Diagnostika signálů řídících jednotek

Matej Juračka

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Matej Juračka**
Osobní číslo:	**A15667**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Diagnostika signálů řídicích jednotek**
Téma anglicky:	**Diagnostics of Control Units Signals.**

Zásady pro vypracování:

1. **Zpracujte přehled použití sběrnice CAN.**
2. **Nastudujte standard sběrnice CAN.**
3. **Nastudujte API ovladačů CAN převodníků firem Eberspächer a Vector.**
4. **Vytvořte utility (dodržte standard POSIX) pro odesílání a příjem po sběrnici CAN.**
5. **Ověřte funkčnost utilit se zařízením od firmy Vector.**
6. **Vytvořte návrh GUI pro vytvořené utility.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MESSMER, Hans-Peter a Klaus DEMBOWSKI. Velká kniha hardware. Brno: CP Books, 2005. ISBN 80-251-0416-8.**
2. **PRATA, Stephen. Mistrovství v C++. Praha: Computer Press, 2001. Všechny cesty k informacím. ISBN 80-7226-339-0.**
3. **KERNIGHAN, Brian W. a Dennis M. RITCHIE. Programovací jazyk C. Dotisk 1.vydání. Brne: ALBATROS MEDIA, 2013. ISBN 978-80-251-0897-0.**
4. **BOSCH. CAN Specification: Version 2.0 [online]. 1991, , 73 [cit. 2017-11-24]. Dostupné z: http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf**
5. **JAVAID, Tariq. New Network Technologies & Challenges for the Future - FlexRay, CAN FD, IP [online]. 2013, , 45 [cit. 2017-11-24]. Dostupné z: https://vector.com/portal/medien/cmc/events/commercial_events/VU_Conference/VUC13/VeC Network-Challenges_Javaid/VeCoUK13_03_Multi-Network-Challenges_Tariq-Javaid_Presentation.pdf**
6. **CAN in Automation (CiA). CAN FD – The basic idea [online]. CAN in Automation, 2017 [cit. 2017-11-24]. Dostupné z: https://www.can-cia.org/can-knowledge/can/can-fd/**

Vedoucí bakalářské práce: **doc. Ing. Martin Sysel, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **21. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 21. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 15.5.2019

MATEJ JURAČKA, v.r.

podpis diplomanta

ABSTRAKT

Diagnostika signálov riadiacich jednotiek cez zbernicu CAN s využitím štandardizovaných protokolov pre prenos a API ovládačov pre prevodník. Cieľom bude vytvorenie programových utilít pre prijímanie a posielanie v jazyku C dodržaním POSIX a vizuálne zobrazenie dát pre zbernicu. Bakalárska práca je tvorená v spolupráci s firmou.

Klíčová slova: Zbernica, CAN , uzol, rámec, C, POSIX, GUI

ABSTRACT

The diagnostic signals of control unit via CAN bus using standardized protocols for transmission and API drivers for converter. The aim will be to creation program utilities for receiving and transmitting in C language by following POSIX and visually display of data for the bus. Bachelor thesis is created in cooperation with the company.

Keywords: bus, CAN, frame, node, C, POSIX, GUI

Týmto by som rád poďakoval svojmu vedúcemu práce doc. Ing. Martinovi Syslovi, Ph.D. a Ing. Petru Mrázkovi, Ph.D. za odborné vedenie, pripomienky, cenné rady a čas, ktorý mi venovali počas vypracovania bakalárskej práce.

OBSAH

ÚVOD

Diagnostika signálov riadiacich systémov v priemysle prebieha medzi zariadeniami pomocou priemyselných zberníc. Tie boli vytvorené pôvodne s cieľom nahradiť prúdové slučky číslicovým rozhraním za účelom riadenie zariadení. Postupnom času sa zbernice vyvíjali a zdokonaľovali. Kvôli potrebe väčšej bezpečnosti posielaných dát, zvyšovania odolnosti voči elektromagnetickému rušeniu, zvýšeniu rýchlosti komunikácie medzi zariadeniami, zníženiu časov oneskorenia používaním zložitejších spôsobov komunikácie, nahradzovaniu materiálov, ktoré neboli dostatočne odolné. Všetky tieto vylepšenia a mnohé ďalšie, slúžia na to, aby sa mohli nasadzovať a využívať zbernice v čoraz náročnejších podmienkach, v ktorých sú potrebné.

