paměťové umístění. Při programování pak lze používat místo adres jen názvy registrů, nebo přímo názvy bitů v těchto registrech. Toto prostředí podporuje programování v jazyce C a jazyce Asembler. Využití vyššího programovacího jazyka C by bylo pro programování pohodlnější, ale výsledek kódu je pak větší a méně optimalizovaný. Proto jsem vybral jazyk Asembler.

4.2.2 PDA

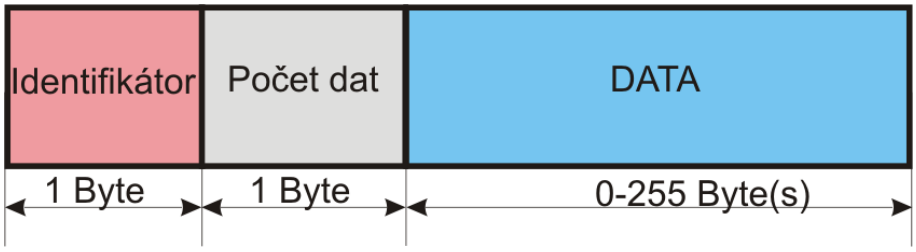
Pro programování PDA se výběr zúžil na dvě řešení. Prvním je profesionální řešení od firmy Metrowerks program Codewarrior verze 9.2, který má velmi propracovaný ladící mód. Druhým je volně šiřitelné řešení (free software) pod GNU (General Public Licence) licencí[11], program Cygwin z balíčku prc-tools ve verzi 2.3. Pro pohodlnější tvorbu formulářů a všeho co souvisí s ovládacími prvky je možné využít speciálních programů. Jsou to například Pendragon Forms 5.0 (14 dní platná demoverze), PDA Toolbox Version 6.0 (30 dní omezena demo verze), Pirlc (GNU licence). Osobně mi nejvíce vyhovoval právě volně šiřitelný program Pirlc. Lze jím jednoduchým způsobem navrhnout design formulářů včetně všech atributů.

Návrh spočívá ve vytvoření ovládacích prvků (menu, tlačítka, posuvníky, atd.) jejich provázání, vložení popisujících textů a také nápovědy. Tyto objekty se pojmenují a vygeneruje se definiční soubor, který je součástí aplikace.

4.3 Komunikační protokol mezi HW a PDA

Uživatelské rozhraní v PDA komunikuje s měřícím zařízením po standardní RS232 sběrnici. V jednom rámci přeneseným po RS232 je vždy 1 byte dat a tak při posílání většího množství dat je nutné hlídat konzistenci přenesených dat. Jelikož je přenášeno nemalé množství dat s různým významem a délkou, bylo potřeba pro přenos dat definovat komunikační protokol. Byl vytvořen a implementován jednoduchý protokol viz. obrázek 4.3. Skládá se z hlavičky, délky dat a dat samotných. Hlavička má význam identifikátoru zprávy(podobně jako CAN), kde každá hodnota představuje nějakou zprávu. Zprávy je možné rozdělit takto:

- ***Datové***, kde se přenáší data získaná ze sledování CAN sběrnice.
- ***Inicializační***, které nastavují parametry analýzy a žádají o poslání příslušných dat.
- ***Chybové hlášení*** dají se rozdělit na:
  - chyba dat – přijatá data mají hodnoty mimo povolený rozsah.
  - chyba počtu dat – při komunikaci se ztratil nějaký rámec a nesouhlasí počet přijatých bajtů dat pro daný povel.



Obrázek 4.1: Obsah zprávy komunikačního protokolu

4.4 Realizace firmwaru v procesoru

4.4.1 Obsluha přerušení

Přerušení dovoluje téměř ihned reagovat na vnější události nebo na události vnitřních zařízení. Protiváhou této rychlé reakce je nutnost precizně ošetřovat stavy v programu. Nelze zaručit, kde přesně se v danou chvíli program nachází, kdy se která činnost přeruší a zda takové přerušení neovlivní výsledek prováděné operace. Jsou zde 4 úrovně priorit přerušení. Důležité operaci je možné nastavit vyšší prioritu. Pokud nastane více požadavků na obsluhu přerušení, přednost získá vždy přerušení s nejvyšší prioritou. Podprogramy pro přerušení jsou psány krátké, aby nedocházelo k blokování programu.

Procesor má 10 zdrojů přerušení v tabulce uvedu jen ty, které používám:

Název	**Adresový vektor**	**Popis**
INT0	0003h	Vnější událost (P3.2 – INT0)
TIMER 0	000Bh	Čítače/časovače (TF0)
UART	0023h	Sériový přenos (TI, RI)
CAN	003Bh	CAN rozhraní

Tabulka 4.1: Tabulka použitých přerušení

4.4.1.1 Nastavení registrů

Pro povolení přerušení je nutné nastavit příslušný bit v registru IEN0, IEN1. Bitem EA(Enable All) v registru IEN0 nastavíme zda se všechny přerušení zakáží, nebo zda se povolí ty které mají povolený příslušný bit.

Dále jsou zde registry IPL0, IPL1, IPH0, IPH1. Slouží k nastavení priority

přerušení. Pro každé přerušení jsou potřeba dva bity (priorita 0–3) jeden v registru IPLx a stejný v registru IPHx.

4.4.2 Sériová komunikace

Tvoří základní komunikační rozhraní spolu s CAN řadičem. Komunikace probíhá asynchronně a v zapojení Null modem viz 3.4. Procesor dovoluje nastavit rychlost až do 460800 Baudu v X2 režimu. Podporované rychlosti ze strany PDA jsou standardní a do 115200 Baudů. Zvolil jsem nejvyšší možnou rychlost, ale došlo ke komplikacím. Rychlost UART je možné nastavit pomocí těchto čítačů 0,1,2. Čítače při použití jako Baudrate generátoru pracují v režimu Auto-Reload. To znamená, že po načítání maximální hodnoty která je 255 (timer0,1), 65535 (timer 2) nedojde k vynulování ale načtení hodnoty z příslušného registru. Čítače 0 a 1 pracují jen jako 8 bitové, čítač 2 jako 16 bitový. Pro vyšší rychlosti je třeba využít čítače 2, pro nižší do 57600 Baudů je to libovolné.

Problém implementace UART je, že je pevně vázán na frekvenci krystalu. Rychlosti UART se odvíjí od frekvence krystalu. Pro UART jsou vhodné hodnoty např. 11,0592 MHz nebo 18,432MHz. Naproti tomu naopak je řadič CAN, který potřebuje pro dosažení rychlostí (např. 100,150,500 kBps) frekvenci v celý číslech např. 12, 16, 24 MHz. Tady vznikl problém, který jsem musel vyřešit snížením rychlosti UART na 57600 Baudu při frekvenci krystalu 16MHz.

