ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

Katedra elektromechaniky a výkonové elektroniky

BAKALÁŘSKÁ PRÁCE

Návrh monitoru automobilových sběrnic

Jan Jendek **2017/2018**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan JENDEK**
Osobní číslo:	**E15B0012K**
Studijní program:	**B2644 Aplikovaná elektrotechnika**
Studijní obor:	**Aplikovaná elektrotechnika**
Název tématu:	**Návrh monitoru automobilových sběrnic**
Zadávající katedra:	**Katedra elektromechaniky a výkonové elektroniky**

Zásady pro vypracování:

1. Prostudujte sběrnice používané v automobilovém průmyslu.
2. Navrhněte monitor vybrané sběrnice, umožňující sledovat komunikaci a vysílat na sběrnici definované zprávy.
3. Využijte pro realizaci vývojový kit. V případě potřeby navrhněte rozšiřující hardware.
4. Vytvořte firmware, který půjde ovládat z PC např. pomocí sériového portu.

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

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

Vedoucí bakalářské práce:	**Ing. Kamil Kosturik, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Datum zadání bakalářské práce:	**10. října 2017**
Termín odevzdání bakalářské práce:	**7. června 2018**

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

L.S.

Prof. Ing. Václav Kůs, CSc.
vedoucí katedry

V Plzni dne 10. října 2017

Abstrakt

Předkládaná bakalářská práce se zabývá automobilovými sběrnicemi. V první části jsou vysvětleny základní principy komunikace jednotlivých sběrnic, jejich fyzické vrstvy a linkové vrstvy. Druhá část bakalářské práce se zaměřuje na praktickou ukázku komunikace skrze CAN sběrnici. Cílem je vytvořit zařízení pro posílání jednoduché zprávy na sběrnici a přijímání zprávy od jiné jednotky. V práci je popsán postup při návrhu hardware, na kterém je prováděno měření. Následně jsou vysvětleny úkony pro softwarové nastavení CAN komunikace, které je třeba splnit pro správnou funkčnost. V závěru je zhodnocení dosažených výsledků.

Klíčová slova

Sběrnice, FlexRay, CAN, LIN, registr, zpráva

Abstract

The bachelor thesis deals with automotive buses. The basic principles of communication of specific buses, their physical and line layers are explained in firts part. The second part of bachelor thesis is focused on practical demonstration of communication through the CAN bus. The goal is to create a device for sending a simple message to the bus and receive a message from another CAN unit. The design of hardware what was used for measurements is described step by step. Subsequently, the CAN communication software settings are explained. There is described what must be fulfilled for proper functionality. The results are evaluated in final part of bachelor thesis.

Key words

Bus, FlexRay, CAN, LIN, register, message

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 6.6.2018 Jan Jendek

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce doc. Ing. Kamilu Kosturikovi, Ph.D. za cenné profesionální rady, připomínky a metodické vedení práce.

Obsah

Seznam symbolů a zkratek

FLEXRAY	Sériová, deterministická datová sběrnice
CAN	Controller Area Network
STM32-Nucleo	Vývojová deska
LIN	Local Interconnect Network
CAN_TX	Vysílací pin budiče
CAN_RX	Přijímací pin budiče
CAN_H	Sběrnicový vodič s vysokou úrovní napětí
CAN_L	Sběrnicový vodič s nízkou úrovní napětí
MAC	Medium Acces Control
LLC	Logical Link Control
GPIO	General Purpose Input/ Output
SOF	Start Of Frame
RTR	Remote Frame
CRC	Cyclic Redundancy check
IDE	Identifier Extension Bit
UART	Universal Synchronous/Asynchronous Receiver and Transmitter
EOF	End Of Frame
TDMA	Time Division Multiple Access
FTDMA	Flexible Time Division Multiple Access
FPO	Floating Point Unit
SDRAM	Synchronous Dynamic Random Access Memory
SRAM	Static Random Access Memory
PSRAM	Pseudo Static Random Access Memory
USB	Universal Serial Bus
CMOS/TTL	Tranzistor Tranzistor Logic
SMD	Surface Mount Device

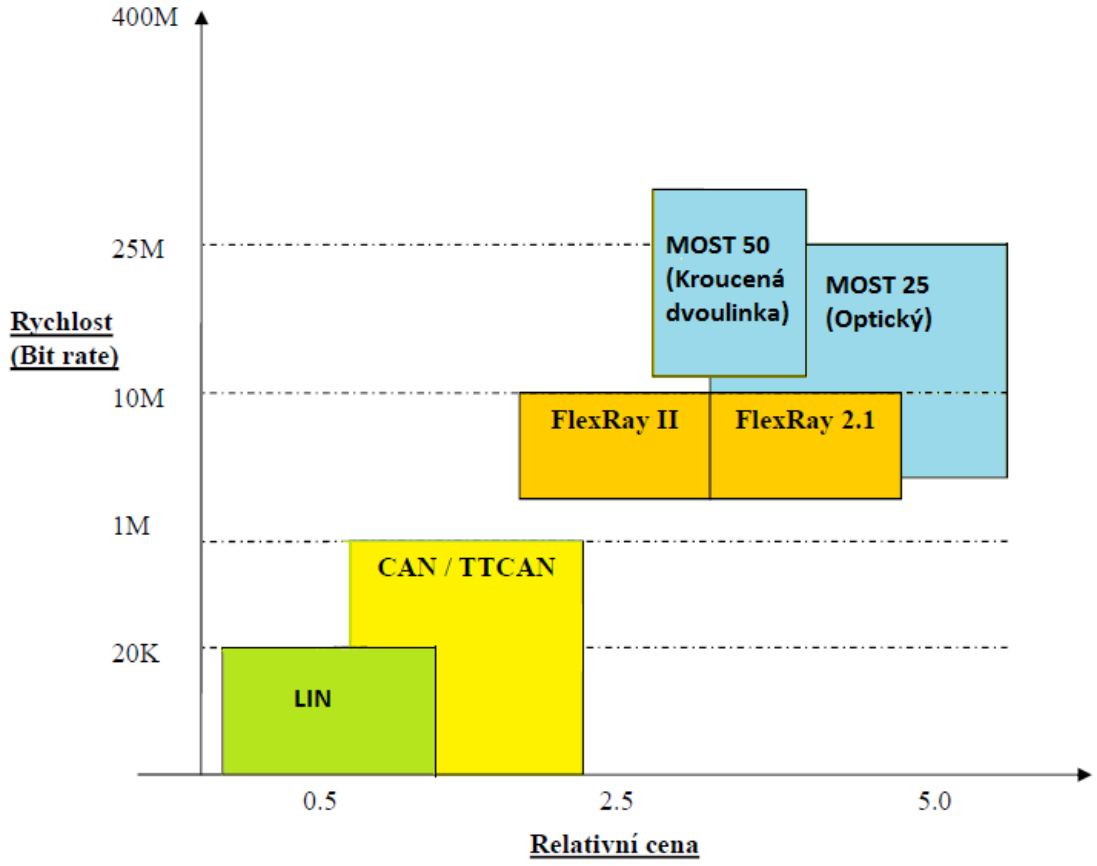
Úvod

Cílem této bakalářské práce je zpracovat přehled nejčastěji používaných automobilových sběrnic a provést návrh hardware a software schopného odesílat a přijímat jednoduché zprávy po CAN sběrnici. Mezi nejčastěji používané sběrnice v automobilovém průmyslu patří CAN, LIN a FlexRay a tyto sběrnice budou v bakalářské práci popsány detailně. Uvedené sběrnice jsou dnes nedílnou součástí všech automobilů díky možnosti přenosu vysokého objemu dat, odolnosti a spolehlivosti. V první části práce jsou vysvětleny základní principy funkčnosti sběrnic, jejich parametry a fyzické a linkové vrstvy těchto jednotlivých sběrnic.

Pro splnění zadání práce je nutné navrhnout hardware a software, který bude schopen monitorovat příchozí zprávy a odesílat předem nadefinované zprávy. Řešením je použít vývojovou desku od firmy STMicroelectronics a k této desce navhrnout plošný spoj pro komunikaci s ostatními zařízeními. K posílání a přijímání jednoduchých zpráv nestačí mít navržený pouze hardware, ale je nutné vyvinout i software pro řízení komunikace. Softwarovou část řeší další kapitola této bakalářské práce. Jedná se o aplikaci, která je schopná posílat definované zprávy po CAN sběrnici pro ostatní zařízení, ale zároveň je schopná také zprávy přijímat. Aplikace je napsána v programovacím jazyce C za použití knihoven od firmy STM. Sériový port nebo software CANoe poskytnutý firmou Vector lze použít pro sledování přijímaných a odesílaných dat.

1. Sběrnice v automobilovém průmyslu

S vývojem automobilového průmyslu bylo mnoho mechanických systémů nahrazeno elektronickými. Nové elektronické systémy umožňovaly zvýšit komfort, bezpečnost a pohodlí (ABS, elektrická okna, tempomat a mnoho dalších). Tato zařízení vyžadují vzájemnou komunikaci, možnost přenosu velkého množství dat a také zajištění bezpečného přenosu informace. Dříve bylo třeba pro propojení těchto elektronických systémů velké množství kabelů a konektorů, a to přinášelo nevýhody z hlediska vyšší hmotnosti auta a zvýšení výrobních nákladů. Tyto problémy lze řešit použitím sběrnicových systémů (CAN, LIN, FLEXRAY a dalších), které byly vyvinuty speciálně pro automobilový průmysl. Na Obr. 1.1 lze vidět porovnání rychlostí a ceny jednotlivých sběrnic.



Obr. 1.1 Porovnání jednotlivých sběrnic

1.1 CAN sběrnice (Control Area Network)

1.1.1 Historie sběrnice CAN

V 80. letech minulého století společnost Robert Bosch GmbH vyvinula sběrnici CAN (Controller Area Network) pro použití v automobilech. Na vývoji hardwaru spolupracovali vývojoví pracovníci společnosti Intel s techniky ze společnosti Mercedes-Benz, kteří měli být budoucí uživatelé. CAN protokol se poté stal velmi rychle standardem v automobilovém průmyslu. Díky dobrým vlastnostem se začala sběrnice CAN používat i v průmyslových aplikacích (komunikace řidících systémů, čidel a mnoho dalších). Standardizací vyšších vrstev komunikačního protokolu vznikly protokoly (CANopen, DeviceNet). V následujících letech se CAN protokol dále vyvíjel, až v roce 1991 firma Bosch vydala specifikaci CAN2.0. Tentýž rok firma Mercedes-Benz uvedla svůj první automobil se sběrnicí CAN. V roce 1993 byl přijat jako mezinárodní standard ISO11898. Tato norma popisuje fyzickou a linkovou vrstvu protokolu CAN. V následujících letech probíhala implementace u dalších automobilových výrobců. *V roce 1996 proběhlo první nasazení sběrnice i v automobilce VW a ŠKODA u vozů Passat a Octavia*.[1] V dnešní době na trh vstupuje nový CAN protokol s názvem CAN FD, který by měl dosahovat vyšších rychlostí a vyššího objemu přenesených dat.

1.1.2 Základní vlastnosti

Sběrnice CAN je sériová datová sběrnice propojující elektronické systémy a zařízení s vysokou mírou zabezpečení proti chybám. Její přenosová kapacita dosahuje až 1 Mbit/s. Protokol CAN je typu multi master, kde každý uzel sběrnice může být master a řídit tak chování jiných uzlů. *Není tedy nutné řídit celou síť z jednoho nadřazeného uzlu, což přináší zjednodušení řízení a zvyšuje spolehlivost (při poruše jednoho uzlu může zbytek sítě pracovat dál)*. [1] Komunikace po sběrnici probíhá mezi dvěma uzly pomocí datové a signalizační zprávy. Signalizaci chyb zajišťují speciální zprávy – zpráva o přetížení a chybová zpráva. Vysílané zprávy po sběrnici jsou přijímány všemi uzly připojenými na sběrnici, protože neobsahují žádnou informací o cílovém uzlu. Zprávy jsou uvozeny identifikátorem, který má délku 11 bitů (standardní formát) nebo 29 bitů (rozšířený formát). Identifikátor udává význam a prioritu přenášené zprávy. Nejnižší hodnota identifikátoru má největší prioritu a naopak. Zprávy se shodnou prioritou neexistují. V okamžiku, když je sběrnice volná a data jsou připravena k přenosu, začne každá stanice s odesíláním své zprávy. Díky bitovému posouzení

příslušného identifikátoru lze vyloučit konflikt, ke kterému by mohlo dojít při přístupu ke sběrnici. [1]

Čtyři oblasti použití sběrnice CAN v automobilech:

- Aplikace multiplexu

Užívá se k řízení komponent v oblast elektroniky komfortu a k regulaci komponent v oblasti karosérie (klimatizace, nastavování sedadel a centrální zamykání). Pro aplikaci multiplexu se používá nízkorychlostní sběrnice CAN Low-Speed s přenosovou rychlostí mezi 10 kBit/s a 125 kBit/s.

