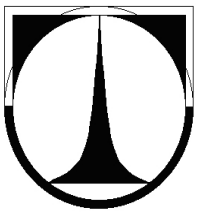
TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií



DIPLOMOVÁ PRÁCE

Liberec 2007 **Tomáš Zeman**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: M 2612 – Elektrotechnika a informatika

Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

Analyzátor CAN sběrnice

CAN bus analyser

Diplomová práce

Autor: **Tomáš Zeman**

Vedoucí diplomové práce: Ing. Pavel Herajn

Konzultant: Ing. Helena Josífková

V Liberci 23. 5. 2007

Zde bude oficiální zadání

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci 23. května 2007

Podpis

Děkuji vedoucímu této diplomové práce Ing. Pavlu Herajnovi za poskytování cenných rad, připomínek a technického zázemí. Dále bych rád poděkoval především Ing. Petru Skalovi za neocenitelnou pomoc při návrhu desky plošných spojů a Ing. Tomáši Mikolandovi za pomoc při jejím osazování, Ing. Tomáši Moravcovi za zpracování návrhu krabičky a laboratoři Katedry výrobních systémů Fakulty strojní za její vyrobení, Martinu Novotnému za trpělivost a snahu při nesčetných diskusích nad problémy této práce.

Můj vděk patří rovněž všem, kteří mě v průběhu této práce podporovali i jinak než odbornými radami.

Abstrakt

Tato diplomová práce popisuje návrh a realizaci analyzátoru CAN sběrnice.

Hlavním cílem bylo vytvořit zařízení, které bude všestranně nenáročné. A to jak na obsluhu, tak na rozměry a spotřebu. Rovněž mělo být snadno připojitelné k osobnímu počítači.

V teoretické části jsou popsány vlastnosti a funkce CAN protokolu, jeho fyzické a linkové vrstvy. Praktická část se zabývá samotným analyzátorem. Nejprve se zaměřuje na jeho hardwarové řešení, následně na řešení softwarové. Hardwarem analyzátoru je elektronické zařízení, řízené mikrokontrolérem, realizující fyzické připojení ke sběrnicím CAN a USB. Softwarové řešení se skládá z firmwaru pro mikrokontrolér a uživatelského rozhraní pro PC, které slouží k ovládání analyzátoru a k zobrazování výstupů z CAN sběrnice.

Klíčová slova: analyzátor, sériová sběrnice, CAN sběrnice, CAN protokol, AT90CAN128

Abstract

The diploma thesis describes design and realization of the CAN bus analyser.

The main goal was to create a broadly undemanding device (concerning easy servicing, small size and low consumption).

The theoretical part concerns basic characteristics of the CAN protocol, its physical and data link layer. The practical part deals with the analyser itself. First, it concentrates on the hardware solution, which is followed by the software solution. The hardware of the analyser is an electronic device operated by a microcontroller, which implements physical connection with the CAN bus and the USB bus. The software solution consists of a firmware for the microcontroller and a user interface for PC, which is used to control the analyser and to display the output from the CAN bus.

Keywords: data-analyser, serial bus, CAN bus, CAN protocol, AT90CAN128

Obsah

Seznam tabulek

Seznam obrázků

Seznam použitých zkratek a symbolů

ADC	–	*Analog-to-Digital Converter*, analogově-číslicový převodník
AVR	–	rodina mikrokontrolérů firmy Atmel s pokročilou RISC architekturou
CAL	–	*CAN Application Layer*, aplikační vrstva CAN protokolu
CAN	–	*Controller Area Network*, průmyslová datová sběrnice
CiA	–	*CAN in Automatization*, mezinárodní nezisková organizace uživatelů a výrobců CAN sběrnice
COM	–	*Communication Port*, sériový port počítače
DPS	–	Deska Plošných Spojů
ESD	–	*Electrostatic Discharge*, elektrostatický výboj
FDM	–	*Fused Deposition Modeling*, metoda vytváření modelů
GND	–	*Ground*, zemění
HLP	–	*High Layer Protocols*, protokol vyšší vrstvy komunikačního modelu
ISO	–	*International Standard Organization*, mezinárodní organizace pro normalizaci
ISP	–	*In-System Programming*, metoda programování mikrokontrolérů uvnitř aplikace
JTAG	–	*Joint Test Action Group*, standard pro testování plošných spojů, programování *Flash* pamětí, apod.
LLC	–	*Logical Link Control*, logické řízení linek
LPT	–	*Line Printer Terminal*, paralelní port počítače
MAC	–	*Medium Access Control*, řízení přístupu k médiu
MAU	–	*Medium Access Unit*, jednotka přístupu k médiu
MDI	–	*Medium Dependent Interface*, rozhraní závisející na médiu
MFC	–	*Microsoft Foundation Classes*, knihovna tříd jazyka Visual C++ pro programování ve Windows
MIPS	–	*Millions of Instructions Per Second*, milionů instrukcí za vteřinu, míra výkonu procesorů

NRZ	–	*Non-Return-to-Zero*, metoda kódování bitu
PC	–	*Personal Computer*, osobní počítač
PCI	–	*Peripheral Component Interconnect*, sběrnice používaná pro připojení periférií k základní desce PC
RISC	–	*Reduced Instruction Set Computer*, architektura procesorů s redukovanou instrukční sadou
SAE	–	*Society of Automotive Engineers*, sdružení automobilových inženýrů
SDS	–	*Smart Distributed System*, aplikační vrstva CAN protokolu
SMD	–	*Surface Mount Device*, součástka umožňující povrchovou montáž
SPI	–	*Serial Peripheral Interface*, sériové komunikační rozhraní
SRAM	–	*Static Ramdom Access Memory*, statická paměť s náhodným přístupem
TTL	–	*Transistor-Transistor Logic*, tranzistorově tranzistorová logika, 0 – 5 V
UART	–	*Universal Asynchronous Receiver-Transmitter*, univerzální asynchronní vysílač/přijímač
USB	–	*Universal Serial Bus*, univerzální sériová sběrnice

ÚVOD

Průmyslová sběrnice CAN (Controller Area Network) je sériovou datovou sběrnicí, která byla původně vyvinuta pro komunikaci a přenos dat mezi řídícími systémy v osobních a nákladních automobilech. Jedná se o diferenciální sběrnici, která byla navržena s důrazem na maximální zabezpečení přenášené informace a splňuje nejvyšší nároky kladené na přenos dat v časově kritických aplikacích. K jejím přednostem tedy patří především spolehlivost, vysoká přenosová rychlost (až 1 Mbit/s), snadná implementace a v neposlední řadě vynikající poměr cena/výkon. Díky těmto vlastnostem našla CAN sběrnice využití i v dalších průmyslových odvětvích a těší se široké oblibě jak mezi uživateli tak mezi výrobci. V současnosti má CAN sběrnice dominantní postavení mezi sběrnicemi určenými pro nasazení v automobilech a využívá ji rovněž stále více výrobců řídících systémů, senzorů, akčních členů a automatizační techniky obecně.

Komunikace na CAN sběrnici probíhá prostřednictvím zpráv. Zařízení, která umožňují tyto zprávy přijímat i odesílat a následně komunikaci na sběrnici analyzovat dle požadavků uživatele, se nazývají analyzátory CAN sběrnice. Zpravidla se skládají z elektronického zařízení, které zprostředkovává komunikaci mezi CAN sběrnicí a PC, a z uživatelského rozhraní pro PC, které slouží k ovládání zařízení, k zobrazení přijatých CAN zpráv a umožňuje nad nimi provádět zmíněné analýzy.

Těchto analyzátorů existuje celá řada, od velkých zařízení připojených k osobním počítačům přes PCI karty až po PCMCIA karty do notebooků. Všechna tato řešení však mají několik nevýhod. Především jsou špatně přenositelná, často vyžadují externí zdroj napájení a nelze je rozšiřovat o další funkční celky.

Cílem této diplomové práce bylo navrhnout a realizovat analyzátor CAN sběrnice, který nebude trpět zmíněnými nedostatky, bude všestranně nenáročný, a to jak na obsluhu, tak na rozměry a spotřebu. Bude určen pro standard fyzické vrstvy *High Speed CAN* a bude podporovat standardy CAN protokolu CAN 2.0A a CAN 2.0B.

Práce je členěna do třech částí. V teoretické části jsou popsány základní vlastnosti CAN sběrnice, její fyzické a linkové vrstvy. Hardwarová část se zabývá elektronickým zařízením a softwarová část popisuje realizaci firmwaru a uživatelského rozhraní pro

PC. Každá z těchto částí obsahuje teoretický úvod, ve kterém jsou vysvětleny základní problémy, principy, požadavky a úvahy.

1 CAN PROTOKOL

V této části jsou popsány základní vlastnosti a funkce CAN protokolu na základě specifikací CAN 2.0A [1] a CAN 2.0B [2]. Rovněž je zde stručně zmíněna jeho historie, přednosti a nejčastější využití v praxi. Blíže je vysvětlena fyzická a linková vrstva protokolu, podoba přenosových rámců a systém zabezpečení proti chybám.

1.1 Úvod

Sériový komunikační protokol CAN (*Controller Area Network*) byl původně vyvinut začátkem osmdesátých let firmou Bosch pro řídící systémy v osobních a nákladních automobilech. Byl navržen s maximálním důrazem na zabezpečení přenášené informace proti chybám a měl splňovat nejvyšší nároky kladené na přenos dat v časově kritických aplikacích. Oficiálně byl CAN protokol představen roku 1986 a brzy se díky svým dobrým vlastnostem rozšířil do dalších průmyslových odvětví.

Roku 1992 byla založena mezinárodní nezisková organizace uživatelů a výrobců CiA (*CAN in Automatization*) [3] s cílem poskytovat technické a marketingové informace a především zajistit budoucí vývoj a standardizaci CAN protokolu. Ten byl standardizován v roce 1993 mezinárodní normou **ISO 11898-1** a zahrnuje fyzickou a linkovou vrstvu protokolu dle referenčního modelu ISO/OSI[1]. Jednalo se o specifikaci CAN 2.0A, která byla později rozšířena na specifikaci CAN 2.0B. CiA rovněž roku 1995 definovala dnes hojně využívaný standard CANopen, jako nadstavbu protokolu aplikační vrstvy CAL (*CAN Application Layer*) a podporuje i další čtyři vzájemně nekompatibilní standardy aplikační vrstvy: CANKingdom, DeviceNet a SDS (*Smart Distributed System*). Vedle těchto standardů existuje i řada čistě firemních návrhů protokolů vyšších vrstev neboli HLP (*High Layer Protocols*).

K nejdůležitějším praktickým přednostem CAN sběrnice patří především vynikající poměr cena/výkon, spolehlivost, vysoká přenosová rychlost, snadná implementace a rozšiřitelnost, flexibilita přenosových rychlostí a dosahu, jednoduchá diagnostika komunikace v síti a standardyzace vyšších vrstev dle referenčního modelu ISO/OSI.

[1] ISO/OSI (*International Standard Organization's Open System Interconnect*) definuje sedmivrstvý komunikační model pro síťové protokoly a distribuované aplikace (**ISO 7498**).

Postupem času si CAN sběrnice získala dominantní postavení mezi sběrnicemi určenými pro nasazení v automobilech (řada dalších takových sběrnic např. VAN či ABUS proto zanikla) a v současnosti je využívána předními výrobci automobilů (koncerny Volkswagen, General Motors atd.) jako sběrnice pro řízení a sběr dat v motorové a komfortní části automobilu. Využívá ji rovněž stále více výrobců řídících systémů, senzorů a akčních členů. Dalšími oblastmi jejího využití jsou obecně: řízení vzdálených zařízení, předávání informací v technologiích, průmyslové informační systémy, systémy inteligentních budov atd.

Integrované řadiče CAN sběrnice a budiče podporující různé fyzické vrstvy a přenosová média jsou dnes běžnou součástí sortimentu předních světových výrobců polovodičové techniky (Intel, Texas, Motorola, Atmel atd.).

CAN protokol je v současnosti jedním z nejúspěšnějších síťových protokolů vůbec a je ideálním prostředkem pro komunikaci v systémech reálného času a v časově kritických aplikacích obecně.

1.2 Základní vlastnosti

CAN protokol je protokol typu *multi-master*, kde každý uzel sítě může být řídící (*master*) a řídit chování jiných uzlů. Není tak třeba žádného centrálního řízení nadřazeným uzlem a zjednoduší se řízení celé sítě. Předností tohoto řešení je i vyšší spolehlivost, protože výpadek některého z uzlů nezpůsobí výpadek celé sítě.

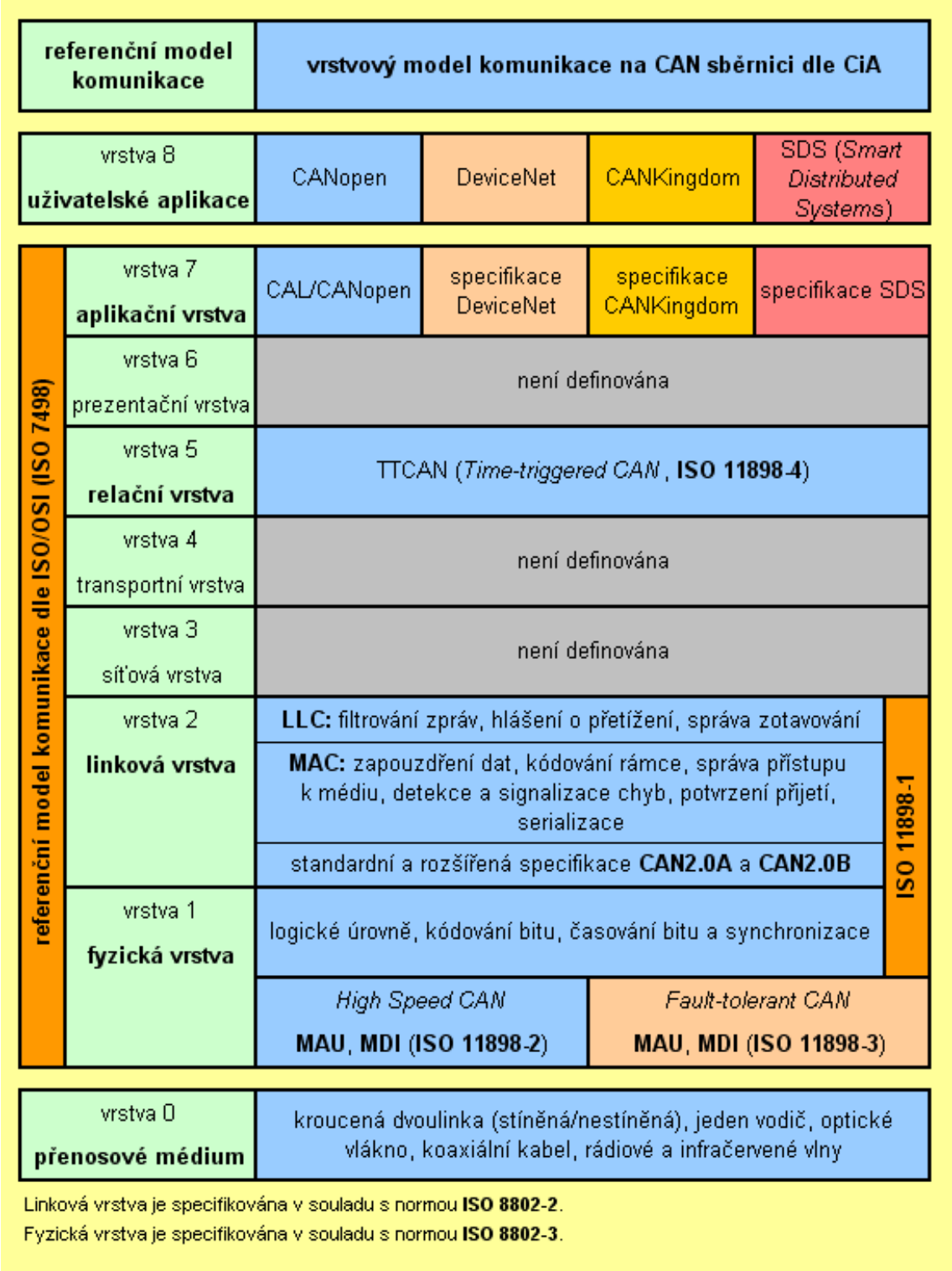
Komunikace mezi uzly probíhá prostřednictvím zpráv (tzv. přenosových rámců, viz kap. 1.5, str. 29) a kolize mezi nimi jsou řešeny na základě prioritního systému (tzv. arbitráže, viz kap. 1.4.1, str. 27), kdy je přenesena vždy zpráva s nejvyšší prioritou. Zprávy nejsou určeny konkrétnímu příjemci a jsou tak přijímány všemy aktivními uzly v síti.

Vrstvový model komunikace na CAN sběrnici dle CiA z pohledu referenčního modelu ISO/OSI je ukázán na obrázku 1. Jednotlivé vrstvy komunikují vždy pouze se sousedními vrstvami a lze je ve stručnosti popsat takto:

Vrstva 8 – Uživatelské aplikace (*User Applications*)[1]

Tato vrstva zahrnuje aplikace koncových uživatelů CAN sběrnice.

[1] Tato vrstva není v referenčním modelu ISO/OSI definována.



Obrázek 1: Vrstvový model komunikace na CAN sběrnici dle CiA

Vrstva 7 – Aplikační vrstva (*Application Layer*)

Pro tuto vrstvu CiA definuje vzájemně nekompatibilní protokoly CAL/CANopen, DeviceNet, CANKingdom a SDS (*Smart Distributed System*) a tato práce se jimi nezabývá.

Vrstva 6 – Prezentační vrstva (*Presentation Layer*)

Není definována.

Vrstva 5 – Relační vrstva (*Session Layer*)

Pro tuto vrstvu CiA poměrně nově definuje protokol TTCAN (*Time-triggered CAN*, **ISO 11898-4**, rok 2000), který je určen speciálně pro systémy řízení v reálném čase a tato práce se jím také nezabývá.

Vrstva 4 – Transportní vrstva (*Transport Layer*)

Není definována.