Hlavnou náplňou tejto práce je vytvoriť utility, ktoré sú schopné komunikovať pomocou správ, ktoré sú tvorené rámcami po vodičoch kanálov zbernice CAN, ktorá sa využíva hlavne v automobilovom priemysle. Bakalárska práca je vyvíjaná v spolupráci s firmou, ktorá sa zameriava primárne na automobilový priemysel.

V prvej časti sú popisované samotné zbernice, v prípade automobilových zberníc sú porovnávané tie najdôležitejšie, ktoré sa využívajú s rozsiahlym popisom CAN zbernice.

Druhá časť je praktická. Najprv je opísaný prevodník pre zbernicu CAN, na ktorom je znázornená komunikácia po zbernici. Ďalej sú opísané vytvorené API firiem Vector a Eberspächer. Nakoniec využitie grafickej nadstavby k uľahčeniu prístupu na zbernicu.

I. TEORETICKÁ ČÁST

1 ZBERNICA

Pod pojmom zbernica sa rozumie sústava signálových vodičov, ktoré zaisťujú prenos dát a riadiacich povelov medzi zariadeniami, ktoré sú prepojené. Zbernica obsahuje spôsob komunikácie, komunikačný protokol [1, 14, 42].

1.1 Parametre zberníc

a) Prenosová rýchlosť – určuje maximálny počet prenesených bitov za 1 sekundu [b/s] závisí na šírke zbernice a frekvencie [1, 14, 42].
b) Šírka zbernice – určuje počet paralelných vodičov prenášajúcich dátový tok [1, 14, 42]
c) Taktovacia frekvencia – prenos informácie po zbernici je riadený hodinovými impulzami. Počet týchto impulzov za 1 sekundu udáva základnú frekvenciu zbernice [kHz, MHz] [1, 14, 42]
d) Bitová šírka adresnej zbernice – udáva adresný rozsah fyzickej pamäte, aký je zbernica schopná adresovať [1, 14, 42]

1.2 Rozdelenie zberníc

1.2.1 Rozdelenie podľa počtu vodičov

- Sériové – prenášajú dáta jedným vodičom, bit po bite pomocou paketu [1, 14, 42]
- Paralelné – prenášajú dáta viacerými vodičmi, po celých bajtoch [1, 14, 42]

1.2.2 Rozdelenie podľa druhu prenášaných signálov

- Riadiace a stavové – určujú, ktoré zariadenia môžu v danom okamihu spolu komunikovať, prípadne typ komunikácie [1, 14, 42]
- Adresové – adresujú príslušné zariadenia [1, 14, 42]
- Dátové – slúžia na prenos dát [1, 14, 42]

1.2.3 Rozdelenie podľa smeru prenosu

- Jednosmerné – polo duplexný prenos [1, 14, 42]
- Obojsmerné – plno duplexný prenos [1, 14, 42]