- Aplikace diagnostiky

Při kontrole pomocí diagnostiky je důležité, aby se síťové propojení mohlo použít k diagnostice všech propojených řídících jednotek. U Diagnostických aplikací jsou přenášeny velké objemy dat. Data jsou přenáše vysokorychlostní sběrnicí CAN High Speed s rychlostí 250 kBit/s a 500 KBit/s.

- Aplikace v oblasti mobilní komunikace

Tato oblast se zaměřuje na spojení multimediálních komponent pro automobil (telefon, audiosystémy, navigační systémy a dalších). Jedná se o sloučení procesů ovládání do jednotného celku, aby se snížilo rozptylování řidiče na nejnižší míru. Přenosová rychlost pro tyto aplikace je do 125 kBit/s.

- Aplikace reálného času

Prostřednictvím sběrnice CAN jsou navzájem propojeny důležité systémy (ESP, řízení převodovky, řízení motoru). Důležitá je doba odezvy pro tyto systémy, a proto je nutné používat vysokorychlostní sběrnici CAN High Speed s přenosovou rychlostí až do 1 MBit/s.

Pro realizaci řidící jednotky se sběrnicí CAN je zapotřebí několik obvodů:

- Mikrokontrolér

Obsluhuje události, dává pokyny pro vysílání zpráv a zpracovává přijatá data.

- Řadič CAN (CAN Controller)

Data obdržená od mikrokontroléru připravuje a předává na budič CAN. Dále realizuje datovou linkovou vrstvu protokolu CAN (rámce, arbitráž, chybové zabezpečení atd.).

- Budič CAN (CAN Transceiver)

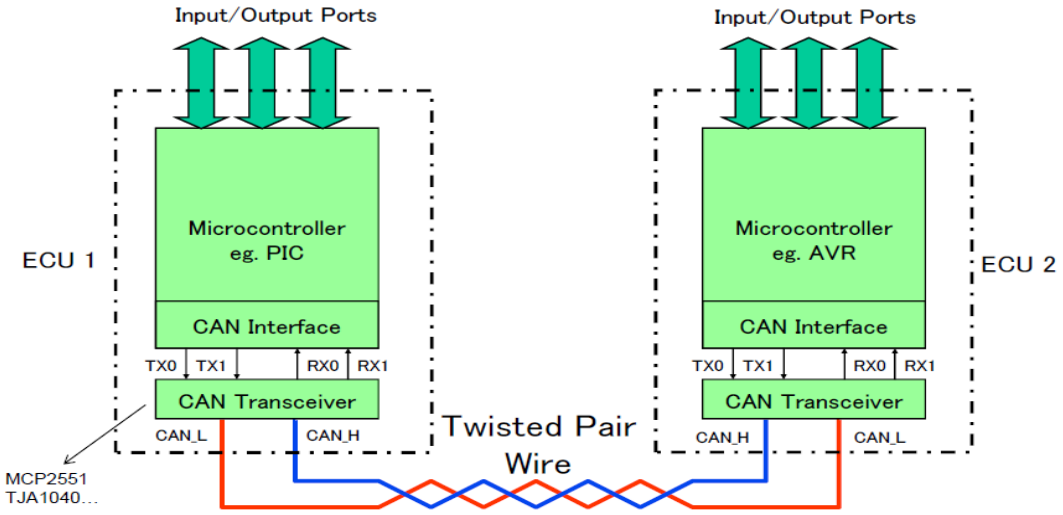
Mění data CAN řadiče na elektrické signály. Přijímá elektrické signály, které mění na data pro řadič. Realizuje fyzickou vrstvu protokolu CAN.

- Vedení datové sběrnice

Slouží k přenosu dat. Vedení datové sběrnice je provedeno pomocí kroucené dvoulinky pro zamezení rušení.

- Ukončení datové sběrnice

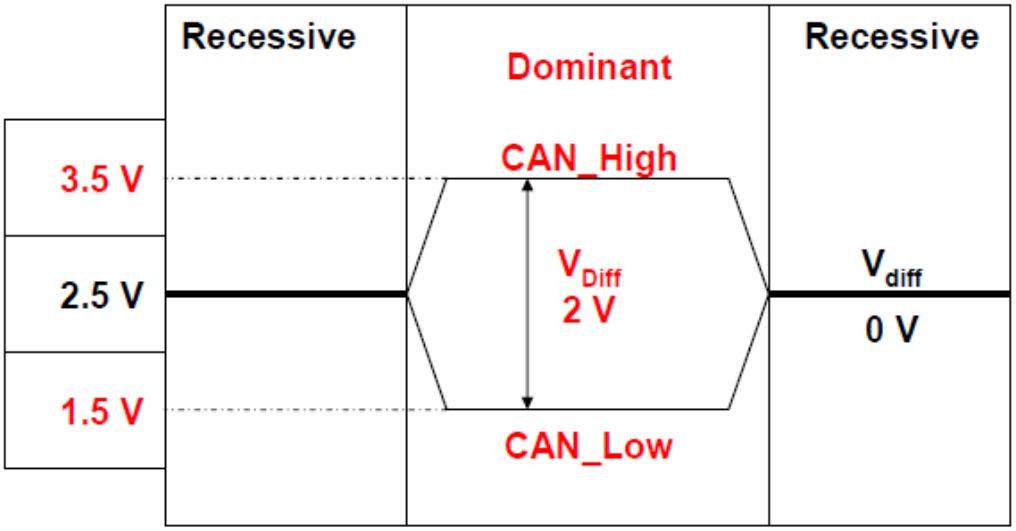
Ukončení datové sběrnice je provedeno pomocí rezistorů o velikosti 120 ohmů, které zabraňují odrazům elektrických signálů



Obr. 1.2 Realizace řídící jednotky se sběrnicí CAN

1.1.3 Fyzická vrstva protokolu CAN

Fyzická vrstva přenosového média je realizována dle normy ISO 11898 CAN standardu. Standard definuje principy časování, kódování bitů, synchronizace a dále elektrické vlastnosti přijímače a budiče. Signálové vodiče označované jako CAN_H a CAN_L tvoří sběrnici. Rozdílovým napětím vznikají na vodičích dvě úrovně označované jako dominantní nebo recesivní. Velikost rozdílového napětí dle normy je pro úroveň dominant“ v rozsahu 3,5V až 5V a pro úroveň „reccesive“ v rozsahu 0V až 1,5V. Napětí vodičů CAN_H a CAN_L je v recesivním stavu stejné (V_{diff} = 0V) a naopak v dominantním stavu rozdílné (V_{diff} = 2V) viz Obr 1.3. Lze tedy říci, že signálové vodiče CAN_H a CAN_L jsou vzájemně logicky invertované. [5]



Obr. 1.3 Napěťové úrovně sběrnice CAN

1.1.4 Linková vrstva protokolu CAN

Linková vrstva je také realizována dle normy ISO 11898 CAN standardu. Můžeme ji rozdělit na dvě podvrstvy MAC (Medium Access Control) a LLC (Logical Link Control). První uvedená je rozhraní mezi fyzickou vrstvou a LLC podvrstvou. Díky možnosti přístupu k médiu může MAC podvrstva provádět důležité úkony jako je kódování dat, vkládání doplňkových bitů, řízení priorit jednotlivých zpráv, potvrzování správně přijatých zpráv a detekci chyb. LLC podvrstva podporuje filtrování přijatých zpráv a hlášení o přetížení jednotlivých zpráv. [3]

1.1.5 Řešení řízení přístupu k médiu

Libovolný uzel může zahájit vysílání v okamžiku, kdy je sběrnice volná. V případě, že libovolný uzel zahájil vysílání jako první, zbylé uzly nemohou přistupovat ke sběrnici a je jim to umožněno až po vyslání kompletní zprávy prvním uzlem. Pokud dojde k detekci chyby v přenášené zprávě, ostatní uzly dostávají výjimku pro odeslání chybového rámce na sběrnici. Přístup ke sběrnici při vysílání více uzlů najednou je dán identifikátorem, který je uveden na začátku každé zprávy. Ten uzel, který posílá zprávu s vyšší prioritou má nižší identifikátor a naopak. Signály měnící se velmi rychle dostávají vyšší prioritu než signály, které se mění relativně pomalu. Velice důležitým článkem je vysílač, který zajišťuje, aby nedocházelo k poškození jednotlivých zpráv. Zjistí-li vysílač, že hodnota vyslaného bitu na sběrnici je jiná než sám vysílá, dojde k přerušení dalšího vysílání. Vysílače se zprávami nižší priority se

automaticky mění na přijímače a opakují svůj pokus o vysílání, jakmile je sběrnice opět volná. Přenos zprávy by se neměl zbytečně prodlužovat. Prodloužení může vzniknout v případě poškození zprávy, protože uzel poškozenou zprávu musí opakovaně odeslat na sběrnici. [1]

1.1.6 Zprávy protokolu CAN

Zprávy jsou přenášeny protokolem CAN sériově. Díky sériovému přenosu dat lze redukovat množství konektorů a vedení a tím dosáhnout vyšší spolehlivosti a úspory hmotnosti. Zprávy jsou posílány postupně a jsou k dispozici všem připojeným řídícím jednotkám ke sběrnici. *Řídicí jednotky mohou začít s vysíláním zpráv současně, pak pořadí vyslaných zpráv závisí na prioritě, která je definována v datovém protokolu*. [1] Každá zpráva má pevně daný identifikátor a to 11 bitový nebo 29 bitový. Identifikátor v podstatě vyjadřuje obsah, zprávy jako jsou např. otáčky motoru, otáčky kola, informace o teplotě oleje a další. Řídící jednotka vyhodnocuje data na základě seznamu akceptovatelných zpráv, v kterém je uložen příslušný identifikátor. Tato data jsou pak vyhodnocena a všechna ostatní jsou ignorována. Zprávy protokolu CAN lze rozdělit na čtyři typy. [3]

Datová zpráva je prvním typem zprávy v protokolu CAN. Zaručuje datovou komunikaci uzlů po sběrnici. Zařízení je schopno odeslat na sběrnicí zprávu dlouhou 0 až 8 datových bajtů. Není nutné pro jednoduché příkazy posílat žádná data. Je možnost pro tyto binární příkazy využít identifikátor zprávy a zkrátit potřebnou dobu přenosu zprávy a tím zajistit lepší propustnost sběrnice pří velkém zatížení.

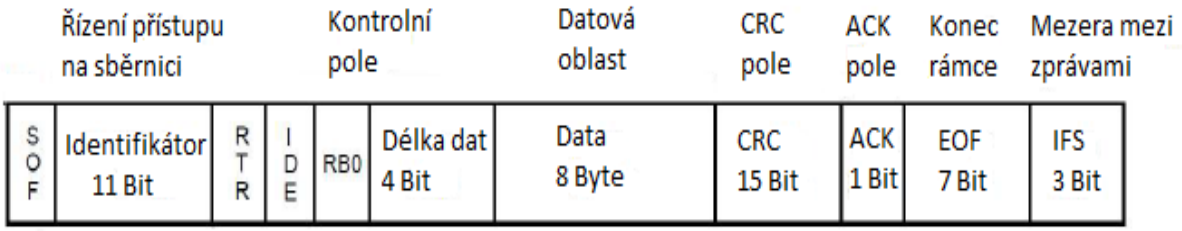
Zprávu na vyžádání dat (Remote Frame) vysílá uzel, který požaduje data. Odpovědí na požadavek jsou data odeslána uzlem, který je má k dispozici. Poslední dvě zprávy lze rozdělit na zprávu o přetížení a zprávu o chybě. Tyto zprávy slouží k eliminaci chybných zpráv a k detekování chyb.