4.4.2.1 Nastavení registrů

Pro nastavení rychlosti UART je třeba nastavit registry TCON(Timer/Counter Control Register), TMOD(Timer/Counter Mode Control), TH0, TL0, TH1, TL1. TCON zajišťuje řízení, spouští čítače(TRx)[1], nastavuje parametry externího přerušení (IEx). Registr TMOD nastavuje režimy běhu(M0x, M1x) a podmínky spuštění (GATEx, C/Tx#). Registry TLx a THx mění význam dle zvoleného režimu:

- Režim 0 – 13 bitový čítač/časovač.
- Režim 1 – 16 bitový čítač/časovač, spojení 8 bitových registů TLx a THx do jednoho 16 bitového.
- Režim 2 – 8 bitový čítač/časovač s auto-reloadem, TLx počítá, v THx je uložena hodnota, která se vloží do TLx při přetečení.
- Režim 3 – dva 8 bitové čítače/časovače.

[1] x znamená hodnotu 0 nebo 1

Pro výpočet UART rychlosti jsem použil čítače/časovače 2, který má obdobné parametry jako předcházející dva, z jedním důležitým rozdílem a tím je 16 bitová předvolba u 3. režimu. Má také nastavovací registry s obdobnými volbami T2CON, T2MOD pro řízení a volbu režimu. Dále TH2 a TL2 pro čítání hodnot a navíc registry pro 16 bitovou p5edvolbu RCAP2H a RCAP2. Do těchto registrů se uloží hodnoty podle vztahu: Baud_rate = (f_{osc} /32)/(65536-RCAP), kde RCAP = RCAP2H,RCAP2L.

Posledním krokem je uložit spočítaná data do registrů RCAP2H,RCAP2L. Dále nastavit T2CON a bitem TR2 spustit čítač.

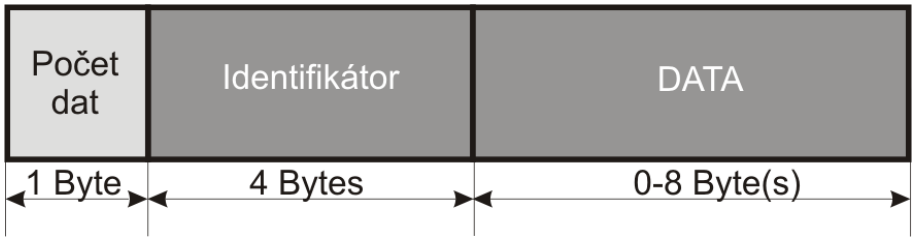
4.4.2.2 RS232 přerušení

Pro přijímání dat a řízení odesílání dat používám přerušení. Při příjmu se nastaví RI příznak a při odeslání dat TI příznak. Oba vyvolají přerušení provedení patřičné události. Přijatá data si ukládám do bufferu. Data je možné ukládat i do jedné proměnné a hned je zpracovávat, ale při zatížení procesoru (např. velké množství dat na sběrnici CAN) se může stát, že se nestihnou data zpracovat a přepíší se dalšími, které přijdou následně. Při odesílání je třeba počkat až data odejdou a pak dávat příkaz k dalšímu přenosu, jinak dojde k přepsání dat. Proto v přerušení používám dva řídící bity Data_received a Data_sent. Postup je takový, že před odesláním dat nastavím bit Data_sent na log 1 a dokud je v log 1 nedovolím odeslat další data. Při odeslání se v přerušení nastaví na log 0. Bit Data_received slouží k detekci příchozích dat a následných operací.

4.4.3 Interní a externí paměť

Interní paměť je pro uložení dat z CAN příliš malá, proto je přidána externí paměť. Interní paměť používám k ukládání lokálních proměnných a pak k uložení parametrů pro nastavení jednotlivých zařízení(sériové komunikace, CAN řadiče, analogově číslicového převodníku).

Externí paměť o velikosti 1Mbit využívám jako buffer pro data získaná při čtení ze sběrnice CAN. Data se ukládají ve formátu viz. 4.2. V procesoru je ještě obsažena paměť označovaná XRAM (Extended RAM). Tato paměť má velikost 1024 bajtů a přistupuje se k ní stejným způsobem jako do externí paměti RAM. Rozdíl, zda se jedná o XRAM nebo o externí RAM je dán bitem EXTRAM.



Obrázek 4.2: Organizace dat při ukládání do externí RAM

4.4.3.1 Nastavení registrů

Pro použití externí paměti RAM v programu procesoru je potřeba dobře nastavit hodnoty v těchto registrech:

AUXR(Auxiliary Register),

Bit číslo	Bit název	Popis
7–6	–	Rezervováno
5	M0	Prodlouží dobu signálů RD a WR^{2}o hodnotu: M0 = 0 ⇒ 6 hodinových period M0 = 1 ⇒ 30 hodinových period
4	–	Rezervováno
3–2	XRS1-0	Nastavení velikosti XRAM
1	EXTRAM	0 – MOVX přistupuje k interní XRAM 1 – MOVX přistupuje k externí RAM
0	A0	0 – Signál ALE 1/6 Fosc 0 – Signál ALE 1/3 Fosc v X2 módu 1 – Signál ALE aktivován při použití instrukcí MOVX, MOVC

Tabulka 4.2: Registr AUXR nastavení parametrů – externí RAM

AUXR1(Auxiliary Control Register 1) zde je zajímavý pouze bit DPS určující, který DPTR0, či DPTR1 je aktivní.

4.4.3.2 Čtení zápis dat

Pro práci s externí pamětí má procesor dva 16 bitové registry DPTR0 a DPTR1. Tato skutečnost velmi usnadní a zrychlí práci s pamětí. Speciálně pak porovnávání a častou kombinaci čtení/zápis dat. Jeden registr se používá pro uložení aktuální

pozice v paměti pro zápis dat a druhý pro čtení dat. Není tedy nutné si pamatovat adresy ve speciálních proměnných a pak je po částech vkládat do DPTR (DPL a DPH). Instrukce MOV pracuje pouze s 1 bajtovými daty, takže nelze přenést adresu naráz. Při programování, viz. příklad níže, se používá jednotný název DPTR pro oba registry. Který z registrů to momentálně je určuje bit DPS v registru AUXR1. (0 - DPTR0; 1 - DPTR1). Registr DPTR0 používám pro zápis a DPTR1 pro čtení. Po resetu procesoru jsou v inicializační části programu oba registry vynulovány.

Pokud se stane, že zápis dat převáží nad čtením a hodnota ve čtecím registru DPTR přeteče a dosáhne hodnoty DPTR pro čtení, dojde k zastavení analýzy a odeslání chybové zprávy do PDA. Jelikož používám 2x větší paměť, než dovede procesor standardně adresovat, horní bit adresy si nastavuji sám podle hodnoty DPTR. Pokud přeteče hodnota DPTR z 0xFFFF(hex) na 0x0000(hex), nastavím nejvyšší bit adresy na log 1 a umožním tak přístup do horní poloviny paměti. Při dalším přetečení opět bit vynuluji a vrátím se zpět do první poloviny paměti. Tuto operaci provádím jak pro čtení pro zápis.

Data se do paměti ukládají přímo v obsluze přerušení po přečtení z CAN sběrnice. Vysílání dat do PDA se děje neustále pokud je volný systémový čas.

