ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

DIPLOMOVÁ PRÁCE

Převodník Ethernet – CAN

Plzeň 2012 Martin Kamenický

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin KAMENICKÝ**
Osobní číslo:	**E09N0269P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Dopravní elektroinženýrství a autoelektronika**
Název tématu:	**Převodník Ethernet - CAN**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Seznamte se standardy komunikace Ethernetu a CAN.
2. Navrhněte vhodnou HW realizaci.
3. Naprogramujte převodník Ethernet - CAN.
4. Změřte dosaženou přenosovou rychlost.
5. Realizaci zhodnotťe a rozeberte případné omezení.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah pracovní zprávy:	**30 - 40 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Seznam odborné literatury:	

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce:	**Ing. Jan Brož** Katedra aplikované elektroniky a telekomunikací
Konzultant diplomové práce:	**Ing. Jan Brož** Katedra aplikované elektroniky a telekomunikací
Datum zadání diplomové práce:	**18. října 2010**
Termín odevzdání diplomové práce:	**11. května 2012**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 17. října 2011

Anotace

Tato práce se zabývá návrhem převodníku Ethernet CAN s použitím mikrokontroléru STM32f105 od firmy STMicroelektronics. V prvních kapitolách je teoreticky rozebráno rozhraní Ethernet a CAN. V další části jsou popsány vlastnosti mikrokontroléru STM32f105. Poté je zde uveden vlastní hardwarový návrh. Ten obsahuje schéma zapojení, návrh desky plošného spoje a výběr součástek. V osmé kapitole je uveden postup programování firmware pro zařízení a popis nastavení jednotlivých periferií mikrokontroléru. V poslední části této práce jsou zhodnoceny výsledné vlastnosti zařízení.

Klíčová slova

Ethernet, CAN, STM32f107,Cortex M3, STMicroelektronics, převodník, ARM

Annotation

This master thesis deals with the Ethernet CAN design using STM32f10 from STMicroelektronics corporate. In the first chapters are discussed feauters of the STM32f105 microcontroller. Then is here listed the main hardware design. This design contains wiring diagram, design of the PCB layout and components selection. In the eighth chaper is listed the firmware programming process of the device and description of the settings for individual microcontroller periphery. In the last section of this master thesis are reviewed the final device properties.

Key words

Ethernet, CAN, STM32f107, Cortex M3, STMicroelektronics, converter, ARM

Martin Kamenický

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni. Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

V Plzni dne 9.5.2012 .....

Podpis

Poděkování

Tímto bych chtěl poděkovat vedoucímu diplomové práce panu Ing. Janu Brožovi za poskytnutí odborného vedení při vypracovávání diplomové práce, panu Ing. Petru Kristovy Ph.D. za cenné rady a pomoc při začátcích programování mikrokontroléru STM32. Také bych chtěl poděkovat mé rodině a všem ostatním, kteří mě podporovali během studia.

Obsah

SEZNAM OBRÁZKŮ 3

SEZNAM TABULEK 4

1 ÚVOD 5

1.1 CÍL PRÁCE 5

2 ZÁKLADY PŘENOSU DAT 5

2.1 PŘENOSOVÝ KANÁL 5
2.1.1 VLASTNOSTI DISKRÉTNÍCH KANÁLŮ 5
2.1.2 VLASTNOSTI SPOJITÉHO KANÁLU 6
2.2 DRUHY PŘENOSU INFORMACE 7
2.2.1 SIMPLEXNÍ PŘENOS INFORMACE 7
2.2.2 POLODUPLEXNÍ PŘENOS 8
2.2.3 DUPLEXNÍ PŘENOS 8
2.3 REFERENČNÍ MODEL ISO/OSI 9

3 POPIS KOMUNIKAČNÍHO ROZHRANÍ CAN 12

3.1 ZÁKLADNÍ CHARAKTERISTIKY SBĚRNICE CAN 12
3.2 MODEL OSI A PROTOKOL CAN 14
3.3 PŘÍSTUP NA CAN SBĚRNICI 14
3.4 ZPŮSOB PŘENOSU JEDNOHO BITU 16
3.5 KÓDOVÁNÍ SIGNÁLU 16
3.6 PROTOKOLY NA SBĚRNICI CAN 17
3.7 TYPY ZPRÁV NA SBĚRNICI CAN 18
3.8 HARDWAROVÁ REALIZACE 20

4 POPIS KOMUNIKAČNÍHO ROZHRANÍ ETHERNET 22

4.1 FYZICKÁ VRSTVA 23
4.2 LINKOVÁ VRSTVA 25
4.2.1 PODVRSTVA LLC – ŘÍZENÍ LOGICKÉHO SPOJE 25
4.2.2 PODVRSTVA MAC 26
4.3 MII ROZHRANÍ 28
4.4 ETHERNET TRANSCEIVER 30
4.5 PROTOKOLY VYŠŠÍCH VRSTEV 34
4.5.1 SÍŤOVÁ VRSTVA 34
4.5.2 TRANSPORTNÍ VRSTVA 34
4.5.3 APLIKAČNÍ VRSTVA 34

5 ZÁKLADNÍ VLASTNOSTI MIKROKONTROLÉRU 35

6 NÁVRH OBVODOVÉHO ZAPOJENÍ 36

6.1 NAPÁJENÍ 36
6.2 MIKROKONTROLÉR 37
6.3 CAN 39
6.4 ETHERNET 40

7 NÁVRH PLOŠNÉHO SPOJE 44

7.1 ZÁKLADNÍ PRAVIDLA PRO NÁVRH DPS V SOULADU S EMC 44
7.2 REALIZACE PLOŠNÉHO SPOJE 45
7.3 SEZNAM POUŽITÝCH SOUČÁSTEK 48

8 SOFTWAROVÉ ŘEŠENÍ 50

8.1 INICIALIZACE OBVODU 50
8.2 NASTAVENÍ GPIO 51
8.3 NASTAVENÍ PŘERUŠENÍ 52
8.4 CAN 53
8.4.1 NASTAVENÍ GPIO A CAN ŘADIČE 53
8.4.2 ODESLÁNÍ A PŘIJETÍ ZPRÁVY 55
8.5 ETHERNET 56
8.5.1 NASTAVENÍ ETHERNETOVÉHO ŘADIČE 56
8.5.2 ODESLÁNÍ PAKETU 59
8.5.3 PŘÍJEM PAKETU 59

9 ZHODNOCENÍ VÝSLEDKŮ 60

10 ZÁVĚR 62

11 POUŽITÁ LITERATURA 63

12 PŘÍLOHA 65

Seznam obrázků

Obrázek 2.1 Jednosměrný simplexní přenos [12] 7
Obrázek 2.2 Obousměrný simplexní přenos [12] 7
Obrázek 2.3 Jednosměrný poloduplexní přenos [12] 8
Obrázek 2.4 Obousměrný poloduplexní přenos [12] 8
Obrázek 2.5 Architektura sítě založená na modelu OSI [4] 9
Obrázek 3.1 Referenční model ISO /OSI a protokol CAN [11] 14
Obrázek 3.2 Arbitrážní přístup na sběrnici [11] 15
Obrázek 3.3 Princip připojení uzlů na sběrnici [11] 15
Obrázek 3.4 Přenos jednoho bitu po CAN [11] 16
Obrázek 3.5 Příklad kódovaní s bitovým Stuffingem [11] 16
Obrázek 3.6 Formát rámce CAN 2.0A Standard [11] 17
Obrázek 3.7 Formát rámce CAN 2.0B Extended [11] 18
Obrázek 3.8 Způsob připojení uzlů na sběrnici [11] 20
Obrázek 3.9 Napěťová úroveň pro CAN – hight speed [11] 21
Obrázek 3.10 Napěťové úrovně pro CAN – low speed [11] 22
Obrázek 4.1 Struktura Ethernet [9] 23
Obrázek 6.1 Schéma zapojení napájení 37
Obrázek 6.2 Schéma zapojení okolí mikrokontroléru 39
Obrázek 6.3 Schéma zapojení CAN 40
Obrázek 6.4 Schéma zapojení Ethernet transceiveru 42
Obrázek 6.5 Konektor RJ45 Vnitřní zapojení [16] 43
Obrázek 7.1 Vrstva TOP 45
Obrázek 7.2 Vrstva GND 46
Obrázek 7.3 Vrstva 3,3V 46
Obrázek 7.4 Vrstva BOTTOM 47
Obrázek 7.5 Rozmístění součástek na straně TOP 47
Obrázek 7.6 Rozmístění součástek na straně BOTTOM 48
Obrázek 9.1 Rámec dat 60

Seznam tabulek

TABULKA 3.1 PŘENOSOVÁ RYCHLOST NA SBĚRNICI CAN V ZÁVISLOSTI NA DÉLCE SBĚRNICE...13
TABULKA 6.1 ZAPOJENÍ KONEKTORU R-LINK.....38
TABULKA 6.2 FUNKCE JEDNOTLIVÝCH PINŮ ETHERNET TRANSCEIVERU.....41
TABULKA 6.3 ZAPOJENÍ ETHERNET TRANSCEIVERU NA PROCESOR.....43
TABULKA 7.1 SEZNAM SOUČÁSTEK 48
TABULKA 8.1 STANDARDNÍ KNIHOVNY MIKROKONTROLÉRU.....50
TABULKA 8.2 PRVKY STRUKTURY PRO NASTAVENÍ CAN 53
TABULKA 8.3 PRVKY STRUKTURY PRO NASTAVENÍ CAN FILTRU 54
TABULKA 8.4 PRVKY STRUKTURY PRO NASTAVENÍ MAC VRSTVY 57
TABULKA 8.5 NASTAVENÍ DMA KANÁLU 58

1 Úvod

1.1 Cíl práce

Cílem této práce je seznámení se standardy komunikací Ethernetu a CAN. Poté navržení vhodné hardwarové realizace převodníku Ethernet – CAN. Po navržení hardwarové realizace je nutno převodník naprogramovat a změřit jeho přenosovou rychlost.

Pro tento účel práce jsem si vybral mikrokontrolér od firmy STMicroelectronics STM32F10105RB s jádrem ARM32. Tento mikrokontrolér jsem si vybral z důvodu, že je vhodný pro komunikaci, protože je vybaven množstvím komunikačních rozhraní.

2 Základy přenosu dat

2.1 Přenosový kanál

Zdroj informace generuje na výstup informaci, která pak vstupuje do kanálu. Tato informace má přenosovou rychlost v_p. Schopnost kanálu přenášet určité maximální množství informace za jednotku času nazýváme informační kapacita. Pro informační kapacitu platí:

$$v_p \leq C \tag{2.1}$$

Podle počtu stavů rozlišujeme dva typy kanálů: diskrétní a spojitý.

2.1.1 Vlastnosti diskrétních kanálů

Modulační rychlost

$$v_m = \frac{1}{a} \tag{2.2}$$

Kde a je délka charakteristického intervalu a jednotkou modulační rychlosti je 1 Bd (baud)

Přenosová rychlost

Udává počet binárních prvků přenesených za jednotku času a lze vypočítat takto:

$$v_p = v_m * \log_2 S \tag{2.3}$$

Kde v_m - je modulační rychlost, která udává počet signálových prvků za jednu sekundu.
S - je počet stavů signálu

Kapacita kanálu

Udává, jakou přenosovou rychlost můžeme dosáhnout v daném prostředí.

$$C = v_{pmax} = B * \log_2 \left(1 + \frac{P_s}{P_š}\right) \tag{2.4}$$

Kde B - šířka pásma
P_s – střední výkon signálu
$P_š$ – střední výkon šumu

2.1.2 Vlastnosti spojitého kanálu

Často se také uskutečňuje komunikace na analogových kanálech. U analogového kanálu také potřebujeme znát jeho informační kapacitu. Pro určení informační kapacity vycházíme z Hartleyova zákona pro maximální množství informace, které kanál přenese. Pro I_{max} platí vztah:

$$I_{max} = B * T * \log_2 \left(1 + \frac{P_s}{P_š}\right) \quad [\text{Sh}] \tag{2.5}$$

Kde: B – šířka kanálu
T – doba přenosu informace
P_s – střední výkon signálu
$P_š$ – střední výkon šumu
I_{max} – maximální množství informace

Protože kapacita kanálu je množství informace přenesených za čas, vypočítá se kapacita kanálu takto:

$$C = \frac{I_{max}}{T} = B * \log_2\left(1 + \frac{P_s}{P_š}\right) \quad [\text{Sh/s}] \qquad (2.6)$$

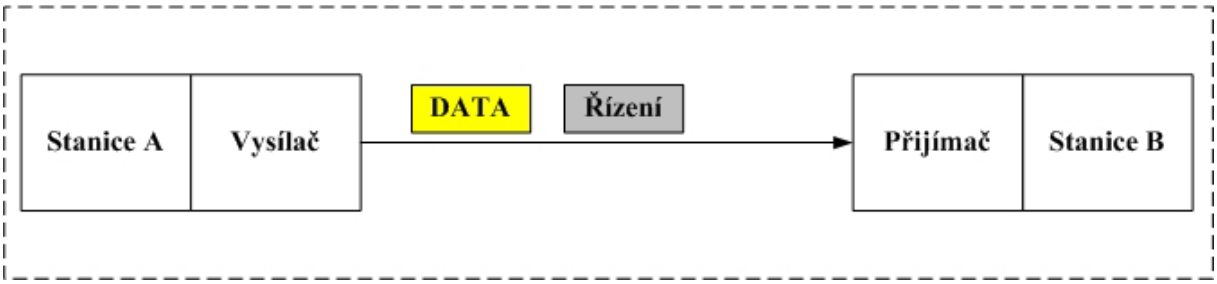
2.2 Druhy přenosu informace

Máme tyto tři druhy přenosu informace. První je simplexní přenos, druhý je poloduplexní přenos a poslední je duplexní přenos informace.

2.2.1 Simplexní přenos informace

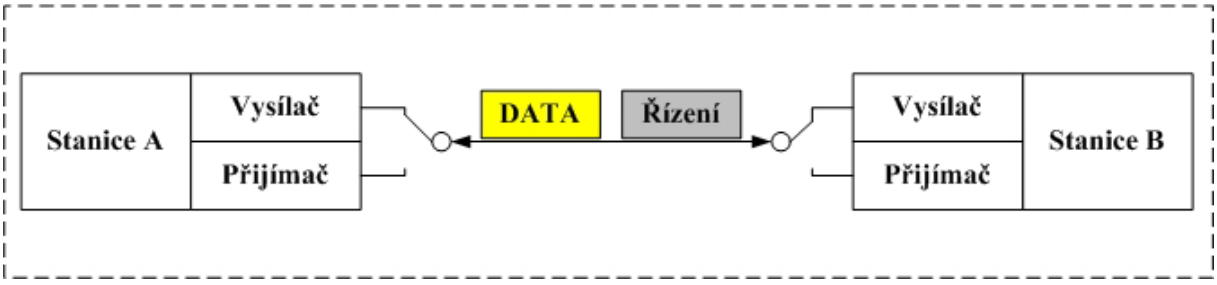
U simplexního přenosu umožňuje koncová stanice přenos dat pouze v jednom směru.

Simplexní přenos může být jednosměrný, kdy se data přenášejí po jednom kanále pouze jedním směrem (Obr. 2.1).



Obrázek 2.1 Jednosměrný simplexní přenos [12]

Druhou možností je obousměrný simplexní přenos, kde se využívá jednoho kanálu, který se střídavě přepíná vysílač A – přijímač B, nebo vysílač B – přijímač A. Znárodněno na obr. 2.2

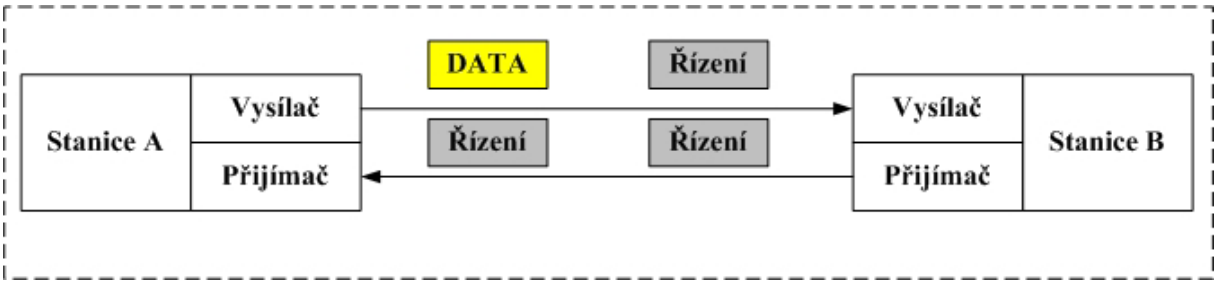


Obrázek 2.2 Obousměrný simplexní přenos [12]

2.2.2 Poloduplexní přenos

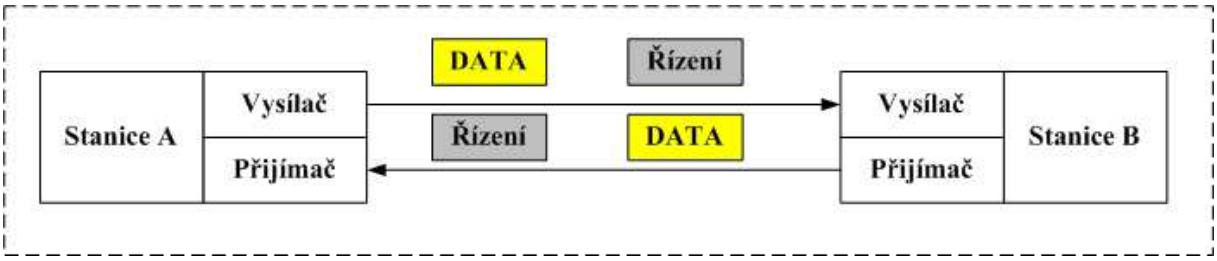
Koncová stanice umožňuje přenos dat střídavě v obou směrech. V systému jsou dva kanály, které ale nelze využívat současně.

Při jednosměrném poloduplexním přenosu se data přenášejí jedním směrem a řídící informace se mohou přenášet v obou směrech. Zpravidla bývá, že zpětný kanál má daleko menší přenosovou kapacitu. Přenos znázorněn na obr. 2.3



Obrázek 2.3 Jednosměrný poloduplexní přenos [12]

Při obousměrném přenosu jsou využity oba kanály pro přenos řízení i dat, ale nemohou být tyto kanály využívány současně. Poloduplexní obousměrný přenos znázorněn na obr. 2.4



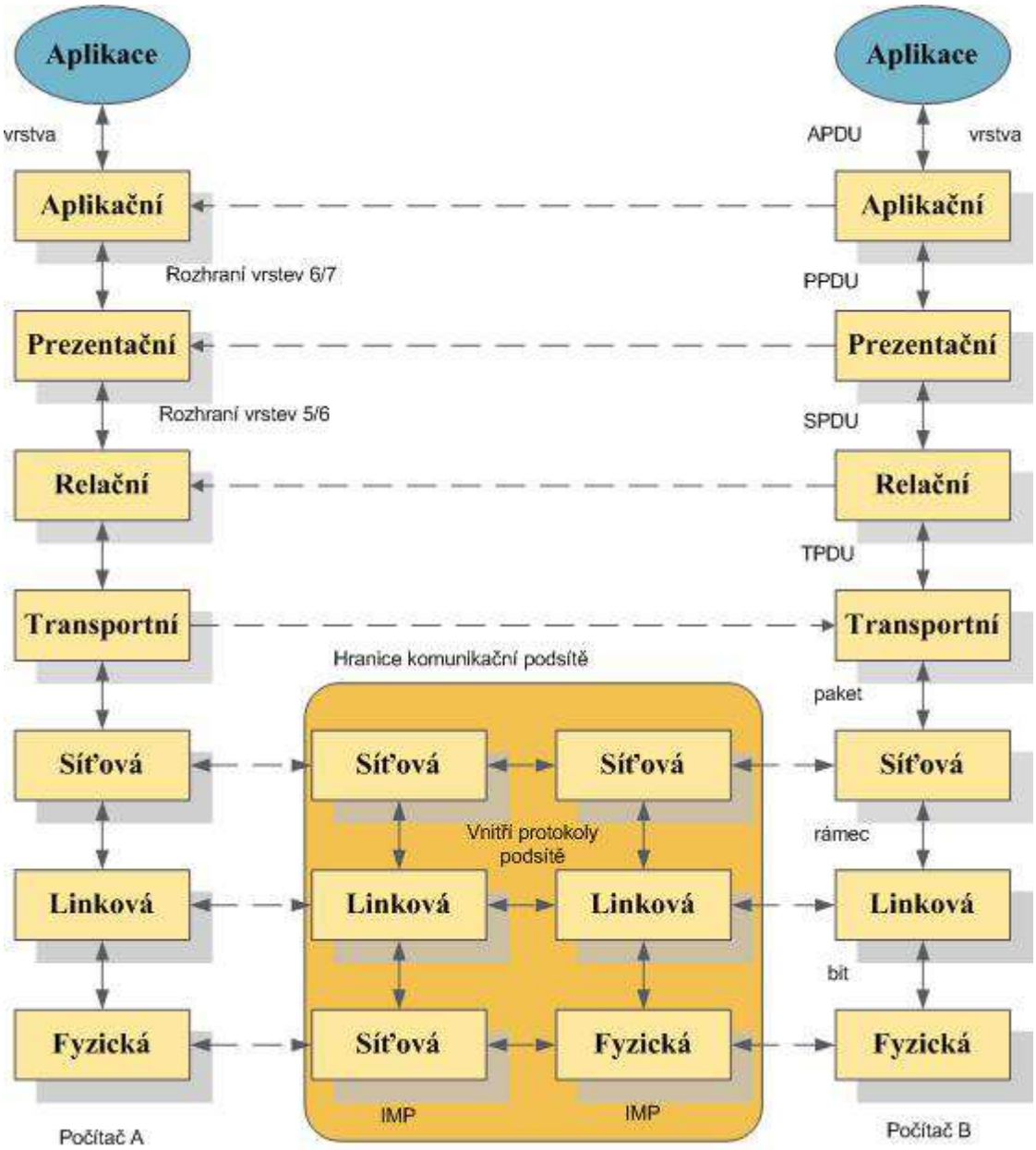
Obrázek 2.4 Obousměrný poloduplexní přenos [12]

2.2.3 Duplexní přenos

Koncové stanice umožňují přenos dat v obou směrech zároveň. Jednosměrný i obousměrný duplexní přenos je stejný jako předchozí poloduplexní s rozdílem, že obě stanice mohou komunikovat zároveň. Znázorněno na obr. 2.4

2.3 Referenční Model ISO/OSI

Referenční model ISO/OSI se stal výchozím modelem pro účely vypracování norem a propojování systémů. V modelu jsou jednotlivé problémy soustředěny do jednotlivých vrstev, tak aby byl minimalizován přenos dat mezi jednotlivými vrstvami. Model OSI má 7 vrstev a jsou znázorněny na obr. 2.5



Obrázek 2.5 Architektura sítě založená na modelu OSI [4]

TPDU – Aplikační protokol datové jednotky
SPDU – Relační protokol datové jednotky
PPDU – Prezentační protokol datové jednotky
APDU - Aplikační protokol datové jednotky

Fyzická vrstva *(Physical Layer)*

Tato vrstva má za úkol přípravu funkčních, mechanických, procedurálních, elektrických a elektronických prostředků pro vytvoření, udržení a ukončení datových obvodů reálného nebo virtuálního typu mezi datovými koncovými zařízeními, zařízeními přenosu dat.