Vrstva 3 – Síťová vrstva (*Network Layer*)

Není definována.

Vrstva 2 – Linková (spojová) vrstva (*Data Link Layer*)

Tato vrstva je popsána samostatnou kapitolou 1.4, str. 27.

Vrstva 1 – Fyzická vrstva (*Physical Layer*)

Z pohledu referenčního modelu ISO/OSI se tato vrstva skládá ze dvou podvrstev. Horní podvrstva je dle CAN protokolu nazývána fyzickou vrstvou, její specifikace je součástí normy **ISO 11898-1**, a je popsána samostatnou kapitolou 1.3, str. 20. Dolní podvrstvou jsou standardy fyzických vrstev, které jsou specifikovány vlastními normami. O těchto standartech pojednává kapitola 1.3.6, str. 24.

Vrstva 0 – Přenosové médium (*Transmission Medium*)[1]

Fyzické médium tvořící sběrnici. Blíže o požadavcích na přenosové médium pojednává kapitola 1.3.5, str. 23.

[1] Tato vrstva není v referenčním modelu ISO/OSI definována.

1.3 Fyzická vrstva

Fyzická vrstva (*Physical Layer*) CAN protokolu definuje vlastní rozhraní k přenosovému médiu a odlišuje se tak od referenčního modelu ISO/OSI, kde je přenosové médium a jeho detailní elektrický a fyzikální popis součástí definice fyzické vrstvy. Je tak ponechána jistá volnost při implementaci a možnost přizpůsobení se potřebám konkrétní aplikace.

1.3.1 Vysílač a přijímač

Vysílač je definován jako zařízení (uzel v síti) vysílající zprávy daného formátu a přijímač je definován jako zařízení schopné tyto zprávy přijímat.

1.3.2 Logické úrovně

Logické úrovně přenášeného signálu jsou reprezentovány recesivní (*recessive*, `r`, log. 1) a dominantní (*dominant*, `d`, log. 0) úrovní. Tyto úrovně jsou vzájemně komplementární a při jejich současném výskytu musí na sběrnici převážit dominantní úroveň. Přenosové médium tak vlastně realizuje funkci logického součinu, viz tab. 1.

U1	**U2**	**S**
r	r	r
d	r	d
r	d	d
d	d	d

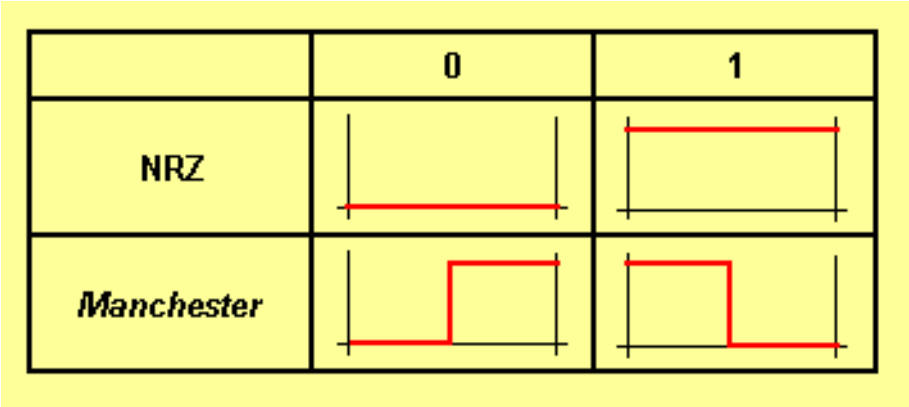
Tabulka 1: Logický součin nad recesivní a dominantní úrovní

Pokud všechny vysílače vysílají recesivní úroveň, pak je logická úroveň na sběrnici recesivní. Pokud vysílá alespoň jeden vysílač dominantní úroveň, je logická úroveň na sběrnici rovněž dominantní. Tato vlastnost je klíčová především pro prioritní systém doručování zpráv (tzv. arbitráž, viz kap. 1.4.1, str. 27).

Skutečná fyzická reprezentace logických úrovní závisí na konkrétní realizaci fyzické vrstvy (viz kap. 1.3.6, str. 24).

1.3.3 Kódování bitu

Pro kódování bitu (*Bit Encoding*) je použita metoda NRZ (*Non-Return-to-Zero*), při které zůstává fyzická úroveň signálu po celou dobu přenosu bitu konstantní. Porovnání této metody s metodou Manchester, viz obrázek 2.



Obrázek 2: Porovnání metod pro kódování bitu – NRZ a Manchester

Při použití tohoto kódování může nastat nežádoucí situace, kdy je na sběrnici delší dobu konstatní úroveň signálu (při sekvenci bitů shodné hodnoty). Tento problém je řešen na úrovni linkové vrstvy a o tomto řešení pojednává kapitola 1.4.2, str. 28.

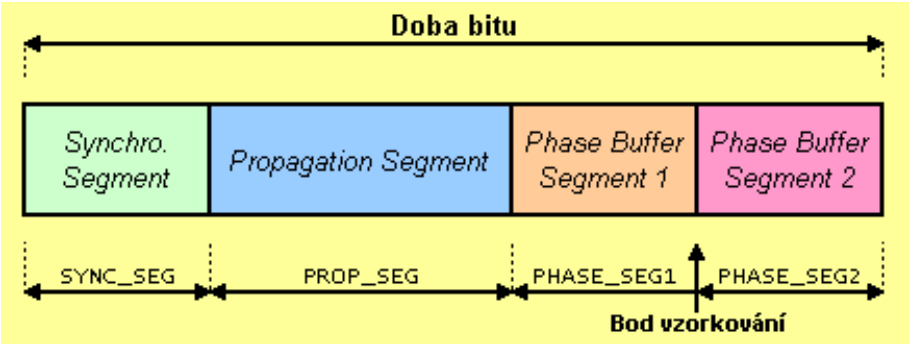
1.3.4 Časování bitu a synchronizace

Přenos jednotlivých bitů v rámci je sériový synchronní a je tedy třeba sofistikovanějších metod synchronizace vysílače/přijímačů.

Přenášené bity následují bezprostředně za sebou a doba potřebná pro přenos jednoho bitu je přesně definována tzv. dobou bitu. Tato doba je dána převrácenou hodnotou přenosové rychlosti sběrnice a je rozdělena na čtyři segmenty, viz obrázek 3.

Synchronization Segment (`SYNC_SEG`)

Slouží k synchronizaci vysílače/přijímačů. Pokud měl předchozí vyslaný bit jinou logickou hodnotu než právě vysílaný, pak se v tomto segmentu změní logická úroveň na sběrnici a vzniklá hrana signálu slouží k synchronizaci. Dále jsou rozlišovány dva typy synchronizace:



Obrázek 3: Časování bitu

- ***Hard Synchronization*** – nastane na začátku rámce, při *recessive-to-dominant* hraně bitu `SOF` (viz kap. 1.5.1, str. 29), kdy je restartována doba bitu.
- ***Resynchronization*** – nastane uvnitř rámce, pokud hrana *recessive-to-dominant* neproběhne uvnitř `SYNC_SEG` segmentu. Pak je prodloužen `PHASE_SEG1` segment (*Bit Lengthening*) či zkrácen `PHASE_SEG2` segment (*Bit Shortening*) daného bitu, čímž se posune bod vzorkování. Kompenzuje se tak například rozdílná frekvence oscilátoru vysílače a přijímače.

Propagation Time Segment (`PROP_SEG`)

Slouží ke kompenzaci doby šíření signálu sběrnicí (od vysílače k přijímačí a zpět), časového zpoždění přijímacích a vysílacích obvodů a doby potřebné k interpretaci analogové úrovně na sběrnici jako logické hodnoty bitu (*Information Processing Time*).

Phase Buffer Segment 1 (`PHASE_SEG1`)

Slouží ke kompenzaci zákmitů signálu na sběrnici a může být prodloužen při resynchronizaci (viz výše).

Sample Point (Bod vzorkování)

Určuje okamžik v čase, kdy je vzorkována analogová úroveň na sběrnici a interpretována jako logická hodnota bitu.

Phase Buffer Segment 2 (`PHASE_SEG2`)

Slouží ke kompenzaci zákmitů signálu na sběrnici a může být zkrácen při resynchronizaci (viz výše).

Jednotlivé segmenty jsou složeny z celočíselných násobků tzv. časového kvanta (*Time Quantum*, TQ), které reprezentuje nejmenší diskrétní jednotku času, kterou jsou schopny vysílače/přijímače rozlišit. Odvozuje se z frekvence jejich oscilátoru, jakožto celočíselný násobek jeho taktů. Pro každý segment je specifikován interval počtu časových kvant, ze kterých se může skládat, viz tabulka 2.

Segment	TQ
SYNC_SEG	1
PROP_SEG	1..8
PHASE_SEG1	1..8
PHASE_SEG2	0..2

Tabulka 2: Počet časových kvant pro jednotlivé segmenty bitu

Celková doba potřebná pro přenos jednoho bitu (doba bitu) se může pohybovat v intervalu od 8 do 25 TQ. To umožňuje velmi flexibilní nastavení přenosové rychlosti sběrnice v závislosti na fyzikálních vlastnostech přenosového média, délky vedení a vlastnostech přijímacích a vysílacích obvodů.

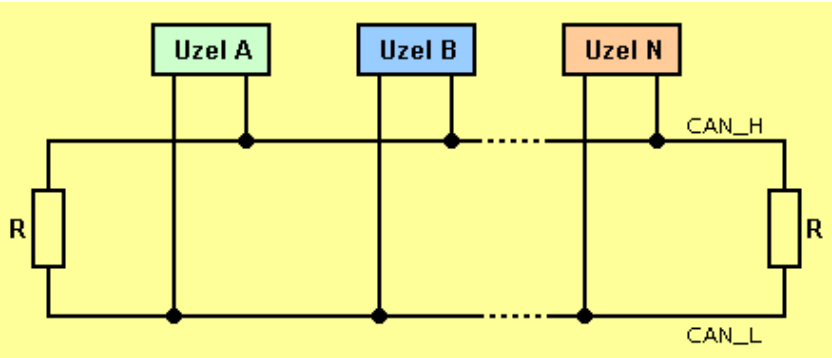
1.3.5 Přenosové médium

Přenosové médium musí především vhodným způsobem fyzicky reprezentovat recesivní a dominantní úroveň tak, aby nad nimi realizovalo funkci logického součinu (viz kap. 1.3.2, str. 20).

Prakticky lze toho médium realizovat mnoha způsoby, například kroucenou dvoulinkou (nestíněnou/stíněnou), jedním vodičem, optickým vláknem, koaxiálním kabelem, rádiovými či infračervenými vlnami atd.

Nejčastěji je však realizováno kroucenou dvoulinkou. Jedná se pak o diferenciální sběrnici tvořenou dvěma vodiči označovanými CAN_H a CAN_L a logická úroveň na sběrnici je definována rozdílem jejich napětí. Tyto vodiče musí být na obou koncích spojeny ukončovacími rezistory, které představují hodnotu charakteristické impedance vedení a zamezují vzniku odrazů, viz obrázek 4.

V případě optického vlákna je dominantní úroveň reprezentována stavem „svítí“ a recesivní úroveň stavem „nesvítí“.



Obrázek 4: Typická struktura CAN sběrnice

1.3.6 Standardy fyzických vrstev

Standardy fyzických vrstev vždy obsahují (v souladu s normou **ISO 8802-3**) dvě podvrstvy: MAU[1] a MDI[2] a v současné době existuje mnoho těchto standardů. Definují elektrické a fyzikální vlastnosti přenosových médií, vysílačů/přijímačů a především definují fyzickou reprezentaci recesivní a dominantní logické úrovně (viz kap. 1.3.2, str. 20).

K nejčastěji využívaným standardům fyzických vrstev patří tyto:

High Speed CAN

Tento standard (vysokorychlostní CAN) je popsán normou **ISO 11898-2** a je nejvíce využívaným standardem fyzické vrstvy. Typickým příkladem využití je například tzv. motorový CAN, který slouží v automobilech k propojení řídící jednotky s ABS, ASR, airbagy apod.

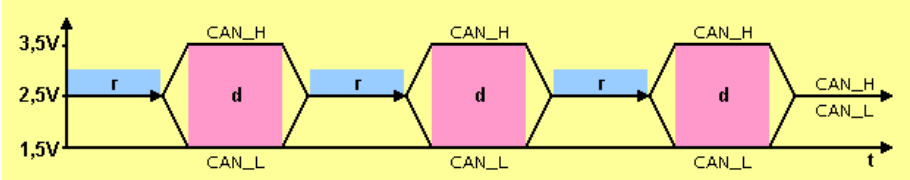
Jako přenosové médium slouží dvouvodičová diferenciální sběrnice na obou koncích zakončená rezistory s nominální hodnotou 124 Ω, s charakteristickou impedancí vedení 120 Ω a odporem max. 70 mΩ/m. Přenosová rychlost této sběrnice je pak 1 Mbit/s při délce vedení max. 40 metrů. V této konfiguraci může být ke sběrnici připojeno maximálně 32 uzlů v závislosti na jejím zatížení. Při přenosové rychlosti sběrnice 50 kbps může být délka vedení až 1 km. Recesivní a dominantní logická úroveň je definována rozdílem napětí obou vodičů, viz tabulka 3.

[1] MAU (*Medium Access Unit*) – jednotka přístupu k médiu.
[2] MDI (*Medium Dependent Interface*) – rozhraní závislé na médiu.

	$U_{CAN\_H} - U_{CAN\_L}$
r	$< 0{,}5$ V
d	$> 0{,}9$ V

Tabulka 3: Fyzická reprezentace logických úrovní standardem *High Speed CAN*

Nominální hodnoty potenciálů vodičů pro dominantní úroveň jsou: U_{CAN_H} = 3,5 V, U_{CAN_L} = 1,5 V, viz obrázek 5.



Obrázek 5: High Speed CAN

Fault-tolerant CAN

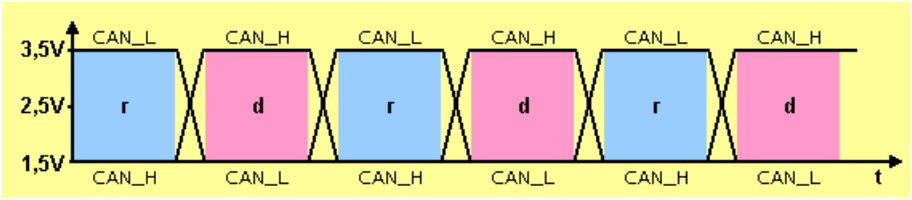
Tento standard fyzické vrstvy (původně *Low Speed CAN*, nízkorychlostní CAN) je popsán normou **ISO 11898-3** a typickým příkladem využití je například tzv. komfortní CAN, který slouží v automobilech k ovládání oken, zrcátek, osvětlení interiéru apod.

Jako přenosové médium rovněž slouží dvouvodičová diferenciální sběrnice. Nominální hodnota zakončovacích rezistorů je však 100 Ω. Přenosová rychlost této sběrnice je 125 kbit/s při délce vedení max. 40 metrů. V této konfiguraci může být ke sběrnici připojeno maximálně 32 uzlů v závislosti za jejím zatížení. Recesivní a dominantní logická úroveň je definována rozdílem napětí obou vodičů, viz tabulka 4.

	$U_{CAN\_H} - U_{CAN\_L}$
r	< 0 V
d	> 0 V

Tabulka 4: Fyzická reprezentace logických úrovní standardem *Fault-tolerant CAN*

Potenciály vodičů U_{CAN_H} a U_{CAN_L} se mohou pohybovat v rozmezí -2 V až 7 V, viz obrázek 6.



Obrázek 6: Faul-tolerant CAN

Hlavní předností této vrstvy je schopnost přenosu signálu i v případě zkratu či přerušení jednoho z vodičů `CAN_H` nebo `CAN_L`. Napěťová úroveň je pak vztažena k zemi (kostře).

Single Wire CAN

Tento standard fyzické vrstvy (jednovodičový CAN) je popsán normou **SAE J2411**[1] a je určen pro aplikace s nízkými požadavky na přenosou rychlost sběrnice a délku vedení.

Jako přenosové médium slouží jednovodičová sběrnice s maximální přenosovou rychlostí 33,3 kbib/s. Typickým příkladem využití je například tzv. *Infotainment CAN*, který slouží v automobilech pro komunikaci zařízení typu rádio nebo GPS navigace s palubní deskou řidiče.

Point-to-point CAN

Tento standard fyzické vrstvy (dvoubodový CAN) je popsán normou **ISO 11992** a vychází z původního standardu *Low Speed CAN* (viz výše). Je však určen speciálně pro komunikaci nákladních automobilů a jejich přívěsů. Pro vozy s jedním přívěsem je definováno dvoubodové (*point-to-point*) propojení, pro vozy s více přívěsy tzv. řetězové (*daisy-chain*) propojení.

Jako přenosové médium slouží nestíněná kroucená dvoulinka, tedy opět diferenciální sběrnice. Její přenosová rychlost je 125 kbit/s při maximální délce vedení 40 metrů.

[1] SAE (*Society of Automotive Engineers*).

1.4 Linková vrstva

Datová linková (spojová) vrstva (*Data Link Layer*) je složena (v souladu s normou **ISO 8802-2**) ze dvou podvrstev:

LLC (*Logical Link Control*)

Pracuje nad přenosovými rámci. Zajišťuje jejich filtrování (*Acceptance Filtering*) a hlášení o přetíženích (*Overload Notification*).

MAC (*Medium Access Control*)

Spravuje samotné rámce. Zapouzdřuje do nich data, zajišťuje jejich kódování, detekci a signalizaci chyb. Dále řídí přístup k přenosovému médiu na základě prioritního systému doručování zpráv a potvrzuje jejich přijetí.