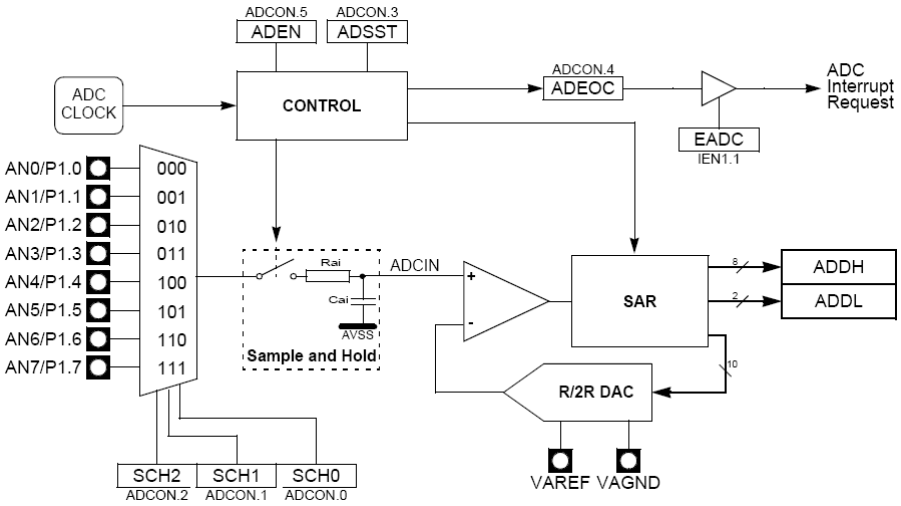
4.4.4 Napětí akumulátorů

Kontrola napětí akumulátoru je řešena dvěma způsoby. První způsob je pomocí integrovaného A/D(analogově digitální) převodníku, takto získáme přesné vzorky napětí. Druhý způsob je pouze dvou stavový(nabité / vybité). Tato hodnota je získávána z DC-DC měniče, který porovnává svoji referenční hodnotu napětí se vstupní hodnotou patřičně upravenou děličem. Výsledek je na výstupním pinu LBO. Hodnota je jako jedna z mnoha zpráv posílána se svým identifikátorem do PDA. Měření napětí se opakuje 10-krát a ve výsledku se vyloučí největší a nejmenší hodnota a je spočítán průměr. Tímto se vyloučí případné možné chyby měření.

4.4.4.1 Nastavení registrů

A/D převodník je možné použít až pro 8 nezávislých vstupů, mezi kterými se přepíná pomocí registru ADCON. Převodník je 10-bitový a může pracovat v těchto režimech:

- ***Standardní konverze*** - v tomto režimu je převod prováděn pouze s přesností 8 bitů. Tento režim používám pro měření, protože nepotřebuji hodnotu napětí



Obrázek 4.3: Zapojení A/D převodníku v procesoru

znát přesně.

- ***Precizní konverze*** - v tomto režimu pracuje převodník se všemi 10 bity. Při převodu dochází k tzv. vypnutí procesoru, aby se snížilo rušení. Procesor je vypnut, porty zůstávají aktivní. V tomto režimu musí být použito pro zjištění hodnoty od A/D převodníku(ADC) přerušení.

Nastavení A/D konverze se děje v následujících registrech:

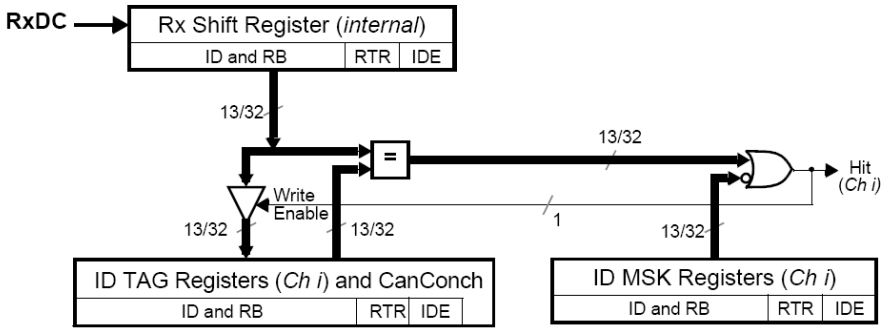
- ADCF(ADC Configuration) – nastavení jednotlivých bitů portu P1, zda se jedná o analogové vstupy nebo standardní datové.
- ADCON(ADC Control Register) – v tomto registru se pomocí bitu ADSST spustí konverze.
- ADCLK(ADC Clock Prescaler) – tento registr nastavuje rychlost převodu danou vztahem $f_{ADC} = f_{cpu\ clock}$ / (4 (or 2 in X2 mode)∗ PRS). Hodnota PRS je 5 bitové číslo (0 – 31).
- ADDH(Data High Byte Register) – v registru se nachází horních 8 bitů dat(9. – 2. bit).
- ADDL(Data Low Byte Register) – název registru je trochu zavádějící, protože nejde o spodní bajt dat, ale jen o nejnižší dva bity(0. a 1. bit).

4.4.5 Nastavení CAN řadiče

Toto nastavení je asi nejsložitější v celém procesoru, představuje 42 konfiguračních registrů. Procesor podporuje standardy CAN 2.0B, jak byly popsány v kapitole 2.

CAN řadič ukládá data do message object[3](dále budu označovat jen objekt). Těchto objektů je celkem 15. Obsahují vždy status registr, kde je definován aktuální stav objektu a informace o identifikátoru, masce a informací týkajících se TTC (Time Trigger Communication). Každý objekt může být nastaven jako vysílací nebo přijímací nezávisle na ostatních.

Nyní bych přiblížil možnosti tohoto převodníku. Pokud chceme vysílat data na sběrnici nastavíme hodnotu identifikátoru odchozí zprávy do některého z 15 objektů a příslušných registrů. Pokud chceme data přijímat je možně stejným způsobem přijímat jeden ty zpráv nebo využít masky, která umožní přijímat určitý rozsah identifikátorů viz obrázek 4.4.



Obrázek 4.4: Princip filtrování zpráv

Zpráva se uloží do interního registru a následně je porovnávána s uloženými ID a maskami v objektech zpráv. K identifikátoru je přidán ještě bit RTR(data, žádost) a IDE (standart, rozšířená). Kterému objektu zpráva vyhoví, tam je uložena. Porovnání probíhá pomocí logického součinu kde se násobí identifikátor s invertovanou hodnotou masky. Výsledek je bud 1 nebo 0(přijmout nebo nepřijmout). Pokud nastavíme masku na hodnotu 0, tak přijmeme všechny zprávy, pokud na hodnotu 7FF(11 bitový identifikátor CAN 2.0A) nebo 1FFFFFFFF(29 bitový identifikátor CAN 2.0B), pak jen příslušný identifikátor.

Další užitečnou funkcí je režim Autobaud. V tomto režimu nemusíme znát rychlost sběrnice CAN a přesto můžeme číst data. Omezením je, že nelze posílat data jen číst. Nastavuje se bitem Autobaud v CANGCON(CAN General Control) registru.