Fyzická vrstva přenáší proud bitů přenosovým médiem.

Linková vrstva *(Link Layer)*

Tato vrstva připravuje funkční a procedurální prostředky k vytvoření, udržení a rušení datových spojů mezi dvěma nebo více síťovými pracovními jednotkami. Datové spojení je tvořeno jedním nebo více reálnými nebo virtuálními datovými obvody. Spojová vrstva spravuje logické vztahy mezi koncovými body spojení, k nimž se vážou pracovní jednotky síťové vrstvy.

Provedení linkové vrstvy může být různé. Záleží na druhu sítě, ale většinou přebírá funkci zabezpečení přenosu dat. Data upravuje do paketů s kódováním, které je schopno rozpoznat a opravit elementární chyby.

Linková vrstva mění proud bitů na spolehlivou cestu přenosu datových bloků rámců.

Síťová vrstva *(Network Layer)*

Tato vrstva vytváří funkční a procedurální prostředky pro výměnu síťových datových služeb mezi dvěma transportními pracovními jednotkami přes síťové spojení. Vytváří nezávislost transportní vrstvy na funkcích směrování a propojování.

Hlavními funkcemi síťové vrstvy jsou směrování, rozpoznávání chyb, a pokračování v přenosu dat po jejich opravě. Síťová vrstva má takzvanou mapu sítě, nebo její část při velké rozsáhlosti sítě.

Síťová vrstva směruje tok dat organizovaných do paketů.

Transportní vrstva *(Transport Layer)*

Tato vrstva připravuje universální transportní obsluhu pro nejblíže vyšší vrstvu a správa pomocných prostředků nejblíže nižší vrstvy. Transportní vrstva osvobozuje svoje uživatele (většinou pracovní jednotky relační vrstvy) od nutnosti určení optimální cesty, kontroly toku dat, přetížení a chyb na této úrovni.

Transportní vrstva zvyšuje kvalitu spojů na požadovanou úroveň.

Relační vrstva *(Session Layer)*

Tato vrstva podporuje spolupráci mezi aplikačními pracovními jednotkami. Služby této vrstvy můžeme rozdělit na dvě skupiny.

- Vazbové služby (dvě pracovní jednotky, služby správy a relace)
- Relační služby (přenos dat, kontrola výměny dat a synchronizace)

Od transportní vrstvy očekává relační vrstva možnost navázání spojení, provedení přenosu a jeho ukončení. Za normálních podmínek musí datový tok proběhnout bez chyb. Pokud by došlo k výskytu nekorigovatelných chyb, musí být relační vrstva o tom informována.

Tato vrstva má velký rozsah výběru jednotlivých funkcí a stanovení kvality služeb. Není však nutné celou tuto nabídku implementovat na všechny aplikace. V ISO 8326 jsou proto funkční jednotky a jejich podmnožiny.

V této vrstvě je třeba se hlavně zabývat, aby byla co nejhladší spolupráce mezi aplikačními procesy. Mechanismy spolupráce musí být jasné a průhledné, jinak se může stát, že uživatelské programy se nepodaří přizpůsobit síťovému prostředí.

Relační vrstva poskytuje informačním systémům nástroje pro řízení a synchronizaci jejich dialogu.

Prezentační vrstva *(Presentation Layer)*

Tato vrstva připravuje služby pro aplikační vrstvu k interpretaci vyměňovaných dat. Připraví výběr ze standardních prezentací a jejich interpretaci tak, aby komunikující aplikační

procesy svoje data nejdříve transformovaly do společného standardního formátu, přenesly je a nakonec je transformovaly zpět.

Prezentační vrstva koordinuje kódování syntaxi přenášených dat.

Aplikační vrstva *(Application Layer)*

Protokoly této vrstvy provádějí komunikaci mezi aplikačními procesy ve spojení se správními funkcemi operačního systému, jež tyto procesy podporují. Z tohoto důvodu lze tvrdit, že u mnohých systémů je aplikační vrstva jen rozšířením běžných operačních systémů na požadavky síťového prostředí.

Tato vrstva je ohledně funkcí velmi rozsáhlá, proto jednotlivé počítače podporují současně jednu nebo dvě funkce.

Aplikační vrstva zpřístupňuje informačním systémům prostředí OSI

3 Popis komunikačního rozhraní CAN

CAN (Control Area Network) je datová komunikační síť pro distribuované řídící aplikace v reálném čase. Byla vyvinuta firmou Bosh v 80 letech a původní její úkol byl snížení počtu kabelů a zabezpečení komunikace mezi snímači, řídícími jednotkami a výkonovými prvky v automobilech. Vzhledem ke svým vlastnostem a hardwarové podpoře se tento systém velice rychle rozšířil a stal se mezinárodním standardem. V současné době se tento systém používá především v automobilovém průmyslu (Mercedes - Benz, Škoda, BMV), v zabezpečovacích systémech, sběr dat v distribuovaných systémech a komunikacích průmyslových aplikací.

3.1 Základní charakteristiky sběrnice CAN

Využívá sdílenou sběrnici s prioritním rozhodováním na základě identifikátoru zprávy, který je v závislosti na protokolu 11 bitový nebo 29 bitový. Využívá multimaster režim, to znamená, že libovolný uzel může začít vysílat komunikační rámec v jakémkoliv okamžiku. Komutace je zajišťována prostřednictvím rámců, které mají maximální délku přenášených dat

datového rámce 8 bytů. Pokud nastane chyba, automaticky se znovu vysílají poškozené rámce. V současné době máme 2 komunikační protokoly.

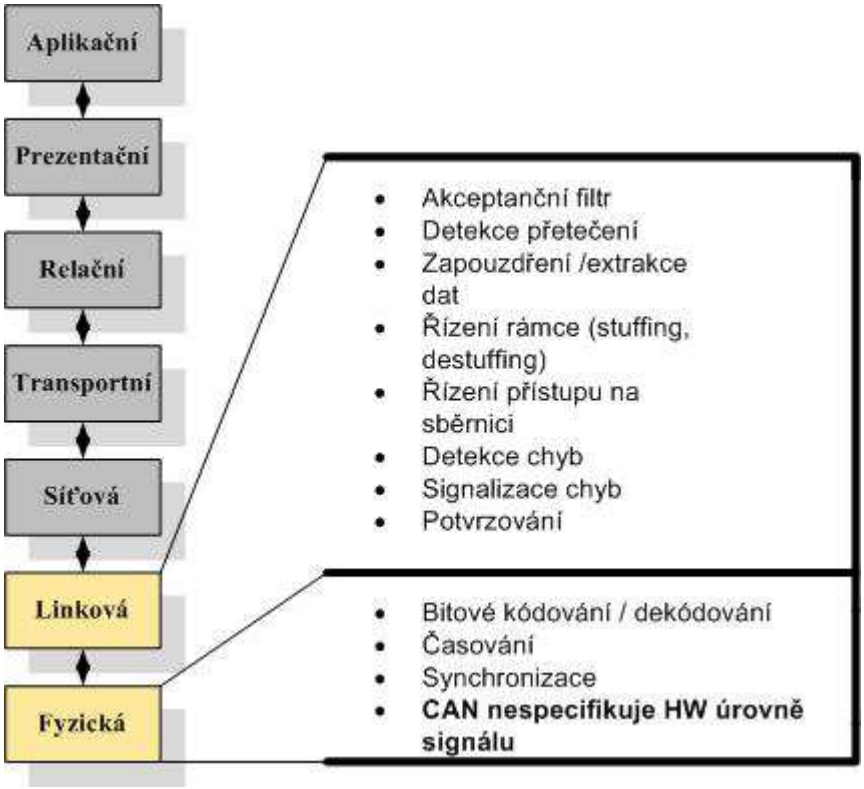
- CAN 2.0A – má délku identifikátoru 11 bitů a maximální rychlost 1 Mbit/s
- CAN 2.0B – má délku identifikátoru 29 bitů a maximální rychlost 1 Mbit/s

Přenosová rychlost na sběrnici CAN není vždy 1 Mbit/s, ale závisí na délce sběrnice. Rychlosti pro jednotlivé vzdálenosti jsou uvedeny v následující tabulce.

Přenosová rychlost	Maximální délka sběrnice
5 kbit/s	10 km
10 kbit/s	6,7 km
20 kbit/s	3,3 km
50 kbit/s	1,3km
100 kbit/s	620 m
125 kbit/s	530 m
250 kbit/s	270 m
500 kbit/s	130 m
1000 kbit/s	40 m

Tabulka 3.1 Přenosová rychlost na sběrnici CAN v závislosti na délce sběrnice

3.2 Model OSI a protokol CAN



Obrázek 3.1 Referenční model ISO /OSI a protokol CAN [11]

Fyzická vrstva

Přenosové médium je elektrické (kroucená dvojlinka), optické, nebo jiné. Přenosové médium musí být schopno přenášet dva logické stavy dominantní 0, nebo recesivní 1.

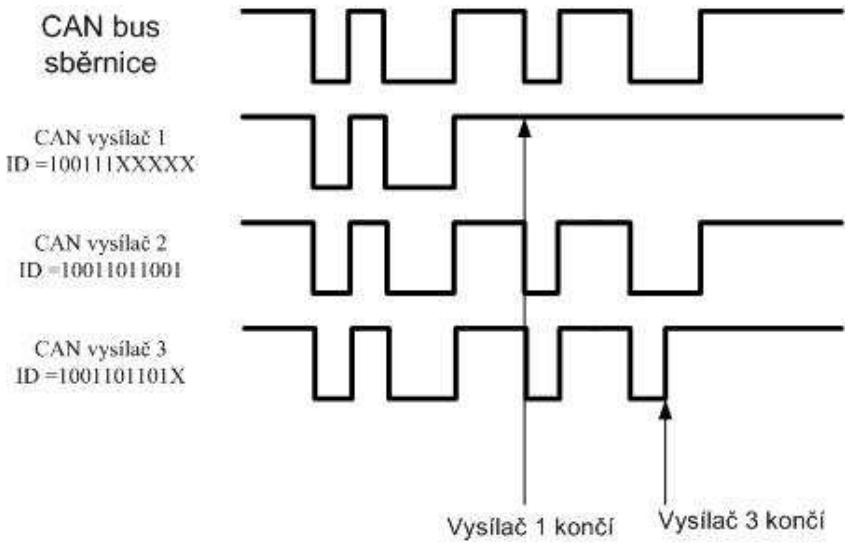
Linková vrstva

Při předávání dat do CAN není žádný uzel adresován. Používá se všesměrové vysílání broadcast. Rozhodnutí o přijetí zprávy zpracovává každý uzel pomocí filtru sám. Kritérium pro rozhodnutí o přijetí je takzvaný identifikátor zprávy a je přenášen se zprávou.

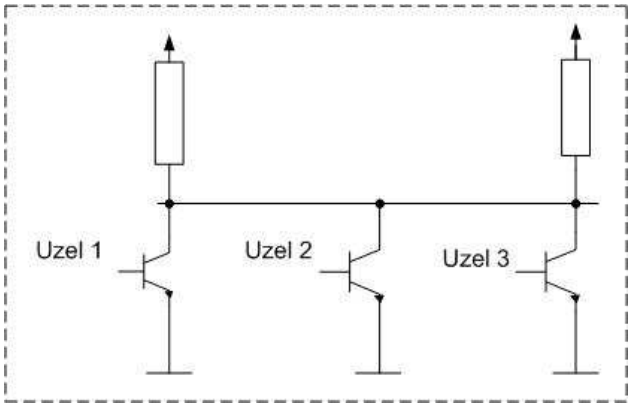
3.3 Přístup na CAN sběrnici

Na sběrnici se mohou vyskytnout dva stavy recesivní log. 1 a dominantní log 0. Recesivní stav je na sběrnici, pokud je sběrnice v klidu. To je vytvořeno pomocí otevřeného kolektoru, znázorněno na obr. (3.3). Pokud chce jakákoliv stanice začít vysílat, na sběrnici se připojí dominantní bit (Log. 0).

Případné konflikty na sběrnici jsou vyřizovány přístupovou arbitráží podle identifikátoru každého uzlu a RTR bitu sledováním každého bitu na sběrnici. Při vysílání každá stanice monitoruje stav sběrnice. Pokud stanice zjistí na sběrnici jiný stav, než vyslala, ihned se odpojí a pokusí se o vysílání, až bude sběrnice v klidu. Tímto je zajištěno, že nedojde k destruktivní kolizi a postupně získá přístup na sběrnici pouze ten uzel, který má nejvyšší prioritu. Arbitrážní přístup je znázorněn na obrázku (3.2)



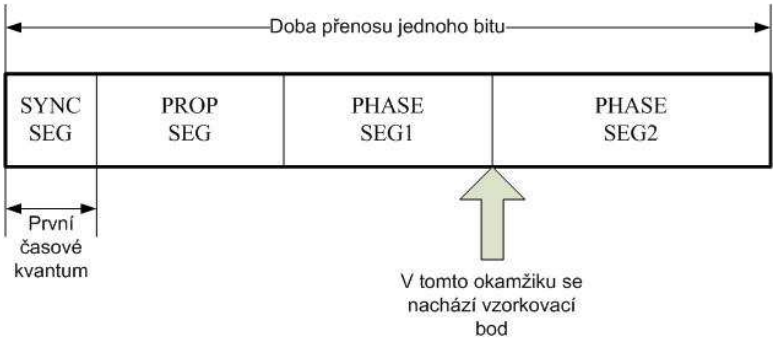
Obrázek 3.2 Arbitrážní přístup na sběrnici [11]



Obrázek 3.3 Princip připojení uzlů na sběrnici [11]

3.4 Způsob přenosu jednoho bitu

Bit je rozložen na 4 části. Každá tato část je celým násobkem časového kvanta, které vychází od kmitočtu oscilátoru hodin řadiče. Tyto časové poměry mezi úseky je možno měnit v závislosti na vlastnostech vedení a zpoždění signálu. Jednotlivé časové úseky jsou znázorněny na obrázku (3.4)



Obrázek 3.4 Přenos jednoho bitu po CAN [11]

SYNC SEG (Synchronisation segment) – očekává se hrana signálu
PROP SEG (Propagation time segment) – Kompenzace doby šíření signálu po sběrnici
PHASE SEG1 a **PHASE SEG2** – mezi těmito intervaly je vzorkovací bod

3.5 Kódování signálu

Kódování dat na sběrnici CAN se provádí pomocí kódu NRZ (non return to zero). Pro datový a remote rámec je zde ještě použito kódování signálu pomocí bitového stuffingu obr. (3.5). To znamená, že při kódování je po každých pěti stejných bitech vložen jeden bit opačné úrovně a při dekódování je zase po pěti stejných bitech vyjmut.

Vyslaná data	010100000 11011111 1100
Vyslaná data s bitovým stuffingem	010100000 **1** 11011111 **0** 1100

Obrázek 3.5 Příklad kódovaní s bitovým Stuffingem [11]

3.6 Protokoly na sběrnici CAN

Standardní formát CAN 2.0A

Obrázek 3.6 Formát rámce CAN 2.0A Standard [11]

SOF (Start of Frame) – Start bit, udává začátek rámce

ID (Identifier) – 11 bitový identifikátor zprávy a zároveň určuje prioritu zprávy

RTR (Remote Transmission Request) – Tento bit určuje, jestli zpráva žádá o data, nebo jestli data obsahuje.

IDE (Identifier Extension) – Tento bit určuje, zda se jedná o standardní formát, nebo rozšířený formát.

R0 (Reserved) – Rezerva 1 bit pro budoucí využití.

DLC (Data Length) – Tyto 4 bity informují o počtu přenášených bytů (0 - 8)

Data Přenášená data (0 – 64 bitů)

CRC (Cyclic Redundancy code) – 15 bitový cyklický redundantní kód, který zabezpečuje přenos dat

ERC (End of CRC) – 1 bit, který ukončuje pole CRC

ACK (Acknowledge Field) – Pokud se tento bit změní z recesivního na dominantní, obdržela alespoň jedna stanice zprávu bez chyb.

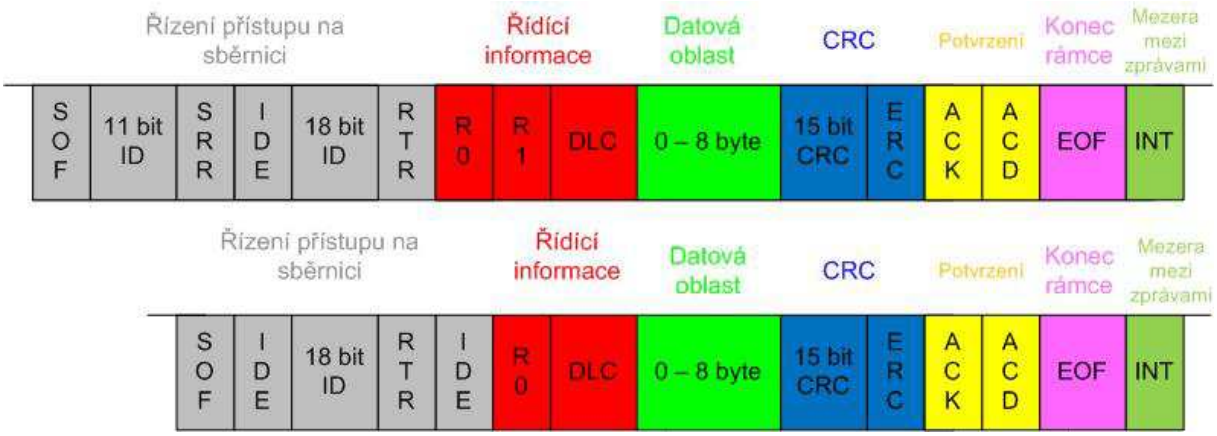
ACD (*Acknowledge Delimiter*) – Tento bit odděluje potvrzovací bit

EOF (End of frame) – Pole 7 bitů ukončuje zprávu. V tuto dobu mohou jednotlivé přijímače informovat vysílač o chybném přijetí zprávy

INT (Intermision field) – Prodleva před vysíláním dalšího rámce, která trvá 3 bity

Bus Idle Nastane po uplynutí INT pokud nezačne žádná stanice vysílat

Standardní formát CAN 2.0B



Obrázek 3.7 Formát rámce CAN 2.0B Extended [11]

Formát rámce CAN 2.0B je shodný s rámcem CAN 2.0A, ale v CAN 2.0B jsou definovaná navíc tři pole.