Datové zprávy lze rozdělit na dva formáty. Standardní formát je definován specifikací 2.0 A a má délku identifikátoru 11 bitů. Rozšíření formát má identifikátor 29 bitů podle CAN specifikace 2.0 B. Oba formáty lze používat v jedné síti, protože jsou kompatibilní. Ze sedmi polí se skládá datová zpráva „Data Frame" viz Obr. 1.4. SOF je prvním polem o velikosti 1 bitu v datovém rámci. Sběrnice se chová v klidovém stavu jako recesivní. Začátek přenosu je proveden synchronizováním stanic a zahájením přenosu pomocí začátku zprávy (SOF – začátek

rámce) spolu s dominantním bitem. Aby nedošlo během přenosu ke ztrátě synchronizmu s odesílatelem, porovnává přijímač všechny recesivní a dominantní úrovně s předem nastaveným časováním bitů.

Další pole o velikosti 11 bitů je identifikátor zprávy zmíněný v textu výše. Součástí identifikátoru je 1 bitové rozhodovací pole RTR (Remote Transmission Request), které má za úkol řídit přístup ke sběrnici. Může nastat situace, kdy začne žádat o přístup na sběrnici uzel se shodným identifikátorem ve stejnou chvíli jako uzel, který data vysílá. Pokud nějaký uzel žádá o zaslání dat, nastaví takový identifikátor zprávy, jako má datová zpráva, jejíž zaslání požaduje. Tím je zajištěno, že pokud ve stejném okamžiku jeden uzel žádá o zaslání dat a jiný data se stejným identifikátorem vysílá, přednost v přístupu na sběrnici získá uzel vysílající datovou zprávu, neboť úroveň RTR bitu datové zprávy je dominant a tudíž má tato zpráva vyšší prioritu. [6] Aby bylo možné rozlišit, zda se jedná o standardní formát nebo rozšířený formát, je v datové zprávě obsaženo kontrolní pole, které obsahuje bit IDE (Identifier Extension Bit). Hodnota IDE bitu 1 znamená rozšířený formát, opačně se jedná o standardní formát. Data odesílaná v datové zprávě jsou obsažena v datovém poli o šířce 0 až 8 bajtů.

Při přenosu dat může dojít k jejímu porušení, a proto je nutné doručená data kontrolovat. K tomu slouží pole CRC (Cyclic Redundancy Check) ve kterém je obsaženo kontrolní slovo rámce. Pro příjemce je důležité, aby dostal potvrzení o správnosti přijatých dat. Pole ACK (Acknowledge) se posílá s recesivní úrovní a v případě správného přijetí příjemcem je přepsáno na dominantní logickou hodnotu. V případě poslání sedmi recesivních bitů se jedná o konec rámce EOF a to představuje konec sdělení. Po framu EOF následuje mezera mezi zprávami IFS, která se skládá ze tří bitů. [1]



Obr. 1.4 Datová zpráva

1.1.7 Rozpoznávání poruch v CAN protokolu

Bit Stuffing neboli vkládání bitů slouží k časové synchronizaci jednotlivých uzlů. V případě, že se na sběrnici vysílá pět bitů po sobě stejné úrovně, vkládá se do zprávy bit opačné úrovně. Toto opatření slouží také k detekci chyb. Funkce Code Check kontroluje obsah ve zprávě a zajistí, že při detekci chyby vkládání bitů je vygenerována chyba vkládání bitů. Pokud je chyba vygenerována, lze rozpoznat poruchu na vedení. Uzel vysílající data zavede po odvysílání pěti bitů se shodnou prioritou jeden bit opačné priority. Uzel přijímací data po přijetí tyto bity vymaže. Vyskytne-li se porucha na vedení, dojde k přerušení probíhajícího přenosu a stanice poté vyšle „Error Frame". Následující frame se skládá ze šesti dominantních bitů. Dojde k porušení pravidla pěti bitů a jednoho opačného a tím se zabrání, aby ostatní stanice přijímaly chybné sdělení.

CAN sběrnice je schopna rozlišit skutečné poruchy od nahodilých. Díky tomu nedojde k přerušení bezchybných sdělení. Kontrola zprávy „Message Frame Check" je jedním ze statistického vyhodnocení chybového stavu. Ve specifikaci je uveden formát zprávy, který musí být v pořádku. V případě detekování nepovolené hodnoty některého bitu se vygeneruje chyba rámce.

Dalším chybovým stavem může být například přetížení komunikace po sběrnici. K detekci slouží zpráva o přetížení „Overload Frame", která oddaluje žádosti o data nebo vysílání dalších datových zpráv. Zpráva o přetížení je charakteristická šesti dominantními bity a následně sedmi bity úrovně reccesive. Zařízení, která nejsou schopna zpracovávat další data kvůli svému vytížení, využívají tyto zprávy o přetížení.

Jedna z nejdůležitějších kontrol poruch je realizace kontrolního součtu CRC (Cyclic Redundancy Check). Jedná se o detekci chyb během ukládání nebo přenosu dat. Kontrolní součet má v datové zprávě vyhrazené své pole na konci zprávy o velikosti 15 bitů. Základním principem je porovnávání přijatých dat se sekvencí vypočtenou z přijatých dat. V případě shody kontrolních dat, k žádné chybě při přenosu nedošlo. V opačném případě došlo k chybnému přenosu a je vygenerována chyba CRC. Uvedený 15 bitový CRC kód, je generován podle polynomu viz vzorec 1.1. [1]

$$x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1 \tag{1.1}$$

2 LIN (Local Interconnect Network)

LIN (Local Interconnect Network) je jednovodičová asynchronní sběrnice. Byla navržena pro automobilový průmysl v roce 1998 skupinou LIN Consortium slouženou z pěti automobilek (Audi, BMW, Mercedes-Benz, Volkswagen, Volvo). Sběrnice LIN je navržena pro použití jednodušších zařízení (polohování sedadel, ovládání klimatizace, stahování oken a mnoho dalších), protože nedosahuje velké přenosové rychlosti jako sběrnice CAN. LIN sběrnice necílí na nahrazení CAN sběrnice, ale pouze jí doplňuje v aplikacích, ve kterých by bylo použití CAN sběrnice finančně neefektivní.

2.1 Základní vlastnosti

Sběrnice pracuje na principu single master /multiple slave. Jedná se o model komunikace, kde jedno zařízení nebo proces má kontrolu nad ostatními zařízeními. Standardem je doporučeno používat maximálně 16 jednotek na síť. Řízení přístupu na sběrnici není problém u LIN komunikace, protože master řídí celý provoz na síti. Je tvořena pouze 1 vodičem (1 wire network), díky tomu jsou sníženy náklady na realizaci vzhledem ke sběrnici CAN. Rychlost není tak vysoká právě kvůli zapojení s 1 vodičem a běžné používané přenosové hodnoty jsou běžně 2400 až 9600 bit/s. Maximální rychlost dosahuje 19200 bit/s. Požadavky na sběrnici LIN nejsou tak vysoké jako na sběrnici CAN. K funkčnosti sběrnice je dostačující běžný jednočipový mikropočítač, který obsahuje interface UART. Jednotky slave nepotřebují ke své činnosti přesný krystalový oscilátor, ale vystačí si s levnými RC oscilátory. Pro běžnou funkcionalitu je zapotřebí mikroprocesor zařizující vysílání a přijímání dat na sběrnici LIN. Mikroprocesor musí obsahovat rozhraní UART/SCI coby hardware řadič sériové komunikace. Dále je zapotřebí budič LIN (LIN transceiver) pracující na napěťové úrovni od 0 až 12 V dle normy. Fyzická vrstva protokolu LIN je realizovaná budičem. Převod signálu z TLL úrovně do fyzické vrstvy je zajištěn tímto budičem. [4]

2.2 LIN rámec (Frame)

LIN (LIN Message Frame) je složen ze dvou rámců. Jedná se o takzvanou hlavičku (Header frame) a odpověď (Response frame). Mezi těmito rámci vzniká variabilní mezirámcová mezera nazývaná „Interframe gap".

Header frame je složen ze tří část:

- SYNC-break (synchronizační pauza)
- SYNC-field (synchronizační pole)
- ID (identifikátor).

2.2.1 Synchronizační pauza

Synchronizační pauza je část zprávy určená ke spolehlivé detekci vyslaných zpráv na sběrnici. Je složena z minimálně 13 po sobě jdoucích nulových bitů. Při přijmutí synchronizační pauzy je zapotřebí zkontrolovat přítomnost nulových bitů z důvodu detekce této zprávy. Může zde vznikat problém s vygenerováním 13 nulových bitů pomocí mikrokontroléru s UART, kde je obvykle implementována jednotka master. Proto se tento problém řeší pomocí snížení přenosové rychlost sériové komunikace vysílače master. Čas potřebný pro vyslání 9 nulových bitů v sériové komunikaci je interpretrován ve slave vysílačích jako 13 nulových bitů. [4]

2.2.2 Synchronizační pole

Slouží k synchronizaci jednotek slave, které synchronizují své hodiny vždy před přijetím nové zprávy. Vše je zajištěno inicializací master jednotky pomocí vyslání hlavičkové zprávy. V zapojení master slave je vyžadována přesná synchronizace mezi jednotkami. Jako časová reference se využije přesný rezonátor v jednotce master, a tím dojde k ušetření finančních nákladů na citlivém rezonátoru nebo oscilátoru v jednotkách slave. Pro synchronizaci rychlosti jednotky slave se použije 5 sestupných a 5 náběžných hran. Odměřením času mezi první sestupnou hranou start bitu až k sestupné hraně 7 bitu synchronizačního bajtu je získána hodnota přenosové rychlosti a tuto hodnotu je nutno vydělit 8, aby byla zjištěna komunikační rychlost masteru. [1]

2.2.3 Identifikační pole

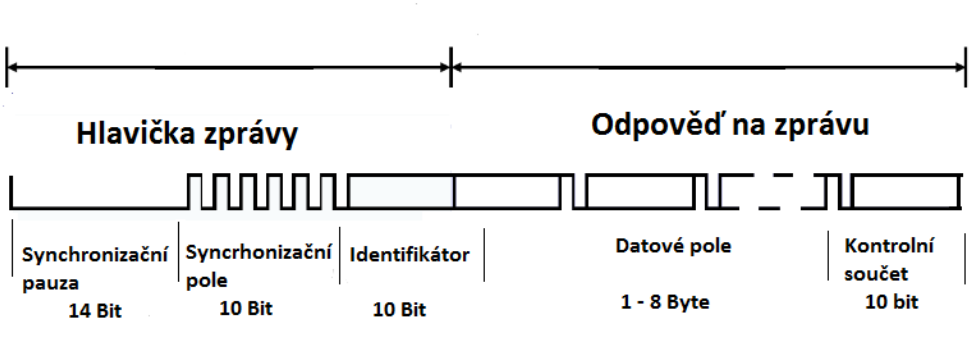
Je poslední částí hlavičkové zprávy. Je chráněno dvoubitovou paritou a značí následující datový rámec. Dva bity jsou určené ke specifikaci délky datového rámce, která může nabývat (2, 4 nebo 8 bajtů). V paměti jsou předem definovány platné ID, protože uzly slave nemají asociované fyzické adresy. Každé ID může mít rozdílný význam pro jiné uzly v síti.

2.2.4 Datový rámec (Response frame)

Následuje hned po hlavičce a je složen z dat o velikosti 0 až 8 bajtů viz Obr 2.1. Pro různé uzly může být různá délka přijatých dat. Po datovém rámci následuje kontrolní součet, který je 1 bajtový.

2.2.5 Sleep command (režim spánku)

Operace režim spánku se využívá ve chvíli, kdy na sběrnici dochází jen ke sporadické komunikaci a je dostačující částečná funkce LIN komunikace. Tuto operaci nemohou zařídit jednotky slave, ale k aktivaci může dojít pouze jednotkou master vysláním řídícího rámce s ID rovnýmu 0x3C s prvním data bajtem rovným 0x00 vyslaného všem jednotkám. Dojde k přepnutí do nízkopříkonového režimu všech slave jednotek až do příchodu budícího (wake up) signálu. Budící signál by měl trvat zhruba 0,25 až 5 ms a měl by nabývat dominantního stavu. Další možností je probuzení samotnou jednotkou slave. [1]



Obr. 2.1 LIN rámec

3 Datová sběrnice FlexRay

Sběrnice FlexRay vznikla v konsorciu firem Semiconductor, Freescale, BMW, Philips, Daimler, Chrysler a také firmou Robert Bosch GmbH. Vysokorychlostní sběrnice FlexRay by měla být využita pro aplikace vyžadující největší možnou provozní datovou rychlost, spolehlivost a vysoký stupeň zabezpečení (řízení podvozku, řídicí jednotky, ovládání brzdového systému ABS). Porovnání ceny a výkonosti sběrnice lze vidět na Obr. 1.1. Nejaktuálnější verze FlexRay protokolu je 3.0.1 vydaná v roce 2010 a stala se ISO standardem. Standard je nyní dostupný pod označením ISO 17458.