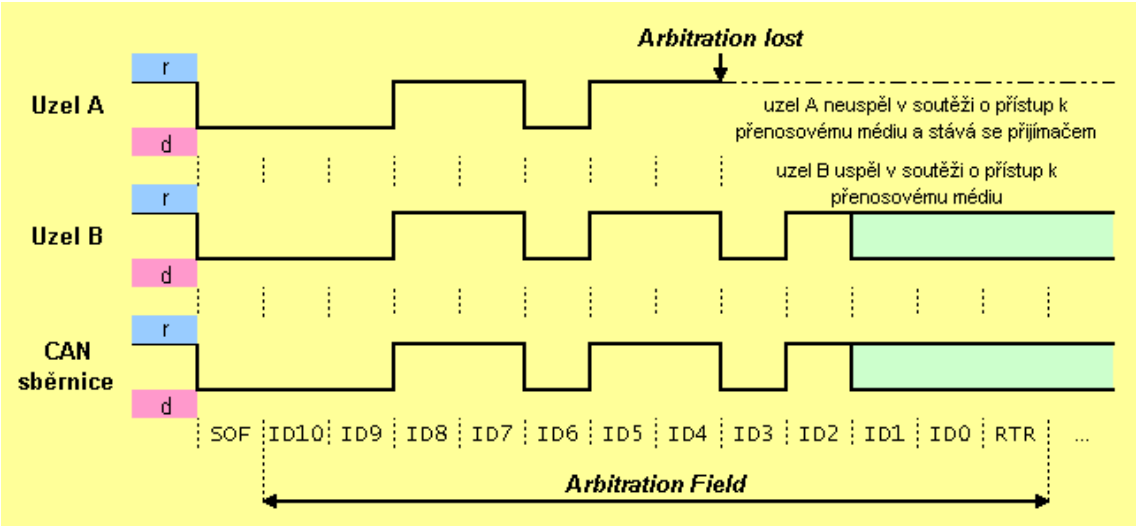
1.4.1 Arbitrážní mechanismus

Arbitráž (*Arbitration*) je proces řízení přístupu k přenosovému médiu. Pokud je sběrnice v klidovém stavu (*Bus Free*), tj. neprobíhá-li na ní žádná komunikace, může kterýkoliv z vysílačů zahájit vysílání. Pokud se o to pokusí více vysílačů zároveň, je tento konflikt řešen pomocí arbitráže nad polem *Arbitration Field* přenášené zprávy (viz kap. 1.5.1, str. 29).

Pro tento proces je zcela klíčová vhodná volba fyzické reprezentace logických úrovní signálu (viz kap. 1.3.2, str. 20), kdy při současném výskytu recesivní a dominantní úrovně musí převážit dominantní úroveň.

Každý vysílač „soutěžící“ o přístup k přenosovému médiu zároveň s vysíláním odposlouchává skutečnou logickou úroveň na sběrnici. Pokud se tato liší od vysílané úrovně, znamená to, že vysílač v této „soutěži“ neuspěl a stává se přijímačem. Takto vysílače „soutěží“ o svou pozici vysílače v síti.

Modelová situace je zobrazena na obrázku 7, kde „soutěží“ vysílače (uzly) A a B o přenosové médium. V okamžiku vysílání bitu `ID3` nesouhlasí logická úroveň na sběrnici s úrovní vysílanou vysílačem A. Tím tento vysílač neuspěl v „soutěži“ o přístup k médiu, stává se přijímačem, a pokusí se data odeslat v nejbližším možném okamžiku, kdy bude sběrnice v klidovém stavu. Vysílač B „zvítězil“, získal přístup k médiu a obhájil svou pozici vysílače v síti.



Obrázek 7: Arbitráž – řízení přístupu k přenosovému médiu

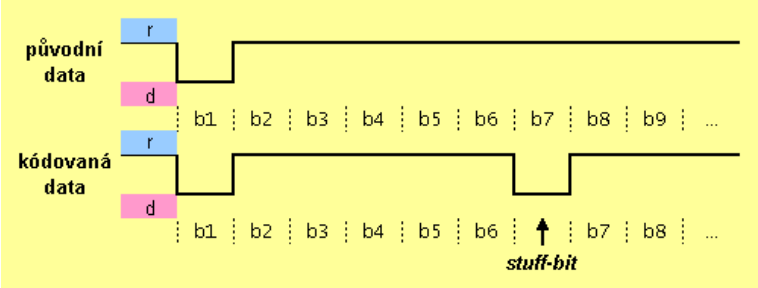
Proces arbitráže je vykonáván vždy nad polem *Arbitration Field* přenášené zprávy, jehož součástí je identifikátor (`ID`), a „zvítězí“ v něm vždy vysílač vysílající zprávu s nižší hodnotou tohoto identifikátoru. Zprávy s nižší hodnotou identifikátoru mají tedy vyšší prioritu než zprávy s vyšší hodnotou identifikátoru. Proto je také tento systém nazýván prioritní systém doručování zpráv.

1.4.2 Kódování rámce

Jak již bylo zmíněno v kapilole 1.3.3, str. 21, při použitém kódování bitu (NRZ) může při přenosu zpráv nastat situace, kdy je na sběrnici delší dobu konstatní úroveň signálu (při sekvenci bitů shodné hodnoty). Aby nedošlo k překročení maximálního časového intervalu mezi dvěma hranami signálu a byla zajištěna správná synchronizace (viz kap. 1.3.4, str. 21), je využito metody vkládání bitů (*Bit Stuffing*). Za každých pět bitů shodné hodnoty je vysílačem vložen komplementární bit (*stuff-bit*) hodnoty opačné. Takto vložený bit je samozřejmě přijímačem odstraněn (*un-stuff*).

Tento princip se neuplatňuje pouze na pole `EOF` přenosového rámce (viz níže), chybový (viz str. 1.5.4, str. 32) a přetěžovací (viz str. 1.5.5, str. 33) rámec.

Příklad tohoto kódování je ukázán na obrázku 8.



Obrázek 8: Kódování dat metodou *Bit Stuffing*

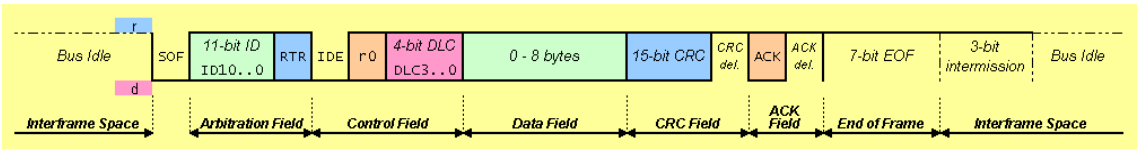
1.5 Přenosové rámce

Komunikace dle CAN protokolu probíhá prostřednictvím přesně definovaných zpráv, tzv. přenosových rámců či telegramů. Tyto se liší svou funkcí, způsobem použití a strukturou.

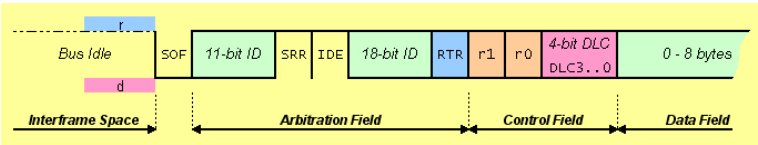
V současné době jsou definovány dva formáty přenosových rámců: standardní (*Standard*, dle specifikace CAN 2.0A) a rozšířený (*Extended*, CAN 2.0B). Liší se pouze délkou pole identifikátoru (`ID`) přenosového rámce a mohou být používány současně.

1.5.1 Obecná struktura přenosového rámce

Přenosový rámec obou formátu se obecně skládá z několika částí, viz obrázek 9 a 10.



Obrázek 9: Obecná struktura standardního formátu přenosového rámce



Obrázek 10: Obecná struktura rozšířeného formátu přenosového rámce

Start of Frame (SOF)

Dominantní bit, jehož hrana *recessive-to-dominant* slouží k synchronizaci přijímačů vzhledem k vysílači (viz kap. 1.3.4, str. 21).

Arbitration Field

Nad tímto polem probíhá proces arbitráže (viz kapitola 1.4.1, str. 27).

- ***Identifier*** (ID) – představuje jedinečný 11 (CAN 2.0A) nebo 29 bitový (CAN 2.0B) identifikátor v rámci celé sítě. Nižší hodnota znamená vyšší prioritu. Sedm nejvyšších bitů (ID10..ID4) nesmí nabývat recesivní úrovně.
- ***Remote Transmission Request*** (RTR) – identifikuje datový a vzdálený rámec. Pokud je bit recesivní úrovně, jedná se o datový rámec, v opačném případě se jedná o vzdálený rámec. Pokud se pokusí dva vysílače vyslat v jednom případě datový a v druhém vzdálený rámec se stejným identifikátorem (ID), uspěje v arbitrážním mechanismu bitem RTR vysílač vysílající vzdálený rámec.

Control Field

- ***Identifier Extension*** (IDE) – identifikuje standardní a rozšířený rámec. Pokud je bit dominantní úrovně, jedná se o standardní rámec, v opačném případě se jedná o rozšířený rámec.
- ***Substitute Remote Request*** (SRR) – bit recesivní úrovně. Tímto bitem je zajištěna vyšší priorita standardních rámců vůči rozšířeným rámcům, pokud je jejich prvních 11 bitů pole *Identifier Extension* shodných.
- **r1**, **r0** – nevyužité bity, vyhrazeny pro budoucí použití.
- ***Data Length Code*** (DLC) – tyto 4 bity (může nabývat hodnot 0 až 7) určují počet bajtů v poli *Data Field*, viz tabulka 5. U vzdáleného rámce je tato hodnota vzdy nulová.

Data Field

Obsahuje 0 až 8 bajtů přenášených dat. Jejich počet je dán hodnotou DLC a u vzdáleného rámce je toto pole prázdné.

CRC Field

- ***Cyclic Redundant Check*** (CRC) - zbytek po dělení všech předcházejících částí rámce polynomem $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + 1$. Tento kontrolní součet s Hammingovou vzdáleností 6, zajišťuje vysoké zabezpečení přenášených dat.

DLC	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

Tabulka 5: Hodnoty DLC a jejich reprezentace logickými úrovněmi

- ***CRC Delimiter*** - oddělovací bit recesivní úrovně. Zajišťuje časovou prodlevu potřebnou pro výpočet a ověření CRC přijaté zprávy.

ACK Field

- ***Acknowledge*** (ACK) – bit recesivní úrovně. Pokud jakýkoliv přijímač v síti přijal odeslanou zprávu, změní během tohoto bitu logickou úroveň na sběrnici na dominantní. Pokud se tak nestane, pokusí se vysílač odestat zprávu znovu.
- ***ACK Delimiter*** – oddělovací bit recesivní úrovně.

End of Frame (EOF)

Ukončení rámce sedmi bity recesivní úrovně. Během vysílání těchto bitů může kterýkoliv uzel, který zaznamenal chybu, vyslat chybový rámec (viz kap. 1.5.4, str. 32). Na toto pole se neuplatňuje kódování *Bit Stuffing* (viz kap. 1.4.2, str. 28).

1.5.2 Datový rámec

Datový rámec (*Data Frame*) slouží k přenosu až osmi bajtů dat a svou strukturou přesně odpovídá obecné struktuře přenosového rámce (viz kap. 1.5.1, str. 29). Žádná data však přenášet nemusí, například při jednoduchých příkazech pro zapnutí/vypnutí nějakého zařízení. Bit RTR v poli *Arbitration Field* je recesivní úrovně.

1.5.3 Vzdálený rámec

Vysláním vzdáleného rámce (*Remote Frame*) daný vysílač žádá o datový rámec se shodným identifikátorem (`ID`) od kterékoliv jiného uzlu v síti.

Jeho struktura odpovídá obecné struktuře přenosového rámce (viz kap. 1.5.1, str. 29), avšak hodnota `DLC` je vždy nulová a rámec tedy neobsahuje žádná data v poli *Data Field* (žádná data nepřenáší, pouze o ně žádá). Bit `RTR` v poli *Arbitration Field* je dominantní úrovně.

1.5.4 Chybový rámec

Chybový rámec (*Error Frame*) je vyslán ihned jakmile kterýkoliv uzel sítě detekuje některou z chyb popsaných v kapitole 1.6, str. 34 a není kódován metodou *Bit Stuffing* (viz kap. 1.3.3, str. 21).

Tento rámec se skládá ze dvou částí. První část vzniká superpozicí tzv. chybových vlajek (*Error Flag*) a druhou tvoří tzv. oddělovač chybového rámce (*Error Delimiter*).

Error Flag

Aktivní uzel v síti se může nacházet v chybově aktivním (*Error Active*) či chybově pasivním (*Error Pasive*) stavu. Mezi těmito stavy uzly přechází na základě stavu dvou vnitřních čítačů: *Transmitter Error Counter* (čítač chyb vysílače) a *Receiver Error Counter* (čítač chyb přijímače). Hodnoty těchto čítačů jsou modifikovány na základě chyb detekovaných při příjmu nebo vysílání přenosových rámců. Pokud uzel generuje příliš mnoho chyb, je odpojen od sběrnice (*Bus Off*), a výstupní obvody jeho budiče přejdou do stavu vysoké impedance.

- ***Error Active*** - v případě detekce chyby na sběrnici vysílá chybový rámec obsahující *Error Flag* tvořený šesti bity dominatní úrovně. Přenášená zpráva je tak poškozena a ostatní uzly v síti začnou rovněž vysílat chybové rámce. Na sběrnici tak superpozicí vznikne sekvence bitů dominantní úrovně, jejíž délka může být 6 až 12 bitů.
- ***Error Pasive*** - v případě detekce chyby na sběrnici vysílá chybový rámec obsahující *Error Flag* tvořený šesti bity recesivní úrovně. Pokud následně detekuje šest bitů shodné úrovně je *Passive Error Flag* dokončen.

Error Delimiter

Skládá se z osmi bitů recesivní úrovně a je vyslán vždy následně po *Error Flag*. Uzel poté čeká na zachycení recesivní úrovně na sběrnici, tj. na ukončení chybového rámce. Pokud k detekci recesivní úrovně nedojde, je znovu vyslán *Error Delimiter* a celý postup se opakuje, dokud nenastanou podmínky pro ukončení chybového rámce.

1.5.5 Přetěžovací rámec

Přetěžovací rámec (*Overload Frame*) je vyslán v případě, že to vyžaduje vnitřní stav přijímače či v případě, že dojde k detekci bitu dominantní úrovně v průběhu pole *Intermission*, a také není kódován metodou *Bit Stuffing* (viz kap. 1.3.3, str. 21).

Tento rámec se skládá ze dvou částí tzv. přetěžovací vlajky (*Overload Flag*) a oddělovače přetěžovacího rámce (*Overload Delimiter*):

Overload Flag

Skládá se ze šesti bitů dominantní úrovně, čímž porušuje konzistenci pole *Intermission*. Pokud ostatní aktivní uzly detekují přetěžovací rámec, odpoví také vysláním tohoto pole.

Overload Delimiter

Oddělovač přetěžovacího rámce se skládá ze šesti bitů dominantní úrovně a je vyslán vždy následně po *Overload Flag*. Uzel poté čeká na zachycení recesivní úrovně na sběrnici, tj. na ukončení přetěžovacího rámce. Pokud k detekci recesivní úrovně nedojde, je znovu vyslán *Overload Delimiter* a celý postup se opakuje, dokud nenastanou podmínky pro ukončení přetěžovacího rámce. Po splnění těch podmínek je navíc vysláno sedm recesivních bitů.

1.5.6 Mezirámcový oddělovač

Mezirámcový oddělovač (*Interframe Spacing*) předchází vyslání datových a vzdálených rámců. Obsahuje pole *Intermission* a *Bus Idle*. Pro chybově pasivní (*Error Pasive*) uzly, které byly v předchozím cyklu vysílači, obsahuje navíc pole *Suspend transmission*.

Intermission

Alespoň 3 bity recesivní úrovně. Toto oddělení přenosových rámců je nutné, aby mohly příjemci uložit přijatou zprávu do vyrovnávací paměti. Žádný uzel nemá oprávnění začít vysílání datového nebo vzdáleného rámce. Jedinou povolenou operací je pouze vyslání přetěžovacího rámce (*Overload Frame*, viz výše).

Buss Idle

Délka tohoto pole není přesně specifikována. V jeho průběhu je sběrnice připravena k přenosu jakéhokoliv přenosového rámce. Pokud některý z uzlů neuspěl v přístupu ke sběrnici, začne přenos ihned po poli *Intermission*. Dominantní úroveň je následně považována za počátek přenosového rámce.

Suspend Transmission

Pokud uzel v chybově pasivním stavu (*Error Pasive*) odeslal zprávu, vyšle 8 bitů recesivní úrovně ještě před tím, než začne zjišťovat, zda je sběrnice volná pro přenos dalších dat. Pokud v jejich průběhu detekuje bit dominantní úrovně (začátek některého z přenosových rámců), stává se tento uzel přijímačem.

1.6 Detekce chyb

Specifikace CAN definuje pět druhů chyb, které mohou nastat při přenosu rámce. Ty pak mohou být rozděleny na chyby na bitové úrovni či chyby na úrovni zpráv.

Chyby na bitové úrovni (*Error at Bit Level*)

- ***Monitoring Error*** – vysílač detekoval na sběrnici jinou logickou hodnotu bitu než právě vysílá. Vyjímku tvoří pole *Arbitration Field* a *Acknowledge*.
- ***Bit Stuffing Error*** – v přenášeném rámci či v polích, na která je uplatněno kódování *Bit Stuffing*, se vyskytla sekvence šesti bitů shodné logické úrovně.

Chyby na úrovni zpráv (*Error at Message Level*)

- ***CRC Error*** – přijímačem vypočtený kontrolní součet (CRC) přijaté zprávy nesouhlasí s kontrolním součtem, který přijatá zpráva obsahuje.
- ***Frame Check Error*** – formát přenášeného rámce se odlišuje od definované struktury (viz 1.5.1, str. 29).

- ***ACK Errors*** – odeslanou zprávu nepřijal žádný uzel v síti a nedošlo tak k detekci bitu dominantní úrovně v poli *Acknowledge*.