SRR (Substitute Remote Request) – přenášen jako recesivní, standardní rámec má vždy větší prioritu než rozšířený.

ID (Identifier) – Druhá část identifikátoru zprávy, která má délku 18 bitů

R1 (Reserved) - Rezerva pro budoucí využití.

Sběrnice CAN dovoluje, aby na sběrnici byli oba typy formátů zpráv. Řadič, který podporuje formát CAN 2.0B dovoluje také přenášet formát CAN 2.0A a vyšší prioritu bude mít řadič CAN 2.0A.

3.7 Typy zpráv na sběrnici CAN

Datová zpráva

Slouží k předávání dat od vysílače k ostatním uzlům.

Žádost o zprávu

Úkolem této zprávy je vyžádání od stanice. Má stejnou strukturu jako datová zpráva, ale neobsahuje žádná data a RTR bit je v recesivním stavu. Jako identifikátor se použije číslo uzlu, od kterého je vyžadována zpráva. Pokud nastane situace, kdy ve stejný okamžik začne druhý uzel vysílat data, dojde při arbitráži až k bitu RTR. Podle RTR se rozhodne, že ve vysílání bude pokračovat ten vysílač, který vysílá data. To je dáno tím, že bit RTR je při vysílání dat v dominantním stavu.

Chybová zpráva

Tato zpráva je vyslána, pokud přijímač rozpoznal chybu. Máme dva druhy chybových rámců

Active error frame

- Error flag – 6 za sebou jdoucí dominantních bitů
- Error delimier – 8 za sebou jdoucích recesivních bitů

Passive error frame

- Error flag – 6 za sebou jdoucí recesivních bitů
- Error delimier – 8 za sebou jdoucích recesivních bitů

Vysláním chybové zprávy Active error frame se poruší správnost dat na sběrnici a vysílač opakuje vysílání. Chybový rámec se vyšle na konci přenosu End of frame pouze z důvodu nesouhlasu CRC. Při opakování přenosu dochází opět k bitové arbitráži, takže se může stát, že některá zpráva bude mít vyšší prioritu než zpráva opakovaná a proto může opakovaná zpráva dorazit se zpožděním.

Detekce a potvrzování chybových stavů

- **Error active** - jednotka může odeslat Active error frame
- **Error Pasive** – Jednotka může vysílat pouze passive error frame. Když je uzel vysílač a detekuje chybu, vyšle pasive error frame, tím je porušen bitový stuffing a dochází k vysílání active error frame uzly, které zprávu vysílají. Pokud je uzel přijímač tak vysláním passive error frame neovlivní činnost na sběrnici, pouze zvýší hodnotu čítače chybových stavů a při správném příjmu opět hodnotu sníží.

- **Bus off –** jednotka nesmí ovlivňovat dění na sběrnici.

Typy chybových stavů

- **Bit error –** vysílač monitoruje sběrnici a hlásí chybu, pokud nejsou na sběrnici vyslaná data.
- **Stuff error –** Přijímač detekuje chybu, pokud je na sběrnici více jak 5 stejných bitů
- **CRC error –** Detekce chyby kontrolního součtu, detekuje přijímač
- **Form error –** Hlídá se správnost polarity polí EOF, Intermission, ACK, ECR,.
- **Acknowledge –** vysílač nedostane potvrzení o přijetí zprávy.

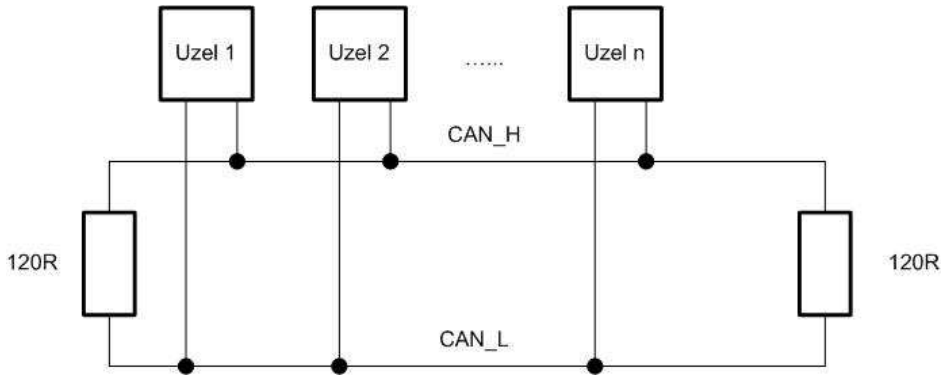
Zpráva o přeplnění

Má stejnou strukturu jako chybová zpráva. Touto zprávou je informován vysílač, že některý z přijímačů nezpracoval předcházející zprávu. Také dochází k vysílání této zprávy, pokud byl objeven dominantní stav v poli intermission. Těchto zpráv může být více za sebou a nezpůsobí opakování zprávy.

3.8 Hardwarová realizace

Princip připojení uzlů na sběrnici

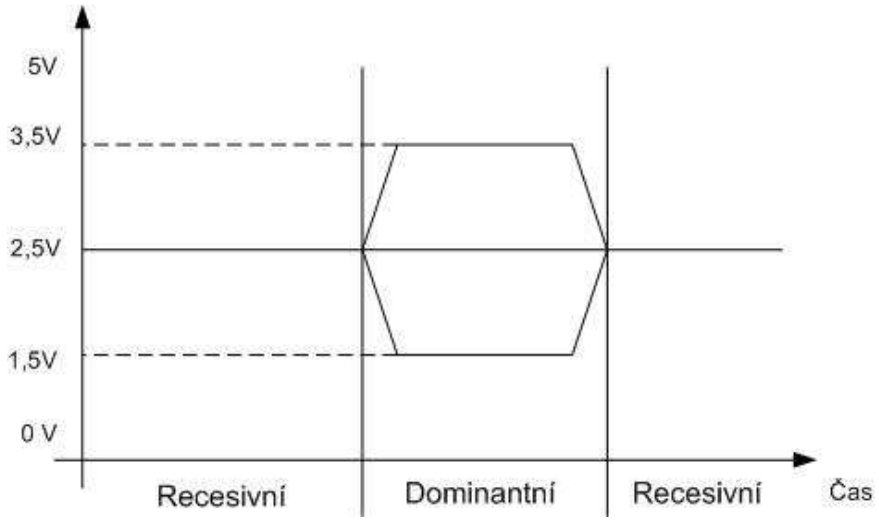
Pro realizaci sběrnice je použito kroucených dvouvodičů o charakteristické impedanci 120Ω. Na těchto dvou vodičích jsou úrovně CAN_H a CAN_L, znázorněno na obrázku 3.8. Maximální počet připojených stanic závisí na typu použitého transceiveru a hlavním parametrem je minimální impedance, kterou je možno transceiver zatížit.



Obrázek 3.8 Způsob připojení uzlů na sběrnici [11]

Napěťové úrovně na sběrnici CAN

Napěťové úrovně se rozlišují podle jednotlivých standardů hight speed a low speed. Tyto jednotlivé standardy nejsou mezi sebou kompatibilní. Znázornění hight speed je na obrázku 3.9 a low speed je na obrázku 3.10.



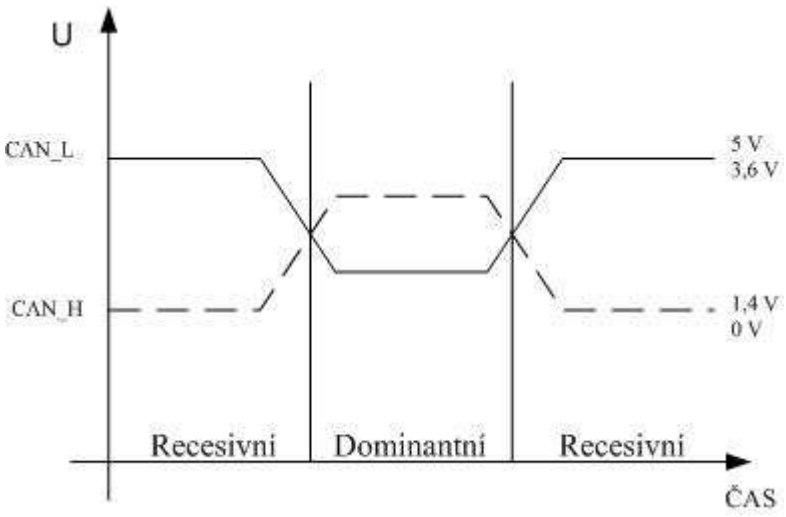
Obrázek 3.9 Napěťová úroveň pro CAN – hight speed [11]

Dominantní stav:

- CAN_H = 3,5 V
- CAN_L = 1,5 V.

Recesivní stav:

- CAN_H = CAN_L =2,5 V



Obrázek 3.10 Napěťové úrovně pro CAN – low speed [11]

Dominantní stav:

- CAN_H = 3,6 V
- CAN_L =1,4 V

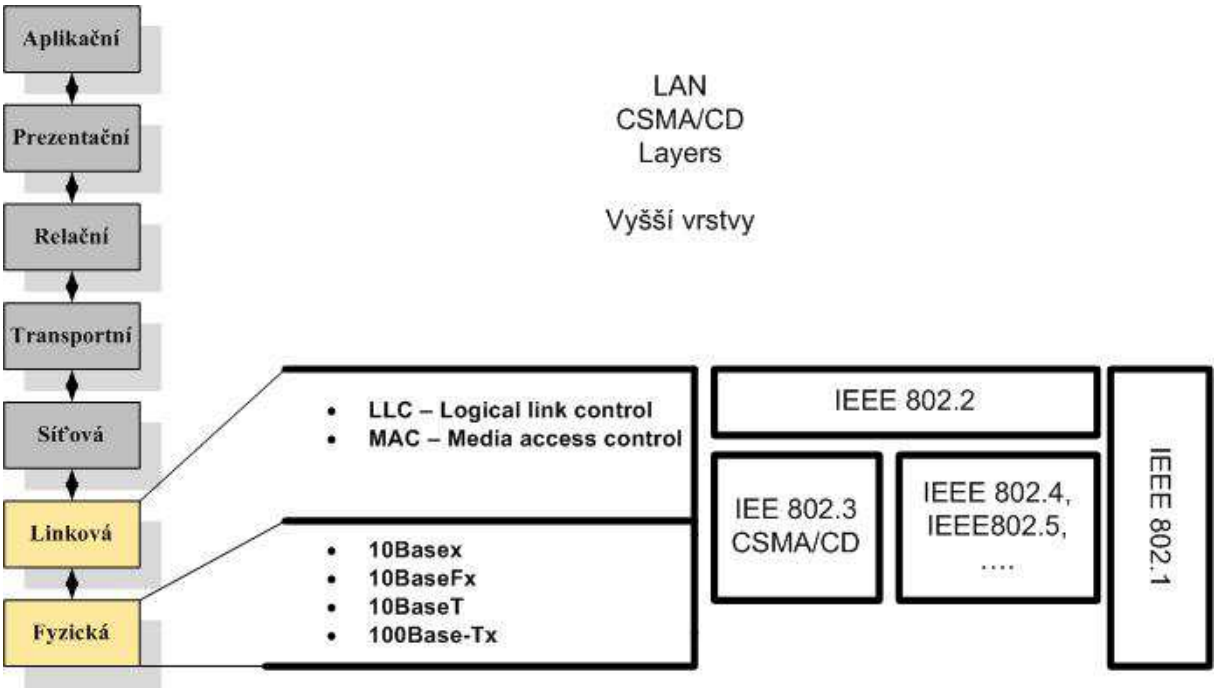
Recesivní stav:

- CAN_H = 0 V
- CAN_L = 5 V.

4 Popis komunikačního rozhraní Ethernet

Jeho počátky sahají do období 1968 až 1972, kdy byla vytvořena první radiová síť Aloha a fungovala na principu CSMA/CD (Multiple Access with Collision Detection). Další vývojový stupeň byl v roce 1972 až 1977od firmy Xerox. Byl přidán princip CS (Carrier sense) a dosahovalo se přenosové rychlosti 2,94 Mbit/s. Na toto navazuje projekt firmy Digital a Intel a je vytvořena specifikace DIX Ethernet – první verze 10 Mbit/s. V roce 1985 byl převzat mezinárodní organizací IEEE a byl uveden jako specifikace 802.x, definující fyzickou a linkovou vrstvu komunikace.

Struktura Ethernet protokolu



Obrázek 4.1 Struktura Ethernet [9]

IEEE 802.1 – specifikuje celkovou strukturu komunikačního uspořádání lokálních a metropolitních sítí. Dále specifikuje všeobecné procedury síťového managementu a návaznost na vyšší vrstvy.

IEEE 802.2 – Má dvě linkové podvrstvy. První je řízení logického spoje LLC (Logical link control) a druhá je MAC klient, která je společná všem přístupovým metodám.

IEEE 802.3 – Linková podvrstva přístupu ke sdílenému médiu MAC a definice fyzických přenosových medií CSMA/CD¨

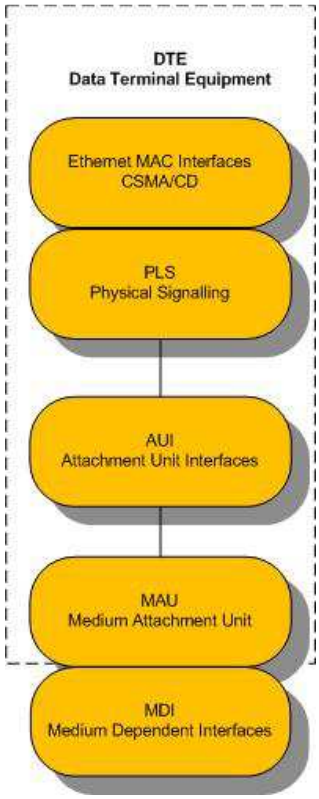
4.1 Fyzická vrstva

Ethernet umožňuje flexibilní volbu přenosového média a topologie sítě. Standart IEEE 802.3 definuje model fyzického rozhraní (obr. 4.2) DTE koncové datové zařízení zabezpečuje pomocí vrstvy Ethernet MAC Interface přístupovou metodu CSMA/CD, formovaní dat do rámců a jejich vysílání. Další vrstva je PLS, která provádí kódování a dekódování přijatých a vysílaných dat kódem Manchester. Signály na rozhraní PLS odpovídají specifikaci rozhraní AUI, které připojuje jednotku MAU. Jednotka MAU realizuje připojení ke komutačnímu

mediu prostřednictvím MDI, který je obvykle konektor.

Celá tato fyzická vrstva je realizována transceiverem. Tyto transceivery pracují v duplexním nebo poloduplexním režimu a nejčastěji podporují verze 10BASE-T a 100BASE-TX. Hlavní rozdíl je v přenosové rychlosti, kde 10BASE-T dosahuje 10MBit/s za použití kódování manchaster a 100BASE-TX dosahuje 125Mbit/s při kódování 4B/5B, MLT-3 a NRZI.

Obě tyto verze používají médium kroucenou dvojlinku UTP kategorie 3 až 5 a konektor RJ45. Maximální délka kabelu UTP může být až 100 metrů. Pro připojení je užito dvou párů vodičů, kde jeden pár je pro příjem a druhý pro vysílání.



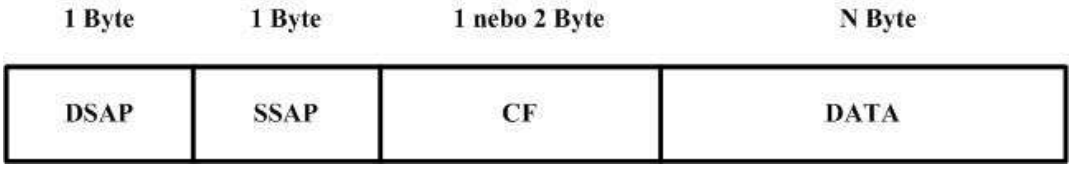
Obrázek 4.2 Model fyzického rozhraní [9]

4.2 Linková vrstva

4.2.1 Podvrstva LLC – řízení logického spoje

Tato vrstva je nezávislá na přístupové metodě a fyzické implementaci sítě. Jejím hlavním úkolem je vytváření a rušení linkových spojení, poskytování přenosových služeb, adresace komunikačních procesů na linkové úrovni, řízení toku dat a kontrola chyb.

LLC určená pro rámec IEEE 802.3 je rozhraní mezi MAC vrstvou a vyššími protokolovými vrstvami. Obecná struktura podvrstvy LLC je znázorněna na obrázku 4.3



Obrázek 4.3 Obecná struktura datového bloku LLC [9]

DSAP (*Destination Service Access Point*) – Cílový přístupový bod. Sedm bitů určuje vlastní SAP a LSB bit určuje typ cílové adresy (individuální – 0, skupinová - 1).

SSAP (Source Service Access Point) – Zdrojový přístupový bod. LSB určuje typ (příkaz – 0, odpověď - 1).

CF (*Control field*) – Řídící pole rámce. Je závisle na typu přenášeného rámce. A toto pole je zrcadlově otočeno proti skutečnému bitovému toku.

DATA Vlastní přenášená data.

LLC vrstva má dvě základní služby

1. **Datagramová služba**

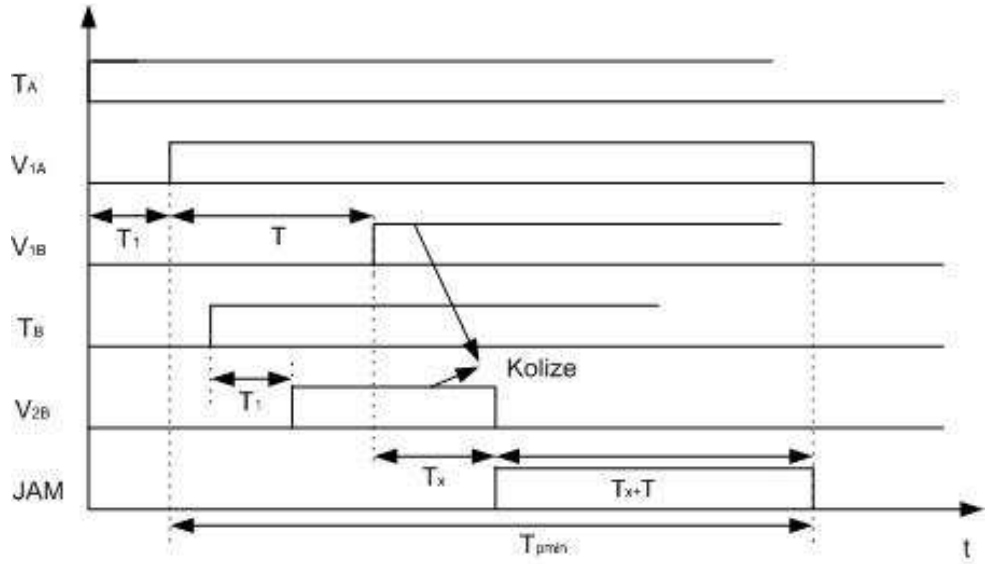
   Tato služba vykonává nepotvrzovaný přenos dat. Jedná se o nezabezpečený přenos dat, kde jsou rámce přenášeny bez potvrzení cílovou stanicí a zabezpečení zajistí vyšší vrstvy.

2. **Služba virtuálních spojení**

   Před započetím přenosu dat musí LLC navázat virtuální spojení mezi dvěma uzly a adresují se jen individuální stanice.

4.2.2 Podvrstva MAC

Tato podvrstva řídí přístup ke společné sběrnici na základě náhodného přístupu CSMA/CD. Z tohoto řízení vyplývá, že každá stanice připojená na sběrnici monitoruje provoz. Jakmile stanice detekuje klidový stav, může začít vysílat. Zároveň na sběrnici může vysílat pouze jedna stanice. Může se stát, že začnou vysílat dvě nebo více stanic najedou. V tom případě dojde ke kolizi. Stanice, která první detekuje kolizi, začne vysílat kolizní posloupnost takzvaný JAM. Po zjištění kolizního stavu se všechny stanice odmlčí a opakují vysílání po náhodném čase. Časový proměny při detekci kolize jsou znázorněny na obrázku 4.4.