[3]Místo pro ukládání dat

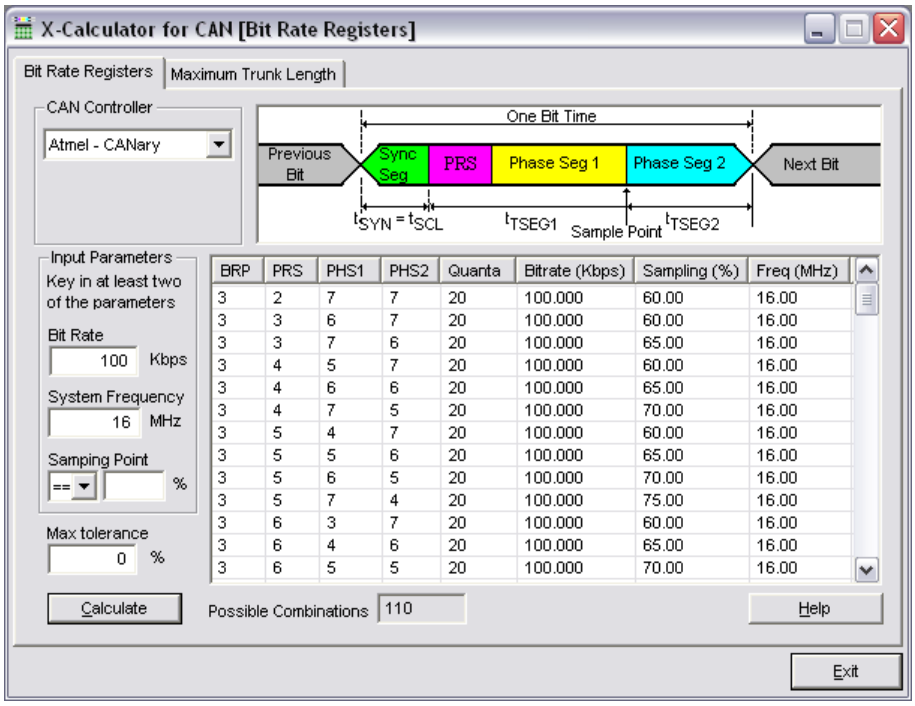
4.4.5.1 Inicializace

Všechny datové registry (identifikátor, maska, data zpráv, status registry) musejí být před použitím vynulovány. Bitem GRES(General Reset) v registru CANGCON se resetuje CAN řadič, po resetu je řadič zakázán. Dále zakážeme přerušení od CAN řadiče(ECAN) a od CAN čítače(ETIM). Potom začneme inicializovat registry. Registr CANPAGE určuje mimo jiné, který objekt je aktivní. Postupně projdeme všechny od 0 – 15 a nastavíme příslušné registry v objektu. Jedná se o tyto registry:

- CANCONCH(Message Object Control) – v tomto registru jsou uloženy informace o zprávě, kolik má dat(DLC), jestli se jedná o standardní nebo rozšířený formát(IDE), je možnost automatické odpovědi na remote frame(RPLV). Dva bity CONCH slouží k nastavení režimu objektu:
  - 00 – objekt je zakázán
  - 01 – objekt je začne vysílat zprávu podle aktuálního nastavení
  - 10 – objekt čeká na příchozí zprávu
  - 11 – objekt čeká na příchozí zprávu, je v režimu buffer. V tomto režimu může přijímat jeden rozsah zpráv více registrů a vytvoří jeden buffer, do kterého se pak zprávy od nejnižšího objektu zapisují.
- CANSTCH(Message Object Status) – informace o stavu objektu význam jednotlivých bitů je níže 4.4.5.2
- CANIDT1(Indentifier Tag) – identifikátor zprávy, standardní (10.–3. bit), rozšířené (28.–21. bit)
- CANIDT2(Indentifier Tag) – identifikátor zprávy, standardní (2.–0. bit), rozšířené (20.–13. bit)
- CANIDT3(Indentifier Tag) – identifikátor zprávy, standardní (nepoužit), rozšířené (12.–5. bit)
- CANIDT4(Indentifier Tag) – RTRTAG bit pro žádost o zaslání dat, rozšířená zpráva zde má navíc (4.–0. bit)
- CANIDM1(Indentifier Mask) – maska zprávy, standardní (10.–3. bit), rozšířené (28.–21. bit)
- CANIDM2(Indentifier Mask) – maska zprávy, standardní (2.–0. bit), rozšířené (20.–13. bit)

- CANIDM3(Indentifier Mask) – maska zprávy, standardní (nepoužit), rozšířené (12.–5. bit)
- CANIDM4(Indentifier Tag) – RTRMSK bit pro žádost o zaslání dat, IDEMSK bit pro maskování rozšířené zprávy, rozšířená zpráva zde má navíc (4.–0. bit)
- CANMSG(Message data) – do tohoto registru zapíšeme data, která chceme odeslat, nebo naopak přečteme přijatá data. Pro všechna data je pouze jeden registr, takže v případě více dat se jen vícekrát zapíše do tohoto registru. Jedná se o 8 bajtový buffer. Pokud chceme přečíst všechna data musíme opakovaně číst z toho registru. (max. 8krát)

Následuje nastavení rychlosti(bit timing) CAN sběrnice. Je možné použít funkci Autobaudrate kdy není třeba nastavovat, ale není pak možnost data na sběrnici zapsat. Nastavení se řeší ve třech registrech CANBT1,2,3(CAN Bit Timing Registers). Registry jsou rozčleněny na bloky po několika bitech a ty reprezentují jednu z hodnot určující délky úseků v jednom bitu. Tyto hodnoty je možné vypočítat pomocí programu X-Calculator, který je volně ke stažení na stránkách firmy Atmel. Zadána je frekvence krystalu, rychlost sběrnice CAN, přesnost a vzorkovací čas. Význam jednotlivých hodnot je vidět přímo v programu viz. obrázek 4.5.



BRP	PRS	PHS1	PHS2	Quanta	Bitrate (Kbps)	Sampling (%)	Freq (MHz)
3	2	7	7	20	100.000	60.00	16.00
3	3	6	7	20	100.000	60.00	16.00
3	3	7	6	20	100.000	65.00	16.00
3	4	5	7	20	100.000	60.00	16.00
3	4	6	6	20	100.000	65.00	16.00
3	4	7	5	20	100.000	70.00	16.00
3	5	4	7	20	100.000	60.00	16.00
3	5	5	6	20	100.000	65.00	16.00
3	5	6	5	20	100.000	70.00	16.00
3	5	7	4	20	100.000	75.00	16.00
3	6	3	7	20	100.000	60.00	16.00
3	6	4	6	20	100.000	65.00	16.00
3	6	5	5	20	100.000	70.00	16.00

Obrázek 4.5: Program X-Calculator - příklad výpočtu

4.4.5.2 Zdroje přerušení

Čtení dat z CAN řadiče se děje výhradně přes přerušení. Přerušení může být vyvoláno následujícími událostmi:

- OVRTIM – přetečení čítače CAN Timer z FFFFh na 0000h. Toto může být např. způsobeno špatným nastavením rychlosti.
- OVRBUF – přetečení bufferu, význam má v případě, že objekty používáme jako jeden buffer o 1–15 položkách podle toho kolik objektu si takto zvolíme. Zprávy jsou pak ukládány do prvního volného objektu z bufferu.
- SERG – Stuff Eror General viz. 2.4.4
- FERG – CRC Error General viz. 2.4.4
- AERG – Acknowledgement Error General viz. 2.4.4
- Message object – bližší význam musíme vyčíst z registru CANSTCH(Message Object Status)
  - DLCW – přijatá zpráva neměla očekávanou hodnotu DLC viz. 2.4.5,
  - TXOK – v pořádku odeslaná data,
  - RXOK – v pořádku přijatá data,
  - BERR – Bit error viz. 2.4.4,
  - SERR – Stuff error viz. 2.4.4,
  - CERR – CRC error viz. 2.4.4,
  - FERR – Form error viz. 2.4.4,
  - AERR – Acknowledgement error viz. 2.4.4.