2 HARDWAROVÉ ŘEŠENÍ

V této části je popsán návrh a realizace hardwarového řešení analyzátoru CAN sběrnice. Požadavky na toto řešení a základní myšlenky návrhu jsou přiblíženy v úvodní části. Následně je uvedeno a vysvětleno jeho blokové schéma. V dalších částech je zdůvodněn výběr elektronických součástek, vysvětlena jejich funkce a použití. U každé z nich je rovněž znázorněno použité pouzdro s popisem vývodů (pinů) a schéma elektrického zapojení. V závěru je popsán návrh desky plošných spojů a její realizace.

2.1 Úvod

Hardwarovým řešením analyzátoru je elektronické zařízení (dále jen zařízení), řízené mikrokontrolérem, realizující fyzické připojení ke sběrnicím CAN a USB. Toto zařízení zachycuje zprávy z CAN sběrnice a odesílá je přes USB sběrnici do osobního počítače (dále jen PC), kde lze tyto zprávy a veškerou komunikaci na CAN sběrnici pohodlně analyzovat v grafickém uživatelském rozhraní (viz kap. 3.4, str. 55). Tato komunikace není samozřejmě pouze jednosměrná a je tedy možné CAN zprávy nejen přijímat, ale i odesílat.

Základním požadavkem na toto zařízení byla především všestranná nenáročnost ze strany koncového uživatele, přenositelnost, jednoduché použití a snadné připojení k PC. Tyto požadavky se odrazily především ve velikosti výsledného zařízení, a tedy například v SMD provedení všech pouzder použitých elektronických součástek, a ve výběru komunikačního rozhraní s PC. Zařízení mělo dále podporovat standard fyzické vrstvy CAN protokolu *High Speed CAN* (viz kap. 1.3.6, str. 24) a mělo být původně napájeno bateriemi typu AAA. Jedním z požadavků tak byla i nízká spotřeba a co možná nejdelší doba provozu. Jak bude dále vysvětleno, tento požadavek se později neuplatnil. Rovněž měla být ponechána možnost toto zařízení rozšířit o další funkční celky, a tedy vyvést nepoužité vývody mikrokontroléru na konektory.

Nejprve bylo nutno zvolit komunikační rozhraní mezi zařízením a PC. V úvahu připadaly tři možnosti: komunikace přes paralelní LPT port, sériový COM port (RS–232) nebo USB sběrnici.

První možnost byla rychle zamítnuta z důvodu celkové zastaralosti LPT portu, malé a stále klesající podpory ze strany výrobců osobních počítačů, především notebooků, a jeho obtížného programování ve 32-bitových operačních systémech Windows. Oproti tomu COM port je stále součástí naprosté většiny PC. Z notebooků je sice také postupně vytlačován, avšak tento problém lze řešit redukcí na USB, a tím „zachránit" přenositelnost zařízení. Předností sériového COM portu je především snadné programování komunikace i ve 32-bitových systémech Windows. Nespornou výhodou obou výše zmíněných rozhraní (oproti USB) je dále možnost jejich přímého propojení s komunikačním rozhraním UART[1], kterým disponuje naprostá většina moderních mikrokontrolérů.

Poslední možností, která připadala v úvahu, byla komunikace přes USB sběrnici. Ta je oblíbená jak mezi výrobci, tak mezi uživateli, a má slibnou perspektivu do budoucnosti. Zmíněný problém nemožnosti přímého propojení USB a UART řeší převodníky USB-UART, které jsou v dnešní době již na vysoké úrovni a ve formě integrovaných obvodů. Výrobci těchto převodníků k nim také zpravidla poskytují ovladače virtuálního COM portu, který je aktivován vždy po připojení zařízení, které daný převodník obsahuje. Zařízení je tak připojeno k USB sběrnici, a může využívat všech jejích výhod. Aplikace k němu přistupují jako ke klasickému COM zařízení. Tím odpadne i složitější programování komunikace přes USB.

Rozhodující výhodou pro USB však byla možnost napájení zařízení přímo z USB portu. Tuto možnost neposkytuje žádné předešlé rozhraní. USB je schopno dodávat proud až 100 mA při napětí 5 V. Takový proud je pro potřeby napájení naprosté většiny externích zařízení zcela dostačující a napětí 5 V odpovídá napájecímu napětí klasických integrovaných obvodů s napěťovou logikou TTL[2]. Tímto odpadl výše zmíněný problém napájení zařízení pomocí baterií a minimalizace jeho spotřeby, a značně se tak zjednodušilo celé jeho řešení a zvýšila přenositelnost.

Pro připojení zařízení k USB sběrnici byla zvolena samice konektoru USB-B a pro připojení zařízení ke CAN sběrnici byl zvolen samec standardně používaného konektoru CANNON 9 (D-Sub).

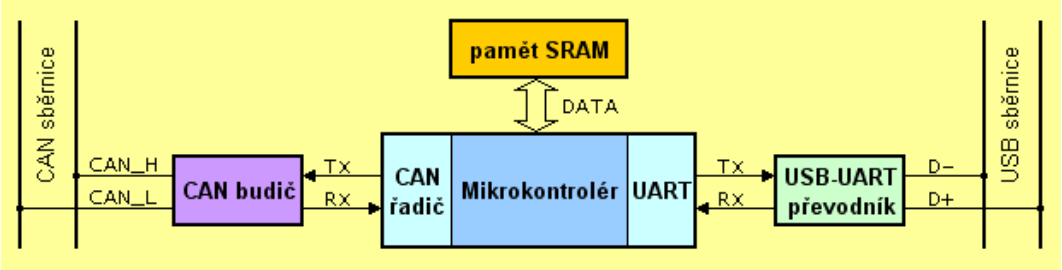
[1] UART (*Universal Asynchronous Receiver-Transmitter*) – univerzální asynchronní vysílač/přijímač

[2] TTL (*Transistor-Transistor Logic*).

Kompletní dokumentace k použitým elektronickým součástkám je vždy volně k dispozici na webových stránkách výrobce dané součástky a je rovněž součástí přiloženého CD-ROM, viz příloha 3.4, str. 62.

2.2 Blokové schéma hardwarového řešení

Blokové schéma hardwarového řešení je ukázáno na obrázku 11.



Obrázek 11: Blokové schéma hardwarového řešení

Jak je vidět z tohoto schématu, zařízení se skládá ze čtyř hlavních částí:

Mikrokontrolér

Propojuje všechny součásti zařízení a řídí jejich činnost na softwarové úrovni programem v sobě obsaženým (firmwarem, viz kap. 3.3, str. 53).

Jeho integrovanou součástí musí být CAN řadič, který na hardwarové úrovni implementuje fyzickou a linkovou vrstvu CAN protokolu (viz kap. 1, str. 16) a plně duplexní UART.

Konkrétně musí zajišťovat tyto funkce: příjem a odesílání CAN zpráv, jejich ukládání a čtení do/z paměti a jejich příjem a odesílání přes UART z/do PC.

USB-UART převodník

Slouží jako rozhraní mezi mikrokontrolérem a USB sběrnicí.

Jeho funkcí je obousměrná transformace USB-UART komunikace.

CAN budič

Slouží jako rozhraní mezi mikrokontrolérem a CAN sběrnicí.

Jeho funkcí je především převod napěťové logiky mezi mikrokontrolérem a CAN sběrnicí v závislosti na standardu fyzické vrstvy CAN (viz kap. 1.3.6, str. 24), pro kterou je tento budič určen.

Zároveň slouží jako galvanické oddělení mikrokontroléru od CAN sběrnice.

Paměť SRAM

Slouží jako vyrovnávací paměť pro přijaté CAN zprávy.

Důvodem jejího použití jsou rozdílné přenosové rychlosti sběrnic CAN a USB (UART). Přenosová rychlost CAN sběrnice může být až 1 Mbit/s. Přenosová rychlost UART může být teoreticky srovnatelná, ale prakticky bude výrazně nižší, typicky například 57600 Bd (cca 56kbit/s).

Interní paměť SRAM mikrokontrolérů je pro tyto potřeby zpravidla nedostačující.

2.3 Mikrokontrolér

Funkce mikrokontroléru byla vysvětlena v kapitole 2.2, str. 38.

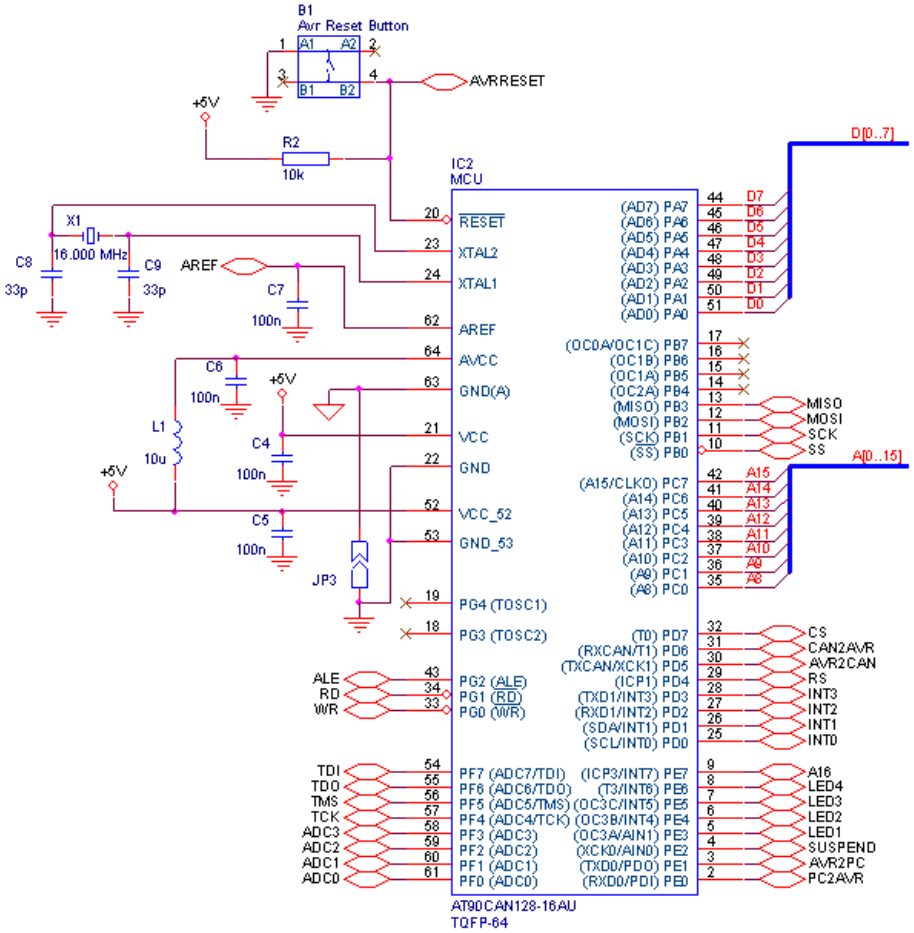
S ohledem na zajištění těchto funkcí byly na mikrokontrolér kladeny následující požadavky: napájecí napětí 5 V (pro přímé napájení z USB), integrovaný UART, integrovaný CAN řadič s podporou standardních i rozšířených rámců a přenosovou rychlostí 1 Mbit/s, možnost připojení externí paměti SRAM a možnost programování jeho firmwaru ve vyšším programovacím jazyce (např. C++).

Těmto požadavkům nejlépe vyhovoval mikrokontrolér AT90CAN128 od firmy Atmel [6]. Tento 8-bitový AVR mikrokontrolér disponuje pokročilou RISC[1] architekturou, solidním výpočetním výkonem (až 16 MIPS[2] při 16 MHz), nízkou spotřebou, a kromě mnoha integrovaných součástí, které nebyly využity, obsahuje: 128kB programové paměti *Flash*, 4kB datové paměti SRAM, 4kB paměti EEPROM, plně duplexní UART, CAN řadič s 15-ti nezávislými objekty CAN zpráv a umožňuje připojení až 64kB externí paměti SRAM a jeho firmware lze programovat ve vyšších programovacích jazycích.

[1] RISC (*Reduced Instruction Set Computer*) – počítač s redukovanou instrukční sadou.
[2] MIPS (*Millions of Instructions Per Second*) – miliónů instrukcí za vteřinu.

Mikrokontroléry rodiny AVR se díky svým přednostem těší velké oblíbenosti mezi uživateli a tedy i dostupnosti nepřeberného množství informací o zkušenostech s nimi, jejich využití, programování a neposlední řadě knižní literatury, např.: [14] a [15].

Označení vývodů mikrokontroléru AT90CAN128 a jejich umístění v pouzdru TQFP-64 je zobrazeno v příloze A, str. 64. Popis jeho vývodů je velmi obsáhlý a nebude zde tedy uveden. Kompletní dokumentace k mikrokontroléru je však volně k dispozici na webových stránkách firmy Atmel.



Obrázek 12: Schéma elektrického zapojení mikrokontroléru AT90CAN128

Schéma elektrického zapojení (viz obrázek 12) obsahuje pouze základní zapojení mikrokontroléru s odrušovacími součástkami, připojeným krystalem (pro CAN řadič je vhodná celočíselná frekvence) a zapojeným resetem přes nízkozdvihové tlačítko. Toto tlačítko je vhodné pro rychlejší a pohodlnější „ladění“ firmwaru, kdy je zpravidla třeba

často mikrokontrolér resetovat. Zbývající vývody jsou pouze pojmenovány a takto budou využity v následujících schématech.

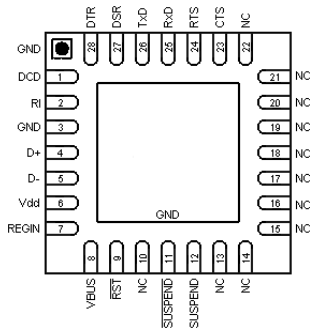
V současné době Atmel nabízí další verze vybraného mikrokontroléru: AT90CAN64 a AT90CAN32. Tyto se od AT90CAN128 liší pouze velikostí interní programové paměti *Flash* (64kB a 32kB) a jsou tak levnější. Tato řada mikrokontrolérů však v době vypracovávání práce nebyla v ČR k dostání. Pouze GM electronic [7], jeden z největších distributorů elektronických součástek v ČR, nabízel závazné objednání mikrokontroléru AT90CAN128.

2.4 USB-UART převodník

Funkce USB-UART převodníku byla vysvětlena v kapitole 2.2, str. 38.

Na tento převodník nebyly kladeny žádné speciální požadavky. Výhodou mohla být pouze jeho dostupnost, snadná implementace, kvalitní ovladače virtuálního COM portu a cena.

Nejčastěji používanými USB-UART převodníky jsou převodníky od firmy FTDI. Jsou velmi oblíbené mezi uživateli, snadno dostupné a relativně bezproblémové. Vybrán však byl převodník CP2102 od firmy Silicon Laboratories [8], který je poměrně nový a zatím nepříliš známý. Je však již v ČR dostupný, ke své funkci potřebuje mnohem méně externích součástek než převodníky od firmy FTDI, jeho ovladače virtuálního COM portu jsou bezproblémové a je dokonce levnější. Jeho jedinou nevýhodou může být bezvývodové pouzdro QFN-28, které může některé uživatele odradit z důvodu obtížnějšího ručního osazování (viz kap. 2.10.2, str. 49).



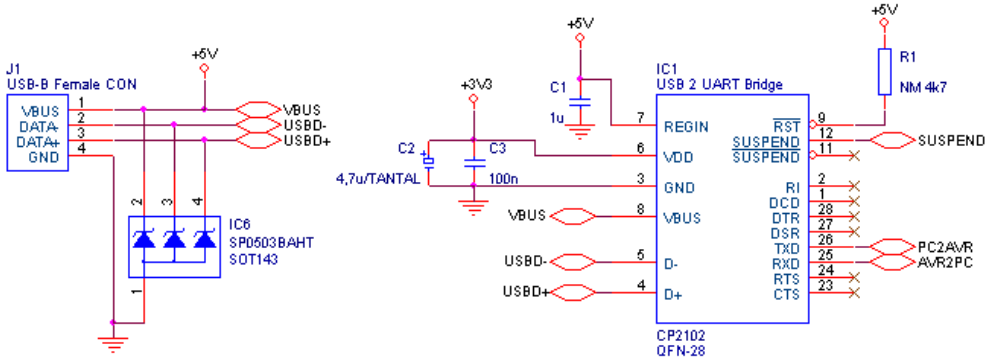
Obrázek 13: USB-UART převodník CP2102 v pouzdru QFN-28

Vývod	č.	Popis
VBUS	8	vstup pro napájecí vodič USB V_{BUS}
D-	5	vstup pro datový vodič USB D_-
D+	4	vstup pro datový vodič USB D_+
REGIN	7	napájecí vstup pro integrovaný napěťový regulátor, 5 V
$\overline{RST}$	9	reset, aktivní log. 0
SUSPEND, $\overline{SUSPEND}$	12, 11	aktivní při vstupu převodníku do USB SUSPEND režimu
TxD	26	výstup dat přijatých z USB
RxD	25	vstup dat k odeslání přes USB
GND	3	zem
V_{dd}	6	vstup/výstup do/z integrovaného napěťového regulátoru

Tabulka 6: Popis vývodů USB-UART převodníku CP2102

Převodník rovněž obsahuje integrovaný napěťový regulátor na 3,3 V, který lze s výhodou využít pro napájení jiných elektronických součástek, v tomto případě CAN budiče (viz kap. 2.6, str. 46).

Označení vývodů USB-UART převodníku CP2102 a jejich umístění v pouzdu QFN-28 je zobrazeno na obrázku 13. Vývody jsou popsány v tabulce 6.



Obrázek 14: Schéma elektrického zapojení USB-UART převodníku CP2102

Schéma elektrického zapojení USB-UART převodníku CP2102 a samice konektoru USB-B je velmi prosté (viz obrázek 14) a vychází z dokumentace k danému převodníku. Součástí tohoto zapojení je obvod SP0503BAHT od firmy Littlefuse [9], který obsahuje tři Zenerovy diody a slouží jako ochrana před elektrostatickým výbojem (*ESD Pro-*

tection). Tento obvod byl u výrobce objednán jako vzorek (*Sample*), avšak k jeho dodání z neznámého důvodu nedošlo.