Obrázek 4.4 časové poměry při detekci kolize [9]

T_a Je požadavek na vysílání stanice 1

V_{1a} Signál vyslaný stanicí 1

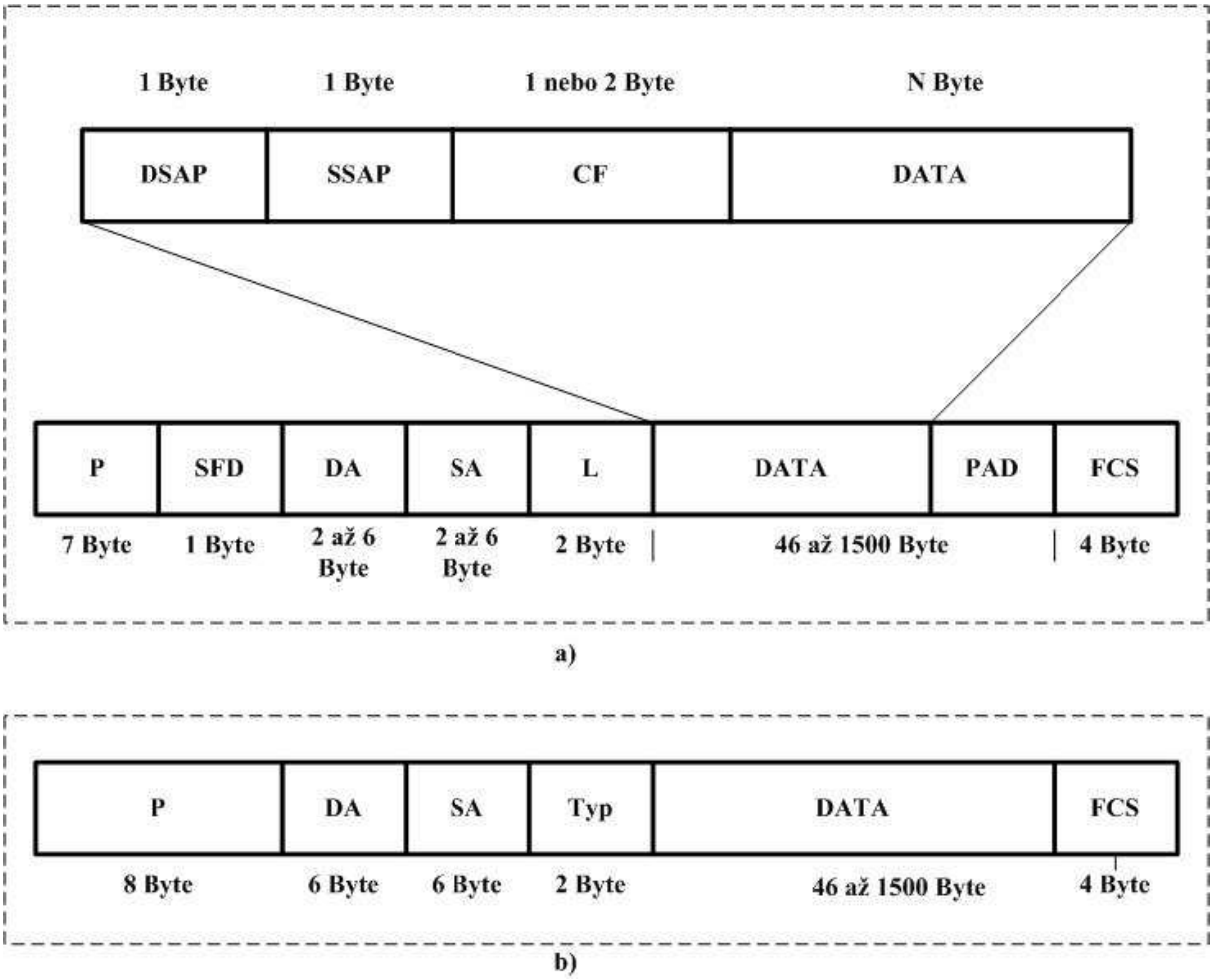
V_{1b} Signál vyslaný stanicí 1 z pohledu stanice 2

T_b Požadavek na vysílání stanice 2

V_{2b} Signál vyslaný stanicí 2

Formát rámce Ethernetu

V Ethernetu jsou využity dva základní typy rámců. První rámec se nazývá Ethernet II a jsou v něm využívány služby vyšších vrstev a vrstva LLC je průchozí. Druhý rámec se nazývá Ethernet 802.3 a služby vrstvy LLC využívá. Tyto rámce sou zobrazeny na obrázku 4.3



Obrázek 4.5 Formát rámce Ethernetu [9]

a) Formát rámce Ethernet IEE 802.3 **b) Formát rámce Ethernet II**

P (Preambule) – Toto pole slouží k označení začátku vysílaného rámce a synchronizaci hodin přijímací stanice. Má délku 64 bitů pro rámec Ethernet II a 56 pro rámec IEEE 802.3.

SFD (Start frame delimiter) - Je takto označován poslední bajt preambule u rámce IEE 802.3. Tento bajt slouží jako oddělovač rámce a má hodnotu 10101011

SA (Source address) – Zdrojová adresa (takzvaná MAC adresa) má 48 bitů, kde poslední bit je vždy nulový

DA (Destination address) - Cílová adresa jedné nebo více stanic. Má 48 bitů a nejnižší bit reprezentuje typ zprávy. Typ může být individuální LSB = 0, skupinová LSB = 1 nebo broadcast, kde všech 48 bitů je jedničkových

L (Length) – U rámce IEE 802.3 určuje délku rámce.

TYP (Type) – V rámci Ethernet II udává protokol vyšší vrstvy. Má velikost 16 bitů a jeho velikost je vždy větší než 1536 oproti standardnímu formátu rámce, kde toto pole určuje skutečnou délku rámce.

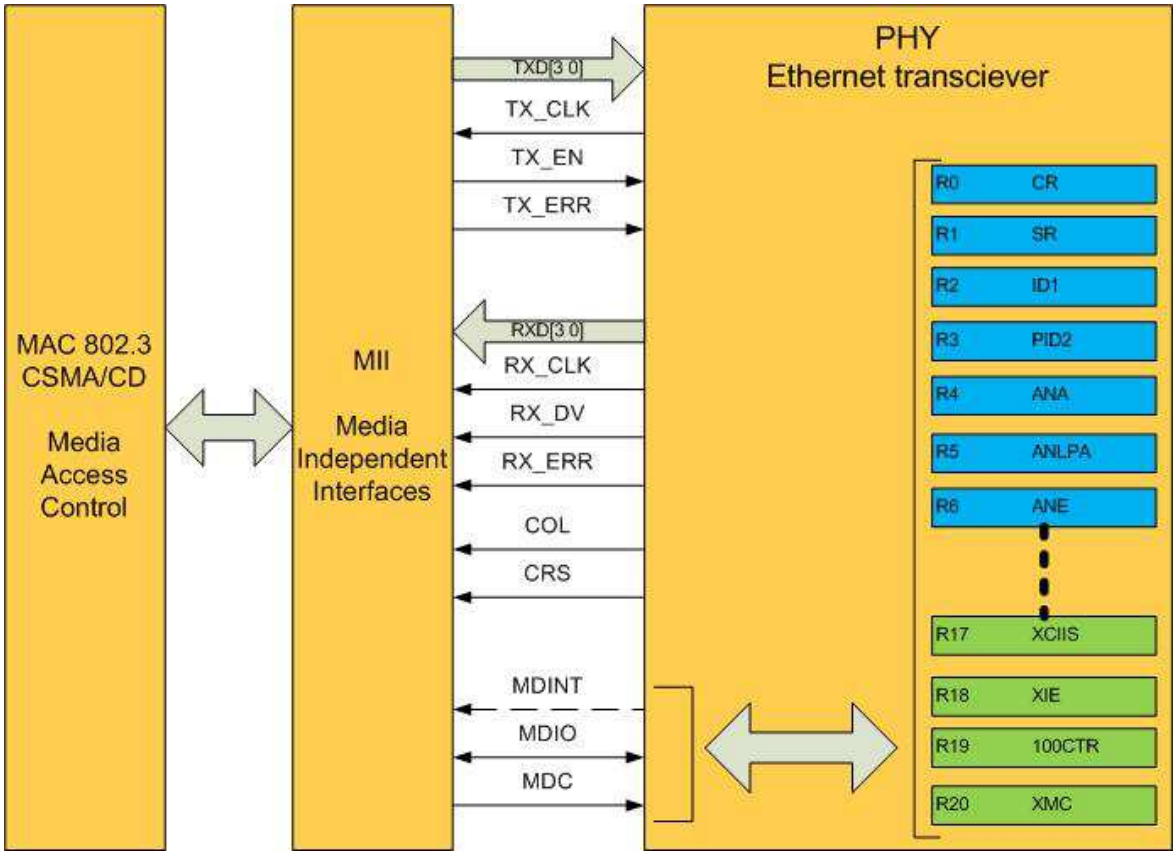
DATA Přenášená data, která jsou předmětem komunikace. Velikost dat je v rozmezí 46 až 1500 bajtů. Minimální počet dat je zde z důvodu detekce kolize.

PAD U rámce IEEE 802.3 je použit jako výplň pro dosažení minimální délky rámce.

FCS Zabezpečení rámce 32 bitovým cyklickým redundantním kódem (CRC).

4.3 MII Rozhraní

MII rozhraní zajišťuje spojení mezi MAC vrstvou a Ethernet transceiverem, který je specifický pro každé fyzické přenosové medium. Rozhraní definuje vzájemné signály, které tyto dvě jednotky propojují. MII rozhraní je zobrazeno na obrázku 4.7 S řídícími signály rozhraní je vhodné se zmínit o standardizované sadě řídících a stavových registrů Ethernet transceiveru, které jsou přístupné přes řídící signály MII rozhraní.



Obrázek 4.7 MII Rozhraní [9]

Signály MII rozhraní můžeme rozdělit do 3 skupin

1. Signály pro přenos datového toku.
2. Signály pro sledování stavu linky.
3. Řídící signály MII rozhraní.

Signály pro přenos obousměrného datového toku

TXD [3:0] – Jedná se o čtveřici datových bitů vyslaných do transceiveru z jednotky MAC.

TX_CLK – Jedná se o hodinový signál, který poskytuje časovou referenci pro přenos vyslaných dat. Využívá se 2,5 MHz pro přenosovou rychlost 10 Mbit/s a 25 MHz pro přenosovou rychlost 100 Mbit/s

TX_EN – Jedná se o povolení vysílání. Tento signál je generován, když jsou na vývodech platná data.

TXD[3:0] - Signál musí být aktivován před první bitovou čtveřicí a musí být aktivní, dokud nejsou všechna data odvysílána.

TX_ERR – Jedná se o chybové hlášení. Tento signál je aktivní, když v průběhu vysílání dojde k chybě a zůstává aktivní po dobu jedné nebo více period hodinového signálu

RXD [3:0] – Jedná se o čtveřici datových bitů vyslaných z transceiveru do jednotky MAC.

RX_CLK – Jedná se o hodinový signál, který poskytuje časové reference pro přenos přijímaných dat. Využívá se 2,5 MHz pro přenosovou rychlost 10 Mbit/s a 25 MHz pro přenosovou rychlost 100 Mbit/s.

RX_DV – Jedná se o potvrzení přijatých dat. Tento signál je generován, když jsou na vývodech RXD[3:0] platná data získaná obnovením a dekódováním datového toku.

RX_ERR – Jedná se o chybové hlášení. Tento signál je aktivní, když při příjmu rámce dojde k chybě. Tento signál zůstává aktivní po dobu jedné nebo více period hodinového signálu.

Signály pro sledování stavu linky

COL – Detekce kolize. Je to asynchronní hladinový signál, který musí být aktivní po celou dobu kolize. Tento signál má smysl pouze u poloduplexního přenosu. Pro duplexní režim je tento signál neaktivní.

CRS – Detekce nosné. Signál je generován, když na vysílací nebo přijímací straně je zaznamenaná aktivita. Signál je asynchronní a je aktivní po celou dobu aktivity. Využívá se pouze u poloduplexního přenosu.

Řídící signály MII rozhraní

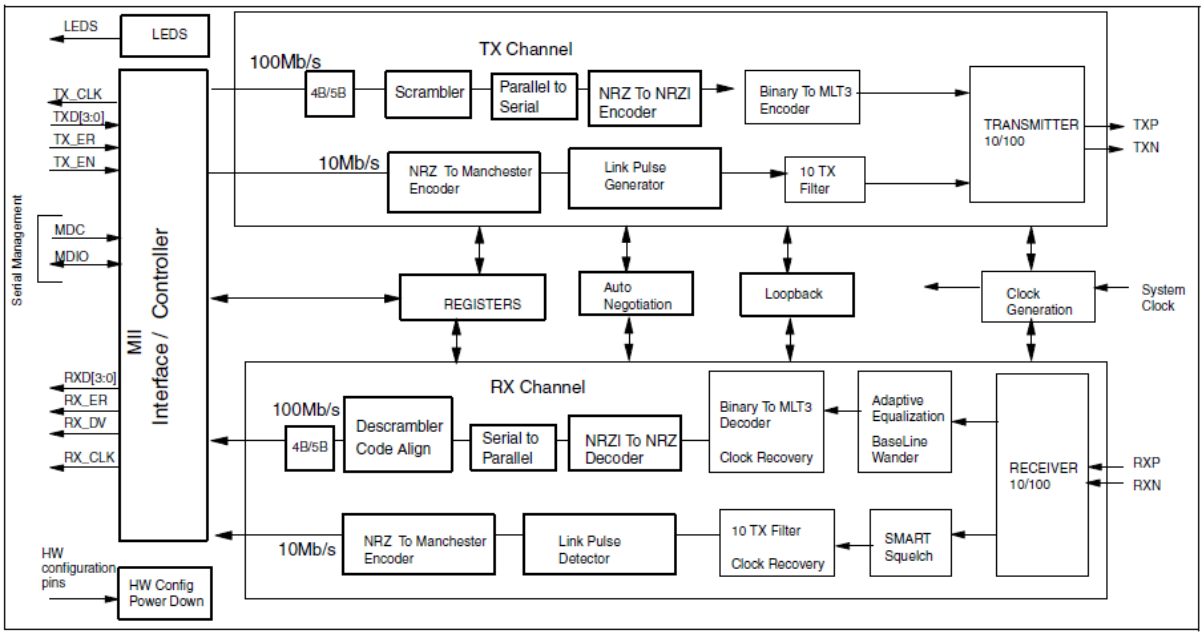
Tyto signály definují protokol a signály při konfiguraci Ethernet transceiveru.

MDIO – Obousměrný datový signál pro přenos řídících a stavových informací transceiveru.

MDC – Slouží k synchronizaci datového toku MDIO. Jedná se o aperiodický hodinový signál o frekvenci maximálně 25 MHz a klidový stav je 0.

4.4 Ethernet transceiver

V této části se podíváme na Ethernet transceiver STE100P. V tomto transceiveru jsou obsaženy dvě specifikace Ethernetu. První je IEEE 802.3i – 10Base - T a IEEE 802.3u – 100Base – TX. Vnitřní zapojení tohoto Ethernet transceiveru je v datasheetu [20] a zobrazeno na obrázku 4.8. V zapojení jsou vidět jednotlivé funkční bloky transceiveru a hlavně rozdílná cesta signálu pro 10 Mbit/s a 100 Mbit/s.



Obrázek 4.8 Ethernet transceiver STE100P [20]

Cesta signálu ve specifikaci IEEE 802.3u

Tento řetězec může dosahovat rychlosti až 100 Mbit/s a je na vysílací straně a v inverzní podobě na přijímací straně. Tento řetězec obsahuje bloky 4B/5B, scrambler, parallel to serial, NRZ to NRZI encoder a Binary to MLT3 encoder.

Blok 4B/5B přiřazuje každé čtveřici bitů pětici bitů tak, aby kombinace obsahovala minimálně dvě 1.

Blok scrambler vytváří z toku dat pseudonáhodnou posloupnost. Toto scramblování zabraňuje vzniku velkých úrovní signálů a zlepšení elektromagnetické kompatibility.

Blok parallel to serial převádí paralelní data na sériová

Blok NRZ to NRZI převádí data z NRZ na NRZI. Princip NRZI je, že pokud následující bit je 1 změní se stav a pokud je 0, zůstane stav nezměněn.

Blok binary to MLT3 encoder stlačuje spektrum signálu více do spodní části z důvodu elektromagnetické kompatibility. Princip je takový, že pokud bit je 1, stav se mění podle cyklu 1- 0 - -1 - 0 - 1 - 0 a pokud je bit 0, stav se nemění.

Cesta signálu ve specifikaci IEEE 802.3i

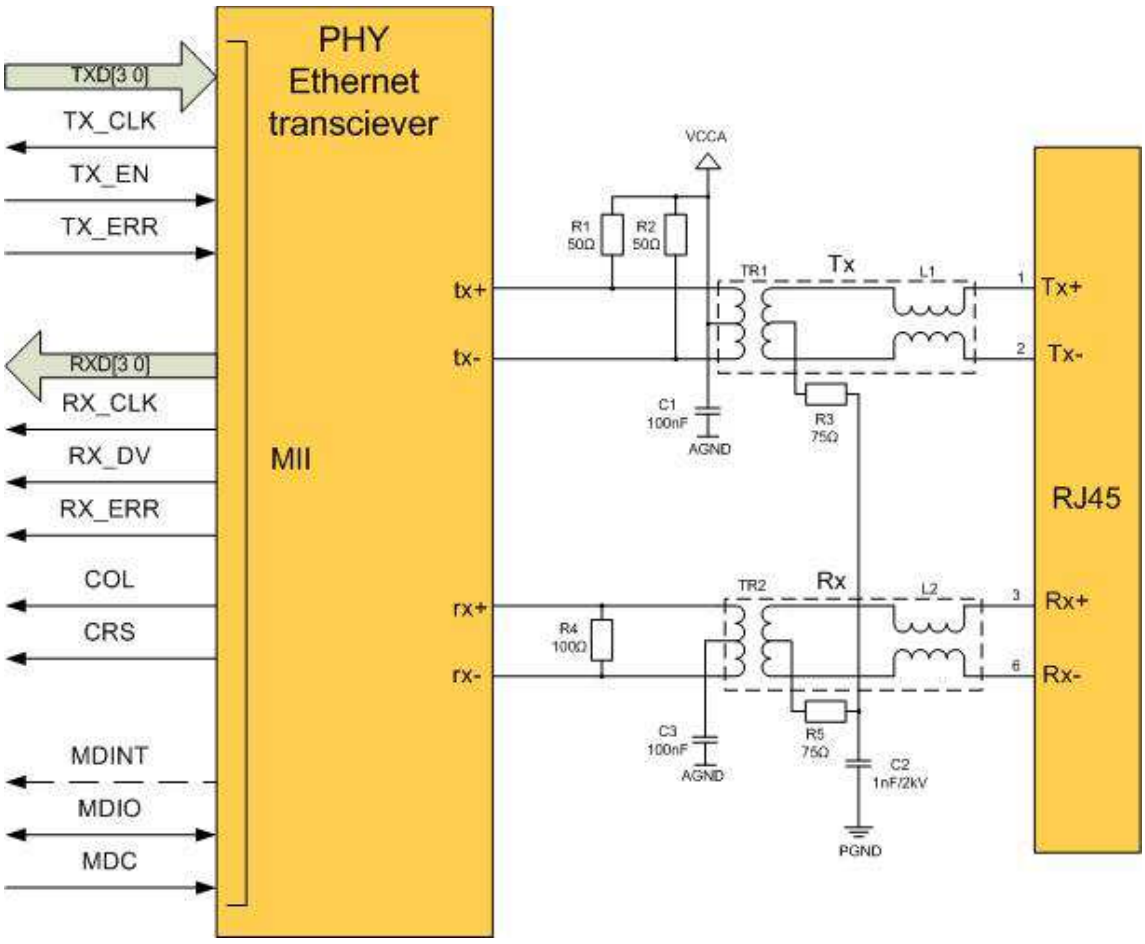
Tento řetězec dosahuje rychlosti 10 Mbit/s a je mnohem jednoduší než předchozí. Tento řetězec obsahuje blok NRZ to Manchester encoder, který převádí data z NRZ do kódu

Manchester. Princip kódování Manchester je takový, pokud je bit 0, na výstupu bude stav 10 a pokud je stav 1, na výstupu máme stav 01. Přechod z 0 na 1 se děje uprostřed periody původního signálu.

Druhý blok je Link pulse generátor a tento blok generuje pulsy pro funkci auto-negotation

Výstupní obvod Ethernet transceiveru

Na obrázku 4.9 je uveden výstupní obvod Ethernet transceiveru. Z obrázku je vidět, že obvod má přijímací část a vysílací část a obě tyto části končí v konektoru RJ45.



Obrázek 4.9 Výstupní obvod Ethernet transceiveru [9]

Vysílací obvod

Vysílaná data jsou generována transceiverem na pinech tx+ a tx- a jsou přivedena k budícímu transformátoru TR1 s převodem 1:1. Transformátor má vyvedený střed, který je

napájen napětím VCCA. Z toho vyplívá, že budící proud protéká přes polovinu vinutí do odpovídajícího vstupu. Sekundární vinutí transformátu je připojeno přes tlumivku na konektor RJ45. Tlumivka a transformátor mají společný magnetický obvod a tlumivka je zapojena tak, aby kompenzovala souhlasnou složku rušícího napětí. Odpory R1 a R2 slouží k impedančnímu přizpůsobení linky. Odpor R3 a kondenzátor C1 opět potlačuje souhlasné rušivé napětí.