1.2.4 Rozdelenie podľa synchronizácie prenosu

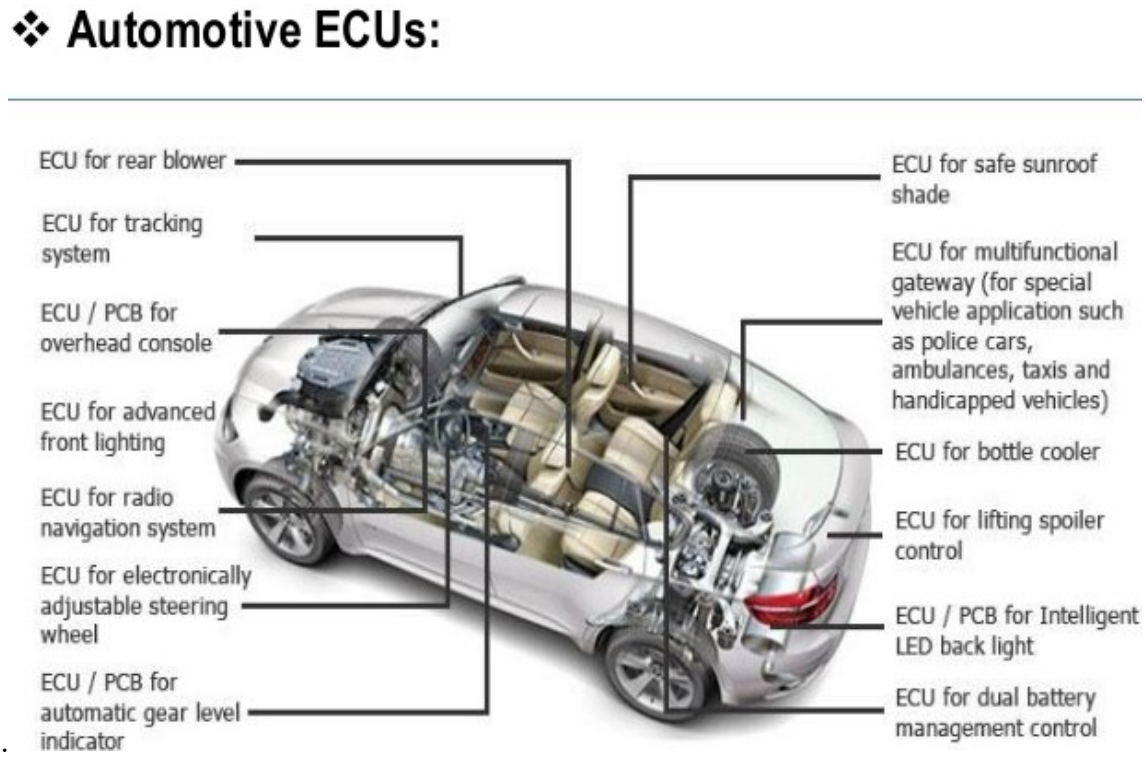
- Synchrónne – dátové prenosy sú riadené hodinovými impulzami. Toto riešenie je nevhodné pre zbernice, ktoré pracujú rôznymi rýchlosťami [1, 14, 42]
- Asynchrónne – používajú na riadenie potvrdzovacie príkazy, ktorými si vysielač a prijímač potvrdzujú vyslanie a prijatie informácie. Nepožívajú sa hodinové signály [42].

1.2.5 Rozdelenie zberníc po fyzikálnej stránke

- Pomocou zmeny elektrického napätia – jednoduché na realizáciu [1, 14, 42]
- Pomocou zmeny elektrického prúdu – väčšia odolnosť proti elektromagnetickému rušeniu [1, 14, 42]

1.3 Triedenie automobilových zberníc

V automobiloch existuje veľké množstvo zberníc, ktoré ovládajú elektronické súčasti automobilu viď obr. 1.



Obrázok 1 – Prehľad použitia zberníc v automobile [22]

Automobilové zbernice sa delia do šiestych kategórií [10, 11]:

- **Trieda A –** multiplexné káblové systémy, ktoré redukujú počet potrebných vodičov pre komunikáciu. Pre prijímanie a vysielanie pomocou multiplexovania signálov sa používa jedno prenosové médium. Tieto zbernice slúžia ako náhrada pre individuálne vodiče, ktoré vykonávajú rovnaké funkcie. Bežne trieda A definuje všeobecne komunikáciu UART (Universal Asynchronou Reciever Transmitter) s prenosovou rýchlosťou pod 10 kb/s.
- **Trieda B –** multiplexné káblové systémy, ktoré využívajú k prenosu uzly. Uzly nahradzujú existujúce modely, ktoré sa prenosu nezúčastňujú. Do triedy B patria menej spoľahlivé (non-critical) zbernice s rýchlosťami prenosu od 10 kb/s do 125 kb/s.
- **Trieda C –** multiplexné káblové systémy, ktoré pomocou vysokorýchlostných prenosov dát v reálnom čase redukujú počet potrebných vodičov pre komunikáciu. Operačné rýchlosti sa pohybujú od 125 kb/s do 1 Mb/s.
- **Emisie a diagnostika –** zbernice, ktoré sa využívajú pre kontrolu a riadenie emisií a zbernice slúžiace na diagnostiku.
- **Mobilné médiá –** multimediálne a komfortné zbernice, cez ktoré komunikujú medzi sebou rôzne multimediálne zariadenia, napr. GPS, rádio, palubný počítač.

- **X-by-wire –** označuje pridávanie elektronických systémov do vozidiel, ktoré boli predtým nahradené mechanickými a hydraulickými systémami.

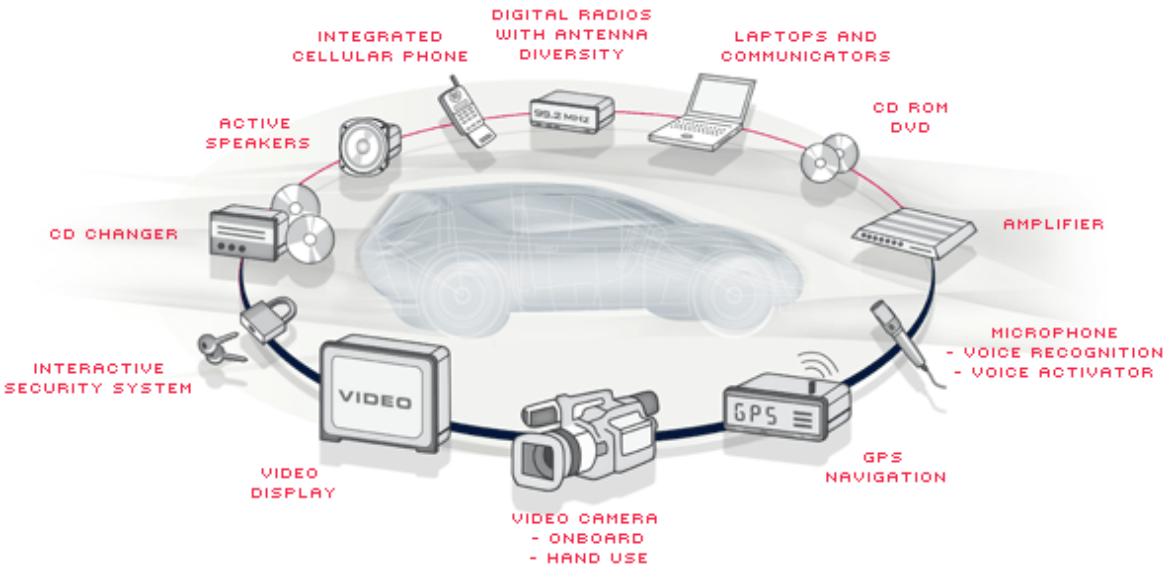
1.4 Prehľad a popis zberníc využívaných v automobiloch

V kapitole 1.4 sú v krátkosti popísané najpoužívanejšie automobilové zbernice.