USB-UART převodník CP2102 byl zakoupen u firmy HT-Eurep Electronic [12].

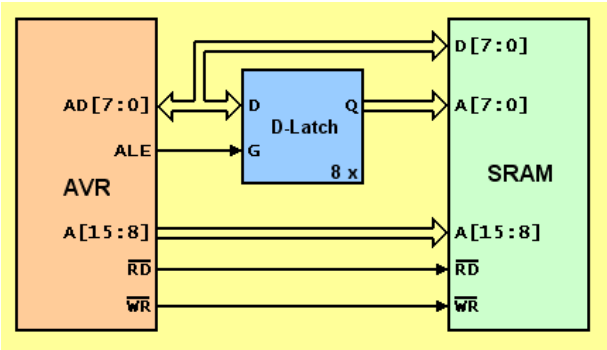
2.5 Externí paměť SRAM

Funkce externí paměti SRAM byla vysvětlena v kapitole 2.2, str. 38.

Mikrokontrolér AT90CAN128 umožňuje adresovat 64k x 8 (64k 8-bitových buněk) paměti a z hlediska programátora je její připojení zcela transparentní. Její použití se nijak neliší od použití interní paměti SRAM mikrokontroléru, tato je pouze definovaným způsobem rozšířena. Přístupová doba k datům v externí paměti je však samozřejmě delší.

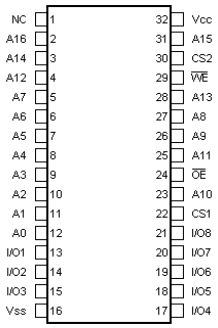
Vzhledem k velice omezené nabídce 64k x 8 pamětí i mezi předními distributory elektronických součástek v ČR byla vybrána paměť o velikosti 128k x 8 bitů (1 Mbit). Jedná se o paměť K6X1008C2D od firmy Samsung [10]. Poslední 16-tý bit (vývod A16, pin 2) adresy paměťové buňky byl připojen na vývod PE7 (pin 9) mikrokontroléru a může být firmwarem nastavován samostatně.

Externí paměť SRAM lze k vybraným ARV mikrokontrolérům připojit v souladu s blokovým schématem uvedeným na obrázku 15 a podrobný popis tohoto zapojení je uveden v dokumentaci ke každému mikrokontroléru, který možnost připojení externí paměti SRAM umožňuje.



Obrázek 15: Blokové schéma připojení externí paměti SRAM k AVR mikrokontroléru

Označení vývodů paměti SRAM K6X1008C2D a jejich umístění v pouzdu SOP-32 je zobrazeno na obrázku 16. Vývody jsou popsány v tabulce 7.



Obrázek 16: Paměť SRAM K6X1008C2D v pouzdru SOP-32

Vývod	č.	Popis
$A_0 \sim A_{16}$	2-12, 23, 25-28, 31	adresové vstupy 0 – 16 bit
$\overline{WE}$	29	zápis dat do paměťové buňky, aktivní log. 0
CS_1, CS_2	22, 30	vstupy pro aktivování paměti
$\overline{OE}$	24	čtení dat z paměťové buňky a zápis na výstup, akt. log. 0
$I/O_0 \sim I/O_7$	13-15, 17-21	vstupy/výstupy dat 0 – 7 bit
GND	16	zem
V_{cc}	32	napájecí vstup

Tabulka 7: Popis vývodů paměti SRAM K6X1008C2D

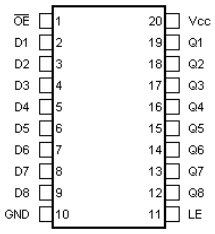
Paměť SRAM K6X1008C2D byla zakoupena u jednoho z největších distributorů elektronických součástek v ČR, GES-ELECTRONICS [11].

2.5.1 D-latch

Funkce D-latch pro připojení externí paměti SRAM k AVR mikrokontrolérům je zcela zřejmá z blokového schématu uvedeného na obrázku 15 a Atmel pro tento účel doporučuje D-latch typu 74AHC. Proto byl vybrán D-latch SN74AHCT573 od firmy Texas [13], který obsahuje osm paralelně zapojených D-klopných obvodů.

Označení vývodů D-latch SN74AHCT573 a jejich umístění v pouzdu TSSOP-20 je zobrazeno na obrázku 17. Vývody jsou popsány v tabulce 8.

D-latch SN74AHCT573 byl u výrobce objednán jako vzorek a s jeho dodáním nebyl žádný problém.

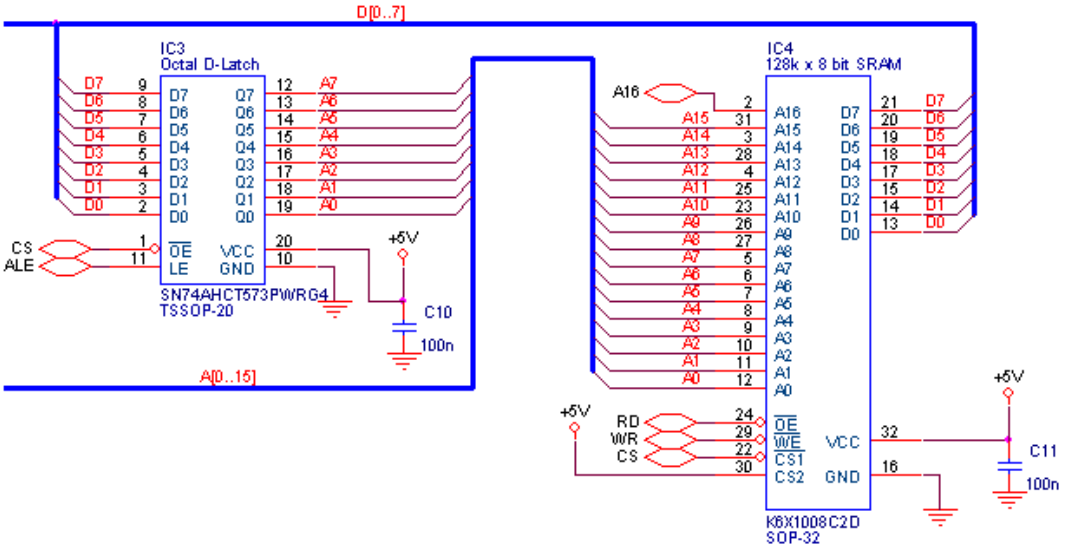


Obrázek 17: D-latch SN74AHCT573PWRG4 v pouzdru TSSOP-20

Vývod	č.	Popis
$D_0 \sim D_7$	2-9	datové vstupy 0 – 7 bit
LE	11	signál pro zápis do vnitřní paměti, aktivní log. 1
$\overline{OE}$	1	aktivace výstupů, aktivní log. 0
$Q_0 \sim Q_7$	12-19	vstupy dat 0 – 7 bit
GND	10	zem
V_{cc}	20	napájecí vstup

Tabulka 8: Popis vývodů D-latch SN74AHCT573

Schéma elektrického zapojení paměti SRAM K6X1008C2D a D-latch SN74AHCT573 je uvedeno na obázku 18.



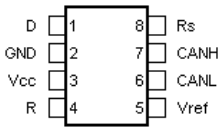
Obrázek 18: Schéma elektrického zapojení paměti SRAM K6X1008C2D a D-latch SN74AHCT573

2.6 CAN budič

Funkce CAN budiče byla vysvětlena v kapitole 2.2, str. 38.

CAN budičů existuje nepřeberné množství od různých výrobců. Nejvíce však zaujal obvod SN65HVD231Q od firmy Texas, díky své bezkonkurenčně nejnižší spotřebě (10 mA) a možnosti různých režimů provozu. Z těchto byl využit režim *Sleep Mode*, ve kterém je budič „uspán“, obvody vysílače a přijímače jsou vypnuty, a spotřeba klesne na 10 μA. Napájecí napětí budiče je 3,3 V, a proto byl k jeho napájení využit integrovaný napěťový regulátor v USB-UART převodníku CP2102 (viz kap. 2.4, str. 41).

Označení vývodů paměti CAN budiče SN65HVD231Q a jejich umístění v pouzdu SOIC-8 je zobrazeno na obrázku 19. Vývody jsou popsány v tabulce 9.



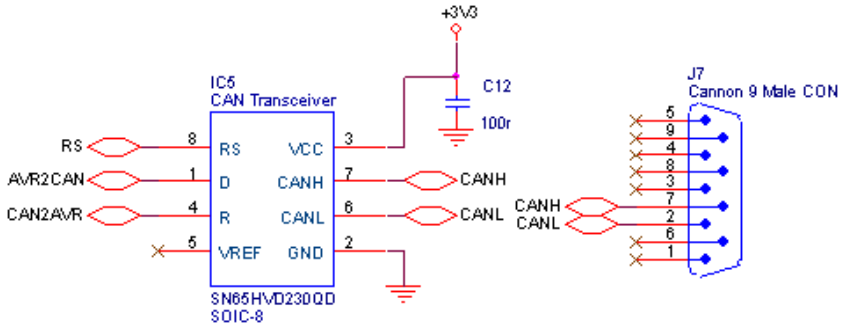
Obrázek 19: CAN budič SN65HVD231QD v pouzdru SOIC-8

Vývod	**č.**	**Popis**
CANL	6	vstup/výstup na CAN sběrnici
CANH	7	vstup/výstup na CAN sběrnici
D	1	vstup dat k odeslání na CAN sběrnici
R	4	výstup dat přijatých z CAN sběrnice
R_s	8	*Sleep mode/Slope Control*
V_{ref}	5	výstup referenčního napětí (odpovídá polovině V_{cc})
GND	2	zem
V_{cc}	3	napájecí vstup

Tabulka 9: Popis vývodů CAN budiče SN65HVD231Q

Schéma elektrického zapojení CAN budiče a samce konektoru `CANNON 9` (D-Sub) je velmi prosté, viz obrázek 20.

CAN budič SN65HVD231Q byl u výrobce objednán jako vzorek a s jeho dodáním nebyl žádný problém.



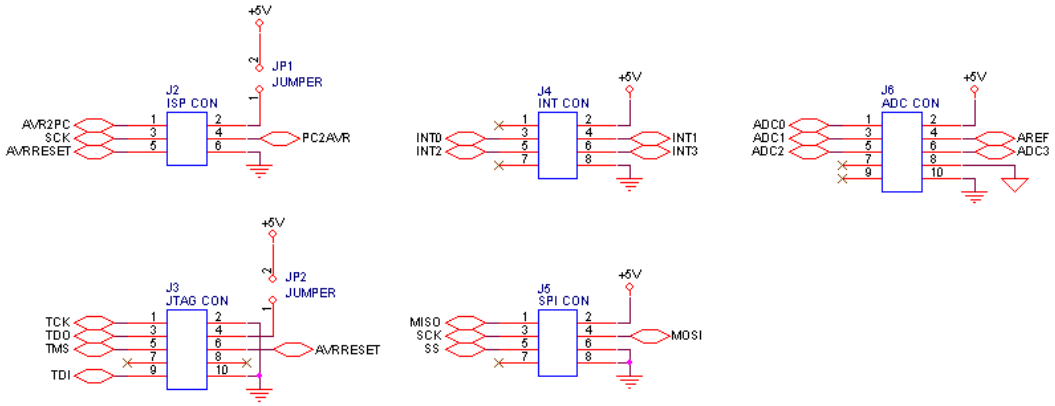
Obrázek 20: Schéma elektrického zapojení CAN budiče SN65HVD231Q

2.7 Konektory

Pro zachování možnosti připojení dalších funkčních celků k zařízení byli nepoužité vývody mikrokontroléru vyvedeny na hřebínkové konektory. Na každý z těchto konektorů je rovněž vyvedeno napájecí napětí 5 V a zem (GND). Každý z konektorů (krom ISP CON) je chráněn proti otočení (*Rotation Save*, při špatném zapojení konektoru se zařízení poškodí) a odpovídá standardnímu zapojení.

Vyvedeny byly konektory pro programování mikrokontroléru (ISP CON, JTAG CON), vnější přerušení (INT CON), SPI rozhraní (SPI CON) a ADC převodník (ADC CON). Při programování mikrokontroléru přes ISP CON či JTAG CON lze pomocí propojek JP1 a JP2 volit mezi napájením zařízení z programátoru či z USB.

Schéma elektrického zapojení konektorů je uvedeno na obrázku 21.



Obrázek 21: Schéma elektrického zapojení konektorů

2.8 Schéma elektrického zapojení

Schéma elektrického zapojení bylo navrženo v návrhovém systému OrCAD, v jeho integrované součásti, schématickém editoru OrCad Capture CIS verze 10.3, a je obsahem příloh B a C, str. 65 a 66.

2.9 Návrh desky plošných spojů

Návrh desky plošných spojů byl vytvořen v návrhovém systému OrCAD, v jeho integrované součásti OrCAD Layout verze 10.3, se snahou o co možná nejmenší rozměry výsledné DPS.

Deska plošných spojů byla navržena jako dvouvrstvá s oboustranným potiskem. Nespadá přesně do žádné třídy přesnosti, použité rozměry se pohybují mezi třídami 4 a 5. Odrušovací součástky jsou umístěny ve spodní vrstvě (*Bottom*) DPS. Rozměry DPS jsou 70 x 50 mm, bližší parametry viz tabulka 10.

Šířka/vzdálenost	**[mils] ([mm])**
napájecí cesty	20 (0,508)
signálové cesty	10 (0,254)
cesta-cesta	10
cesta-prokov	10
cesta-ploška	10
prokov-prokov	10
prokov-ploška	10
ploška-ploška	10
průměr prokovek	0.016 (0,4)

Tabulka 10: Parametry DPS

Pro pohodlné ruční osazování SMD součástek typu rezistor, kondenzátor apod., pro ně bylo zvoleno pouzdro velikosti 1206 (12 x 6 mils).

Rozmístění součástek na DPS viz příloha D, str. 67. Rozvržení vodivých cest na DPS viz příloha E, str. 68.

2.10 Realizace desky plošných spojů

2.10.1 Výroba DPS

Výroba DPS byla zadána soukromé firmě PragoBoard [5]. Tato firma disponuje vyspělými technologiemi výroby DPS, zaručuje jejich vysokou kvalitu a rychlost dodání. V neposlední řadě má výhodnou cenovou nabídku a poskytuje slevy pro studentské zakázky.

Specifikace požadavků pro zakázku výroby DPS je uvedena v příloze G, str. 70. Výrobní podklady pro všechny vrstvy (`TOP`, `BOT`, `SMT`, `SMB`, `SST`, `SSB`) a souřadnicovou vrtačku (`DRL`) jsou součástí přiloženého CD-ROM (viz Obsah přiloženého CD-ROM, str. 62). V souboru `info.txt` je dále popsán význam těchto souborů a jejich formát.

Dodaná deska plošných spojů byla skutečně vysoké kvality. Jeden problém se však vyskytl. Nebyly vyvrtány pravé otvory pro diody `D1-4`, ačkoliv ve výrobních podkladech uvedeny byly. Tento drobný problém byl však snadno vyřešen pomocí ruční vrtačky. Fotografie neosazené DPS viz příloha H, str. 71.

2.10.2 Osazení DPS

Deska plošných spojů byla osazována ručně. Všechny pájecí plošky byly výrobcem pocínovány a pájení mikropájkou tak bylo poměrně snadné. Vyvstaly pouze dva problémy.

Prvním problémem bylo osazení součástky CP2102 (USB-UART převodník) v pouzdru QFN-28. Pouzdra QFN jsou bezvývodová (*Leadless*), a proto je lze jen velmi obtížně pájet mikropájkou. Mnoho lidí se tak snaží těmto pouzdrům vyhnout, avšak jejich osazení je poměrně jednoduché. Pájecí plošky však musí být precizně provedeny a je vhodné, aby alespoň o jeden milimetr přesahovaly okraj pouzdra. Pak je stačí pomazat tenkou vrstvou cínové pasty, precizně usadit součástku, lehce ji přimáčknout a horkovzdušnou pistolí opatrně zapájet. Poté je nutno odstranit případné zkraty mezi pájecími ploškami, které jsou převážně způsobeny drobnými kuličkami cínu, nejlépe ostrým hrotem, případně pomocí mikropájky a většího množství kalafuny. Tyto součástky je zpravidla vhodné osazovat jako první, protože horkovzdušná pistole celou DPS a především

okolí součástky silně prohřeje a mohlo by tak dojít k poškození či odpájení již osazených součástek.

Druhým problémem byly poměrně malé pájecí plošky pro vývody součástky K6X1008C2D (paměť SRAM) v pouzdru SOP-32. Tento problém byl vyřešen lehkým přihnutím vývodů („nožiček“) součástky k jejímu pouzdru. Tak bylo získáno potřebné místo na pájecích ploškách a součástka byla snadno zapájena. Drobnou komplikací (způsobenou špatnou manipulaci s DPS) bylo stržení pájecí plošky vývodu 32 (V_{cc}). Tato komplikace však byla snadno vyřešena pomocí drátové propojky.

Fotografie osazené DPS viz příloha I, str. 72.

2.10.3 Krabička

Původně nebylo uvažováno o vložení zařízení do krabičky, a tak s ní ani nebylo počítáno při návrhu DPS, jejich rozměrů, případných děr pro šrouby, ani jiných úchytů. Tím byla téměř vyloučena možnost umístění zařízení do některé ze standardních krabiček, které jsou běžně k dostání.

Pro snazší a bezpečnější manipulaci se zařízením je však krabička velmi vhodná a byla tedy vyrobena „na míru“. Navržena byla v systému NX Unigraphics a posléze vyrobena z plastu ABS strojem Prodigy firmy Stratasys technologií FDM (*Fused Deposition Modeling*) na Katedře výrobních systémů Fakulty strojní.

Fotografie krabičky viz příloha J, str. 73.