3.1 Základní vlastnosti

Moderní sériová vysokorychlostní sběrnice FlexRay je deterministická a odolná proti poruchám. Její přenosová rychlost dosahuje až 10 Mbit/s a stanice obsahuje dva nezávislé komunikační kanály A a B. Kanál B slouží jako záložní v případě selhání systému na kanálu A. Je možné používat tyto dva kanály současně a tím zvýšit přenosvou rychlost až na 20 MBit/s. Systém FlexRay je složen z několika dalších uzlů a prostřednictvým fyzického přenosového média propojuje všechny zbylé uzly. Sběnice může být založena na mnoho síťových topologiích mezi které patří například aktivní hvězda, kaskádová hvězda, dvoukanálová sběrnice nebo mix těchto zmíněných topologií. FlexRay je postavena na „Time-triggered communication" architektuře. Znamená to, že veškeré akce na sběrnici jsou statické a časově definované v tomto systému. Princip časové kontroly umožňuje deterministickou datovou komunikaci, a díky tomu je možná snadná implementace různých koncepcí na ní založené, jako je odolnost proti chybám, která je dosažena synchroním spouštěním akcí.

Pro implementaci "Time-triggered" komunikace se využíva metoda TDMA (Time Division Multiple Access). TDMA metoda nedovoluje nekontrolovatelně přistupovat ke sběrnici jako v případě CAN komunikace. FlexRay uzly odpovídají přesně nadefinovanému komunikačnímu plánu a ke každému cyklu se přidělí určitá časová pozice každé FlexRay zprávy a dojde k přířazení času pro odesílání těchto zpráv. [6]

3.2 Přístup ke sběrnici

Přístup ke sběrnici je řešen dvěma různými způsoby. FTDMA metodou (Flexible Time Division Multiple Acess) a TDMA metodou (Time Divison Multiple Acess).

- TDMA

Tato metoda je založena na předem definované komunikaci a je řešena statickými sloty určitě délky. Během komunikace po sběrnici FlexRay jednotky dostávají přístup na sběrnici dle předem daných plánů.

- FTDMA

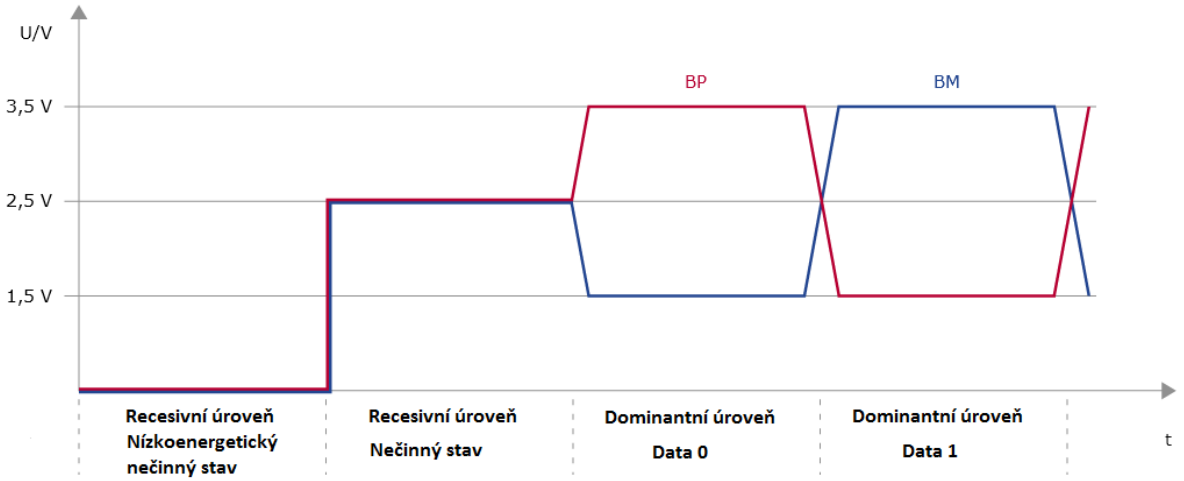
Je vhodná tam, kde TDMA metoda není ideálním řešením. To se může stát při sporadickém přenosu nebo při asynchronních procesech. Základem je dynamický segment pro zprávy, které nejsou posílány v předem daném intervalu, ale jsou posílány na základě událostí.

3.3 Fyzická vrstva

Fyzická vrtva u FlexRay komunikace je také založena na přenosu diferenciálních napětí. Přenos probíhá po nestíněné kroucené dvoulince BP (kladné) a BM (záporné). Velká přenosová rychlost by mohla způsobovat odrazy signálů na vedení, a proto je nutné řešit zakončení sběrnice pomocí ukončovacího terminátoru s impedancí mezi 80 až 110 Ω. Nabízí se možnost konce sběrnic ukončit pomocí takzvaného "split bus terminátoru", aby se nemusely všude implementovat zakončovací terminátory. Rozdělené sběrnice fungují poté jako filtr typu dolní propust, který nám odfiltruje vysokofrekvenční signály.

Fyzická vrstva definuje čtyři stavy na sběrnici viz Obr. 3.1. První je nízkoenergetický nečinný stav, který se objeví v případě přepnutí všech FlexRay vysílačů do nízko energetického režimu. Tento stav je charakteristický napětím 0 V. Nečinný stav je charakterizovaný napětím 2,5 V a diferenčím napětím 0 V. Napěťový rozsah pro nečinný stav je od 1,8 až 3,2 V. Následující stavy Data 0 a Data 1 jsou specifické rozdílovým napětím +2 V a -2 V. V případě Dat 0 je diferenční napětí +2 V a na pinu BP je napětí 3,5 V a na pinu BM je napětí 1,5 V.

V opačném případě bude diferenční napětí -2 V a hodnoty na BP a BM se prohodí. Dále je třeba zmínit, že dominantní stav je charakterizován diferenčním napětím 0 V a recesivní stav představuje napětí jiné než 0 V. [9]



Obr 3.1 Fyzická vrstva

3.4 FlexRay rámec

FlexRay datový rámec je složen ze tří částí:

- Hlavička (Header)

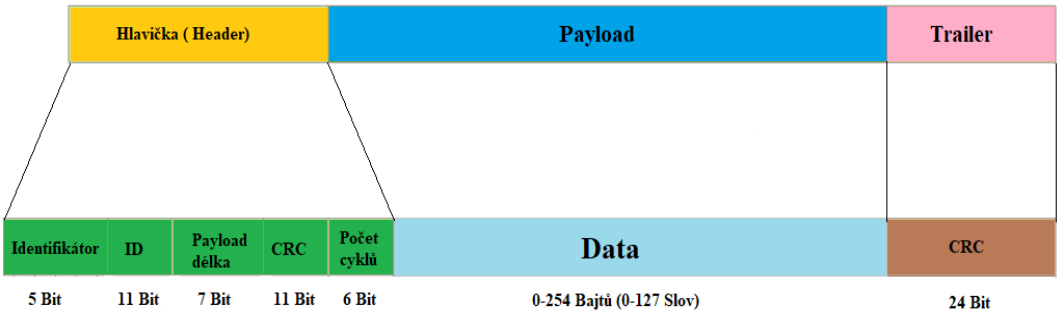
Hlavička se skládá ze 40 bitů. První část hlavičky je 5 bitová a obsahuje 1 bitové indikátory sloužící k přesné specifikaci posílané zprávy viz Obr 3.2. Identifikátor má délku 11 bitů a jediné nepoužitelné ID je 0x00, které se používá k identifikaci nevalidních zpráv. Další zajímou částí je 7 bitová payload délka obsažená v hlavičce. Ta udává přesnou délku následně přenášených dat, která mohou mít obsah až 254 bajtů. Následuje 11 bitová CRC sekvence vypočtená z identifikátoru, payload délky a je dána Flexray specifikací. Jako poslední část hlavičky je 6 bitový počet cyklů, pomocí kterého jsou zprávy posílány v definovaných cyklech.

- Payload

Jednou zprávou lze přenést až 254 uživatelských bajtů. Pro statické sloty je délka fixní, ale pro dynamické zprávy délka není fixní a payload se může pro různé hodnoty měnit.

- Trailer

Trailer obsahuje 24 bitovou CRC (Cyclic Redundancy Check) sekvenci. V tomto případě je CRC sekvence velmi výkonná. Počítá se z hlavičky a payload dat - výpočet probíhá na základě generování polynomu definovaného ve FlexRay specifikaci. [6]



Obr. 3.2 FlexRay Rámec

4 Praktická část

4.1 Vývojová deska Nucleo STM32F429ZI

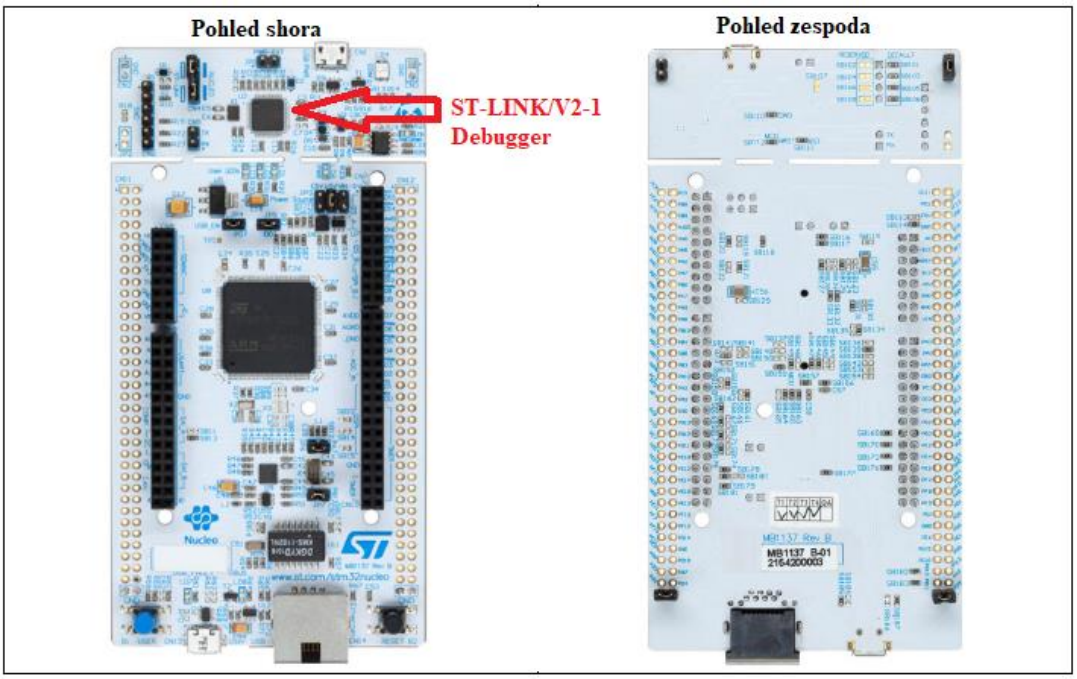
Úkolem bylo zprovoznit komunikaci mezi zařízeními prostřednictvím CAN sběrnice. Pro tento účel byl potřeba mikrokontrolér, který má HW i SW podporu CAN komunikace. Vhodnou volbou došlo na vývojový kit od firmy STMicroelectronics s názvem Nucleo STM32F429ZI. Velice výkonný 32 bitový procesor Arm Cortex M4 pracující na frekvenci až 180 Mhz je mozkem tohoto zařízení. Jádro Cortexu-M4 je vybaveno funkcí FPO (Floating Point Unit) umožňující vykonávat složitejší operace s čísly s pohyblivou řádovou čárkou. Další veliká výhoda kromě velké pracovní frekvence je adaptivní real time akcelerátor, díky kterému je možno přístupovat k flash paměti bez čekání. Flash paměť dosahuje až 2 MB velikosti a umožňuje číst data, zatímco se data zapisují. Tato paměť umožňuje flexibilní externí paměťové řadiče s až 32 bitovou datovou sběrnicí (SDRAM, SRAM, PSRAM a mnoho dalších). Pro některé použití aplikací dovoluje na I/0 (vstup, výstup) napětí od 1.7 V do 3.6 V. Podpora CAN protokolu byla využita pro zpracovávanou bakalářskou práci. Vývojová deska podporuje také Ethernet protokol a mnoho dalších, v dnešní době potřebných funkcí. Na Obr. 4.1 je zobrazena celá vývojová deska s ST morpho konektory. Na tyto konektory bude navržena deska umožnující komunikaci mezi Nucleo deskou a dalšimi CAN jednotkami.