Jednotlivá přerušení musí být povolena, nastavení je registrech:

- CANGIE(General Interupt Enable) – nastavení povolení přerušení od přijatých, odeslaných dat a chyb.
- CANIE1 a CANIE2(CAN Enable Interupt Message Object 1,2) – nastavení povolení přerušení od určitého objektu 0–15.

Při jednom z těchto přerušení, pokud je povoleno, je zavolána obsluha přerušení. Princip obsluhy popisují následující body:

1. Nejdříve musíme uložit aktuální číslo objektu, je nutné jej zase na konci obnovit, aby nedošlo k poškození dat, která tam jsou uložena.
2. Hledáme, kdo přerušení vyvolal. Postupně projdeme bity v registrech CANSIT1,2. Zde je nastaven bit u příslušného objektu který přerušení vyvolal. Pokud není nastaven žádný bit jedná se o obecné přerušení. Potom hledáme význam v registru CANGIT(General Interupt).
3. Nastavení příslušného objektu zpráv, v případě General Interupt tuto sekci přeskočíme. Analyzujeme registr CANSTCH viz. 4.4.5.2.
4. Provedeme uložení do externí paměti.
5. Vynulujeme registr CANSTCH.
6. Pokud se jednalo o General Interupt, provedeme patřičné kroky a registr CANGIT vynulujeme.
7. Nastavíme původní aktivní objekt v CANPAGE a obsluhu ukončíme.

4.4.6 Hlavní program

Program v procesoru pracuje následujícím způsobem. Po resetu procesoru se nejdříve nastaví vektory přerušení na obslužné podprogramy. V další části se provede inicializace proměnných a nastavení registrů pro obsluhu sériového portu a čítačů/časovačů. Následuje smyčka, ve které se testují tyto stavy:

- Příchozí data od PDA – vyhodnotí se typ dat a nastaví se příslušné registry.
- Data připravená k odeslání do PDA – vyhodnotí původ dat a je zavolán příslušný podprogram pro odeslání dat.
- Chybové stavy – v tomto případě se pošle do PDA číslo chyby.

Ve smyčce se testují řídící bity, které jsou nastavovány v jednotlivých přerušeních. V obsluze přerušení dochází jen k uložení dat do paměti a nastavení příslušného řídícího bitu.

4.5 Uživatelské rozhraní v PDA

4.5.1 Úvod

Programování aplikací pro operační systém PalmOS je možné např. v jazyku C nebo C++. Kromě podpory standardních metod a funkcí C nebo C++ jsou k dispozici ještě speciální optimalizované funkce. Tyto funkce jsou např. pro práci s řetězci, konverzi číselných hodnot, práce s čísly v pohyblivé desetinné čárce a také práce s pamětí. Tyto funkce jsou zahrnuty v hlavním hlavičkové souboru PalmOS.h. Tento soubor je základem každé aplikace.

4.5.2 Operační systém PalmOS

Operační systém nepoužívá žádný souborový systém jak jsme zvyklí u osobních počítačů. Používá systém databází, tyto databáze jsou popsány v kapitole 4.5.4. PalmOS není multitaskový operační systém, to znamená, že v jednom okamžiku může běžet pouze jedna aplikace. Při žádosti o spuštění jiné aplikace, je poslán signál stop aktivní aplikaci, která uloží aktuální stav svých proměnných a ukončí se. Při přepínání mezi aplikacemi můžeme nabýt dojmu, že jsou aktivní obě. To je způsobeno tím, že přepnutí mezi aplikacemi je rychlé a aplikace má možnost se otevřít do stavu, kdy byla ukončena, včetně například rozepsané poznámky.

4.5.3 Základní datové typy

Při vývoji aplikací je dobré používat datové typy definované hlavičkovém souboru PalmTypes.h. Jsou přehledně rozdělené a značené. Výhodou použití těchto proměnných je také to, že není nutné provádět různé, protože funkce v PalmOS pracují právě s těmito typy. Základní datové typy jsou uvedeny v tabulce 4.3

4.5.4 Operační paměť

Jelikož operační systém nepoužívá žádný souborový systém, jsou aplikace a jejich data uloženy v databázích. Paměť je rozdělena na dvě části dynamickou (obdoba paměti RAM u PC[4]) a na paměť pro uložení databází.

Kvůli bezpečnosti dat, jsou databáze umístěny v části paměti, která není přímo přístupná pro zápis, nebo čtení. Do paměti lze přistupovat pouze přes obslužné

[4]osobní počítač

Název	Velikost Byte(s)	Popis
Int8	1	Celočíselná hodnota se znaménkem
Int16	1	Celočíselná hodnota se znaménkem
Int32	1	Celočíselná hodnota se znaménkem
UInt8	1	Celočíselná hodnota bez znaménka
UInt16	1	Celočíselná hodnota bez znaménka
UInt32	1	Celočíselná hodnota bez znaménka
Boolean	1	Logické hodnoty true, false (C/C++ bool)
Coord	2	Označení souřadnic na display
Err	2	Číslo chyby
MemPtr	4	Ukazatel do paměti (C/C++ void)
MemHadle	4	Ukazatel pohyblivého bloku v paměti
LocalID	4	Ukazatel databázového záznamu

Tabulka 4.3: Základní typy proměnných

funkce PalmOS. V případě neoprávněného přístupu aplikace do této paměti je vyvolán reset celého PDA a k poškození dat nedojde. Databáze jsou dále děleny na dva typy:

- Databáze záznamů – slouží k uložení uživatelských dat z různých aplikací.
- Databáze prostředků – určena pro uložení aplikací.

Obsah jednotlivých položek databáze je definován pomocí struktury dat. Například pro uložení filtru CAN zpráv v tomto analyzátoru je použita tato definice:

```
typedef struct CANID {
        UInt16 id[2];      /* Číslo identifikátoru */
        UInt16 mask[2];    /* Maska identifirátoru */
        Char name[50];     /* Název identifikátor */
} CANID;
```

Výhodou tohoto uspořádání je, že aplikace je spuštěna přímo z místa v paměti kde je uložena. Tato metoda šetří paměť a je rychlá, protože není nutné nikam kopírovat data aplikace.

Dynamická paměť i paměť pro uložení databází je členěna na bloky (chunck), které mohou být následujícího typu:

- Přemístitelné – operační systém může proměnnou tohoto typu libovolně přesouvat v paměti v rámci optimalizace. Pokud požíváme proměnnou tohoto typu je nutné před čtením, nebo zápisem dat provést uzamknutí v paměti. Funkce pro zamčení a uvolnění jsou *MemHandleLock()* a *MemHandleUnlock()*. Tento typ proměnných se alokuje pomocí funkce *MemHandleNew()* a uvolňuje funkcí *MemHandleFree()*.