3 SOFTWAROVÉ ŘEŠENÍ

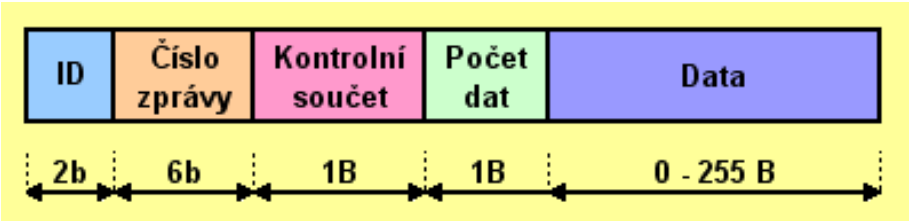
V této části je popsán návrh a realizace softwarového řešení analyzátoru CAN sběrnice. Toto řešení se skládá z několika částí a každé je věnována samostatná podkapitola.

3.1 Úvod

Softwarové řešení analyzátoru se skládá ze dvou hlavních částí. První částí je firmware mikrokontroléru, druhou pak uživatelské rozhraní pro PC. Tyto dvě části mezi sebou komunikují prostřednictvím komunikačního protokolu, který bylo rovněž třeba navrhnout.

3.2 Komunikační protokol

Pro potřebu komunikace mezi firmwarem (viz kap. 3.3, str. 53) a uživatelským rozhraním pro PC (viz kap. 3.4, str. 55) bylo nutno definovat komunikační protokol. Ten měl při bajtově orientovaném přenosu (USB-UART) zajišťovat kontrolu konzistence přenášených dat s různou délkou a významem. Toho bylo uspokojivě dosaženo definováním zprávy s přesně danou strukturou, viz obrázek 22.



Obrázek 22: Struktura zprávy

`ID` – identifikátor typu zprávy

Tyto dva bity identifikují tři typy zpráv:

- **Iniciační zpráva** (`00b`) – slouží pro autorizaci komunikace při jejím zahájení a pro nastavení parametrů analýzy. Přehled druhů iniciačních zpráv viz příloha K, str. 74.
- **Datová zpráva** (`01b`) – slouží čistě pro přenos dat, tj. CAN zpráv. Tato se pouze celá vloží do datové části zprávy a je tedy definován pouze jeden druh datové zprávy (žádný jiný druh dat není třeba přenášet).

- **Chybová zpráva** (`10b`) – slouží především pro oznamování chyb vzniklých při komunikaci či běhu programu (firmwaru). Tyto zprávy jsou jednoznačně definovány svým druhem a tedy nemusí, ale mohou obsahovat datovou část. Přehled druhů chybových zpráv viz příloha K, str. 74.

Druh zprávy

Číslo v těchto šesti bitech (`0` až `63`) identifikuje druh zprávy. Pro každý typ zprávy tedy může být definováno až 64 druhů zprávy, například může být definováno až 64 druhů chybové zprávy. Datová zpráva má však pouze jeden druh a jeho číslo je `0`.

Kontrolní součet

Kontrolní součet v tomto bajtu je doplňkem do nulového jednobajtového součtu všech bajtů zprávy. Slouží k ověření neporušené konzistence doručených dat, která mohla být narušena během přenosu. Na tomto místě by bylo jistě vhodnější využít například CRC kód, který by případnou chybu odhalil mnohem spolehlivěji. Algoritmus jeho výpočtu je však poměrně náročný, a proto k tomu z důvodu obav nad přílišným zatížením mikrokontroléru nedošlo.

Počet dat

Číslo v tomto bajtu udává počet bajtů přenášených dat (v poli data). Může jich tedy být 0 až 255.

Data

0 až 255 bajtů dat.

Podrobný popis všech druhů zpráv, jejich parametrů a funkcí by byl natolik obsáhlý, že zde nebude uveden. Příloha K, str. 74 obsahuje alespoň základní přehled podporovaných druhů zpráv.

Implementované zprávy zdaleka nevyčerpávají všechny možnosti mikrokontroléru a CAN řadiče, avšak díky vhodnému řešení firmwaru mikrokontroléru (viz kap. 3.3, str. 53) a komunikačního protokolu lze nové druhy zpráv a jejich funkce implementovat velmi snadno.

3.3 Firmware mikrokontroléru

Firmware je program, který řídí činnost mikrokontroléru, jeho integrovaných součástí a periférií k němu připojených.

Původně bylo zamýšleno programovat firmware ve vyšším objektově orientovaném jazyce C++, zjednodušit a zrychlit tak jeho vývoj. Z obav nad výkonností takového řešení byl však nakonec zvolen programovací jazyk assembler [14] [15], který má nejblíže samotnému hardwaru a umožňuje tak maximální využití výkonu mikrokontroléru. Vývoj firmwaru probíhal ve vývojovém prostředí AVR Studio verze 4 SP4, které je volně ke stažení na webových stránkách výrobce mikrokontroléru Atmel [6].

K programování interní programové paměti *Flash* přímo z AVR Studia byl využíván ISP programátor Biprog [16], který dokáže spolupracovat s programem AVRProg i s pluginem STK500.

Projekt firmwaru se skládá z několika bohatě komentovaných zdrojových souborů:

`ca.asm` – hlavní soubor.

Obsahuje hlavní smyčku programu. Dále vkládá všechny `'*.inc'` soubory a definuje proměnné v programovém a datovém segmentu.

`can128def.inc` – soubor definic pro mikrokontrolér AT90CAN128.

`cadef.inc` – soubor definic.

Obsahuje definice registrů, I/O registrů, řídících a stavových vlajek, konstant a identifikátorů UART zpráv.

`caint.inc` – vektory přerušení a jejich obsluhy.

`camac.inc` – makra.

`casub.inc` – podprogramy.

`caumh.inc` – obsluhy příchozích UART zpráv.

Hlavní funkcí firmwaru je obsluha komunikačního rozhraní UART a CAN řadiče.

UART implementuje na hardwarové úrovni pouze bajtově orientovanou komunikaci (data jsou z pohledu programu a programátora přenášena po bajtech). Z tohoto důvodu

bylo nutné definovat komunikační protokol (viz kap. 3.2, str. 51) a implementovat jeho podporu softwarově. Komunikace tak probíhá ve formě tzv. UART zpráv. Řešení tohoto problému bylo poměrně náročné, vyžádalo si velké množství času a věnuje se mu naprostá většina zdrojového kódu firmwaru. Podařilo se však implementovat plně duplexní komunikaci dle navrženého protokolu.

CAN řadič implementuje na hardwarové úrovni fyzickou a linkovou vrstvu CAN protokolu (viz kap. 1, str. 16) a komunikace tedy probíhá ve formě CAN zpráv (data jsou z pohledu programu a programátora přenášena po celých zprávách). Na softwarové úrovni je pouze nutné vykonat poměrně složité nastavení registrů CAN řadiče a dále je již jeho obsluha poměrně snadná. Probíhá pomocí obsluhy přerušení, která je vyvolána například při doručení CAN zprávy.

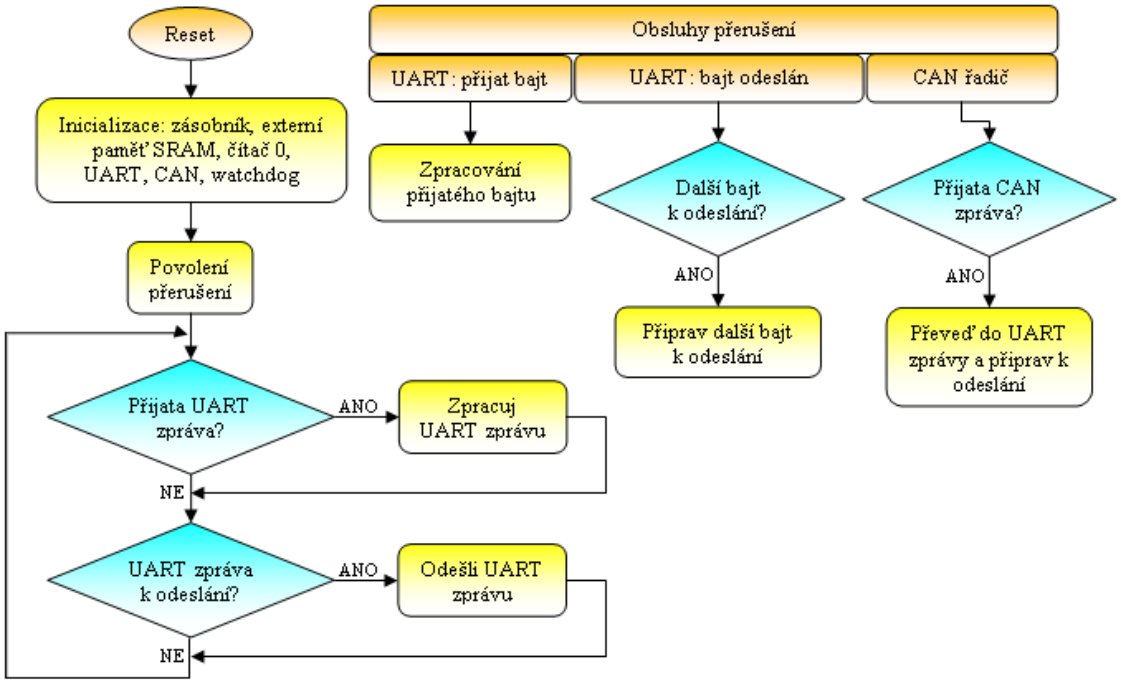
Veškerá komunikace je realizována pomocí obsluh přerušení. Obsluhy přerušení UART se omezují pouze na uložení/načtení bajtu do/z kruhového zásobníku a jejich spotřeba systémového času mikrokontroléru je tak minimální. Naproti tomu je obsluha přerušení CAN řadiče poměrně náročná a při velkém počtu přijímaných CAN zpráv spotřebovává většinu systémového času. Přijaté CAN zprávy jsou v ní vloženy do datové UART zprávy (`UM_DATA`), uloženy do paměti SRAM, a ukazatel na ně se zařadí do fronty UART zpráv k odeslání. Ve volném systémovém čase mikrokontroléru (v hlavní smyčce programu) se pak tyto zprávy postupně odesílají.

V hlavní smyčce programu je zajištěna UART komunikace a veškerá režie s ní spojená. Tuto režii zajišťuje vrstvový systém podprogramů, které mezi sebou sdílí datové členy a nastavují stavové vlajky. Na základě těchto stavových vlajek je hlavní smyčka programu větvena.

Navázání komunikace probíhá pomocí tzv. autobaudu, kdy program čeká na přijetí znaku `'U'`, který je v binárním kódu reprezentován sekvencí bitů `01010101`. Pomocí časovače je změřen časový úsek potřebný pro přenos této sekvence a ze získaného údaje je následně vypočtena a nastavena přenosová rychlost UART. Pro ověření správnosti nastavení je vysláná identifikační zpráva zařízení `UM_CAID` a čeká se na přijetí stejné zprávy z druhé strany. Pokud se tyto shodují, je komunikace úspěšně navázána. Pokud se neshodují, nebo zpráva není za definovanou dobu vůbec přijata, začně navazování komu-

nikace znovu od začátku. Celou tuto proceduru je třeba opakovat v případě chybného kontrolního součtu příchozí zprávy.

Velmi zjednodušený vývojový diagram firmaware je ukázán na obrázku 23.



Obrázek 23: Vývojový diagram firmwaru

3.4 Uživatelské rozhraní pro PC

Uživatelským rozhraním pro PC je aplikace s grafickým rozhraním (dále jen aplikace) pro 32-bitové operační systémy Windows od firmy Microsoft. Aplikace komunikuje prostřednictvím protokolu (viz kap. 3.2, str. 51) s firmwarem (viz kap. 3.3, str. 53) mikrokontroléru a zpřístupňuje tak uživateli snadné využívání jeho funkcí.

Tato aplikace měla uživateli poskytovat možnost pohodlně analyzovat komunikaci na CAN sběrnici přehledným zobrazováním přijatých CAN zpráv, jejich filtrováním, vyhledáváním apod. Měla rovněž nabízet možnost CAN zprávy odesílat, jednotlivě i opakovaně.

V takovém rozsahu však aplikaci nebylo možné zrealizovat. Její vývoj by zabral neúměrné množství času, které nebylo k dispozici. Tato skutečnost tak poskytuje možnost pro budoucí navázání na tuto práci.

Pro vývoj aplikace byl použit objektově orientovaný programovací jazyk C++ s využitím knihovny tříd MFC (*Microsoft Foundation Classes*) [17] a vývojové prostředí Microsoft Visual C++ 6.0. Sériová komunikace [19] byla programována pomocí vynikající volně šiřitelné knihovny *Serial Library for C++* [18]. Tato objektově orientovaná knihovna poskytuje snadný přístup ke všem nastavením a funkcím COM portu ve vlastním vláknu (*Thread*), a umožnuje tak běh komunikace na „pozadí“ aplikace. Existuje rovněž i její verze přizpůsobená MFC.

Činnost aplikace byla nakonec omezena na využití nejzákladnějších funkcí, které firmware mikrokontroléru poskytuje, a lze ji shrnout do následujících bodů:

1. **Navázání komunikace**
   Vysláním znaku 'U' (autobaud), porovnáním přijaté zprávy se zprávou `UM_CAID` a odesláním zprávy `UM_CAID`.

2. **Nastavení CAN řadiče**
   Prostřednictvím iniciační zprávy `UM_ASET`, která obsahuje především informace o požadované přenosové rychlosti.

3. **Nastavení objektu CAN zprávy**
   Prostřednictvím iniciační zprávy `UM_MOBSET`, kterou lze nastavit objekt zprávy na příjem, odesílání apod.

4. **Zahájení analýzy**
   Vysláním iniciační zprávy `UM_ASTART`.

5. **Příjem zpráv**
   Prostřednictvím datové zprávy `UM_DATA`.

6. **Odeslání zprávy**
   Prostřednictvím datové zprávy `UM_DATA`.

7. **Zastavení analýzy**
   Vysláním iniciační zprávy `UM_ASTOP`.

Přijaté CAN zprávy se ukládají do souboru `'can_msg.dat'` v hexadecimálním zápisu jednotlivých bajtů.

ZÁVĚR

Cílem diplomové práce bylo navrhnout a realizovat analyzátor CAN sběrnice. Výsledné řešení se skládá z hardwarové a softwarové části.

Hardwarová část v podobě elektronického zařízení, realizovaného na desce plošných spojů, umožňuje fyzické připojení ke sběrnicím CAN a USB. Jako zdroj napájení pro toho zařízení a komunikační rozhraní s PC byla zvolena USB sběrnice. Elektronické součástky byly vybrány nejen s ohledem na technické požadavky, ale i s ohledem na jejich dostupnost. Ta byla u některých součástek problematická, a tak musely být objednány jako vzorky přímo od výrobců.

Navržené elektrické zapojení bylo obtížné testovat, protože většina použitých součástek byla v SMD provedení pouzdra, která nelze vložit do nepájivého pole. Testována tak byla pouze kritická část zařízení, a sice mikrokontrolér s připojenou externí pamětí SRAM. Pro tento účel byla vyrobena deska plošného spoje, na kterou byl osazen pouze vybraný mikrokontrolér AT90CAN128. Vývody tohoto mikrokontroléru byly vyvedeny na hřebínkové konektory, které bylo možné vložit do nepájivého pole společně s pamětí SRAM v DIP provedení. Toto zapojení bylo následně testováno, byla na něm vyvinuta převážná část firmwaru mikrokontroléru, a posloužilo jako základ pro návrh konečné podoby desky plošných spojů. Se zbývajícími součástkami, jejichž zapojení nemohlo být testováno, nebyl později žádný problém.

První softwarovou částí je vyvinutý firmware, který dokáže využít všech funkcí, které integrovaný CAN řadič v mikrokontroléru nabízí. Rovněž implementuje komunikaci s PC na základě jednoduchého komunikačního protokolu, který byl pro tento účel navržen. ISP programátor sloužící pro zápis firmwaru do programové paměti mikrokontroléru byl sestaven na nepájivém poli. Firmware byl realizován v programovacím jazyce assembler a bylo tak maximálně využito výkonu mikrokontroléru.

Poslední částí řešení je aplikace představující grafické uživatelské rozhraní pro PC. Tuto aplikaci se z časových důvodů nezdařilo realizovat v zamýšleném rozsahu. Přesto implementuje základní funkce komunikace se zařízením analyzátoru prostřednictvím virtuálního COM portu, příjmu a odesílání CAN zpráv. Aplikace byla programována

v objektově orientovaném jazyce C++ s využitím knihovny tříd MFC a může být solidním základem pro navázání na tuto práci.

Pro zařízení byla vyrobena krabička a celé řešení bylo úspěšně testováno v laboratoři Ústavu mechatroniky a technické informatiky.

Použitá literatura

[1] *CAN Specification Version 2.0 part A*
Bosch, 1991.

[2] *CAN Specification Version 2.0 part B*
Bosch, 1991.

[3] *CAN in Automatization*
Dostupné z: `http://www.can-cia.org`

[4] *CAN dokumentace* [cit. 10. března 2007]
Dostupné z: `http://fieldbus.feld.cvut.cz/can/`

[5] *PragoBoard*
Dostupné z: `http://www.pragoboard.cz`

[6] *Atmel Corporation*
Dostupné z: `http://www.atmel.com`

[7] *GM electronic*
Dostupné z: `http://www.gme.cz`

[8] *Silicon Laboratories*
Dostupné z: `http://www.silabs.com`

[9] *Littlefuse*
Dostupné z: `http://www.littlefuse.com`

[10] *Samsung Electronics*
Dostupné z: `http://www.samsung.com`

[11] *GES-ELECTRONICS*
Dostupné z: `http://www.ges.cz`

[12] *HT-Eurep Electronic*
Dostupné z: `http://www.hte.cz`

[13] *Texas Instruments*
Dostupné z: `http://www.ti.com`

[14] Váňa, V.: *Mikrokontroléry Atmel AVR – popis procesoru a instrukční soubor*
Praha: BEN, 2003. ISBN 80-7300-083-0

[15] Váňa, V.: *Mikrokontroléry Atmel AVR – assembler*
Praha: BEN, 2003. ISBN 80-7300-093-8

[16] *Biprog* [cit. 2. listopadu 2006]
Dostupné z: `http://web.quick.cz/ruckl/biprog/biprog.html`

[17] Prosise, J.: *Programování ve Windows pomocí MFC*
Praha: Computer Press, 2000. ISBN 80-7226-309-9

[18] *Serial Library for C++* [cit. 30. října 2006]
Dostupné z: `http://www.codeproject.com/system/serial.asp`

[19] Vacek, V.: *Sériová komunikace ve WIN32*
Praha: BEN, 2003. ISBN 80-7300-086-5

[20] Hulmut, K., Patrick D.: LaTeX– *podrobný průvodce*
Brno: Computer Press, 2004. ISBN 80-722-6973-9

Obsah přiloženého CD-ROM

Přiložený CD-ROM obsahuje tyto složky:

DiplomovaPrace

Tato diplomová práce ve formátu PDF.