4.1.1 Embedded ST-LINK/V2-1 debugger

Nucleo STM32F429ZI deska má ve své horní částí, jak je vidět na Obr 4.1 hardwarovou podporu pro debuggování. Při návrhu monitoru sběrnice byl využit softwarový nástroj pro hledání chyb a ladění samotného programu. Jedná se o nejnovější verzi tohoto debuggeru. Implementuje USB rozhraní pro snadnou komunikaci s PC, virtuálním COM portem, velkokapacitním paměťovým rozhraním a dále umožňuje lepší řízení spotřeby pro STM32 aplikace. [7]

4.1.2 Vývojové prostředí

Pro softwarovou část návhru bylo použito návrhové studio Atollic STUDIO for ARM od firmy STMicroelectronics. Je komerčně vylepšené pro C/C++ IDE, dále založené na otevřených zdrojových součástech s profesionálními rozšířenimi, funkcemi a nástroji. Prostředí je zdarma, ovšem pro pokročilejší funkce nabízí placenou verzi TrueSTUDIO Pro. Placená verze umožňuje sledování dat v reálném čase, vícejádrové debugování, vyčítání historie přístupu k paměti a mnoho dalších specialit.



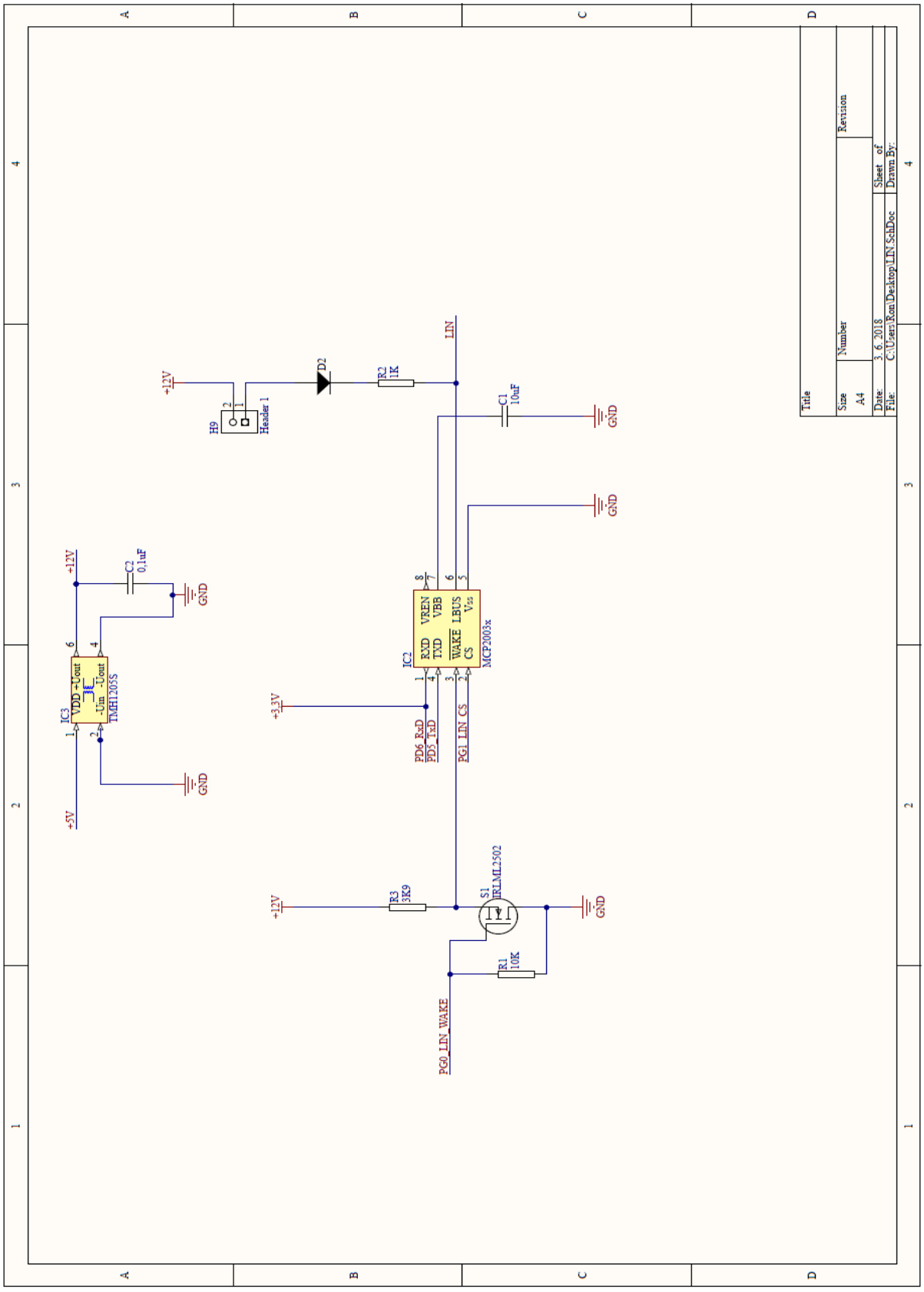
Obr. 4.1 Zobrazení vývojové desky

4.2 Návrh plošného spoje

Z důvodu použití vývojové desky od firmy STM, bylo nutné navrhnout a vyrobit plošný spoj, díky kterému bude možné komunikovat mezi vývojovou deskou a dalšími zařízeními. Pro návrh desky byl využit návrhový systém plošných spojů Altium Designer australské společnosti Altium Ltd. Tento velmi mocný návrhový systém nabízí mnoho funkčností, od návrhu schématu až po zprostředkování výstupních dat pro výrobu plošného spoje.

V prvé řadě, bylo třeba vybrat vhodné součástky, ze kterých by bylo možné obvod sestavit. Plošný spoj lze rozdělit na dvě části.

První část obsahuje zapojení pro budoucí použití LIN komunikace viz Obr 4.2. LIN budič byl zvolen MCP2003 od společnosti Microchip. Jedná se o součástku zaručující rozhraní mezi mikrokontrolérem a LIN sběrnicí. Má za úkol překládat logické úrovně CMOS/TTL na logickou úroveň LIN a naopak. Dosahuje nejvyšší přenosové rychlosti až 20 Kbaud. Dále obsahuje externí nebo interní ochrany proti zničení. Mezi ně lze zařadit externí ochranu proti přepólování a interní ochrany jako například ESD, tepelná ochrana. LIN budič lze provozovat v různých režimech. První z nich je "Power-down mode" Jedná se o mód se sníženou spotřebou, kde je vše vypnuto až na budící sekci. V našem zapojení na pinu 3 "WAKE" viz Obr.4.2 máme připojené dva klasické SMD rezistory R3 3K9Ω. Mosfet tranzistor, který nám umožňuje probouzet LIN komunikaci v případě režimu snížené spotřeby. Tranzistor je řízen z portu STM Nucleo vývojové desky a to konkrétně z pinu PG0, který je pracuje jako IO port. Na výstupním pinu 6 LIN budiče je připojená dioda D2 s rezistorem R2 1KΩ skrze jumper H9 na napětí 12V. Pokud chceme využívát budič jako master, necháme jumper zapojený. V opačném případě bude LIN budič pracovat v módu slave. MCP2003 má rozsah pracovního napájecího napětí 6 V až 27 V, proto jsme využili DC-DC měnič viz Obr 4.2. Tento stejnosměrný konvertor je schopen vytvořit námi potřebné napětí větší než 5 V. [10]

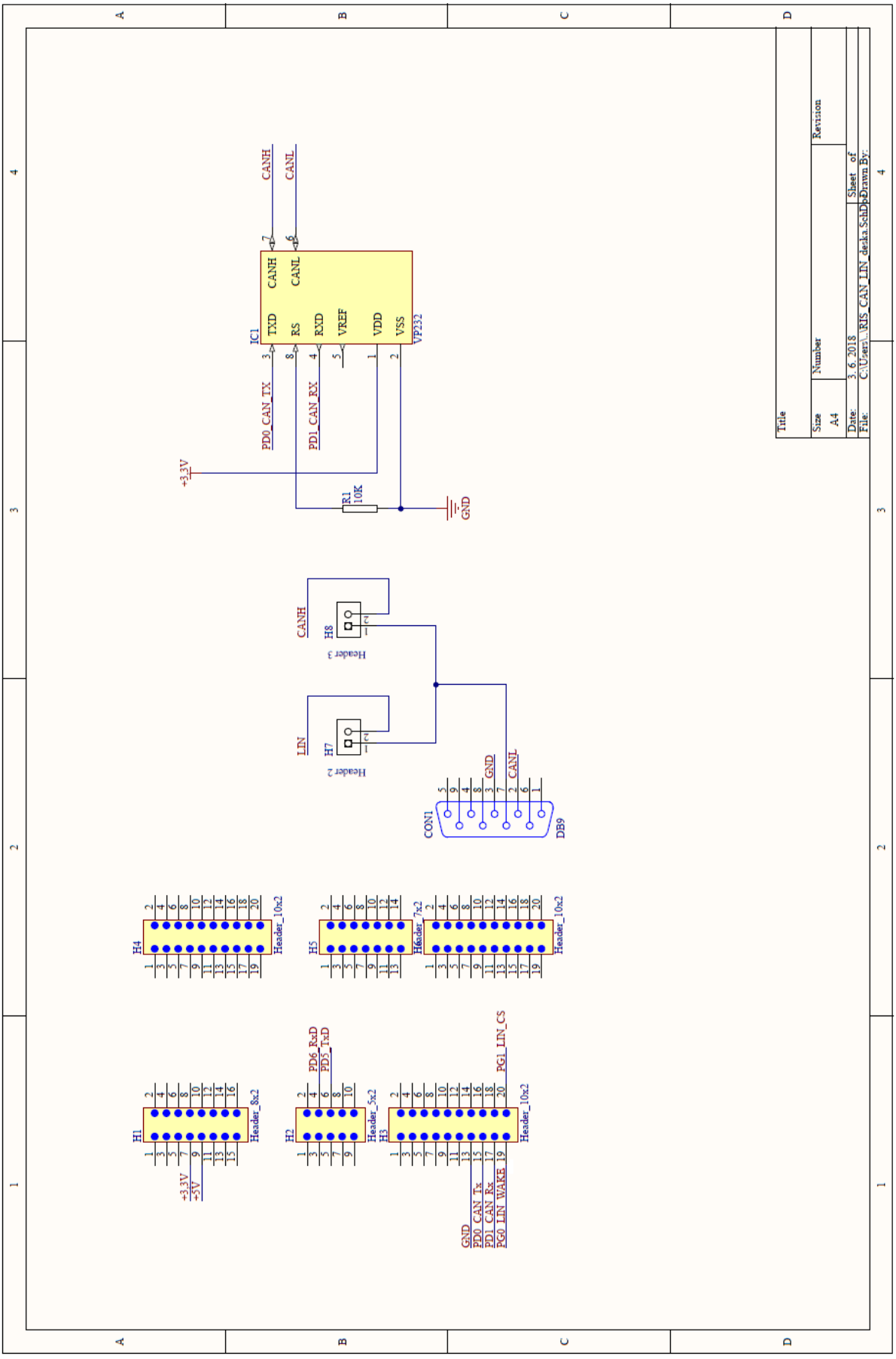


Obr. 4.2 LIN zapojení s budičem MCP2003

Druhá část obsahující stávající zapojení pro CAN komunikaci viz Obr 4.3. Zapojení CAN budiče vypadá v tomto případě jednodušší než u LIN. CAN budič byl použit SN65HVD23x označovaný také jako VP232 od firmy Texas Instruments. Budič komunikuje skrze fyzickou vrstvu dle ISO standardu 11898. Maximální komunikační rychlost dosahuje až 1Mbit/s. Budeme využívat čtvrtinu jeho maximální komunikační rychlosti, tedy 250 kBit/s. Je navržen pro provoz v obzvláště náročných podmínkách, a proto obsahuje ochranné prvky, jako jsou například ochrana proti přehřátí, ochrana proti přepětí. Vhodným zapojením lze dosáhnout ruzných provozních módů. Vstupní napájení 3.3 V je pro nás velice výhodné, protože můžeme využít na ST morpho kontektoru H1 pin 9. High-speed je prvním z provozních módů. Tento mód byl využit pro naši aplikaci. Nastavení probíhá pomocí připojení pinu 8 skrze R1 10K Ω na společnou zem. Tento odpor omezuje náběžné a sestupné hrany, protože je uměrný výstupnímu proudu. Tím získáme dobu přeběhu 15 V/ms. V případě vysoké logické úrovně na pinu 8 budič automaticky přejde do sleep modu do doby, než přijde nízká logická úroveň a tím se celý obvod probudí. Zapojené piny 3 PD0_CAN_TX a 4 PD1_CAN_RX slouží pro přenos dat mezi mikrokontrolérem a budičem. VDD pin 1 +3,3V slouží pro napájení budiče a VSS pin 1 vede na zem GND. Piny 7 CANH a 6 CANL na CAN budiči vedou do konektoru D-sub9, který slouží pro komunikaci s externí jednotkou. Zapojení tohoto konektoru lze vidět v Tab.1.1. [11]