Přijímací obvod

Přijímací obvod je principiálně stejný jako vysílací, ale signál je veden směrem od konektoru RJ45 do transceiveru. Tlumivka zde opět potlačuje souhlasné rušivé napětí. Transformátor TR2 nemá nyní napájen střed, protože piny rx+ a rx- jsou vstupní a obsahují komparační obvody. I zde je linka impedančně přizpůsobena. O impedanční přizpůsobení se stará odpor R4.

Přenosové medium

K propojení dvou stanic se používají dva druhy kabelů. První typ je přímý kabel. Tento kabel se používá k propojení koncové stanice se směrovačem. Druhý typ kabelu je křížený kabel. Ten se využívá k propojení dvou koncových stanic. Nejčastější typy kabelů, které používáme, jsou UTP a STP. Oba typy mají charakteristickou impedanci 100 Ω a maximální vzdálenost mezi jednotlivými uzly může být 100 metrů. Konektory, které se používají pro tyto kabely, jsou RJ45 a jsou zobrazeny na obrázku 4.10

Obrázek 4.10 Konektor RJ45

4.5 Protokoly vyšších vrstev

4.5.1 Síťová vrstva

Síťová vrstva má za úkol logické adresování síťových uzlů a zajišťuje přenos mezi nimi. Dále umožňuje komunikaci mezi jednotlivými uzly sítě, které nejsou přímo propojeny. Proces, který vyhledá nejvhodnější cestu, se nazývá routing (směrování) a pro tento účel bylo vyvinuto několik protokolů.

- Protokol intersítě (IP)
- Protokol mapování adres (ARP)
- Protokol řídících hlášení (ICMP)
- Protokol správy skupin (IGMP)
- Směrovací protokoly OSPF, IGRP a RIP

4.5.2 Transportní vrstva

Transportní vrstva modelu OSI zajišťuje segmentaci toku dat do bloků a poskytuje efektivní přenosové služby. Tato vrstva přidává dále informace, které slouží k opětovnému složení datové zprávy na straně příjemce. Rozhraní mezi transportní a aplikační vrstvou se označuje číslem portu. Čísla portů mohou být v rozsahu mezi 0 až 65535. V transportní vrstvě se využívají dva protokoly

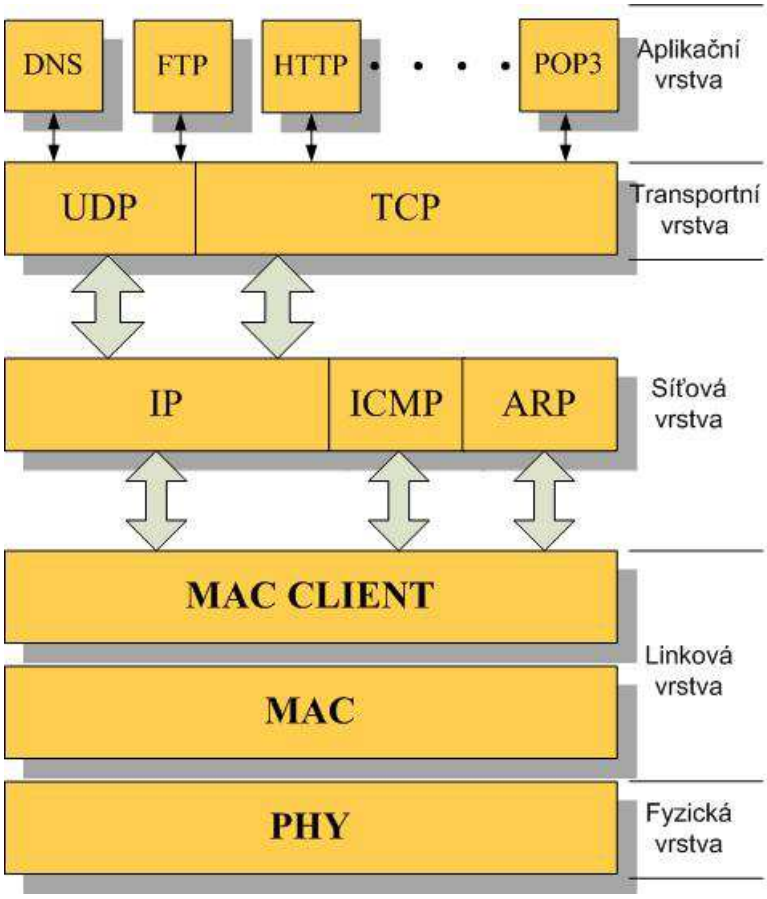
- **User datagram protokol (UDP)** – transportní služba bez spojení

- **Transmission control protokol (TCP)** – transportní služba se spojením a řízením koncového spojení

4.5.3 Aplikační vrstva

V Ethernetu nejsou využívání služby Relační a Prezentační vrstvy a proto je lze považovat za transparentní. Aplikační vrstva zprostředkovává přístup ke komunikačním systémům a v této vrstvě se používají tyto protokoly HTTP, FTP, SMTP, POP3, DHCP, NTS a mnoho dalších.

Složení celého Ethernetu je na obrázku 4.6



Obrázek 4.6 Návaznost protokolových vrstev Ethernetu [9]

5 Základní vlastnosti mikrokontroléru

STM32F105xx a STM32f107xx procesory obsahují vysoce výkonné Cortex™-M3 32bit RISC jádro pracuje na frekvenci 72 MHz. Tento procesor má vysokorychlostní vestavěnou paměť flash až 256 kB a SRAM 64 kB a rozsáhlý počet vstupů a výstupů a periferií je připojen ke dvěma APB sběrnicím. Tyto vlastnosti dělají tyto procesory vhodné pro širokou škálu aplikací, jako jsou motorové pohony a ovládání aplikace, průmyslové aplikace, PLC, měniče, tiskárny, skenery a domácí audio zařízení.

Základní vlastnosti mikrokontroléru STM32F105RB

- Jádro: ARM 32-bit Cortex™-M3 CPU, maximální frekvence 73 MHz
- Flash paměť 64 až 256 KB.
- Operační paměť SRAM 64 KB
- Napájení od 2 do 3,6 V
- Krystal 3 – 25 MHz, interní oscilátor RC 8MHz, interní RC s kalibrací 40 kHz
- Při poklesu napájení jsou zálohovány registry baterií
- 2 x 12 bitový A/D převodník (16 kanálů) s převodem 1 µS a s rozsahem 0 až 3,6 V
- 2 x 12 bitový D/A převodník
- 12 DMA kanálů
- 80 I/O portů, 5 V tolerance
- JTAG rozhraní, sériová linka
- Až 4 16 bitové časovače, 1 x 16 bitový čítač pro řízení motoru
- 2 x časovače watchdog.
- Až 14 komunikačních rozhraní
    - 2 x I^2C
    - 5 x USART s IRDA, LIN, ISO7816
    - 3 x SPI (18 Mbit)
    - 2 x CAN 2.0B
    - USB 2.0
    - 10/100 Etherntet MAC s DMA MII/RMII

6 Návrh obvodového zapojení

V této kapitole budou popsány jednotlivé bloky převodníku ETHERNET-CAN, jeho obvodového zapojení a nastavení.

6.1 Napájení

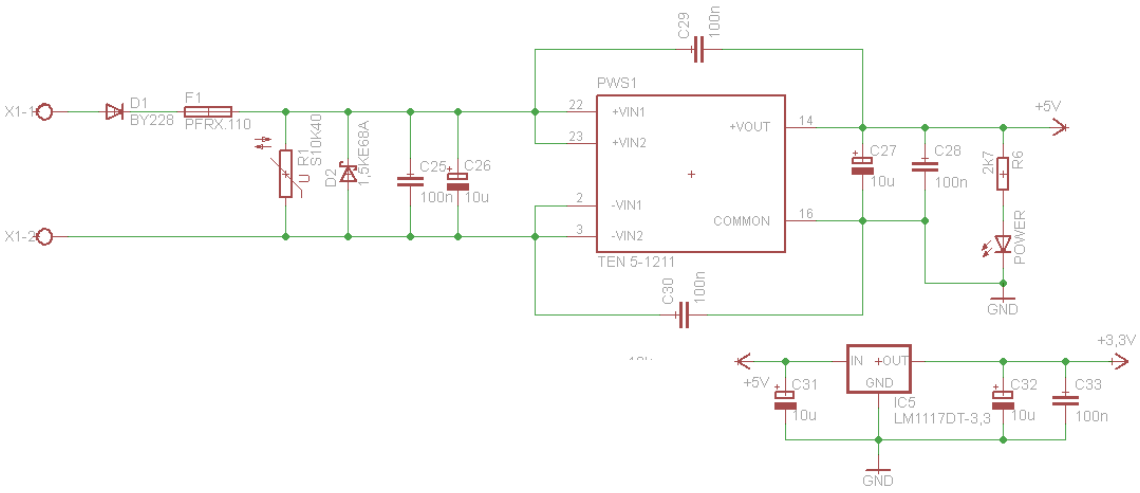
Napájení přivedeme na konektor X1 a můžeme přivést stejnosměrné napětí od 9 do 18V. Signalizace zapnutého napájení je provedena zelenou led diodou POWER_ON.

Popis zapojení:

Prve se podíváme na napájení přes konektor X1. Hned za konektorem je daná dioda D1, která slouží proti přepólování vstupního napětí. Za diodou je umístěna vratná pojistka F1. Tato pojistka je na proud 1 A. Proti přepěťovým špičkám zde máme varistor R1 a transil D2. Pro filtraci napětí máme zde 2 kondenzátory (keramický C25 o kapacitě 100nF a tantal C26 o kapacitě 10 uF). Po filtraci napětí následuje DC/DC měnič TEN 5-1211. Tento DC/DC měnič má vstupní napětí od 9 do 18 V a výstupní napětí má 5 V. Tento měnič je opatřen bezpečnostními kondenzátory C29 a C30 o kapacitě 100 nF a pro napětí 2 kV. Na výstupu měniče máme dva kondenzátory. Tantalový C27 o kapacitě 10 uF a keramický C28 o kapacitě 100 nF. Za těmito kondenzátory je umístěna led dioda, která signalizuje zapnuté napájení.

Pro napájení mikrokontroléru a dalších obvodů, které potřebují napětí 3,3 V, je zde použit LDO stabilizátor typu LM1117DT-3,3. Tento stabilizátor je napájen 5 V a je opatřen filtračními kondenzátory C31, C32 a C33 o kapacitě 10 uF a 100 nF.

Schéma zapojení napájení je zobrazeno na obrázku 6.1



Obrázek 6.1 Schéma zapojení napájení

6.2 Mikrokontrolér

Na obrázku 6.2 je zobrazeno schéma zapojení okolí mikrokontroléru. Zde jsou zakresleny a popsány jednotlivé části obvodů mikrokontroléru.

Napájení

Procesor je napájen napětím 3,3 V. Toto napětí je přivedeno na piny 13, 19, 32, 48, 64 a zem je připojena na piny 12, 18, 31, 47, 64. Na každý tento vstup je připojen kondenzátor o kapacitě 100 nF. Dále má procesor piny Vbatt, ke kterému je připojena baterie. Tato baterie složí jako záložní zdroj, když dojde k výpadku napájení.

Reset

Dále má procesor vstup resetu. Na vstup resetu (NRST) je připojeno tlačítko a kondenzátor C36 o kapacitě 100nF. Tento kondenzátor zde slouží pro odstranění zákmitů. Dále je signál veden na Ethernet transceiver STE100p a R-Link.

Krystaly

Na mikrokontroléru jsou připojeny dva krystaly. První je Q2 o hodnotě 25 MHz. Tento je připojen na piny 5 a 6 a slouží jako hlavní hodiny pro celý procesor. Druhý krystal Q3 o hodnotě 32 kHz je připojen na pin 3 a 4

JTAG

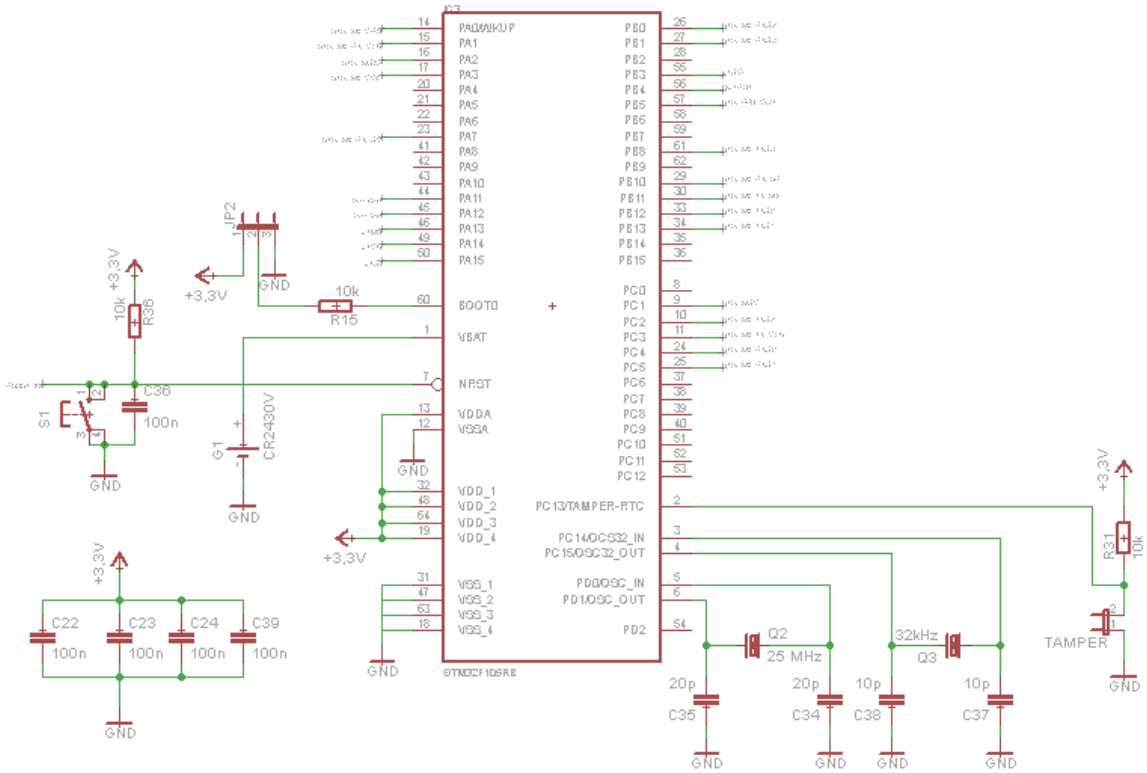
Na programování procesoru v desce jsem si zvolil JTAG programátor R-Link. Pro tento programátor je na desce konektor o 20 pinech. V tabulce 6.1 je uvedeno zapojení jednotlivých pinů.

Tabulka 6.1 Zapojení konektoru R-Link

Pin	Signál	Pin	Signál
1	3,3V	11	
2	3,3V	12	GND
3	NJTRST	13	JTDO
4	GND	14	GND
5	JTDI	15	RESET_IN
6	GND	16	GND
7	JTMS	17	
8	GND	18	GND
9	JTCK	19	
10	GND	20	GND

Tamper

Vstup tamper je na pinu dva a tento vstup je připojen na propojku. Touto propojkou určujeme, zda bude na vstupu tamper logická 1, nebo logická 0.

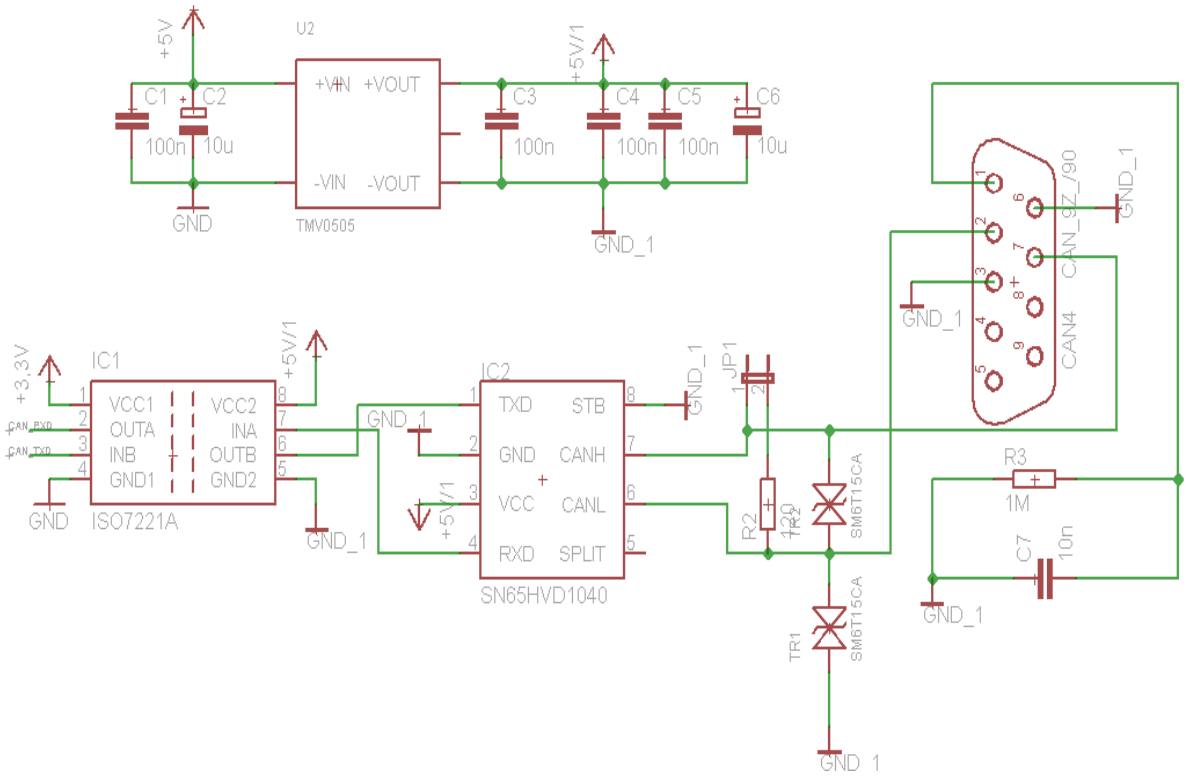


Obrázek 6.2 Schéma zapojení okolí mikrokontroléru

6.3 CAN

Pro napájení transceiveru je použit DC/DC měnič TMV0505. Dále je zde obvod pro galvanické oddělení pomocí izolátoru IC1. Tento izolátor zároveň upravuje napěťové úrovně z 3,3 V logiky procesoru na 5 V logiku CAN transceiveru. Tento transceiver má nastavitelnou strmost hran. Tato strmost se nechá nastavit pomocí odporu, který bychom připojily na pin STB na transceiveru a na zem. Vedení je možno zakončit odporem R2 o velikosti 120 Ω pomocí propojky JP1. Proti přetížení sběrnice jsou v obvodu transily. Pro připojení kabelu nám slouží konektor cannon 9.