- Nepřemístitelné – operační systém nemůže danou proměnnou přesouvat. Mezi tyto proměnné patří staticky definované proměnné a dynamické proměnné alokované pomocí funkce *MemPtrNew()*;

4.5.5 Základy pro tvorbu aplikací

Základem programu je smyčka ve které se testuje zda nedošlo k nějaké události (stisknutí tlačítka, dotyk pera na displej). V této smyčce běží PDA v úsporném režimu, tím je dosaženo delší doby provozu.

4.5.5.1 Základní funkce PilotMain

Při programování v C/C++ začínáme program funkcí main(), v PalmOS je to funkce PilotMain(). Tato funkce se volá při žádosti o spuštění programu. Možné důvody volání jsou v tabulce 4.4, aplikace dostane informaci o důvodu volání v parametru funkce PilotMain.

Tato funkce musí obsahovat tyto bloky:

- Start programu – zde se provede kontrola potřebné paměti, pustí se hlavní formulář viz. 4.5.5.2.

- Zpracování událostí – smyčka zpracování událostí. Tato část musí zpracovávat požadavek na ukončení. Zpracování dalších událostí záleží na druhu aplikace.

- Ukončení programu – zde se provede zavření všech formulářů, uložení uživatelských dat a dealokace dynamických proměnných pokud byly použity.

4.5.5.2 Formulář

Základním prvkem aplikace je formulář, který můžeme navrhnout některým z programů pro design formulářů viz. kapitola 4.2.2. Většina aplikací má více for-

Důvod spuštění	Popis
sysAppLaunchCmdNormalLaunch	Normální start programu. Uživatel spustil program poklepáním na ikonu.
sysAppLaunchCmdExgAskUser	Spuštěno na základě výzvy po infraportu od jiného zařízení.
sysAppLaunchCmdExgReceiveData	Spuštěno na základě přijatých dat po infraportu určených této aplikaci.
sysAppLaunchCmdExgSyncNotify	Aplikace spuštěna po synchronizaci sync[5].
sysAppLaunchCmdExgSystemReset	Před restartem systému jsou volány všechny aplikace tímto kódem
sysAppLaunchCmdExgTimeChange	Je zavolána aplikace při změně systémového času

Tabulka 4.4: Možné příčiny startu aplikace

mulářů, pokud tedy v programu přistupujeme k ovládacím prvkům formuláře(listbox, edit, button, atd.), můžeme pracovat pouze s ovládacími prvky aktivního formuláře.

4.5.6 CAN analyzátor

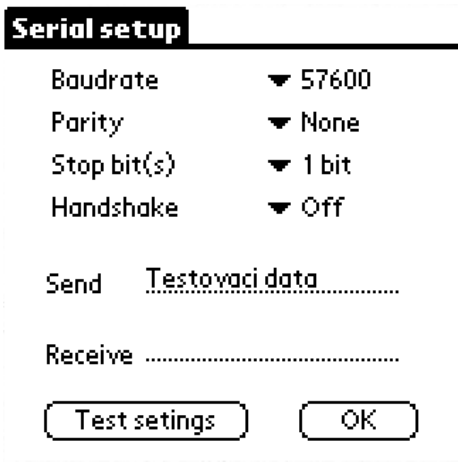
CAN analyzátor se skládá z několika konfiguračních formulářů pro nastavení parametrů CAN řadiče, další formuláře slouží k nastavení parametrů zobrazování a ukládání dat získaných z CAN sběrnice. Pro ukládání dat je použito databází, které lze propojit s tabulkou a data přehledně zobrazovat bez nutnosti data přesouvat do jiných ovládacích prvků.

Pro otestování funkčnosti komunikace mezi hardwarem a PDA slouží formulář *Serial*. Zde je možné vyslat data a porovnat je s příchozími daty obr. 4.6.

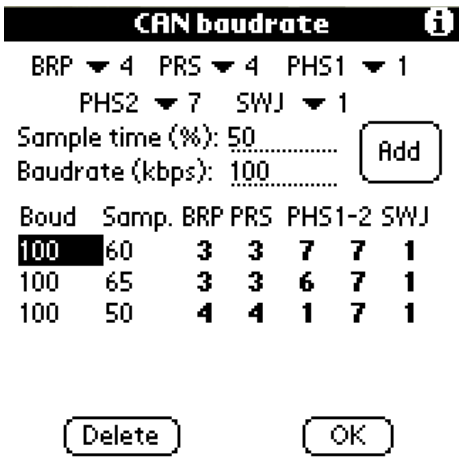
Následují formuláře pro nastavení rychlosti CAN sběrnice, časování a metod analýzy. Pro výběr rychlosti(baudrate) a vzorkovacího okamžiku (sample time) ve formuláři *CAN - modes* obr. 4.8, je nutné nejdříve zadat hodnoty ve formuláři *CAN baudrate* obr. 4.7. Tyto hodnoty je možné spočítat v programu XCalculator obr. 4.5.

Pro nastavení zobrazování a ukládání dat slouží *CAN - interface*, zde je možné nastavit jakým stylem data ukládat. Jestli každou zprávu na nový řádek, nebo přepisovat již zobrazenou zprávu aktuálními daty. Dále je možnost ukládání do různých tabulek. Je možné nastavit časové omezení analýzi.

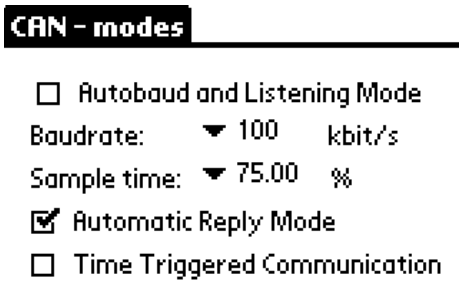
Vysílání zpráv na sběrnici CAN je řešeno ve formuláři *Send CAN Message* na obr. 4.10. Vysílat je možné libovolnou zprávu, nebo vybrat některou z uložených