- **Obrazky** – veškeré použité obrázky ve formátu PNG. Pouze fotografie jsou ve formátu JPG.

Dokumentace

Dokumentace ke všem použitým integrovaným obvodům ve formátu PDF.

Projekty

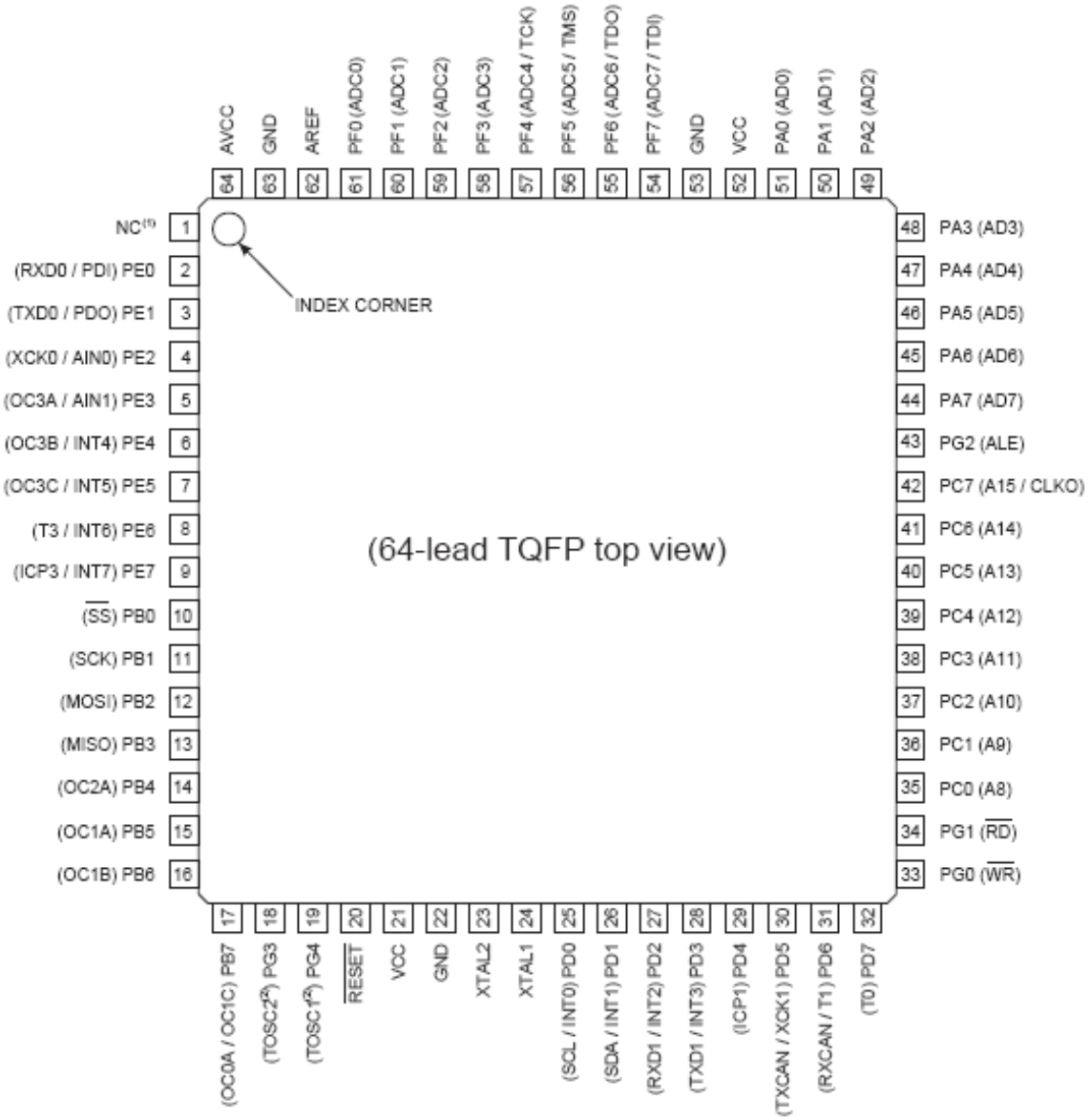
- **AvrStudio** – firmware pro mikrokontrolér AT90CAN128.
- **OrCad** – schéma elektrického zapojení a z něho vytvořený návrh desky plošných spojů hardwaru analyzátoru.
- **VisualC++** – uživatelské rozhraní pro PC.

Software

Používaný software.

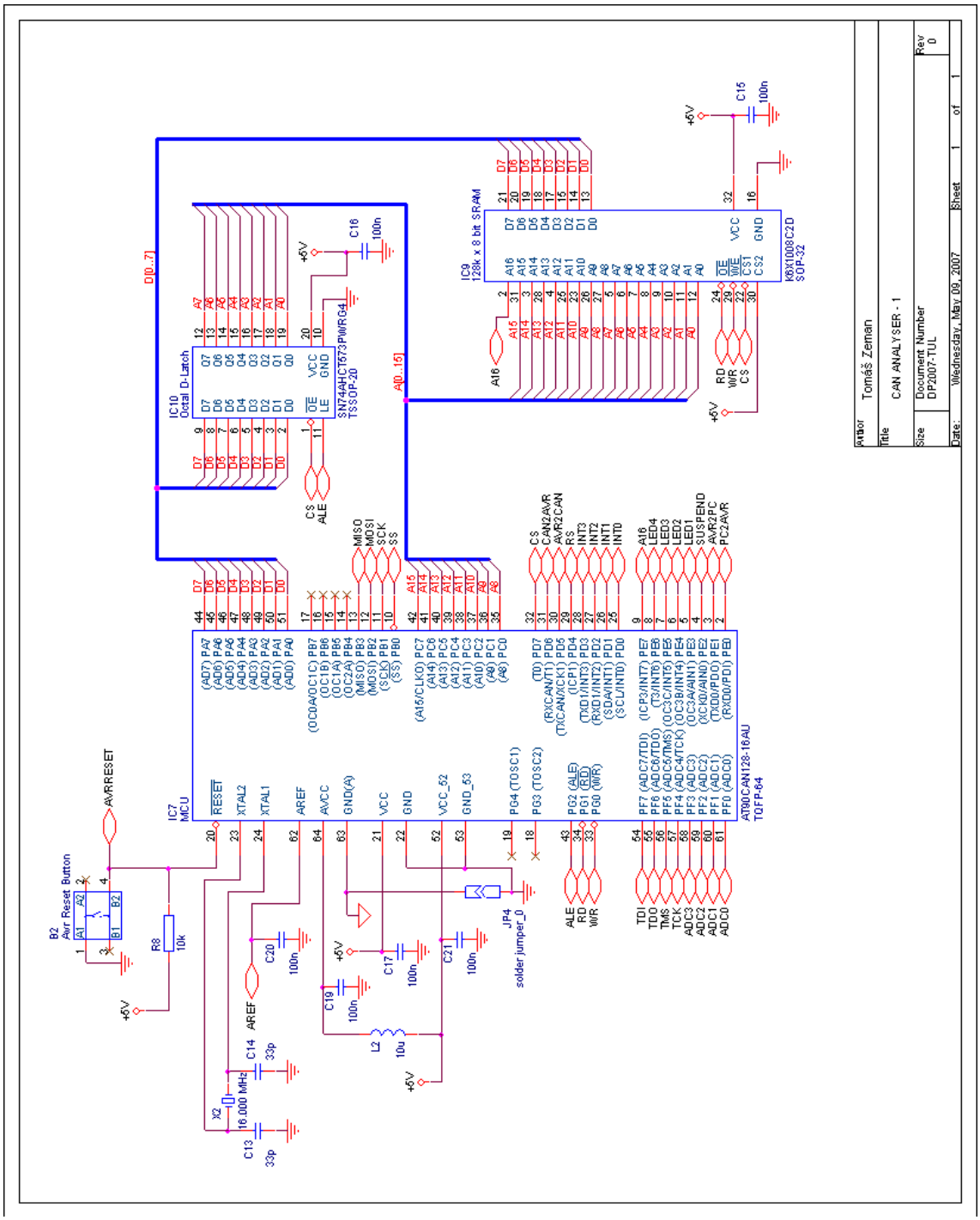
PŘÍLOHY

A Mikrokontrolér AT90CAN128-16AU v pouzdru TQFP-64



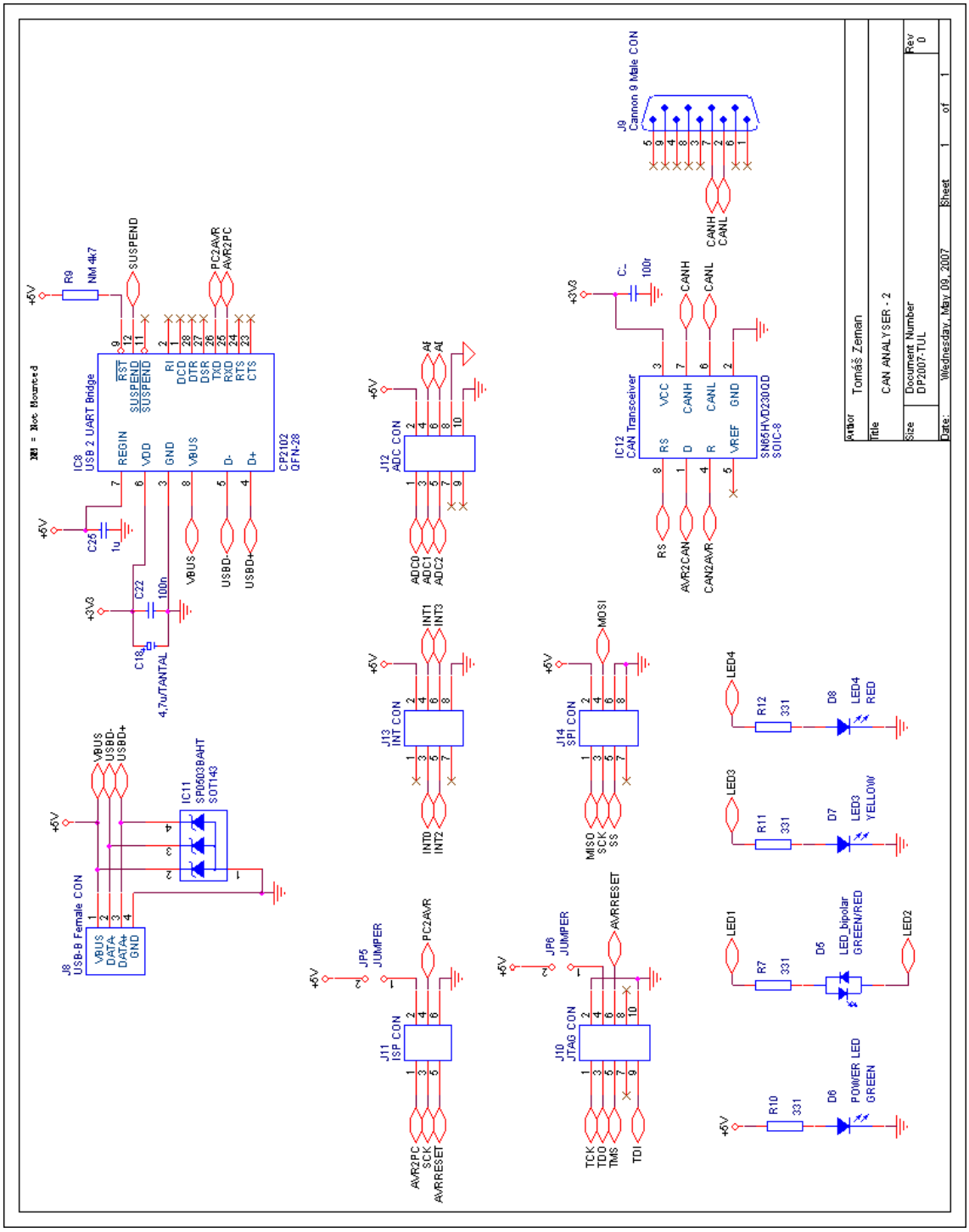
Obrázek 24: Mikrokontrolér AT90CAN128-16AU v pouzdru TQFP-64

B Schéma elektrického zapojení – část první



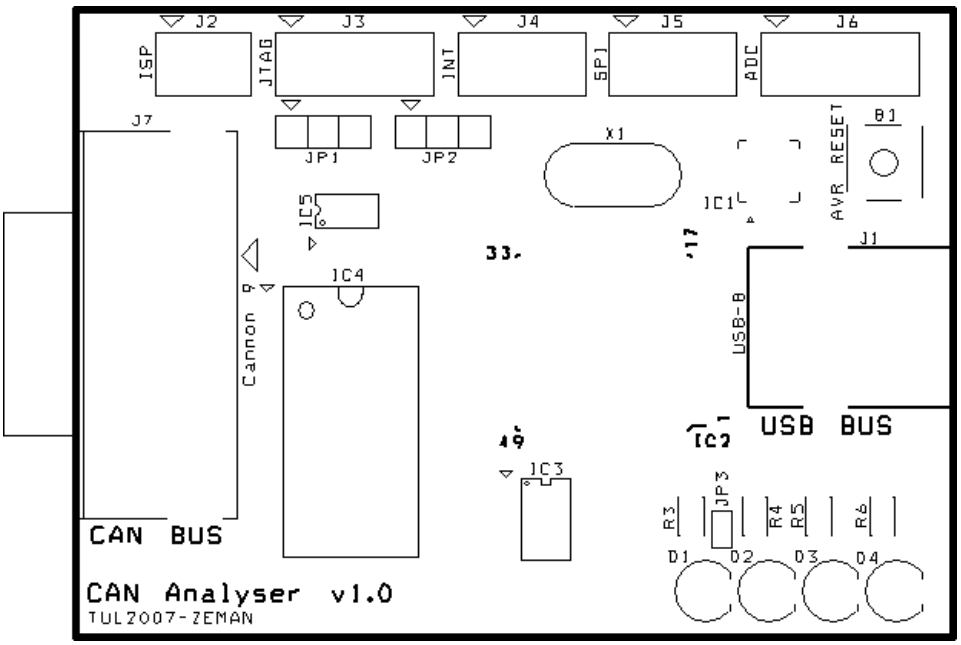
Obrázek 25: Schéma elektrického zapojení – mikrokontrolér s externí pamětí SRAM