Pin	Zapojení
1	Nepřipojen
2	CAN Low
3	GND
4	Nepřipojen
5	Nepřipojen
6	Nepřipojen
7	CAN High/LIN
8	Nepřipojen
9	Nepřipojen

Tab 1.1 Zapojení DB konektoru



Obr. 4.3 Zapojení CAN budiče a zapojení konektorů

4.3 Aplikace pro komunikaci skrze CAN sběrnici

Arm Cortex M4 podporuje základní rozšíření bxCAN pro CAN sběrnici. Je schopen zvládat vélké množství příchazejících zpráv, aniž by zatěžoval procesor. Podporuje obě verze CAN protokolu 2.0A i 2.0B. Aby bylo možné posílat a přijímat data po CAN sběrnici je zapotřebí nastavit příslušné registry jednotlivých periférií a správný bxCAN operační mód. BxCAN periférii a její komponenty lze vidět na Obr 4.11.

4.3.1 Inicializace periférií

K nakonfigurování zařízení bylo třeba nastavit více periférií než jen bxCAN. K zobrazení přicházejících dat bude využita sériová komunikace USART2 a dále indikační LED diody pro zobrazení, zda data byla odeslána nebo přijata. K fungování USART a CAN periférie je třeba nastavit hodiny. Nastavení probíha pomocí funkce: SystemClock_Config (void). V této funkci nastavujeme hodiny pro CPU, AHB a APB sběrnice. V našem případě je třeba nastavit hodiny pro sběrnici APB1 a AHB1, protože naše periférie jsou zde přivedeny. Pro nastavení parametrů pro hodiny a oscilátor využíváme TypeDef struktury s názvem RCC_OscInitStruct a RCC_ClkInitStruct. Externí oscilátor s frekvencí 8 MHz skrze HSE (High Speed Clock) se PPL násobičkou dostaneme na frekvenci 72 MHZ vnitřní sběrnice. Předděličkou se dostaneme na finální frekvenci APB1 sběrnice 36 MHZ, kterou budeme využívat pro nastavování CAN sběrnice.

```
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

    // Inicializace CPU, APB a AHB hodin pro sběrnice

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 72;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 3;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
    // Systick přerušení
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

    // nastavení systick
      HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);
```

Obr. 4.4 Nastavení hodin pro periférie

Pro správnou funkčnost bxCANu je třeba využít GPIO (General Purpose Input/Output) piny. Tyto piny nastavujeme příslušnými registry. Registry nám dovolují nastavit vice módů (vstupní, výstupní push pull, výstupní open drain, alternativní push pull nebo open drain). Využijeme inicializační funkci pro GPIO periferie: GPIO_Init(void). Z datasheetu víme, že CAN1_RX a CAN1_TX jsou zapojený na pinech PD 0 a PD 1. Nejdříve musíme nastavit hodiny pro tento port. Nastavení probíhá pomocí 32 bitového RCC_ AHB1ENR (Reset and Clock Control) registru, kde na pozici 3 příslušící našemu portu nastavíme bit GPIOD EN. Dále budeme nastavovat konkrétní registry našeho GPIOD portu. V prvé řadě nastavíme GPIOD_MODER registr určující, v jakém režimu bude fungovat náš port. Nastavíme ho do alternativního módu zapsáním hodnoty 10 neboli 0x02 v HEX na pozici 0 a 1, protože se odkazujeme na piny PD0 a PD1. Registr GPIOD_PUPDR nastavíme do 0, tedy do no pull down/pull up. Rychlost hrany nastavíme pomocí registru GPIOD_OSPEEDR na velmi vysokou rychlost, tedy zapíšeme na naše pozice 0 a 1 hodnoty 11 tedy 0x03 v HEX. V poslední řadě musíme dořešit nastavení alternativního módu. Z datasheetu víme, že pro CAN musíme nastavit alternativní mód 9. Toho docílíme 32 bitovými registry AFR. Tyto registry jsou dva a to AFRL

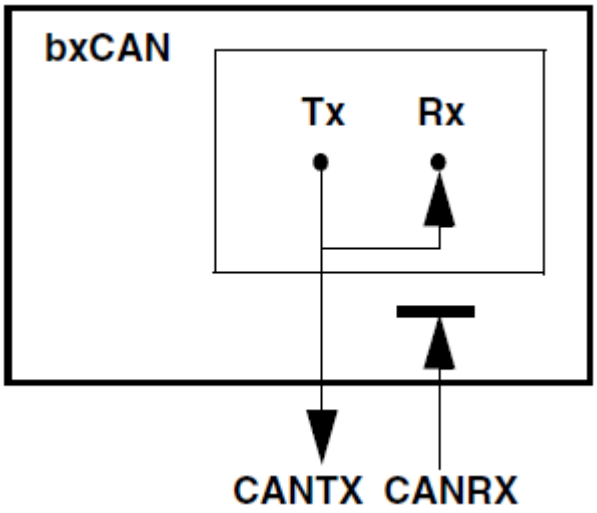
(platí pro AF 0-7) a AFRH (platí pro AF 8-15), v našem případě budeme využívat registr AFRH, protože potřebujeme alternativní mód 9. Zapíšeme tedy na pozici 0 a 1 1010 - 0x09 v HEX.

Pro indikaci využijeme LED diody umístěné na Nucleo vývojové desce. Jedná se o červenou a zelenou LED diodu. Podíváme se do vnitřního zapojení naší desky v datasheetu a zde vidíme, že pro červenou a zelenou LED diodu náleží pin PB 14 a pin PB 0. Jedná se tedy o port GPIOB. Inicializace LED diod provedeme pomocí funkce: LED_Init(void)**.** Nastavení probíhalo podobně jako při inicializaci GPIO pro CAN. Podmínkou: if (!(RCC-> AHB1ENR & RCC_AHB1ENR_GPIOBEN)) kontrolujeme, zda nejsou aktivní hodiny pro GPIOB. Pokud nejsou aktivní hodiny, aktivujeme je pomocí RCC_AHB1ENR. Dále provedeme reset puls periférií pomocí RCC_AHB1RSTR registru. Nastavení MODER regitru bude v tomto případě do režimu output, tedy 0x01 na pozici 14 a 0**.** OTYPER registr dostaneme do výstupu push-pull, tedy do defaultního stavu. Náběžné hrany budou opět nastaveny na velmi vysokou rychlost.

Inicializace sériové komunikace probíhá pomocí funcke: USART2_Init (void). U sériové komunikace musíme opět nastavit alternativní GPIO porty. Inicializace proběhla již ve funkci: GPIO_Init(void)**.** Použijí se stejné registry jako v GPIO nastavení CANu s tím rozdílem, že alternativní mód bude AF 7. [8]

4.3.2 Operační módy bxCAN

Hlavní módy podporované bxCAN jsou celkem 3. První podporovaný je takzvaný inicializační mód. Do inicializačního módu je možné se dostat nastavením příslušných bitů v CAN registrech. Jedná se o registry CAN_MCR (CAN master control register) a CAN_MSR (CAN master status register). Tyto registry mají velikost 32 bitů**.** K nastavení inicializačního módu musíme v CAN_MCR registru nastavit INRQ bit, který se nachází na nulté pozici v registru. Pokud dojde v CAN_MSR registru k nastavení INAK bitu, je potvrzeno nastaven nicializačního módu. Toto potvrzení probíhá automaticky a je řešeno na úrovni hardware vstvy.



Obr 4.5 Zapojení Loop back modu

Důležité bylo v prvé řadě otestovat komunikaci v takzvaném "Loop back modu" Jedná se o speciální testovací mód sloužící k automatickému otestování funkčnosti bxCANu. Princip spočívá v ignoraci vstupního CANRX pinu a provede se vnitřní zpětná vazba z výstupního pinu Tx na vstupní pin Rx. Testování pomocí tohoto módu proběhlo z důvodu ověření funkčnosti komunikace bez další připojené CAN jednotky.

Pro vzájemnou komunikaci dvou a více CAN jednotek je musíme přejít z Loop back módu do normálního. K nastavení může dojít po úspěšné inicializaci bxCANu. V registru CAN_MCR provedeme vymazání INRQ bitu, který byl nastaven v inicializačním módu. Tím se přepneme do normálního módu a dojde k synchronizaci s přenosem dat na sběrnici CAN. Potvzení o nastavení je opět řešeno hardwarovou vrstvou a dojde k vymazání INAK bitu v CAN_MSR registru. Sleep mode neboli režim spánku je poslední z podporovaných módů. Nastavení probíhá pomocí SLEEP bitu v CAN_MCR registrech. Tento režim nebude v naší práci využit, ovšem z praktického hlediska se hojně využívá ve všech jednotkách v automobilovém průmyslu. Důvod je ten, že se sníží spotřeba elektrické energie, avšak software může stále přistupovat k datům na sběrnici. [9]

4.3.3 Inicializace CAN

Zavoláním funkce: CAN_Init() provedeme kompletní inicializaci periférie a přípravu na vysílání a přijímání zpráv. Skrze vytvořené funkce byl nastaven normální mód pro použitý CAN, protože LOOPBACK mód jsme využili pouze na otestování.

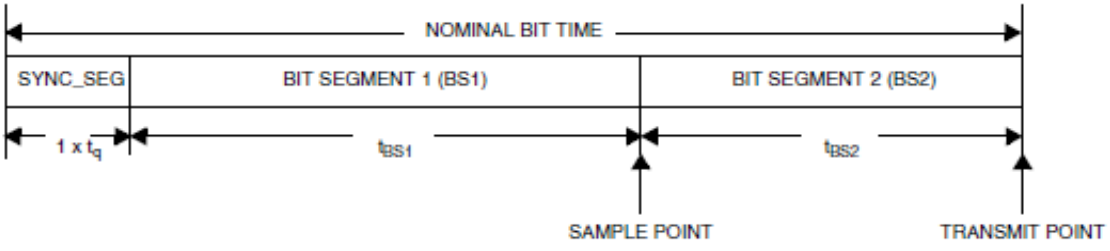
Přepnutí do inicializačního módu řeší funkce: HAL_CAN_Init(&can), která má návratovou hodnotu. Tato funkce nastavuje v registrech potřebné bity, jak již bylo zmíněno v kapitole operační módy bxCAN. Pokud inicializace proběhla v pořádku, funkce by měla vrátit hodnotu HAL_OK, v opačném případě vrátí HAL_ERROR. Funkce je ošetřena podmínkou pro případ vrácení špatné návratové hodnoty, tím se zavolá jiná funkce Error_Handler zajišťující chybový stav. Nesmíme zapomenout na nakonfigurování akceptačních filtrů. Jedná se o jednu z nejdůležitějších vlastností řadiče protokolu CAN umožňující schopnost filtrovat zprávy, které jsou potřebné od těch, které jsou nepotřebné. Bez akceptačních filtrů by nebylo možné přijímat zprávy. Nepotřebné zprávy jsou odfiltrovány a jsou propuštěny jen ty potřebné. Akceptační filtry mohou být nastaveny pouze v inicializačním módu.