Obrázek 4.6: Komunikace mezi PDA a procesorem

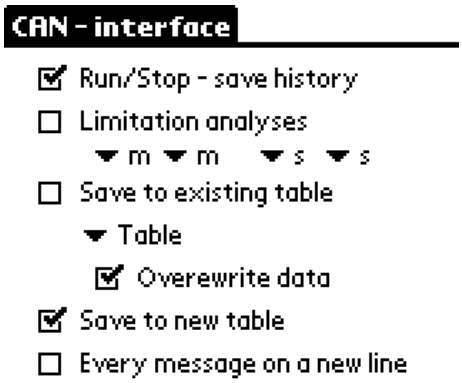


Obrázek 4.7: Nastavení rychlosti CAN sběrnice a časování bitů

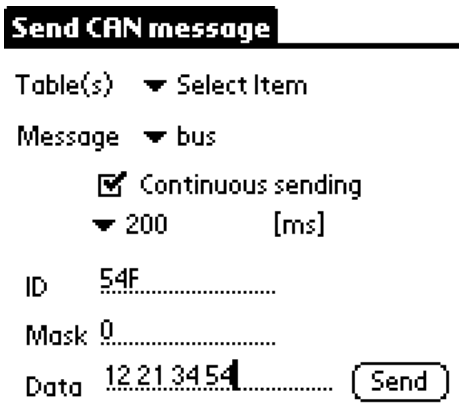


Obrázek 4.8: Nastavení módů CAN převodníku

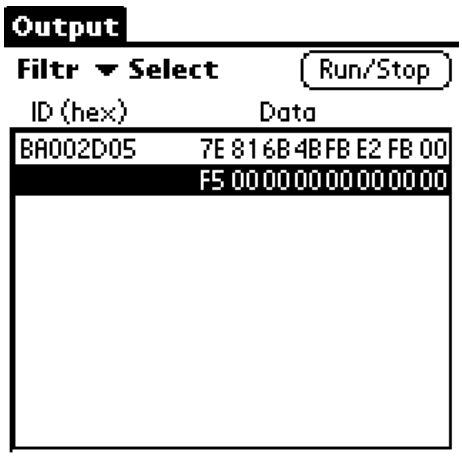
tabulek a v ní zprávu, která byla přijata. Existuje také možnost periodicky vysílat zprávu s volitelnou periodou. Pro zobrazení dat z CAN sběrnice je používán formulář *Output*. Data se zde zobrazují dle nastavení 4.9.



Obrázek 4.9: Nastavení zobrazování a ukládání zpráv



Obrázek 4.10: Vysílání zpráv na CAN sběrnici



Obrázek 4.11: Vysílání zpráv na CAN sběrnici

Kapitola 5

Závěr

Cílem diplomové práce bylo navrhnout a realizovat mobilní CAN analyzátor. Důraz byl kladen na malé rozměry a nízkou spotřebu. Realizace celé práce měla několik částí.

V první fázi bylo nutné prostudovat možnosti realizace, zvolit zařízení pro uživatelské rozhraní a vybrat jednotlivé elektronické součástky. Byla vyrobena vývojová verze plošného spoje, kde byly pro možnosti dalšího testování všechny důležité vývody jednotlivých elektronických obvodů vyvedeny na konektory. Na tomto testovacím zařízení jsem provedl vývoj firmwaru pro procesor a testoval jeho komunikaci s programem v PDA.

Během testování bylo zjištěno, že není možné použít původně navrhovanou rychlost komunikace mezi procesorem a PDA. Bylo proto nutné vyměnit krystal udávající rychlost procesoru, za účelem lepšího nastavení časování sběrnice CAN. Tento zásah měl vliv na rychlost přenosu dat ze sběrnice CAN do PDA, která musela být snížena, více v kapitole 4.4.2. Na funkce analyzátoru to nemělo vliv, jen je omezena doba analýzy čtení všech dat ze sběrnice při jejím značném zatížení.

Přes tento malý problém se povedlo implementovat analýzy dat s maximálními možnostmi, které interní CAN řadič v procesoru nabízí. Zařízení se povedlo úspěšně začlenit do standardní „krabičky" určené pro externí zařízení. Odběr proudu z akumulátorů se při klidovém stavu, kdy procesor není zatížen, pohybuje pod 10 mA. V případě čtení dat z CAN sběrnice se hodnota odebíraného proudu pohybuje max. 25mA v závislosti na zatížení sběrnice CAN. Tyto hodnoty považuji za velmi dobré, analyzátor dokáže s nabitými akumulátory pracovat okolo 20 hodin. Nabíjení akumulátorů (kapacita 700mAh) je v rozmezí 4 – 5 hodin. S těmito výsledky je možné říci, že se povedlo splnit požadavky kladené na mobilní analyzátor.

Literatura

[1] Doc.Ing.Ondřej Vysoký, CSc.: *Elektronické systémy II*. Vydavatelství ČVUT, Praha 1999.

[2] Haasz, V., Sedláček, M.: *Elektrická měření*. Vydavatelství ČVUT, Praha 1998.

[3] FOIT, J., Hudec, L.: *Součástky moderní elektroniky*. Vydavatelství ČVUT, Praha 1998.

[4] Záhlava, V.: *OrCad 10*. Vydavatelství Grada Publishing, a.s., Praha 2004.

[5] Lesný, P.: *Úvod do programování PalmOS*: *http://www.palmos.wz.cz*

[6] ATMEL: *http://www.atmel.com*

[7] CAN in Automation: *http://www.can-cia.com*

[8] CAN dokumentace: *http://fieldbus.feld.cvut.cz/can/*

[9] Flash Memory: *http://www.intel.com/design/flash/articles/what.htm*

[10] Keil Software: *http://www.keil.com*

[11] GNU licence: *http://www.gnu.org/licenses/licenses.html*

[12] Dokumentace Palm OS: *http://www.palmos.com/dev/support/docs/palmos/*

Dodatek A

Struktura přiloženého CD ROM

Přiložený CD ROM v jednotlivých složkách obsahuje:

DiplomováPráce – Tato diplomová práce ve formátu PDF.

Dokumentace – Dokumentace k používanému hardwaru a softwaru ve formátu PDF.

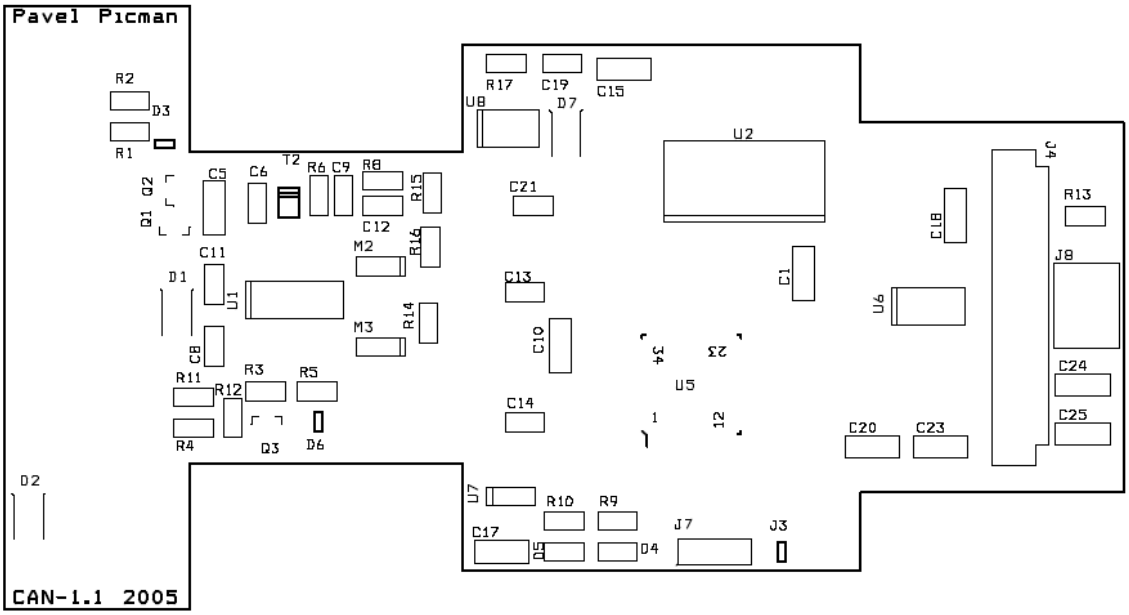
Obrázky – Veškeré použité obrázky ve formátech PNG.