C Schéma elektrického zapojení – část druhá

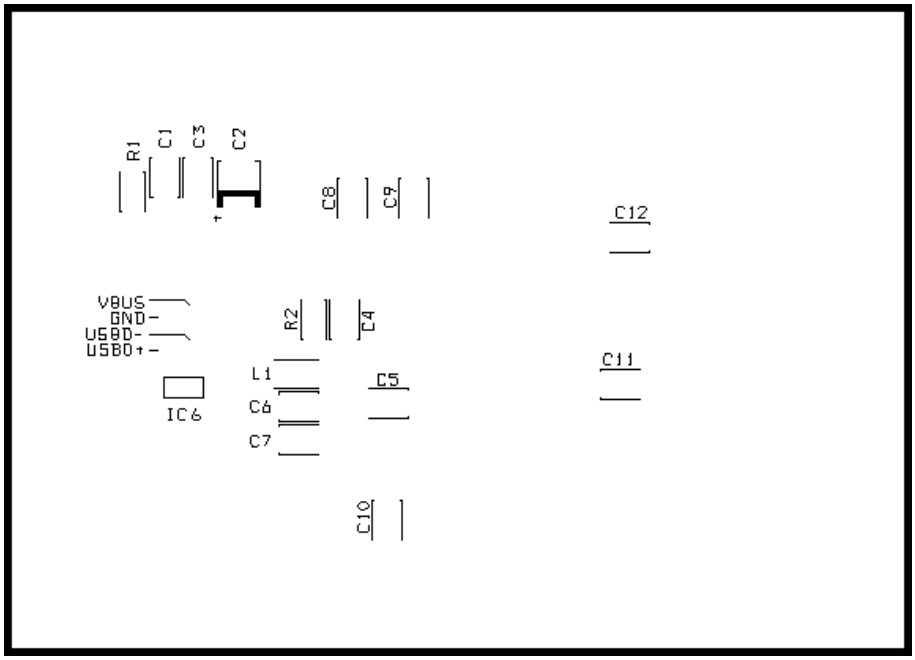


Obrázek 26: Schéma elektrického zapojení – převodník, budič, diody a konektory

D Rozmístění součástek na DPS

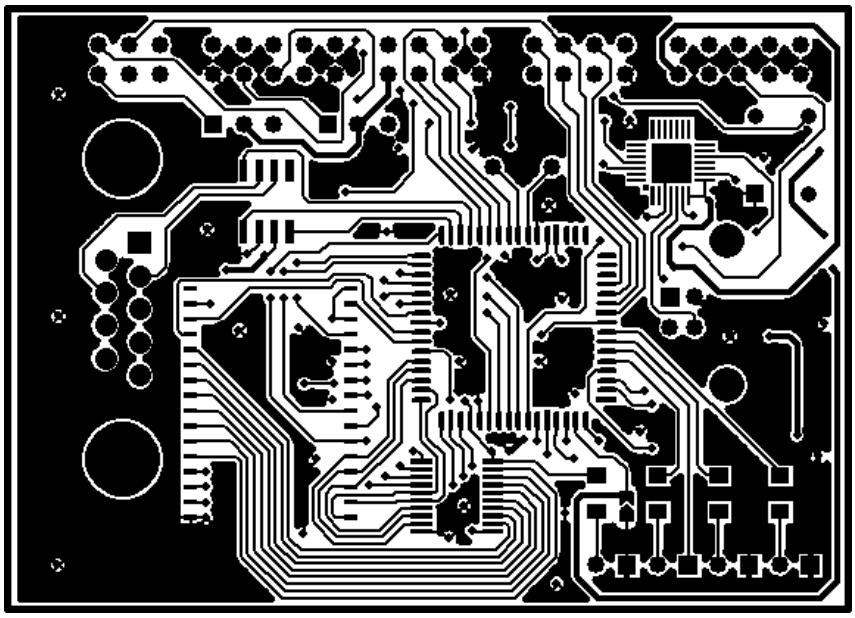


Obrázek 27: Rozmístění součástek na DPS – horní strana

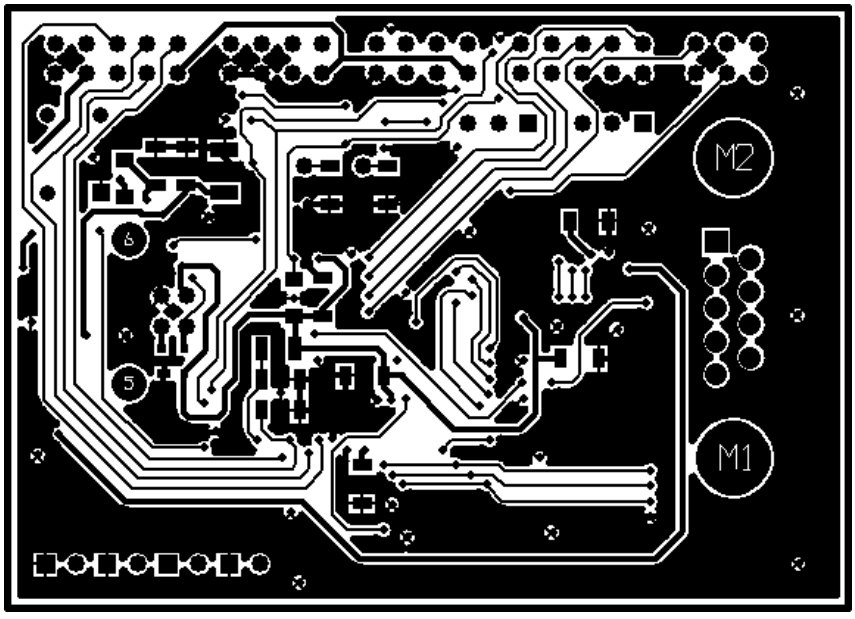


Obrázek 28: Rozmístění součástek na DPS – spodní strana

E Rozvržení vodivých cest na DPS



Obrázek 29: Rozvržení vodivých cest na DPS – horní strana



Obrázek 30: Rozvržení vodivých cest na DPS – spodní strana

F Seznam použitého materiálu

Číslo položky	Název	p.	Hodnota	Pouzdro	Popis
1	IC1	1	CP2102	QFN-28	USB-UART převodník
2	IC2	1	AT90CAN128-16AU	TQFP-64	mikrokontrolér
3	IC3	1	SN74AHCT573PWG4	TSSOP-20	8 x D-latch
4	IC4	1	K6X1008C2D	SOP-32	128k x 8 bit paměť SRAM
5	IC5	1	SN65HVD230QD	SOIC-8	CAN budič
6	IC6	1	SP0503BAHT	SOT143	3 x Zenerova dioda
7	X1	1	16.000 MHz		krystal
8	B1	1	Avr Reset Button		tlačítko
9	J1	1	USB-B Female CON		konektor
10	J2	1	ISP CON		konektor
11	J3	1	JTAG CON		konektor
12	J4	1	INT CON		konektor
13	J5	1	SPI CON		konektor
14	J6	1	ADC CON		konektor
15	J7	1	Cannon 9 Male CON		konektor
16, 17	JP1, JP2	2	JUMPER		propojka
18	D1	1	POWER LED		led dioda
19	D2	1	LED_bipolar		bipolární led dioda
20, 21	D3, D4	2	LED3, LED4		led dioda
22	R1	1	NM 4k7	SMD 1206	rezistor
23	R2	1	10k	SMD 1206	rezistor
24-27	R3-R6	4	331	SMD 1206	rezistor
28	C1	1	1u	SMD 1206	kondenzátor
29	C2	1	4u7/TANTAL	SMD 1206	kondenzátor
30-34, 37-39	C3-C7	8	100n	SMD 1206	kondenzátor
35-36	C8-C9	2	33p	SMD 1206	kondenzátor
40	L1	1	10u	SMD 1206	cívka

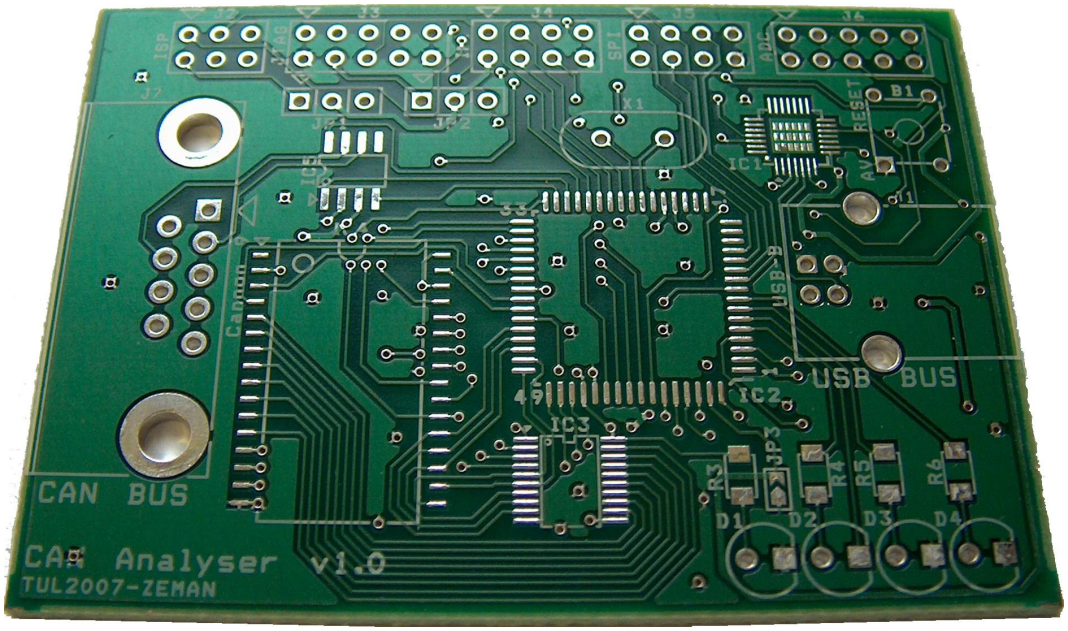
Tabulka 11: Seznam použitého materiálu

G Specifikace požadavků pro zakázku výroby DPS

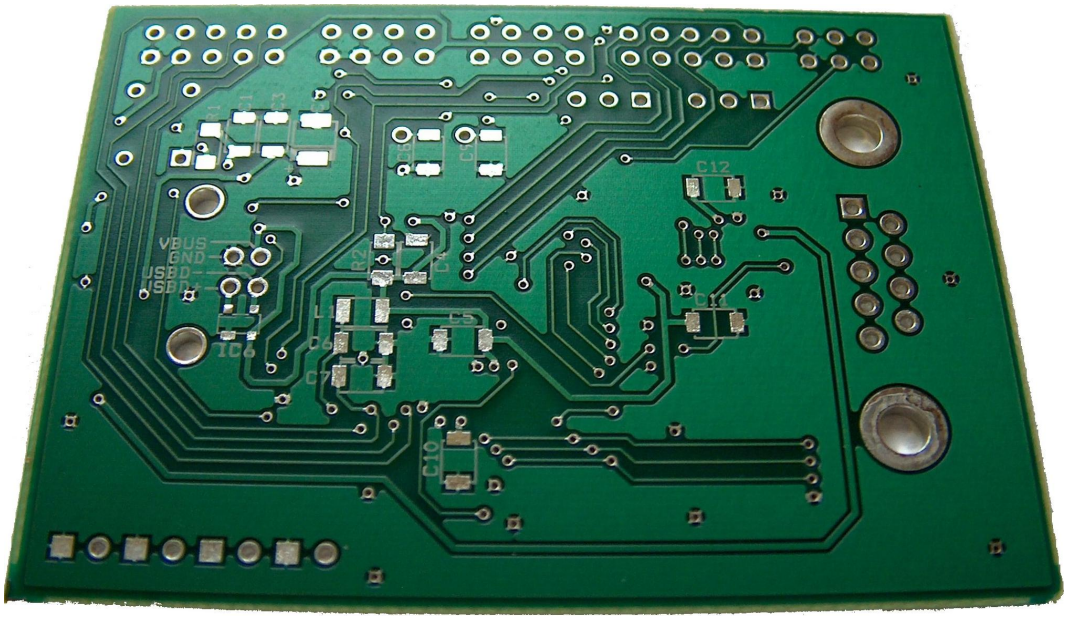
Zákazník	Tomáš Zeman
Škola	Technická Univerzita v Liberci
Mobilní telefon	
E-mail	
Název DPS	CAN Analyser
Typ	oboustranná
Počet kusů	2 ks
Rozměry	70 x 50 [mm]
Materiál	FR4
Tloušťka materiálu	1,5 [mm]
Tloušťka mědi	35/35 [um]
Povrch	cín+HAL
Maska	
Potisk	obě strany
Barva potisku	bílá
Vrtání	konečné průměry otvorů
Opracování	stříhání
Převzaté podklady	TOP, BOT, SMT, SMB, SST, SSB, DRL
Převzetí	osobně
Vrácení podkladů	uložit u výrobce
Datum objednání	
Datum dokončení zakázky	

Tabulka 12: Specifikace požadavků pro zakázku výroby DPS

H Neosazená DPS – fotografie

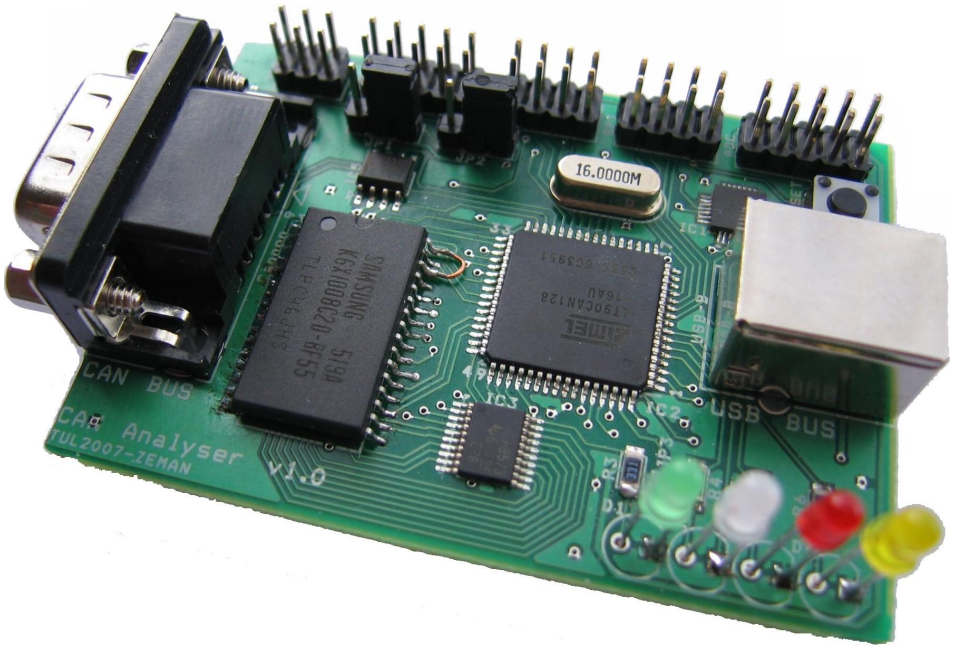


Obrázek 31: Neosazená DPS – horní strana

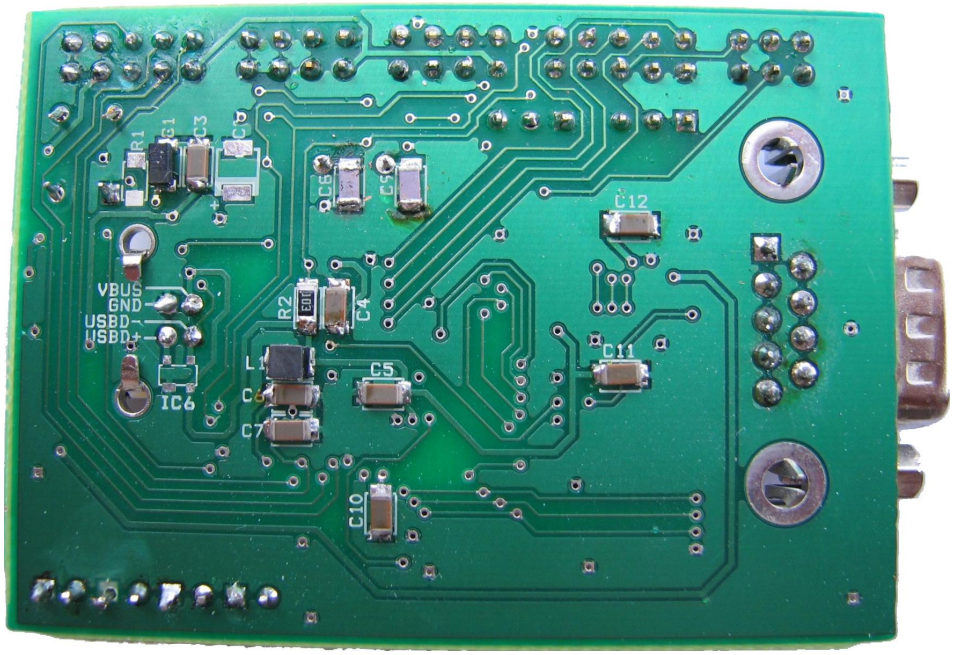


Obrázek 32: Neosazená DPS – spodní strana

I Osazená DPS – fotografie



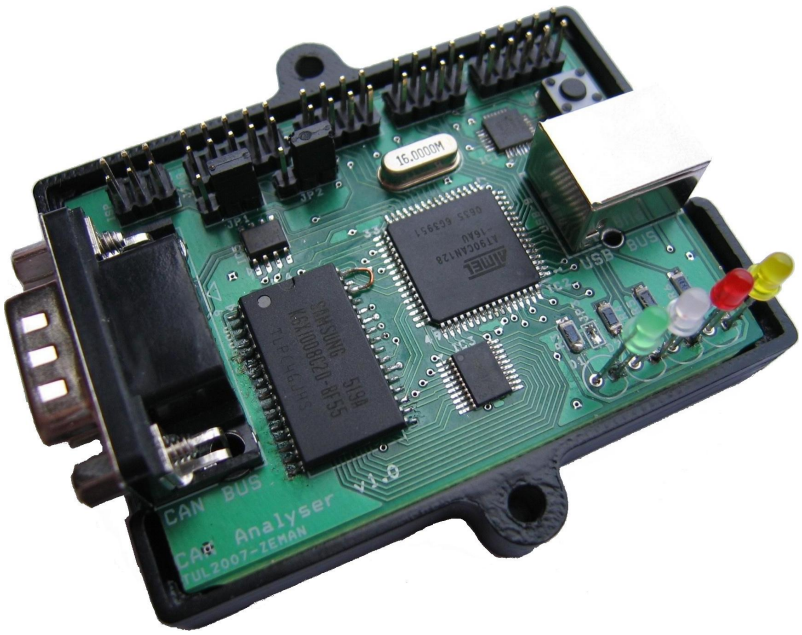
Obrázek 33: Osazená DPS – horní strana



Obrázek 34: Osazená DPS – spodní strana

J Analyzátor CAN sběrnice – fotografie

Obrázek 35: Analyzátor CAN sběrnice – krabička



Obrázek 36: Analyzátor CAN sběrnice – otevřená krabička

K Druhy zpráv

Druh	Název	Data	Popis
0	UM_CAID	CAN Analyzer v1.1	identifikační zpráva analyzátoru
1	UM_ASTART		zahájení analýzy
2	UM_ASTOP		zastavení analýzy
3	UM_ASET	{BRP, PRS, PHS1, PHS2, SJW, SMP}	nastavení CAN řadiče
4	UM_MOBSET		nastavení objektu CAN zprávy
5	UM_ALISTEN		nastavení *Listening* módu

Tabulka 13: Druhy iniciačních zpráv

Druh	Název	Data	Popis
0	UM_NOTCOMPLETE	Message not complete!	zpráva není kompletní
1	UM_BADCRC	Bad message CRC!	chybný kontrolní součet zprávy
2	UM_IDUNKNOWN	Message ID unknown!	neznámý identifikátor zprávy
3	UM_BADFORMAT	Bad message format!	chybný formát zprávy
4	UM_OUTOFMEMORY	Out of memory!	zaplnění paměti SRAM
5	UM_RUNTIME	Runtime error!	běhová chyba firmwaru

Tabulka 14: Druhy chybových zpráv

T_{bit} = T_{sync} + T_{prs} + T_{phs1} + T_{phs2} = 8..25 TQ

T_{sync} = 1 x T_{scl} = (BRP[5..0] + 1)/clk_{io} = 1 TQ

T_{prs} = (1..8) x T_{scl} = (PRS[2..0] + 1) x T_{scl}

T_{phs1} = (1..8) x T_{scl} = (PHS1[2..0] + 1) x T_{scl}

T_{phs2} = (1..8) x T_{scl} = (PHS2[2..0] + 1) x T_{scl}

T_{sjw} = (1..4) x T_{scl} = (SJW[1..0] + 1) x T_{scl}