Nastavení parametrů probíhá pomocí struktury: CAN_Filter_TypeDef, kterou jsme si přiřadili k názvu: nastaveníFiltru. Filtr jsme nenastavovali na konkrétní ID, protože v našem případě nechceme filtrovat žádné zprávy. Funkce: HAL_CAN_ConfigFilters (&can, &nastavenifiltru) má dva vstupní parametry a také návratovou hodnotu jako v případě CAN initu. V této funkci neprobíhá nic jiného než přiřazení parametru z naší struktury k jednotlivým registrům a poté přepsání potřebného bitu pro nastavení. Funkce je opět ošetřena podmínkou pro případ chybového stavu. Používané registry pro filtry jsou 32 bitové. V registru CAN_FM1R se nastavuje jaký mód použít pro identifikátor. V nastavení máme uvedeno IDMASK to přísluší v registru zapsání 0 na všechny pozice kromě 4 bitů, které jsou rezervovány, a není možné je použít. Postup pro další registry je podobný.

```
static void CAN_Init(void)
{
    CAN_FilterTypeDef nastaveniFiltru;

      can.Instance = CAN1;
      can.Init.Mode = CAN_MODE_NORMAL; // jaký mód máme použít (NORMAL nebo LOOPBACK)
      //can.Init.Mode = CAN_LOOPBACK;

      // nastavení časování CAN
      can.Init.Prescaler = 8;
      can.Init.SyncJumpWidth = CAN_SJW_1TQ;
      can.Init.TimeSeg1 = CAN_BS1_12TQ;
      can.Init.TimeSeg2 = CAN_BS2_5TQ;

      can.Init.TimeTriggeredMode = DISABLE;
      can.Init.AutoBusOff = DISABLE;
      can.Init.AutoWakeUp = DISABLE;
      can.Init.AutoRetransmission = ENABLE;
      can.Init.ReceiveFifoLocked = DISABLE;
      can.Init.TransmitFifoPriority = DISABLE;

      // vstup do inicializačního módu
      if (HAL_CAN_Init(&can) != HAL_OK)
      {
        _Error_Handler(__FILE__, __LINE__);
      }
      // nastavení akceptačních filtrů
        nastaveniFiltru.FilterBank = 0;
        nastaveniFiltru.FilterMode = CAN_FILTERMODE_IDMASK;
        nastaveniFiltru.FilterScale = CAN_FILTERSCALE_32BIT;
        nastaveniFiltru.FilterIdHigh = 0x0000;
        nastaveniFiltru.FilterIdLow = 0x0000;
        nastaveniFiltru.FilterMaskIdHigh = 0x0000;
        nastaveniFiltru.FilterMaskIdLow = 0x0000;
        nastaveniFiltru.FilterFIFOAssignment = CAN_RX_FIFO0;
        nastaveniFiltru.FilterActivation = ENABLE;
        nastaveniFiltru.SlaveStartFilterBank = 14;
```

Obr 4.6 Inicializační funkce CAN_Init

Je důležité v inicializaci provést nastavení bitového časování CAN komunikace. Logika časování bitů umožňuje sledovat sériovou komunikaci. Je důležité provést toto nastavení, abychom mohli zjistit přenosovou rychlost. Logika má za úkol provést vzorkování, upravit toto vzorkování na hranu začínajícího bitu a resynchronizovat na následujících hranách. Tato operace nám rozdělí bit na tři části (SYNC_SEG) synchronizační segment, (BS1) bitový segment 1, (BS2) bitový segment 2. [9]



Obr. 4.7. Časování CAN komunikace

$$Přenosová\ rychlost = \frac{1}{NominamBitTime} \quad (4.1)$$

$$NominalBitTime = 1 \times t_q \times t_{BS1} \times t_{BS2} \quad (4.2)$$

$$t_{BS1} = t_q \times (TS1[3:0] + 1) \quad (4.3)$$

$$t_{BS2} = t_q \times (TS2[2:0] + 1) \quad (4.4)$$

$$t_Q = t_{PCLK} \times (BRP[9:0] + 1) \quad (4.5)$$

t_{BS1} – bitový segment 1

t_{BS2} – bitový segment 2

t_Q - synchornizační segment

t_{PCLK} – časová perioda hodin APB sběrnice

BRP [9:0], TS1[3:0] a TS2[2:0] jsou definovány v CAN_BTR registru. Pro výpočet naší přenosové rychlosti byla použita tabulka na Obr. 4.8.

								fPCL	36	MHz	-> tPCLK	0,02778	µs							
								CAN_BTR			tq	tBS1	tBS2	CAN-Bit		Samplepoint	NominalBitTime	Baudrate
CAN_Prescalar=	CAN_BS1=			CAN_BS2=				BRP	TS1	TS2	[µs]	[µs]	[µs]			[%]	[µs]	[kbit/s]
2	CAN_BS1_	12	tq	CAN_BS2_	5	tq		1	11	4	0,05556	0,66667	0,27778	19	tq	72,22	1	1000,00
4	CAN_BS1_	12	tq	CAN_BS2_	5	tq		3	11	4	0,11111	1,33333	0,55556	21	tq	72,22	2	500,00
8	CAN_BS1_	12	tq	CAN_BS2_	5	tq		7	11	4	0,22222	2,66667	1,11111	25	tq	72,22	4	250,00
16	CAN_BS1_	12	tq	CAN_BS2_	5	tq		15	11	4	0,44444	5,33333	2,22222	33	tq	72,22	8	125,00
20	CAN_BS1_	12	tq	CAN_BS2_	5	tq		19	11	4	0,55556	6,66667	2,77778	37	tq	72,22	10	100,00
40	CAN_BS1_	12	tq	CAN_BS2_	5	tq		39	11	4	1,11111	13,3333	5,55556	57	tq	72,22	20	50,00
80	CAN_BS1_	12	tq	CAN_BS2_	5	tq		79	11	4	2,22222	26,6667	11,1111	97	tq	72,22	40	25,00

Obr 4.8 tabulka pro výpočet přenosové rychlosti

Pro zahájení vysílání a přijímání bylo třeba přejít z inicializačního módu do normálního. Výstup z inicializace provedeme funkcí: HAL_CAN_Start(&can). Provede se vyčištění INRQ bitu v CAN_MCR registru. Musíme čekat na potvrzení, že přepnutí proběhlo. Docílíme toho sledováním INAK bitu v CAN_MSR registru, jakmile se INAK bit změní na hodnotu 0,

přepnutí proběhlo úspěšně a funkce nám vrací parametr HAL_OK. Je zde vnořený časový limit pro případ, že k přepnutí po určitém časovém intervalu nedojde a vyhodnotíme chybový stav.

4.3.4 Přenos a příjem po CAN sběrnici

Pro vysílání a přijímání dat je třeba splnit některé úkony. V prvé řadě budeme řešit posílání dat. Interface mezi softwarem a hardwarem pro CAN zprávy je řešen implementací takzvaných maliboxů. Tyto mailboxy jsou v našem bxCANu celkem tři. Mailbox obsahuje všechny informace ohledně zprávy (data, status, identifikátor, délka a další). K poslání zprávy je třeba mít prázdný mailbox. Informace o tom, zda je mailbox prázdný či není lze vyčíst z registru CAN_TSR, který obsahuje infromace o stavu přenosu.

```
// Nastavení zprávy pro přenos
 TxHeader.StdId = 0x00;
 TxHeader.RTR = CAN_RTR_DATA;
 TxHeader.IDE = CAN_ID_STD;
 TxHeader.DLC = 8;
 TxData[0] = 0x00;
 TxData[1] = 0x00;
 TxData[2] = 0x00;
 TxData[3] = 0x00;
 TxData[4] = 0x00;
 TxData[5] = 0x00;
 TxData[6] = 0x00;
 TxData[7] = 0x00;
```

Obr. 4.9 Nastavení zprávy pro přenos

Před tím, než začneme posílat zprávu je třeba si jí předem připravit. Přírava probíhá na konci funkce CAN_Init, která je ukázana na Obr 4.7. Je nutné nastavit vysoký počet parametrů. V první řádce nastavujeme hodnotu identifikátoru v HEX hodnotě. O jaký typ identifikátoru se jedná (standardní nebo rozšířený) se nastavuje ve třetí řádce pod parametrem IDE. Pro naše účely byl použit standardní identifikátor. Dalším parametrem je RTR označující, zda se jedná o datový rámec nebo o požadavek na data. Proměnná CAN_RTR_DATA znamená použití datového rámce. Specifikovali jsme si délku rámce DLC 8. Následující řádky obsahují konkrétní data, které budeme posílat na sběrnici. Data jsme inicializovali na hodnotu 0x00.V

jiné částí programu, v nekonečné smyčce jsme k datům přiřadili generátor náhodných čísel a tyto čísla převádíme do šestnáctkové soustavy. Docílíme tím generování náhodných dat.

V nekonečné smyčce voláme funkci: HAL_CAN_AddTxMessage (&can, TxHeader, TxData, &TxMailbox) pro vyslání dat na sběrnici. Funkce má tři vstupní parametry a má též návratovou hodnotu. V první částí kontroluje, zda jsou všechny tři mailboxy prázdné. Kontrola probíha pomocí CAN_TSR registru, kde využíváme nastaveného bitu TME0, TME1 aTME2. Pokud jsou bity nastavené, není zde čekající požadavek na posílání dat pro konkrétní mailbox.

Následuje vybrání jednoho z dostupných prázdných mailboxů. Vybraný mailbox je třeba uložit a dochází ke konkrétnímu nastavení zprávy pro poslání na sběrnici. Toto nastavení se bere z parametrů, které jsme si nastavili v inicializační funkci pro CAN. V 32 bitovém registru CAN_TIxR nastavením IDE bitu do 0 získáme standardní identifikátor v opačném případě rozšířený. Budeme používat datový rámec, takže RTR bit nastavíme do 0. Pro poslání dat na sběrnici musíme naplnit dva registry CAN_TDLR a CAN_TDHR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

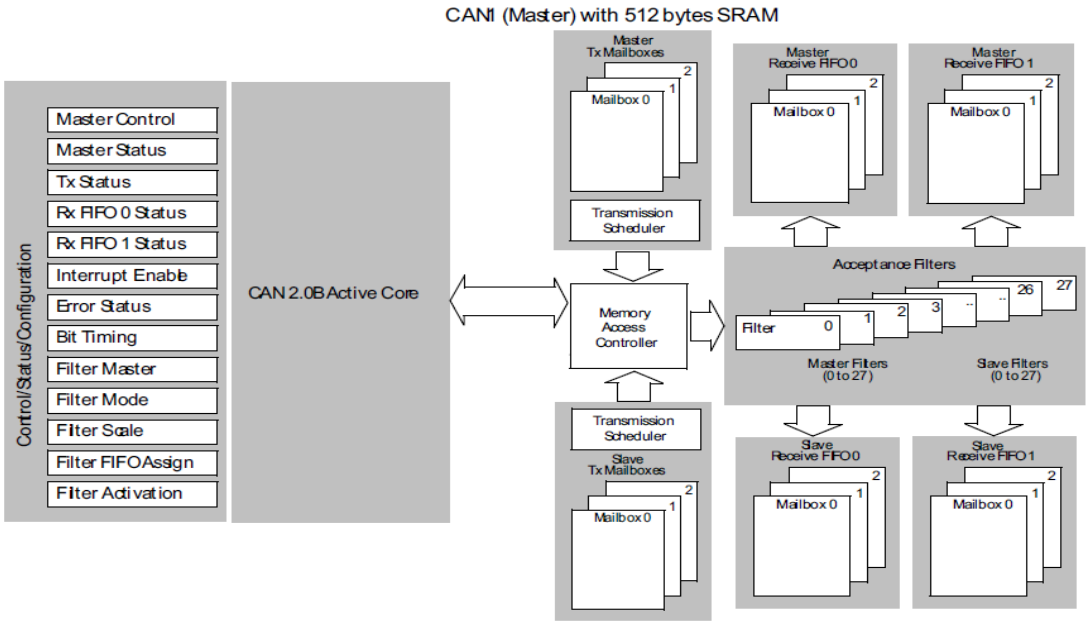
Obr. 4.10. Registru CAN_TDLR

První zmíněný registr slouží pro naplnění Txdat [0] až [3], druhý pro Txdata [4] až [7]. Po naplnění všech dat do registrů musíme poslat požadavek na vyslání zprávy na sběrnici. Tento požadavek proběhne nastavením TXRQ bitu v registru CAN_TIxR. Jakmile se zpráva odešle a vyčístí se náš mailbox, hardware automaticky tento bit vyčistí a vrátí na původní hodnotu. Naše funkce vrátí HAL_OK v případě, že vše proběhlo v pořádku.