Projekty – Kompletní projekt pro mikroprocesor v prostředí KEIL a kompletní projekt aplikace PDA.

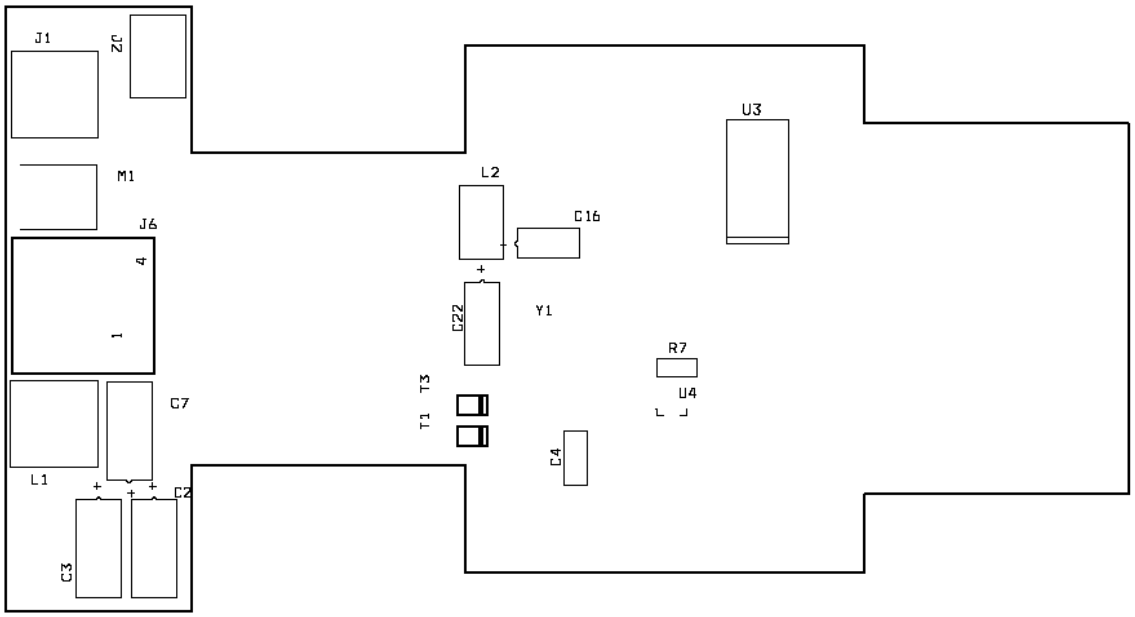
Dodatek B

Realizace výsledné desky

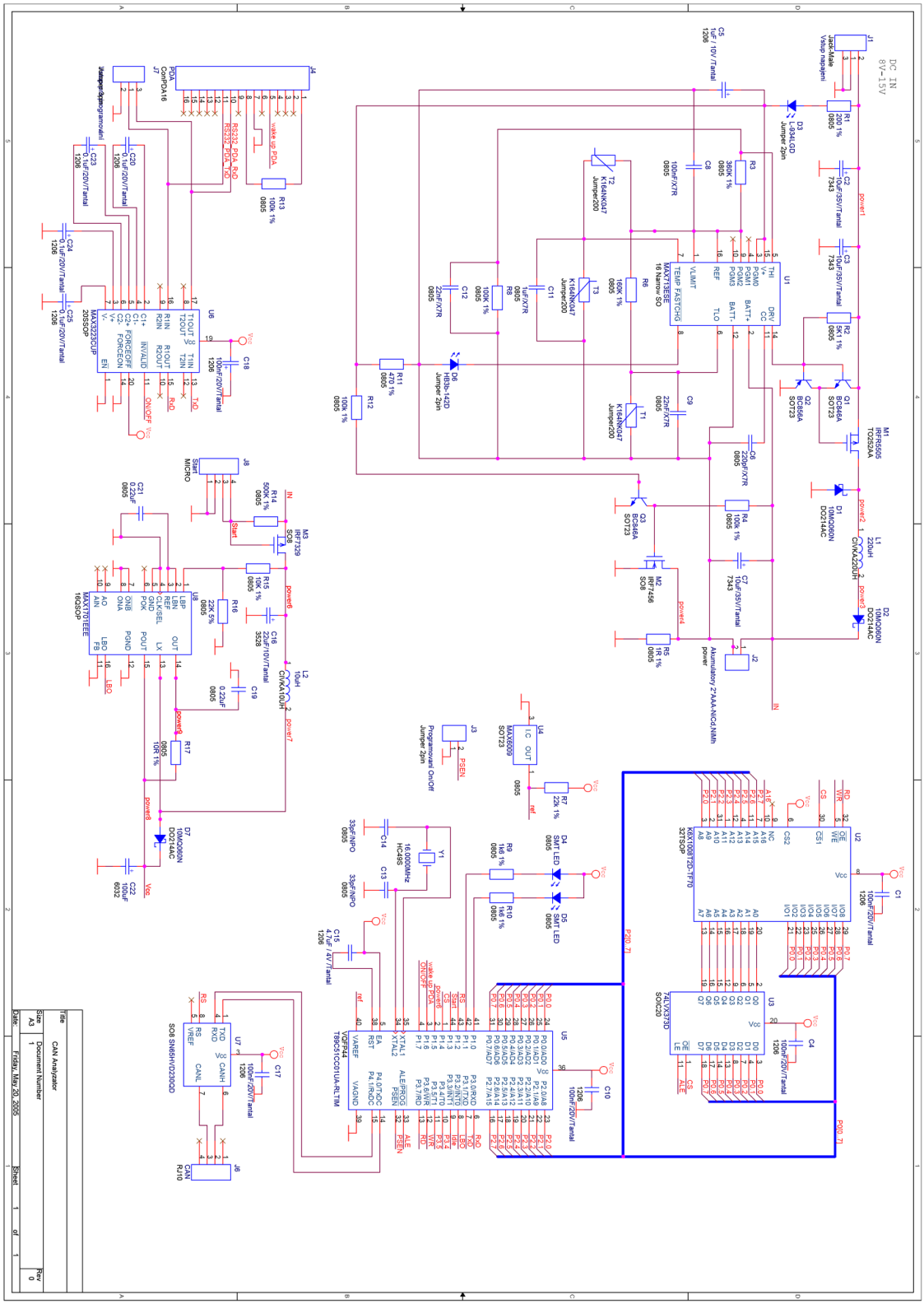
- Schéma zapojení elektroniky hardwaru je na obr. B.3.
- Rozmístění součástek na horní(TOP) straně plošného spoje je na obr. B.1.
- Rozmístění součástek na spodní(BOTTOM) straně plošného spoje je na obr. B.2.



Obrázek B.1: Rozmístění součástek – horní strana



Obrázek B.2: Rozmístění součástek – spodní strana



Obrázek B.3: Schéma zapojení elektroniky CAN analyzátoru