Připojení sběrnice CAN k procesoru je za pomoci dvou signálů. První je CAN_RxD, který je připojen na pin 44 a druhý CAN_TxD je připojen na pin 45. Zapojení CAN transceiveru je na obrázku 6.3



Obrázek 6.3 Schéma zapojení CAN

6.4 Ethernet

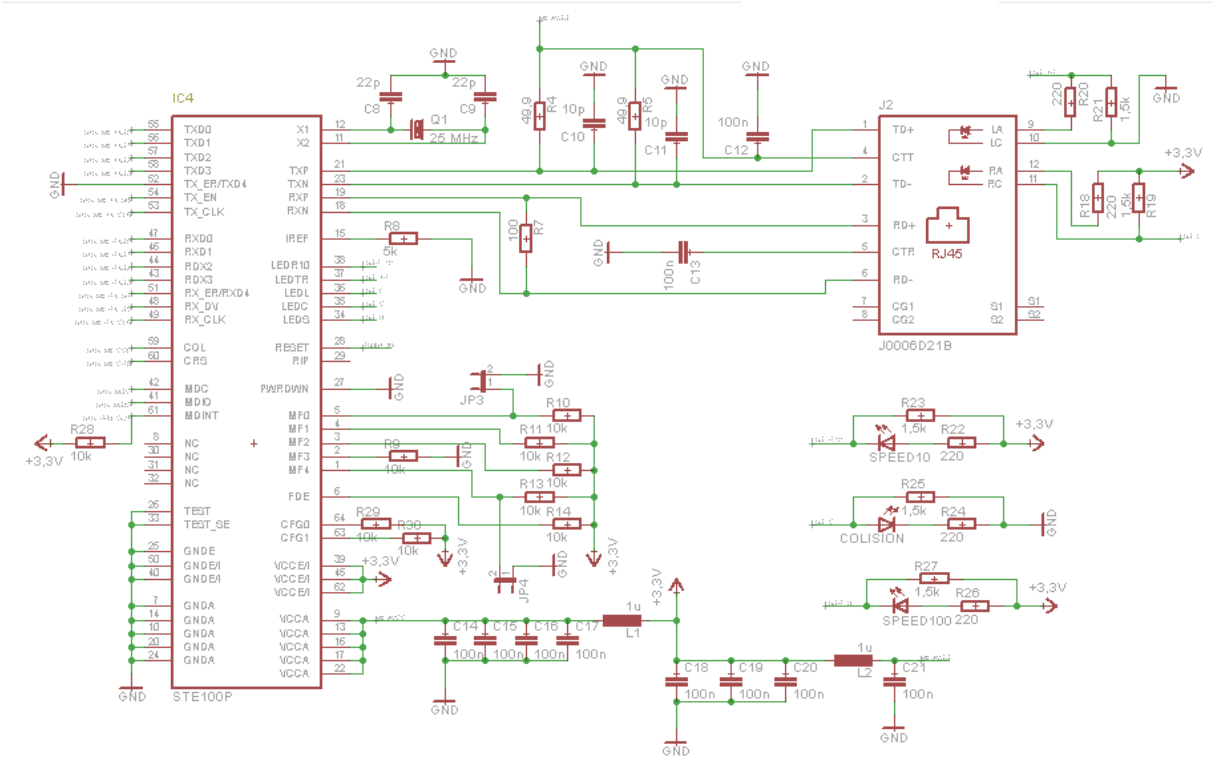
Pro zapojení Ethernetu jsem použil transceiver STE100P. Tento obvod je napájen napětím 3,3 V a má rozdělenou digitální a analogovou část. Digitální část je napájena přímo napětím 3,3 V a analogová je napájena přes cívku L1. Dále je přes cívku L2 napájen střed transformátoru. Jako zdroj hodinového signálu nám zde slouží krystal Q1 na frekvenci 25 MHz. U tohoto transceiveru máme možnost volit rychlost přenosu dat 100 Mbit/s nebo 10 Mbit/s a nebo zvolit funkci auto-negotitation. K tomuto nastavení slouží propojky JP3 a JP4. Tento transceiver obsahuje MII rozhraní, vstupní signály na sběrnici a signály pro nastavení vnitřních registrů. Popis jednotlivých signálů tohoto Ethernet transceiveru, je uveden v následující tabulce.

Tabulka 6.2 funkce jednotlivých pinů ethernet transceiveru

Pin	Název	Popis
52 58 57 56 55	Txd4 Txd3 Txd2 Txd1 Txd0	Piny pro přenos dat
54	Tx_en	Tento vstup je aktivní, když jsou na vstupech txd platná data
53	Tx_clk	Hodinový signál pro přenos 25MHz pro 100Mbit/s 2.5MHz pro 10Mbit/s
52	Tx_er	Tento vstup je aktivní pokud došlo v přenosovém datovém toku k chybě
51 43 44 46 47	Rxd4 Rxd3 Rxd2 Rxd1 Rxd0	Piny pro příjem dat
48	Rx_dv	Tento signál potvrzuje, že přijatá data jsou platná
51	Rx_er	Tento signál udává, že přijatá data jsou chybná
49	Rx_clk	Hodinový signál pro příjem 25MHz pro 100Mbit/s 2.5MHz pro 10Mbit/s
59	COL	Kolize – signál je aktivní po celou dobu kolize
60	CRS	Při half-duplexu určuje, zda je vysílací nebo přijímací medium k dispozici Pro full duplex určuje, jen jestli je volné přijímací medium
42	mdc	Řídící hodinový signál pro mdo sériový datový kanál
41	mdio	Řídící datový vstup/výstup. Obousměrný sériový datový kanál PHY rozhraní
61	mdint	Přerušení od řídícího kanálu
12	x1	Referenční vstup hodin 25 MHz
11	x2	Referenční výstup hodin 25 MHz
21 23	txp txn	Rozdílové výstupy 100BASE-TX nebo 10BASE-T
19 18	rxp rxn	Rozdílové vstupy 100BASE-TX nebo 10BASE-T
15	iref	Vstup referenčního proudu
38	ledr10	Výstup aktivní při detekci rychlosti 10 Mbit/s
37	ledtr	Při aktivitě na Rx/Tx je na tomto pinu frekvence 10 Hz
36	ledl	Signál je aktivní, pokud link test je pozitivní
35	ledc	V režimu full duplex je aktivní, pokud došlo ke kolizi
34	leds	Výstup aktivní při detekci rychlosti 100 Mbit/s
64	cfg0	Nastavení řízení 0
63	cfg1	Nastavení řízení 1
28	reset	Resetovací signál
29	rip	Aktivní pokud reset proběhl v pořádku
8,30, 31,32	nc	Nezapojené piny
26 33	test test_se	Testovací piny. Při normálním provozu by měly být připojeny na zem
27	pwrdown	Vypnutí transceiveru
6	fde	Povolení režimu fullduplex

5	mf0	Vypíná / zapíná auto negotion
4	mf1	Povoluje nrz – nrzi převod
3	mf2	Povoluje kódování 4B5B
2	mf3	Zapíná / vypíná scrambler
1	mf4	Rychlost 10 / 100 Mbit/s

Popis zapojení Ethernetu



Obrázek 6.4 Schéma zapojení Ethernet transceiveru

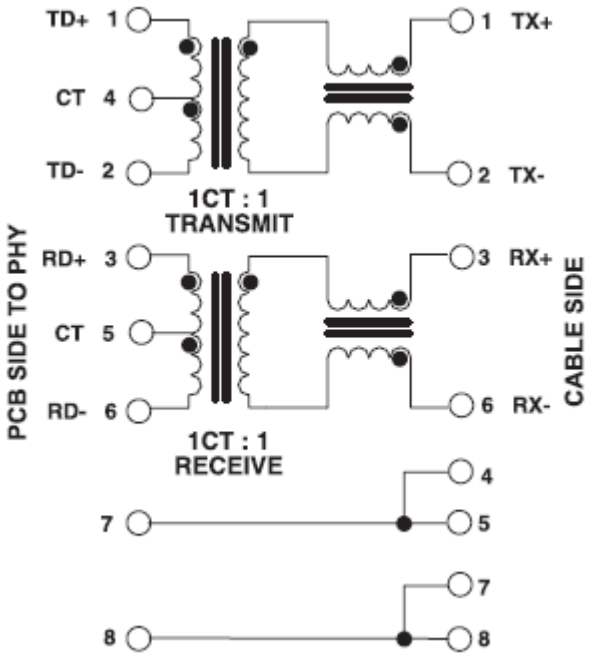
Na předchozím obrázku můžeme vidět schéma zapojení Ethernetu. Jako zdroj hodinového signálu je zde krystal Q1 o frekvenci 25 MHz. Dále můžeme propojkou JP3 zvolit jestli zapneme, nebo vypneme auto-negotiation a propojkou JP4 volíme rychlost 10 Mbit/s, nebo 100 Mbit/s. Signály Rx a Tx jsou přímo přivedené na výstupní konektor. Dále je na schématu vidět zapojení LED diod, které informují o stavu sběrnice. Připojení Ethernet transceiveru na procesor je v následující tabulce.

Tabulka 6.3 Zapojení Ethernet transceiveru na procesor

Brána procesoru	Signál
PB12	MII_TXD0
PB13	MII_TXD1
PC2	MII_TXD2
PB8	MII_TXD3
PB11	MII_TX_EN
PC3	MII_TX_CLK
PC4	MII_RXD0
PC5	MII_RXD1
PB0	MII_RXD2
PB1	MII_RXD3
PB10	MII_RX_ER/RXD4
PA7	MII_RX_DV
PA1	MII_RX_CLK
PA3	MII_COL
PA0	MII_CRS
PC1	ETH_MDC
PA2	MDIO

Vnitřní zapojení konektoru RJ45

Konektor RJ45 jsem použil J0006D21BNL. Jedná se konektor s oddělovacím transformátorem a dvěma signalizačníma led diodami.



Obrázek 6.5 Konektor RJ45 Vnitřní zapojení [16]

7 Návrh plošného spoje

Při návrhu DPS elektronických zařízení se musí sledovat několik základních aspektů. Samozřejmě se předpokládá, že zařízení bude bezpečné a spolehlivé. Dalším důležitým aspektem je i estetická stránka návrhu a v dnešní době je navíc důležité zohlednit důležitý aspekt EMC.

7.1 Základní pravidla pro návrh DPS v souladu s EMC

Základním požadavkem EMC je snížení vyzařování ze zařízení. Jelikož velikost vyzařování závisí hlavně na velikosti proudu, kmitočtu a ploše proudové smyčky patří mezi základní návrhová pravidla tyto vlastnosti.

1. Minimalizace hodnot proudu – volíme vhodné typy součástek, impedanční přizpůsobení a co nejmenší počet synchronně řízených obvodů
2. Minimalizace kmitočtového spektra - vyhnout se používání zbytečně rychlých součástek a nepoužívat zbytečně rychlou datovou komunikaci. Při návrhu je také vhodné filtrování a blokování napájení.
3. Minimalizace ploch proudových smyček – nutno vhodně rozmístit součástky na DPS, správně blokovat jejich napájení, vhodně vést spoje zemnění, vhodná konfigurace napájení I/O. Využívat řazení vrstev u vícevrstvých plošných spojů a použít ochranné paralelní spoje.
4. Filtrace I/O – filtraci použít pro ochranu I/O před ESD a přechodovými jevy, zamezí vyzařování do vedení.
5. Hodinové signály co nejkratší – Hodinový signál by měl být veden jako první, nebo druhý hned po napájení.
6. Vedení sběrnic – sběrnice by měly být co nejkratší a vodiče sběrnice by měly mít stejnou délku.
7. Maximalizace zemnící plochy - dbát na to, aby rozlitá měď byla i pod součástkami a vývody konektorů. Nejlépe je použít celou jednu zemnící vrstvu.

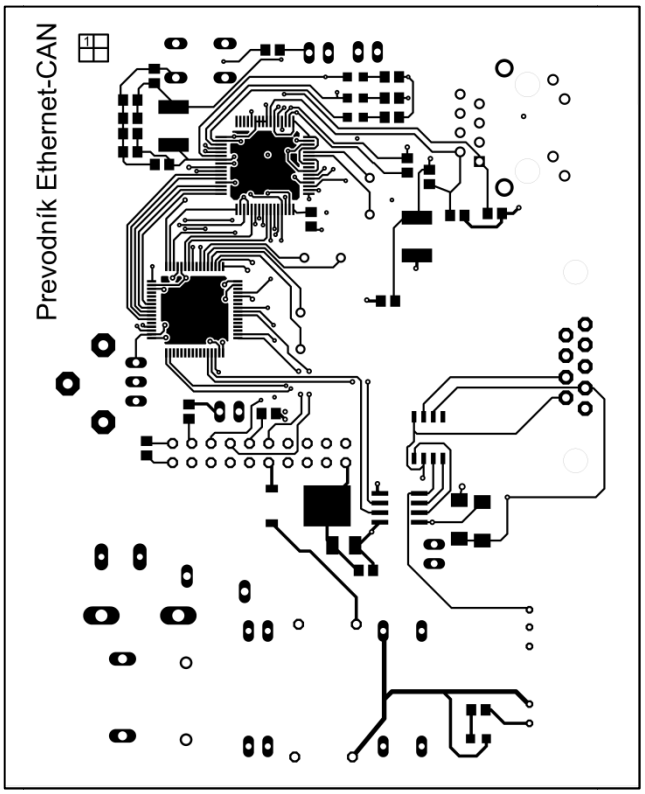
7.2 Realizace plošného spoje

Převodník Ethernet-CAN je realizován na 4 vrstvé desce, z důvodu vyšších kmitočtů. Signálové cesty jsou vedeny pouze na vrstvě první TOP (obr. 7.1) a čtvrté BOTTOM (obr. 7.2). Vrstva dvě a tři slouží pouze pro rozvod napájení. 3,3 V je vedeno ve třetí vrstvě (obr. 7.3) a GND je realizováno ve vrstvě druhé (obr. 7.4). Dále je zde ještě napájení pro DC/DC měnič. Toto napájení je realizované na vrstvě BOTTOM. Na obrázcích 4.2 a 4.3 je také vidět izolační mezera, která odděluje jednotlivé potenciály. Rozmístění součástek na desce plošného spoje je zobrazeno na obrázku 4.5 (BOTTOM) a 4.6 (TOP).

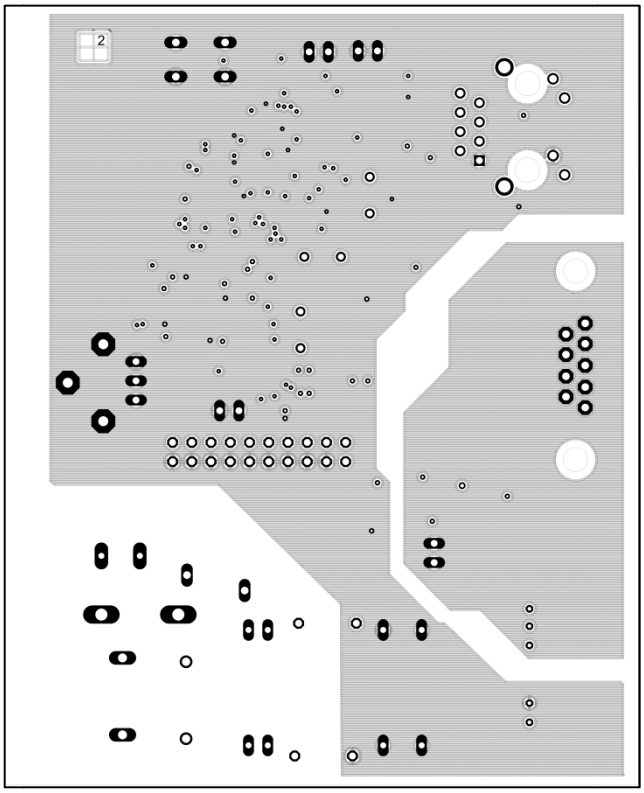
Při návrhu plošného spoje byly respektovány obecná pravidla pro EMC, tvary jednotlivých cest signálu a rozložení součástek na DPS.

Plošný spoj jsem navrhoval v programu Eagle ve verzi 5.6. Některé součástky nebyly součástí standardní knihovny Eaglu a proto jsem je musel dokreslit ručně. Soubory těchto knihoven jsou uložené na přiloženém CD.

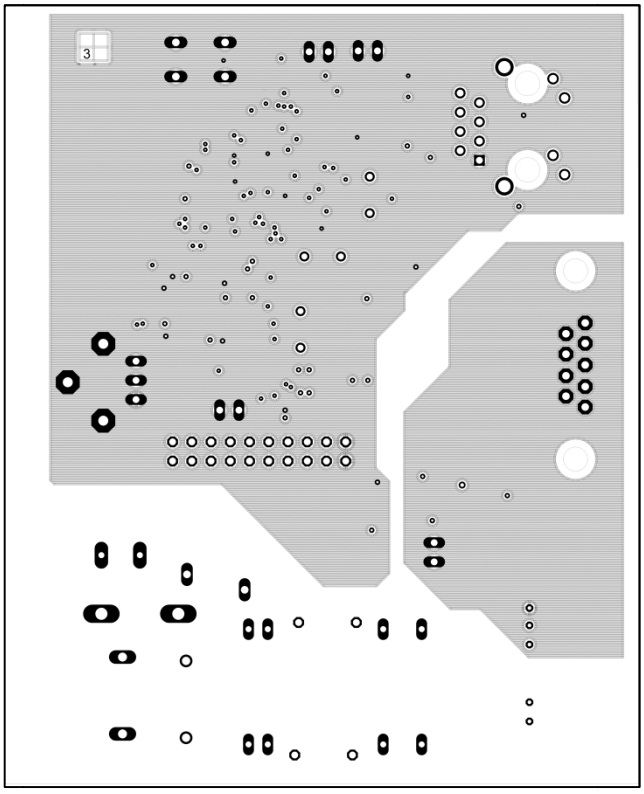
Data pro výrobu DPS, které byly z programu generovány, jsou ve formátu gerber RS274.



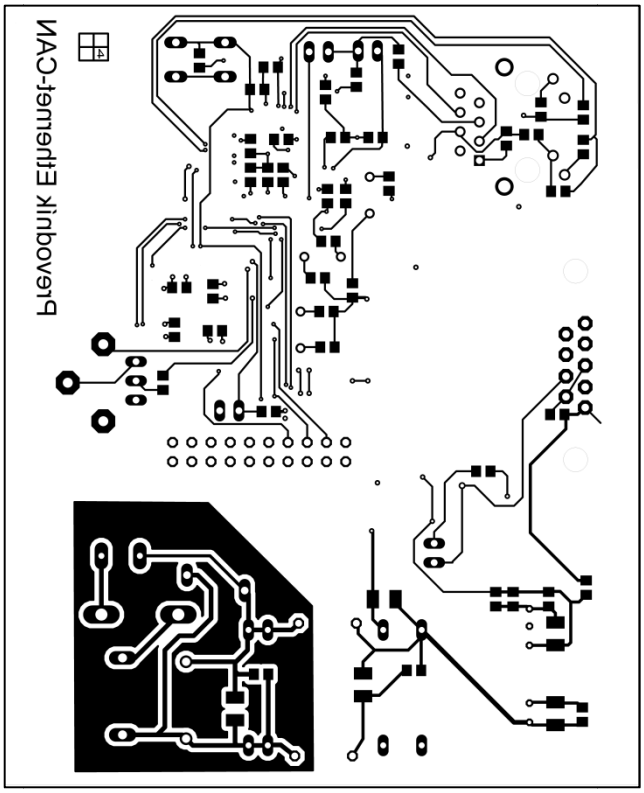
Obrázek 7.1 Vrstva TOP



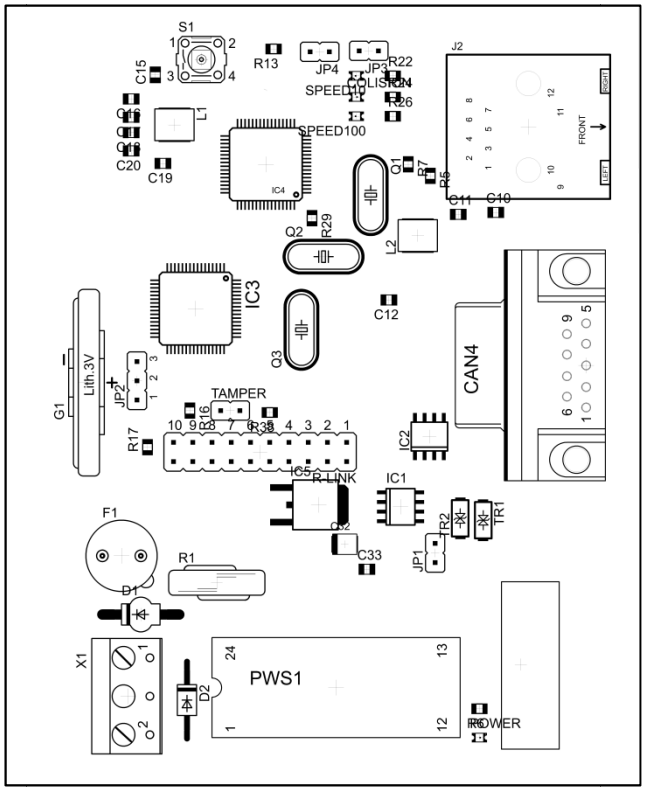
Obrázek 7.2 Vrstva GND



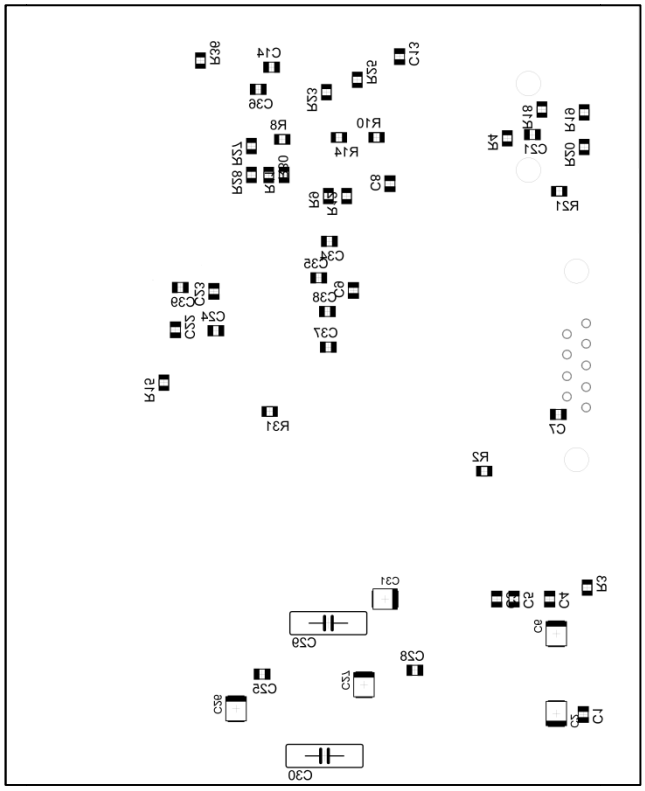
Obrázek 7.3 Vrstva 3,3V



Obrázek 7.4 Vrstva BOTTOM



Obrázek 7.5 Rozmístění součástek na straně TOP



Obrázek 7.6 Rozmístění součástek na straně BOTTOM

7.3 Seznam použitých součástek

Tabulka 7.1 Seznam součástek

Název	Popis	Hodnota	pouzdro
C1	Kondenzátor	100n	0805
C2	Kondenzátor	10u	SMC_B
C3-C5	Kondenzátor	100n	0805
C6-C7	Kondenzátor	10u	SMC_B
C8-C9	Kondenzátor	22p	0805
C10-C11	Kondenzátor	10p	0805
C12-C25	Kondenzátor	100n	0805
C26-C27	Kondenzátor	10u	SMC_B
C28	Kondenzátor	100n	0805
C29-C30	Kondenzátor	100n	C075-032X103
C21-C320	Kondenzátor	10u	SMC_B
C33	Kondenzátor	100n	0805
C34-C35	Kondenzátor	20p	0805
C36	Kondenzátor	100n	0805
C37-C38	Kondenzátor	10p	0805
C39	Kondenzátor	100n	0805
CAN3-CAN4	Konektor cannon	-	Cannon 9