Pro přijímání dat máme dva výstupní mailboxy nazývané FIFO. Jsou prázdné do té doby, než přijde platná zprává. FIFO má tři stavy, které nastávají při přijmu zprávy. První nastane při příjmu první zprávy a nastaví se stav “pending 1”. Dochází k zapsání hodnoty do FMP [1:0] = 01 v CAN_RFR registru. Zpráva se stane dostupnou ve výstupním FIFO mailboxu. Je nutné zprávu uvolnit pomocí RFOM bitu v CAN_RFR registru. Nedojde-li k uvolnění,

následující validní zpráva se uloží do FIFO pod stavem “pending 2” s hodnotou FMP [1:0] = 10. Ukládací proces se opakuje do té doby, než přijde další zpráva dostávající FIFO stav do “pending 3” s hodntou FMP [1:0] = 11. Software musí uvolnit mailbox pro další zprávu, jinak dojde k nenávratné ztrátě zprávy. [12]

Funkcí: HAL_CAN_GetRxMessage (can, CAN_RX_FIFO0, &RxHeader, RxData) řešíme příjímání dat ze sběrnice. Provedeme kontrolu obsahu FIFO mailboxů. Není-li jeden z mailboxů prázdný, můžeme začít s vyčítáním konkrétních dat z přijaté zprávy. Vyčítání dat probíha pomocí CAN registrů pro příjem zpráv. Pro získání identifikátoru a požadavku na přenos použijeme CAN_RIxR registr, kde vyčteme IDE a RTR bit. V případě IDE hodnoty 0 se jedná o standardní identifikátor, v opačném případě o rozšířený. Pro RTR hodnota 0 znamená požadavek o data. Délka dat DLC je vyčítána z CAN_RDTxR registru, hodnotu můžeme získat 1 až 8, protože maximální dělka dat u sběrnice CAN je 8. Pokud získáme hodnotu 0, jedná se požadavek o data, který neobsahuje datovou část. Dále vyčteme 16 bitový časovač TIME [15:0] udávající zachycenou hodnotu, kdy došlo k detekci SOF (začátku rámce). Následuje vyčtení konkrétních dat. Data se vyčítají skrze podobné registry jako v případě vysílání źpráv na sběrnici. Konkrétně se jedná o registry CAN_RDLxR a CAN_RDHxR. První uvedený obsahuje data [0] až [3] a druhý [4] až [7]. Po přiřazení dat do našeho RxData pole musí dojít na uvolnění FIFO mailboxu. Nastavením příslušného bitu v registru RFM uvolníme zprávu a jsme připraveni na příjem další.



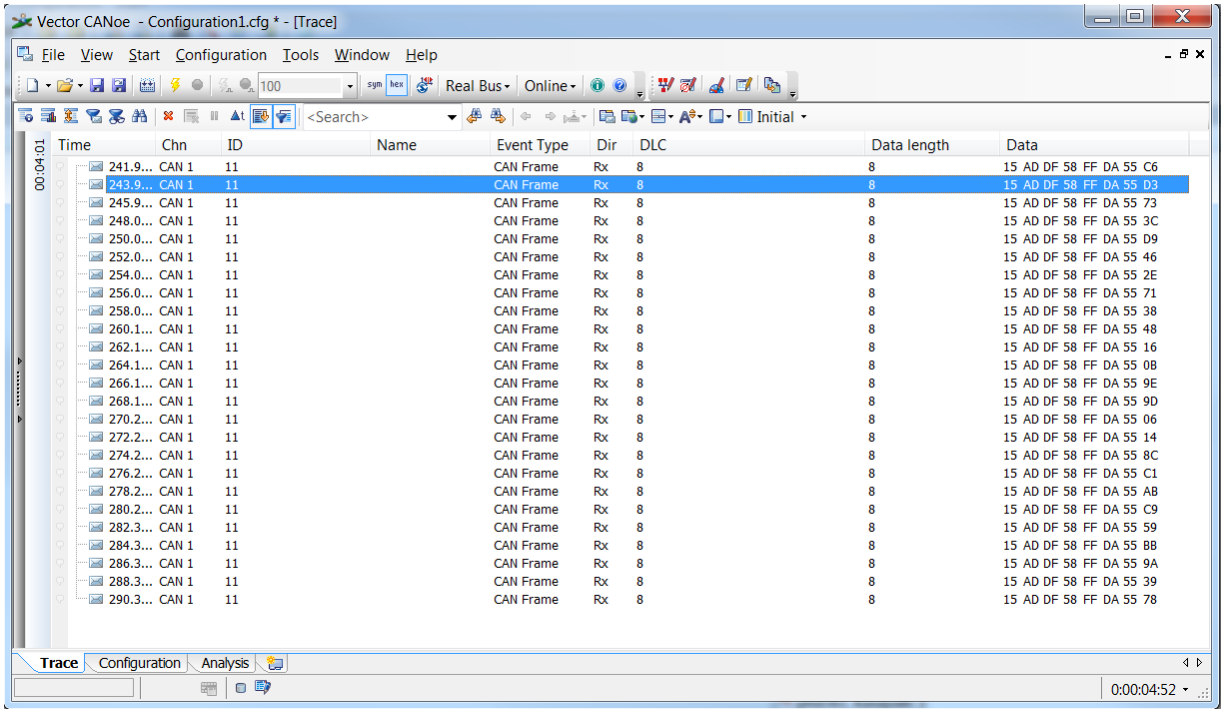
Obr. 4.11. Zobrazení bxCAN

5 Zhodnocení dosažených výsledků

Práce se zabývá návrhem monitoringu CAN sběrnice. Konkrétně se jedná o posílání jednoduché zprávy na sběrnici a následný příjem zprávy od jiného zařízení. Bylo nutné navrhnout plošný spoj pro komunikaci mezi zapůjčenou Nucleo F429ZI deskou a jiným CAN zařízením. Vývojová deska osazená 32 bitovým procesorem ARM Cortex-M4 s podporou CAN řadiče je vhodným řešením pro tento úkol. Na desce plošného spoje je umístěn CAN budič VP232 od firmy Texas Instruments sloužící k fyzickému připojení použitého CAN řadiče v procesoru. Pro komunikaci mezi zařízeními byl použit konektor D-Sub 9, který na pinech 2 a 7 má přivedené sběrnicové vodiče CAN_L a CAN_H. Návrh softwaru je řešen pomocí dostupných knihoven od firmy STMicroelectronics a jejich nástroje STM32CubeMX. Zároveň umožňuje monitorovat situaci na sběrnici a v případě přijetí zprávy od jiného zařízení, zobrazí přijatou zprávu na příslušný COM port. Softwarový nástroj CANoe od firmy Vector byl použit pro data, které posíláme. Bylo zapotřebí propojit vytvořený hardware s jiným zařízením, aby mohlo dojít ke komunikaci mezi dvěma jednotkami. Zařízení VN 7610 obsahující CAN řadič s budičem bylo využito pro realizaci komunikace s vytvořeným plošným spojem.

Obr 5.1. Vector VN 7610

Softwarový nástroj CANoe má grafický interface pro sledování komunikace. Na Obr 5.2 lze vidět průběh komunikace. Jedná se o zprávu s identifikátorem 0x11 a délkou dat 8 bajtů.



Obr 5.2. Výsledná data posílaná našim zařízením

6 Závěr

Cílem této bakalářské práce bylo prozkoumat automobilové sběrnice a provést návrh monitorovacího zařízení pro vybranou sběrnici. V práci jsou splněny všechny body zadání. V teoretické části probíhá shrnutí základních informací a principů o nejpoužívanějších automobilových sběrnicích. Z teoretických poznatků lze vyvodit srovnání těchto sběrnic. Díky přijatelné ceně a rychlosti je nejpoužívanější sběrnicí CAN. Nové automobily by měly implementovat vylepšený formát CAN FD, který by se měl stát v budoucnu standardem. V aplikacích, kde není třeba velká rychlost je možno využít pomalější a levnější sběrnici LIN. Opačnou variantu je sběrnice FlexRay disponující velmi vysokou rychlostí a mírou zabezpečení, cena je ovšem podstatně vyšší, než u CAN a LIN sběrnic.

Úkolem praktické částí bylo navržení desky plošného spoje a software pro jednoduchou komunikaci. Plošný spoj byl připraven pro dvě varianty sběrnic CAN a LIN, ale dále se práce zabývala pouze variantou pro CAN. V průběhu návrhu plošného spoje došlo k prohození napájecího pinu CAN budiče s pinem pro vysílání dat. Tato chyba však byla na plošném spoji opravena, a byla tak zajištěna správná funkcionalita hardware.

Při návrhu software došlo z počátku k chybnému nastavení přenosových rychlostí sběrnic mikroprocesoru. Toto nastavení způsobilo nefunkčnost datové komunikace sběrnice CAN. Po nalezení a následném vyřešení této chyby bylo dále za potřebí, zajistit správné přenosové rychlosti na straně přijímacích a odesílacích zařízení. V samotném závěru práce byla komunikace otestována prostřednictvím nástroje pro analýzu CAN sběrnice.

Celkovým výsledkem práce bylo nasimulování reálné zprávy o zvolených parametrech běžného automobilu se sběrnicí CAN. Princip využití řídicí jednotky v autmobilu, tedy v reálné situaci, je stejný jako při posílání naší zprávy. V reálném automobilu je ovšem komunikace mnohem složitější – chodí zde mnoho zpráv od různych řídicích jednotek a celý systém je tedy komplexnější.

Seznam literatury a informačních zdrojů

[1] VLK, František. *Automobilová elektronika 2. Systémy řízení podvozku a komfortní systémy* 1. vyd. Brno: Vlastním nákladem, 2006, 308 s. ISBN 80-239-7062-3

[2] MACHÁČEK, Miroslav. *Controller Area Network (CAN)*. [cit. 2018-05-01].

[3] POLÁK, Karel. Sběrnice CAN BUS – Controller Area Network [online].16.6.2003. [cit. 2018-05-01]. Dostupné z: http://www.elektrorevue.cz/clanky/03021/index.html

[4] SUTORÝ, Tomáš. Sběrnice LIN – Local Interconnect Network [online].10.3.2004. [cit. 2018-05-01]. Dostupné z: http://www.elektrorevue.cz/clanky/04012/index.html

[5] VECTOR. *CAN - Controller Area Network*. [online] [cit. 2018-05-01]. Dostupné z: https://elearning.vector.com/index.php?&wbt_ls_seite_id=1329976&root=378422&seite=vl_can_introduction_en

[6] VECTOR. *FlexRay* [online] [cit.2018-05-01]. Dostupné z: https://elearning.vector.com/index.php?&wbt_ls_seite_id=1329976&root=378422&seite=vl_can_introduction_en

[7] STMicroelecotronics. *User manual* [online] [cit.2018-05-01]. Dostupné z: http://www.st.com/content/ccc/resource/technical/document/user_manual/group0/26/49/90/2e/33/0d/4a/da/DM00244518/files/DM00244518.pdf/jcr:content/translations/en.DM00244518.pdf

[8] STMicroelectronics. *ARM Cortex-M4 Datasheet* [online] [cit.2018-05-01]. Dostupné z: http://www.st.com/content/ccc/resource/technical/document/datasheet/ef/92/76/6d/bb/c2/4f/f7/DM00037051.pdf/files/DM00037051.pdf/jcr:content/translations/en.DM00037051.pdf

[9] STMicroelectronics. *M4 Reference manual* [online] [cit.2018-05-01]. Dostupné z: http://www.st.com/content/ccc/resource/technical/document/reference_manual/3d/6d/5a/66/b4/99/40/d4/DM00031020.pdf/files/DM00031020.pdf/jcr:content/translations/en.DM00031020.pdf

[10] Microchip. *MCP 2003 datasheet* [online] [cit.2018-05-01]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/22230a.pdf

[11] Texas Instrument. *VP 232 datasheet* [online] [cit.2018-05-01]. Dostupné z: http://www.ti.com/lit/ds/slos346h/slos346h.pdf

Seznam použitých obrázku a tabulek

PŘÍLOHY

Příloha 1: plošný spoj DPS

Příloha 2: obsah přiloženého CD:

- použité datasheety
- soubory plošného spoje z prostředí Altium Designer
- bakalářská práce ve formátu PDF
- testovací firmware z prostředí Atollic TrueSTUDIO

PŘÍLOHY 1: Plošný spoj DPS