Colision	LED	-	0805
D1	Dioda	BY228	SOD64-10
D2	Transil	1,5KE68A	DO41-10
F1	Pojistka	PFRX.110	19560
G1	Pouzdro baterie	CR2430V	CR2430V
IC1	Isolátor	ISO7221A	SO-08
IC2	CAN transceiver	SN65HVD1040	SOIC8
IC3	Procesor	STM30F105RB	TQFP64
IC4	Ethernet transceiver	STE100p	TQFP64
IC5	Stabilizátor	LM111DT-3,3	TO252
J2	Konektor RJ45	J0006d21B	J0-A/B/F
JP-JP4	Lišta s kolíky 2x1	-	-
L1-L2	Tlumivka	1u	L5650M
Power	LED	-	0805
PWS1	DC/DC měnič	TEN30510	TEN3XXXX
Q1-Q2	Krystal	25 MHz	HC49/S
Q3	Krystal	32 KHz	HC49/S
R-LINK	Lišta s kolíky 2 x 10	-	-
R1	Varistor	S10K40	S10K40
R2	Odpor	120	0805
R3	Odpor	1M	0805
R4-R5	Odpor	49,9	0805
R6	Odpor	2k7	0805
R7	Odpor	100	0805
R8	Odpor	5k	0805
R9-R17	Odpor	10k	0805
R18	Odpor	220	0805
R19	Odpor	1,5k	0805
R20	Odpor	220	0805
R21	Odpor	1,5k	0805
R22	Odpor	220	0805
R23	Odpor	1,5k	0805
R24	Odpor	220	0805
R25	Odpor	1,5k	0805
R26	Odpor	220	0805
R27	Odpor	1,5k	0805
R28-R31	Odpor	10k	0805
R36, R38	Odpor	10k	0805
S1	Tlačítko	-	B3F-10XX
SPEED10	LED	-	0805
SPEED100	LED	-	0805
TAMPER	Lišta s kolíky 1 x 2	-	-
TR1-TR2	Transil	SM6T15CA	DO214AA
U2	DC/DC měnič	TMV0505	SIP7
X1	Svorkovnice	-	-

8 Softwarové řešení

8.1 Inicializace obvodu

Pro vytvoření softwaru jsem si vybral univerzální programovací prostředí společnosti Raisonance Ride 7. Toto prostředí nabízí možnost programovat mikrokontroléry rodiny ARM a osmibitové 8051 a STM8. Toto prostředí nabízí neomezený compilér s možností debugobání a je v podporované společností ST. Jedinou nevýhodou tohoto prostřední je použití pouze RLink programátoru.

Základ programu tvoří referenční knihovny od ST microcontrolers dostupné na [24]. Tyto knihovny obsahují základní datové struktury, funkce a makra pro ovládání jednotlivých periferií mikrokontroléru. Při použití těchto knihoven není nutno znát podrobně jednotlivé registry mikrokontroléru. Výčet standardních knihoven je uveden v následující tabulce.

Tabulka 8.1 Standardní knihovny mikrokontroléru

Knihovna	Funkce
stm32f10x_adc	Obsluha A/D převodníku
stm32f10x_bkp	Obsluha backUp registrů
stm32f10x_can	Obsluha CAN sběrnice
stm32f10x_dma	Obsluha DMA kanálu
stm32f10x_exti	Obsluha přerušení
stm32f10x_flash	Obsluha flash
stm32f10x_gpio	Obsluha vstupních/výstupních pinů
stm32f10x_i2c	Obsluha I2C
stm32f10x_pwr	Obsluha napájení
stm32f10x_rcc	Obsluha resetu a clock control
stm32f10x_rtc	Obsluha hodin reálného času
stm32f10x_spi	Obsluha SPI
stm32f10x_tim	Obsluha čítačů časovačů
stm32f10x_usart	Obsluha Usart
stm32f10x_wwdg	Obsluha watchdog
CMSIS	Pro práci s jádrem procesoru

Po vytvoření nového projektu je potřeba vložit všechny potřebné knihovny. První knihovna, kterou budeme vkládat, je startup_stm32f10x_cl. Tento soubor je napsaný v assembléru a obsahuje počáteční nastavení stack pointru, vektoru přerušení, nastavení hodin a pamětí. Další knihovny, které je nutno vložit jsou core_CM3, systém_stm32f10x. Tyto knihovny jsou součástí CMSIS (Cortex Microcontroller Software Interface Standard) a umožňují práci s Cortex M3

jádrem. Po těchto knihovnách vložíme do projektu standardní knihovny pro ovládání periferií popřípadě další knihovny, které potřebuje pro náš projekt.

Jádra ARM jsou navrhována s ohledem na spotřebu. Z tohoto důvodu jsou po spuštění mikrokontroléru všechny periferie vypnuté. Proto musíme prve jednotlivé periférie povolit a přivést do nich hodinový signál.

V microkoltroléru jsou obsaženy dva externí oscilátory. První je rychlý oscilátor HSE o frekvenci 3 až 25 MHz a druhu pomaloběžný LSE o frekvenci 32,768 kHz, který je určen pro vnitřní jednotku reálného času. Dále je možno použít interní RC oscilátor o frekvenci pro HSI 8 MHz a pro LSI 40 kHz. Kmitočet HSE a HSI je poté ještě možno upravovat pomocí fázového závěsu a předděliček. Maximální hodnota, kterou můžeme pomocí těchto předděliček nastavit je 72 MHz.

8.2 Nastavení GPIO

V referenčním manuálu [14] zjistíme, že procesor má tři sběrnice AHB, APB1 a APB2 a každá periferie je na jednu z těchto sběrnic zapojena. Nyní musíme vybrané periferie povolit a přivézt do nich hodinový signál. To provedeme zavolání funkce *RCC_AHBPeriphClockCmd, RCC_APB1PeriphClockCmd a RCC_APB2PeriphClockCmd*, který mají v parametru adresu bitů zastoupené makrem. Nyní bude uveden příklad zdrojového kódu pro povolení a přivedení hodinového signálu do periferií. Tento uvedený kód není úplným nastavením hodin procesoru, ale pouze povolením přivedením hodin na periferie CAN a portu B a D.

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOB, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN1 | RCC_APB1Periph_CAN2, ENABLE);
```

Po přivedení hodin do jednotlivých bran je potřeba jednotlivé piny nastavit. Nejprve je nutné definovat rychlost, jakou bude pin pracovat a jeho parametry. Pro toto nastavení je připravena knihovna stm32f10x_gpio. V této knihovně je připravena struktura *GPIO_InitDef*, do které se zapíší parametry pinu. Zapsání do struktury se provede zavoláním funkce *GPIO_Init(),* která má parametr adresu struktury a název portu. Pokud budeme potřebovat přemapovat piny, nesmíme zapomenout pustit hodiny do jednotky AFIO. Přemapování se

provede příkazem *GPIO_PinRemapConfig()*. Nyní následuje ukázka kódu pro nastavení pinu 6 na bráně B na alternativní funkci CAN2.

```
GPIO_InitTypeDef  GPIO_InitStructure;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOB, &GPIO_InitStructure);

GPIO_PinRemapConfig(GPIO_Remap_CAN2, ENABLE);
```

GPIO_Pin - Nastaví číslo pinu.

GPIO_Mode - Nastaví funkční mód pinu. Tyto módy můžou být:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

GPIO_Speed - Nám určuje rychlost, jakou bude pin pracovat. Tyto rychlosti mohou být 2, 10, 50 MHz.

8.3 Nastavení přerušení

Systém přerušení (NVIC) má definováno 68 maskovatelných přerušení. Tímto můžeme nastavit 16 stupňů priority. Přerušení může být externí událost EXTI. Tímto můžeme detekovat sestupnou nebo vzestupnou hranu signálu. Pro nastavení přerušení máme k dispozici knihovnu misc. V toto knihovně máme strukturu *NVIC_InitTypeDef* a k zapsání parametrů do struktury nám slouží funkce *NVIC_Init()*, která má jako parametr adresu struktury.

```
NVIC_InitStructure.NVIC_IRQChannel = CAN1_RX0_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
```

NVIC_IRQChannel – Určuje IRQ kanál, který chceme povolit nebo zakázat

NVIC_IRQChannelPreemptionPriority – Určuje přednostní prioritu pro kanál.

IRQChannelSubPriority – Určuje podružnou prioritu pro kanál.

NVIC_IRQChannelCmd – Povoluje přerušení.

8.4 CAN

8.4.1 Nastavení GPIO a CAN řadiče

Nejprve je nutno pustit hodinový signál do jednotek CAN1, CAN2, AFIO a portů B a D. Poté je potřeba nastavit vstupní a výstupní piny. Jako Rx použijeme pin 0 na portu D a pin 5 na portu B. Pro Tx použijeme pin 1 na portu D a pin 6 na portu B. Vstupní piny budou nastaveny na mód input-pull-up. Výstupní piny nastavíme na mód Alternate function push-pull o frekvenci 50 MHz. V závěru je potřeba provést přemapování pinů jednotky CAN.

Po nastavení pinů je potřeba nastavit CAN řadič. Pro toto nastavení máme opět připravenou knihovnu stm32f10x_can. V této knihovně nám pro nastavení CAN řadiče slouží struktura *CAN_InitTypeDef*, do které se zapíší jednotlivé parametry řadiče zavoláním funkce *CAN_Init()*, kde se předá v parametru adresa struktury a výběr jednotky CAN. Nyní následuje ukázka kódu pro nastavení CAN řadiče.

Tabulka 8.2 Prvky struktury pro nastavení CAN

Prvek struktury	**Nastavená hodnota**
CAN_TTCM	DISABLE
CAN_ABOM	DISABLE
CAN_AWUM	DISABLE
CAN_NART	DISABLE
CAN_RFLM	DISABLE
CAN_TXFP	ENABLE
CAN_Mode	CAN_Mode_Normal
CAN_SJW	CAN_SJW_1tq
CAN_BS1	CAN_BS1_3tq
CAN_BS2	CAN_BS2_2tq
CAN_Prescaler	60

CAN_TTCM Povoluje nebo zakazuje komunikaci v době spouštění.

CAN_ABOM Povoluje nebo zakazuje automatické vypínání řízení sběrnice.
CAN_AWUM Zapíná nebo vypíná automatické probouzení sběrnice.
CAN_NART Zapíná nebo vypíná automatické opakování vysílání zprávy, pokud se nepovede zprávu odvysílat.
CAN_RFLM Povoluje nebo zakazuje přijímání zpráv do FIFO v uzamčeném režimu.
CAN_TXFP Povoluje nebo zakazuje přenos FIFO priority.
CAN_Mode Určuje operační mód CAN. Tyto módy mohou být normal, loopback, silent a silent loopback.
CAN_SJW Určuje maximální počet časových kvant. Maximální počet může být 4.
CAN_BS1 Určuje počet kvant v bitové části 1. Počet kvant může být v rozmezí 1 – 16.
CAN_BS2 Určuje počet kvant v bitové části 2. Počet kvant může být v rozmezí 1 – 8.
CAN_Prescaler Určuje délku časového kvanta. Je v rozmezí 1 až 1024 a udává přenosovou rychlost.

Po nastavení CAN řadiče nám ještě zbývá nastavit CAN filtr. Pro ten opět máme v knihovně strukturu *CAN_FilterInitTypeDef*, do které se zapíší parametry filtru funkcí *CAN_FilterInit()*, která má parametr adresu struktury. V následující tabulce jsou jednotlivé prvky struktury pro nastavení CAN filtru.

Tabulka 8.3 Prvky struktury pro nastavení CAN filtru

Prvek struktury	**Nastavená hodnota**
CAN_FilterNumber	ENABLE
CAN_FilterMode	CAN_FilterMode_IdMask
CAN_FilterScale	CAN_FilterScale_32bit
CAN_FilterIdHigh	0x6420
CAN_FilterIdLow	0x0000
CAN_FilterMaskIdHigh	0x0000
CAN_FilterMaskIdLow	0x0000
CAN_FilterFIFOAssignment	0
CAN_FilterActivation	ENABLE

CAN_FilterNumber Určuje, který filtr bude použit. Filtry mohou být 1 až 13.

CAN_FilterMode Určuje režim filtru, který má být inicializován. Filtr může být Idlist, Idmask.
CAN_FilterScale Určuje měřítko filtru. Měřítko můžeme mít 16 bitů nebo 32 bitů.
CAN_FilterIdHigh Určuje prvních 16 bitů identifikačního čísla filtru.
CAN_FilterIdLow Určuje druhých 16 bitů identifikačního čísla filtru.
CAN_FilterMaskIdHigh Určuje masku filtru číslo nebo identifikační číslo, v závislosti na režimu**.** Prvních 16 bitů konfigurace. (MSB)
CAN_FilterMaskIdLow Určuje masku filtru číslo nebo identifikační číslo, v závislosti na režimu**.** Druhých 16 bitů konfigurace. (LSB)
CAN_FilterFIFOAssignment Určuje, která FIFO bude přiřazena k filtru (FIFO 0 nebo FIFO 1)
CAN_FilterActivation Zapíná nebo vypíná filtr.

8.4.2 Odeslání a přijetí zprávy

Pro přijetí a odesílání dat mámě dvě struktury. První je pro příjem *CanRxMSQ* a pro odeslání *CanTxMsg*. Pro odeslání zprávy zavoláme funkci *CAN_Transmit()*, kde v parametru jí předáme adresu na strukturu *CanTxMsg* a periferii CAN, která má zprávu odeslat. Příchozí data na CAN vyvolají přerušení a z přerušení se zavolá funkce *CAN_Receive()*, kde v parametru je periferie CAN, která data přijala, adresa na strukturu *CanRxMsg*, kam budou uložena data.

Prvky struktury pro vysílání dat:
CanTxMsg TxMessage;

- TxMessage.StdId - Identifikátor zprávy
- TxMessage.ExtId - Rozšířený identifikátor zprávy
- TxMessage.RTR - Data nebo žádost o data
- TxMessage.IDE - Standardní nebo rozšířený rámec
- TxMessage.DLC - Počet bytu v rámci
- TxMessage.Data[i] - Data připravena k vysílání

Prvky struktury pro příjem dat:

CanRxMsg RxMessage;

- RxMessage.StdId - Identifikátor zprávy
- RxMessage.ExtId - Rozšířený identifikátor zprávy
- RxMessage.IDE - Standardní nebo rozšířený rámec
- RxMessage.DLC - Počet bytů v rámci
- RxMessage.FMI - Určuje index filtru zprávy
- RxMessage.Data[i] - Přijatá data

8.5 Ethernet

8.5.1 Nastavení Ethernetového řadiče

Při konfiguraci Ethernetu je nejprve potřeba povolit na AHB sběrnici ETH_MAC jednotku, ETH_MAC_TX a ETH_MAC_RX kanál a pustit do nich hodinový signál. Dále je nutno nastavit piny MII rozhraní. Vstupní piny budou nastaveny na mód Input – flouting a výstupní budou v módu Alternate function push-pull. Všechny piny budou nastaveny na frekvenci 50 MHz. Pro piny MII rozhraní je potřeba ještě provést přemapování pinů.

Po základním nastavení pinů a jednotek mikrokontroléru je potřeba nakonfigurovat Ethernet řadič a DMA jednotku. Pro nastavení Ethernetu na mikrokontroléru máme knihovnu stm32_eth. V této knihovně je struktura *ETH_InitTypeDef*, která nakonfiguruje MAC a PHY jednotku. Po nakonfigurování MAC řadiče není potřeba programovat žádnou další komunikaci, protože zápis do fyzické vrstvy zajistí řadič. Nyní následuje ukázka kódu, který nastaví rozhraní jako MII a provede reset Ethernetového řadiče.

```
ETH_InitTypeDef ETH_InitStructure;
GPIO_ETH_MediaInterfaceConfig(GPIO_ETH_MediaInterface_MII);
RCC_MCOConfig(RCC_MCO_HSE);
ETH_DeInit();
ETH_SoftwareReset();
while (ETH_GetSoftwareResetStatus() == SET);
```

Pro základní nastavení řadiče naplníme strukturu základním nastavením příkazem *ETH_StructInit(&ETH_InitStructure),* který má v parametru adresu struktury. Po základním nastavení dojde v programu ke změně některých parametrů MAC vrstvy a dma kanálu. Poté zapíšeme data do struktury zavoláním funkce *ETH_Init(),* které v parametru předáme adresu struktury a adresu fyzické vrstvy. Poté povolíme přerušení od přijatého rámce příkazem *ETH_DMAITConfig(ETH_DMA_IT_NIS | ETH_DMA_IT_R, ENABLE)* a nastavíme jednotku NVIC ETH_IRQn s prioritou 1. Nakonec zavoláme funkci *ETH_Start(),* která aktivuje přenosový stavový automat a příjmový stavový automat. Poté bude nastaven přenos DMA do vyrovnávací paměti a zapne se přijímání a vysílání. Nyní se podíváme na strukturu pro MAC jednotku a DMA kanál. Jednotlivé prvky struktur jsou uvedeny v následujících tabulkách.

Tabulka 8.4 Prvky struktury pro nastavení MAC vrstvy

Prvek struktury	**Nastavená hodnota**
ETH_AutoNegotiation	ETH_AutoNegotiation_Enable
ETH_Speed	ETH_Speed_100M
ETH_ReceiveOwn	ETH_ReceiveOwn_Enable
ETH_LoopbackMode	ETH_LoopbackMode_Disable
ETH_Mode	ETH_Mode_HalfDuplex
ETH_RetryTransmission	ETH_RetryTransmission_Disable
ETH_AutomaticPadCRCStrip	ETH_AutomaticPadCRCStrip_Disable
ETH_DestinationAddrFilter	ETH_DestinationAddrFilter_Normal
ETH_ReceiveAll	ETH_ReceiveAll_Disable
ETH_SourceAddrFilter	ETH_SourceAddrFilter_Disable
ETH_BroadcastFramesReception	ETH_BroadcastFramesReception_Enable;
ETH_PromiscuousMode	ETH_PromiscuousMode_Disable
ETH_MulticastFramesFilter	ETH_MulticastFramesFilter_Perfect
ETH_UnicastFramesFilter	ETH_UnicastFramesFilter_Perfect

ETH_AutoNegotiation Zapíná nebo vypíná automatické nastavení rychlosti (10 nebo 100 Mbit/s).

ETH_Speed Nastavuje rychlost přenosu dat. Tento parametr je nepotřebný, pokud je aktivován Auto negotiation.

ETH_ReceiveOwn Dovolí příjem rámců, kdy je aktivní signál TX_EN

ETH_LoopbackMode Povolí nebo zakáže interní zpětnou smyčku

ETH_Mode Nastavuje mód přenosu half duplex nebo full duplex. Tento parametr je nepotřebný, pokud je aktivován Auto negotion

ETH_RetryTransmission Při režimu half duplex v případě kolize se pokusí opakovat přenos.

ETH_AutomaticPadCRCStrip Zapíná nebo vypíná automatické odstranění doplňku na minimální velikost rámce a CRC.

ETH_DestinationAddrFilter Výběr filtru pro unicast a multicast rámce.

ETH_ReceiveAll Zapíná nebo vypíná příjem všech platných rámců, nebo ty jejích cílová MAC adresa splňuje podmínky nastavené filtrem.

ETH_SourceAddrFilter Výběr filtru pro zdrojové adresy.

ETH_BroadcastFramesReception Povoluje nebo zakazuje příjem broadcast rámců.

ETH_PromiscuousMode Zapíná nebo vypíná promiskuitní mód.

ETH_MulticastFramesFilter Výběr multicastového filtru.

ETH_UnicastFramesFilter Výběr unicastového filtru

Tabulka 8.5 Nastavení DMA kanálu

Prvek struktury	**Nastavená hodnota**
ETH_DropTCPIPChecksumErrorFrame	ETH_DropTCPIPChecksumErrorFrame_Enable
ETH_ReceiveStoreForwar	ETH_ReceiveStoreForward_Enable
ETH_FlushReceivedFrame	ETH_FlushReceivedFrame_Disable
ETH_TransmitStoreForward	ETH_TransmitStoreForward_Enable
ETH_ForwardErrorFrames	ETH_ForwardErrorFrames_Disable
ETH_ForwardUndersizedGoodFrames	ETH_ForwardUndersizedGoodFrames_Disable
ETH_ReceiveThresholdControl	ETH_ReceiveThresholdControl_64Bytes
ETH_SecondFrameOperate	ETH_SecondFrameOperate_Enable
ETH_AddressAlignedBeats	ETH_AddressAlignedBeats_Enable
ETH_FixedBurst	ETH_FixedBurst_Enable
ETH_RxDMABurstLength	ETH_RxDMABurstLength_32Beat
ETH_TxDMABurstLength	ETH_TxDMABurstLength_32Beat
ETH_DescriptorSkipLength	0x0
ETH_DMAArbitration	ETH_DMAArbitration_RoundRobin_RxTx_2_1

ETH_DropTCPIPChecksumErrorFrame Povolí nebo zakáže shození TCP-IP paketu s chybou CRC.

ETH_ReceiveStoreForwar Povolí nebo zakáže uložení a předání přijatého rámce do DMA.

ETH_FlushReceivedFrame Povolí nebo zakáže vyprazdňování přijatých rámců.

ETH_TransmitStoreForward Povolí nebo zakáže uložení odeslaného rámce v DMA.

ETH_ForwardErrorFrames Povolí nebo zakáže přenos chybných rámců do DMA.

ETH_ForwardUndersizedGoodFrames Povolí nebo zakáže Rx FIFO, aby předala krátké rámce

ETH_SecondFrameOperate Povolí nebo zakáže obsloužit druhý rámec, pokud nebyl potvrzen status prvního rámce.
ETH_AddressAlignedBeats Povolí nebo zakáže zarovnání adres.
ETH_FixedBurst Povolí nebo zakáže AHB rozhraní pevných přenosů burst.
ETH_RxDMABurstLength Určuje maximální počet taktů, které budou předány do jedné Rx DMA transakce
ETH_TxDMABurstLength Určuje maximální počet taktů, které budou předány do jedné Tx DMA transakce
ETH_DescriptorSkipLength Určuje počet slov mezi dvěma deskriptory (kruhový režim)
ETH_DMAArbitration DMA Tx/Rx arbitr

8.5.2 Odeslání paketu

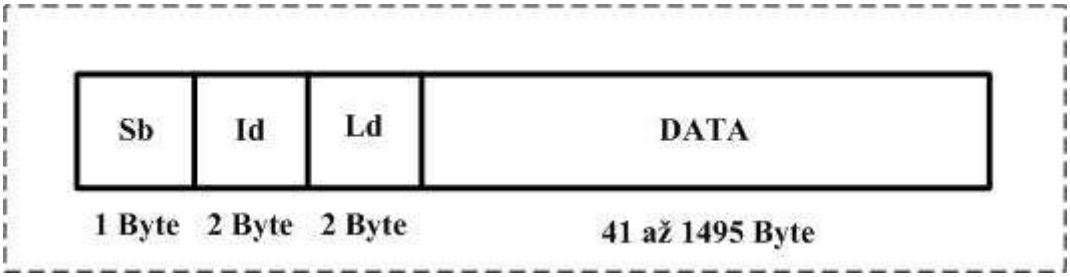
Pro odesílání paketu máme v knihovně STM32_Eth funkci ETH_*HandleTxPkt()*, které musíme předat v parametru ukazatel na data a délku dat. Tato funkce nastaví DNA kanál a odešle paket poté, co se uvolní linka. Před zavoláním této funkce musíme v programu sestavit kompletní rámec. K sestavení kompletního rámce nám slouží funkce *posli_eth()*. Tato funkce sestaví kompletní Ethernetový rámec a odešle pomocí funkce *ETH_HandleTxPkt()*.

8.5.3 Příjem paketu

Příjem paketů bude probíhat v obsluze přerušení *ETH_IRQHandler()*. Po vyvolání přerušení se zavolá funkce pro příjem paketu *ETH_HandleRxPkt()*. Této funkci předáme parametr adresu pole, kam má být přijatý rámec uložen a funkce vrací délku rámce. Přijatý rámec už neobsahuje preambuly a CRC součet, protože ty už zkontrolovala MAC jednotka a není potřeba je dále předávat. Po přijetí dat funkcí ETH_*HandleRxPkt()* se zavolá funkce *zapis_data_eth()*, která uloží přijatý data do bufferu a nastaví proměnou prcan do jedničky. Další zpracování dat už probíhá v těle programu.

9 Zhodnocení výsledků

Tento převodník bude fungovat jako převodník mezi CAN sběrnicí a ovládacím PC, které bude připojeno přes rozhraní Ethernet. Princip tohoto převodníku je, že převodník přijímá z Ethernetu jednotlivé rámce, které jsou ukládány do bufferu, který slouží k vyrovnání rychlostí mezi jednotlivými rozhraními. Pro odesílání dat do převodníku z Ethernetu jsem upravil pole dat Ethernetového rámce.



Obrázek 9.1 Rámec dat

Sb – Servisní byte, ve kterém se přenáší zaplnění bufferu převodníku.

Id – Zde se přenáší identifikátor zprávy pro sběrnici CAN

Ld – Zde je uložena délka přenášených dat.

Po přijetí rámce z Ethernetu převodník uloží data do bufferu. Buffer je realizován strukturou. Struktura má 3 prvky. První je identifikátor zprávy. Druhý je délka dat a třetí jsou vlastní data. Dále po každém přijatém rámci převodník kontroluje v zaplnění bufferu a posílá tuto informaci zpět do PC, který podle toho odesílá další rámce.

```
typedef struct
    {
    uint16_t    Id;
    uint16_t    delka;
    uint8_t     data[1495];
    }BUFFERETH;
```

Po přijetí a uložení dat do bufferu z Ethernetu převodník pošle data na sběrnici CAN. Odesílání probíhá tak, že z bufferu se vytáhne identifikátor zprávy a délka dat. Poté pod tímto identifikátorem dojde k odeslání všech dat, jejich počet určuje prvek délka. Toto se opakuje, dokud nebudou odeslány všechny přijaté rámce z bufferu.

Přenos data z CAN na Ethernet je podobný. Při přijetí z CAN je do bufferu uložen identifikátor zprávy a data. Délka dat je zvýšena o počet přijatých dat. Po přijetí všech dat ze sběrnice CAN program začne připravovat odeslání na Ethernet. Nejprve program vytvoří rámec tak, že prve se vloží cílová adresa pak zdrojová adresa a délka celého rámce. Poté dojde ke vložení informace o zaplnění bufferu. Nyní následuje vlastní vložení dat z bufferu. Prve se vloží identifikátor zprávy a poté délka dat v bufferu. A nakonec se vloží vlastní data a dojde vyslání rámce na sběrnici.

Aby bylo možné přijímat a vysílat data, musíme mít v PC obslužnou aplikaci. Tato aplikace se stará o vytváření, odeslání a příjem rámců. Dále je úkolem této aplikace kontrolovat zaplnění bufferu v převodníku podle zpráv, které převodník posílá a podle toho odesílat rámce. Tato aplikace byla napsána v objektovém jazyce C#. Pokud chceme přes aplikaci odeslat paket, vyplníme pole Id, data a ještě kolikrát chceme paket odeslat a stiskneme tlačítko „odeslat“. Zachycování paketů probíhá neustále, pokud přijde paket od našeho zařízení, aplikace zobrazí zaplnění bufferu, Id a přijatá data. Dále aplikace obsahuje tlačítko „vymaž“, které vymaže okna pro zobrazování přijatých dat. Pro fungování aplikace je nutné mít v počítači nainstalované ovladače WinPcap. Náhled aplikace je přidán do přílohy.

Přenosová rychlost převodníku bude záležet na přenosové rychlosti na sběrnici CAN. Další zpoždění v přenosu signálu nastane při zpracovávání dat v převodníku. To znamená při přijímání zprávy, kopírovaní dat do bufferu, kopírování dat z bufferu a odeslání.

10 Závěr

Cílem této práce bylo obvodové navržení zařízení převodníku Ethernet – CAN a naprogramování firmwaru procesoru pro převodník. Pro toto zařízení jsem si vybral mikrokontrolér STM32f105 od firmy STMicroelectronic. V první části této práce jsem se pokusil teoreticky rozebrat problematiku komunikačních rozhraní Ethernet a CAN. V další části této práce jsem se věnoval obvodovému návrhu a návrhu plošného spoje. Při návrhu plošného spoje jsem se rozhodl, že plošný spoj z důvodu vyšších přenosových rychlostí navrhnu čtyřvrstvý, kde vnitřní dvě vrstvy použiji pouze na napájení. Bohužel z časových důvodů už jsem nestihl tuto desku nechat vyrobit a osadit a proto jsem se rozhodl, že firmware pro mikrokontrolér odzkouším na vývojové desce STM3210C-EVAL.

Vzhledem k tomu, že se jedná o mé první seznámení s procesory s architekturou ARM, rozhodl jsem se plně využít knihovny od STMicroelectronic. Za pomocí těchto knihoven lze nastavit jednotlivé periferie procesoru, aniž bychom museli znát podrobně všechny registry procesoru. Popis nastavení použitých periferií jsem také uvedl v této práci.

Po nastavení periferií procesoru jsem začal řešit jakým způsobem přenášet data. Napsání vlastního Ethernetového stacku se ukázalo jako velmi časově náročná záležitost, proto jsem si připravil rámec, který budu vkládat do pole dat Ethernetového rámce a odesílat na broadcast adresu. Při odesílání dat na CAN jsem si zvolil, že budu data posílat pouze standardními rámci přenosovou rychlostí 100 kbit/s.

Firmware tohoto zařízení umí přeposílat datové zprávy ze sběrnice CAN na Ethernet a naopak. Dále má v sobě buffer, který vyrovnává rozdílné rychlosti obou sběrnic. V praxi by se toto zařízení nechalo využít například pro monitorování nebo řízení CAN sběrnice z PC. Pro funkčnost tohoto zařízení je nutno, abychom měli napsanou ovládací aplikaci do PC. Z časových důvodů moje ovládací aplikace do PC nemá všechny plánované funkce, a proto jsem ji zjednodušil pouze na odeslání a příjem rámce.

11 Použitá literatura

[1] HEROUT, Pavel. *Učebnice jazyka C*. 4 přepracované vydání. České Budějovice. Kopp, 2005. 271 s. ISBN 80-7232-220-6.

[2] NĚMEC, Karel. FEI VUT. *Datová komunikace*. Brno, 1999, 167 s.

[3] PROKEŠ, Aleš. FEI VUT. *Komunikační systémy*. Brno, 2002, 38 s.

[4] HERMAN, Ivo. FEI VUT. *Komunikační technologie*. Brno, 2002, 107 s.

[5] BIGELOW, Stephen J. *Mistrovství v počítačových sítích*: *správa, konfigurace, diagnostika a řešení problémů*. Vyd. 1. Překlad Petr Matějů. Brno: Computer Press, 2004, 990 s. ISBN 80-251-0178-9.

[6] FILKA, M. *Přenosová média*. Brno, 2002, 60 s.

[7] DOSTÁLEK, Libor. *Velký průvodce protokoly TCP/IP a systémem DNS*: *Vysvětlení nejpoužívanějších protokolů a konfigurace současných sítí. Internet i vnitřní podnikové sítě. Delegace domén, přidělování*. 2. aktual.vyd. Praha: Computer Press, 2000, 426 s. ISBN 80-7226-323-4.

[8] HÁJEK, Jakub. *Hardware pro komunikační gateway s STR912FW44*. Plzeň, 2008. Diplomová práce. FEL ZČU.

[9] KRIST, Petr. *Ethernet*, Západočeská univerzita v Plzni, 2010

[10] KRIST, Petr. *Protokoly vyšších vrstev*, Západočeská univerzita v Plzni, 2010

[11] KOSTURIK, Kamil, *CAN bus*, Západočeská univerzita v Plzni, 2010

[12] KOSTURIK Kamil, *Řídící a informační sběrnice*, Západočeská univerzita v Plzni, 2010

[13] ZAHÁLKA, Michal, *Převodník Ethernet – USB.* Plzeň, 2011. Diplomová práce. FEL ZČU.

[14] STMICROELECTRONIC. *STM32F105xx and STM32F107xx Errata sheet* [online]. 2011 [cit. 2012-02-15]. Dostupné z: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/ERRATA_SHEET/CD00238166.pdf

[15] TEXAS INSTRUMENTS. *Datasheet ISO7221* [online]. 2011 [cit. 2012-02-15]. Dostupné z: www.ti.com/lit/ds/symlink/iso7221a.pdf

[16] PULSE ELECTRONICS. *Datasheet Tab-DOWN RJ45* [online]. 2011 [cit. 2012-02-08]. Dostupné z: http://www.gme.cz/dokumentace/833/833-105/dsh.833-105.1.pdf

[17] STMICROELECTRONICS. *AN3102*: *Application note* [online]. 2009 [cit. 2012-04-08]. Dostupné z: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/APPLICATION_NOTE/CD00255062.pdf

[18] STMICROELECTRONICS. *RM0008*: *Reference manual* [online]. 2011 [cit. 2012-04-08]. Dostupné z: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/REFERENCE_MANUAL/CD00171190.pdf

[19] TEXAS INSTRUMENTS. *Datasheet SN65HVD1040* [online]. 2011 [cit. 2012-04-08]. Dostupné z: http://www.ti.com/lit/ds/symlink/sn65hvd1040.pdf

[20] STMICROELECTRONICS. *Datasheet STE100p* [online]. 2006 [cit. 2012-04-08]. Dostupné z: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00001918.pdf

[21] ZÁHLAVA, Vít, *Současné metody profesionálního návrhu plošných spojů*., ČVUT Praha, [online]. [cit. 2012-04-08]. Dostupné z: http://www.roznovskastredni.cz/dwnl/pel2005/03/zahlava.pdf

[22*]* Začínáme s STM32 VL Discovery 1-32. [online]. [cit. 2012-01-02]. Dostupné z: http://mcu.cz/news.php?extend.2769

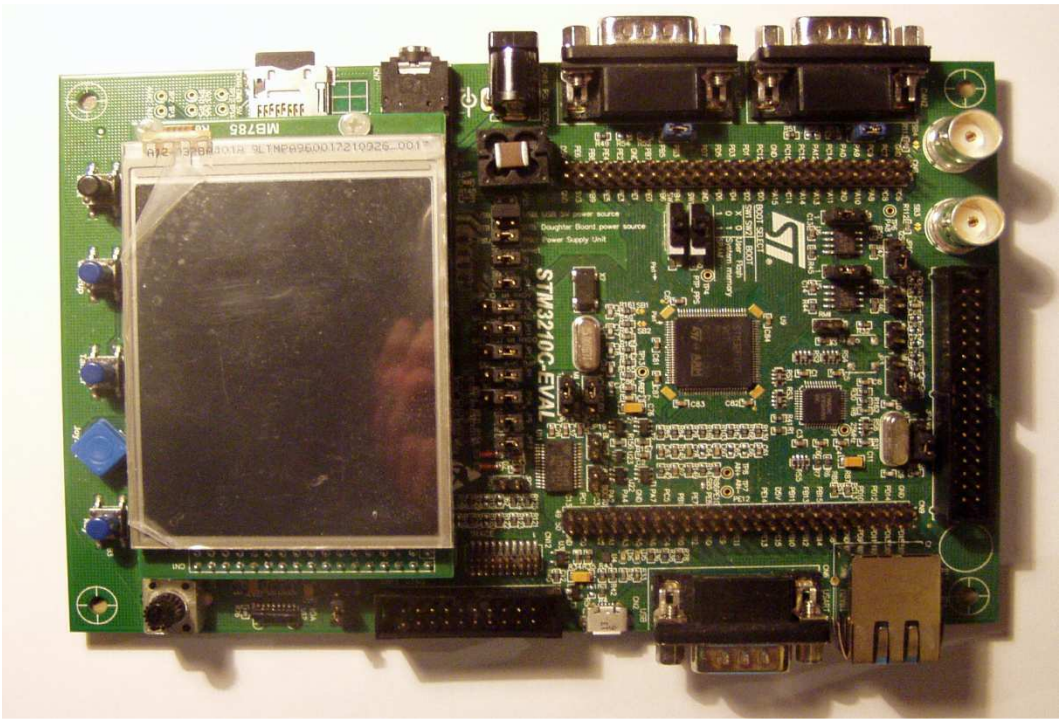
[23] RAISONANCE. *Ride7 for ARM*: *RAISONANCE Tools for the STRx and STM32 families* [online]. 2007 [cit. 2012-05-08]. Dostupné z: http://elmicro.com/files/raisonance/gettingstartedarm_ride7.pdf

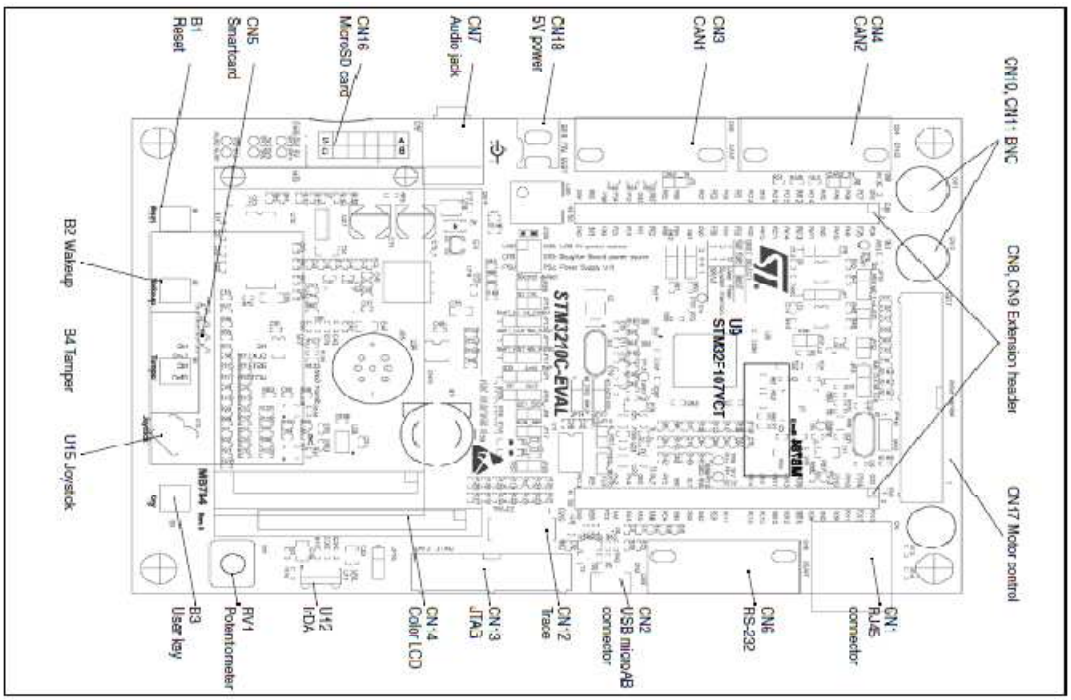
[23] STMICROELECTRONICS. *Datasheet SM6T6V8A* [online]. 1998 [cit. 2012-03-08]. Dostupné z: http://www.datasheetcatalog.org/datasheet/SGSThomsonMicroelectronics/mXtutyt.pdf

[24] STM32F107VC. STMICROELECTRONICS. *STMicroelectronics web* [online]. [cit. 2012-04-08]. Dostupné z: http://www.st.com/internet/mcu/product/221020.jsp

12 Příloha

Hardware STM3210C-EVAL





Obslužná aplikace do PC