1.4.1 MOST zbernica

MOST-Bus (Media Oriented Systems Transport) je vysokorýchlostná multimediálna zbernica využívaná v automobilovom priemysle, ktorá využíva plastové optické vlákno alebo elektrický vodič ako prenosové médium. Bola vyvinutá v roku 1998 organizáciou MOST Cooperation, pri ktorej zrode stáli spoločnosti Audi, BMW, Daimler-Chrysler, atď. Zbernica je typu bod-bod a jej sieťová topológia je realizovaná ako kruh. Na rozdiel od iných zberníc, ktoré sa využívajú v automobilovom priemysle, MOST zbernica je definovaná vo všetkých siedmych vrstvách ISO/OSI modelu pre dátovú komunikáciu. Definície pre fyzickú vrstvu

(elektrické a optické parametre), aplikačnú vrstvu, sieťovú vrstvu a riadenie prístupu na médium sú uvedené v špecifikácií MOST Bus specification obr. 2. Na zbernicu je možné pripojiť nízko inteligentné zariadenia, ako digitálne rádio prijímače, GPS navigácie a pod. Toto spojenie kruhového tvaru vzniká tak, že optický výstup jedného zariadenia je pripojený pomocou optického vlákna do optického vstupu ďalšieho zariadenia. Pomocou dátových rámcov sú prenášané dáta cez optické vlákna alebo elektrický vodič. Prenos je buď asynchrónny alebo synchrónny, záleží na tom aké zariadenia spolu komunikujú [25, 26, 27].

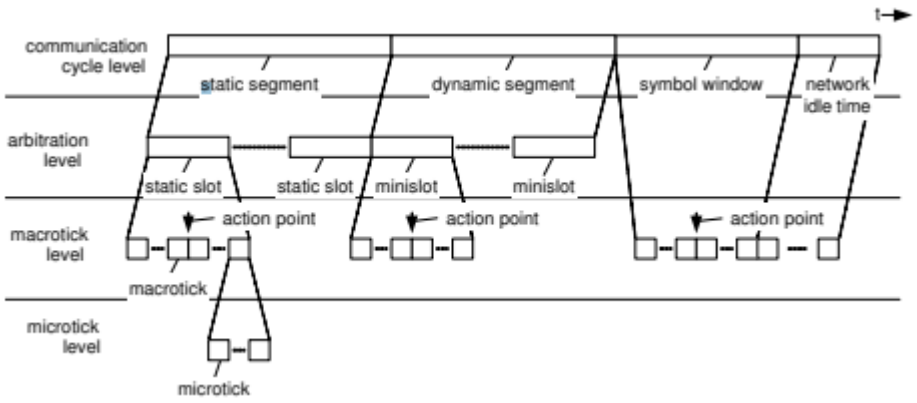


Obrázok 2 - Prepojenie zariadení MOST zbernicou [26]

1.4.2 FlexRay

FlexRay-BUS patrí medzi komunikačné protokoly s vysokorýchlostným dátovým tokom jednotlivých riadených zariadení. Bola vyvinutá konzorciom FlexRay v roku 1999. Je navrhnutá tak, aby bola rýchlejšia a zároveň spoľahlivejšia ako zbernice s podobným zameraním CAN a LIN. Jej nevýhodou je vysoká cena. Má vyššiu odolnosť voči elektromagnetickému rušeniu. Zbernica FlexRay podporuje dátové prenosy s rýchlosťou až do 10 Mb/s. Patrí medzi Multi-Master zbernice. Podporuje výhradne zbernicové topológie typu hviezdy a tzv. party line, kruhu a zbernice. Môže obsahovať dva nezávislé dátové kanály pre chybovú toleranciu. V prípade, že nastala chyba, komunikácia môže pokračovať s obmedzenou šírkou pásma ak je jeden z kanálov nefunkčný. Komunikácia prebieha v cykloch, ktorý sa rozdeľuje na dve časti viď obr. 3 [5, 8, 9]:

- Statický segment – je prerozdeľovaný do úsekov pre jednotlivé komunikačné typy, každý úsek má pridelený vlastný poskytuje silnejší vymedzenie ako jej zbernicový predchodca CAN
- Dynamický segment – funguje podobne ako CAN, uzlami preberá kontrolu podľa toho či je dostupný, čo umožňuje.



Obrázok 3 – Komunikačný cyklus FlexRay zbernice [24]

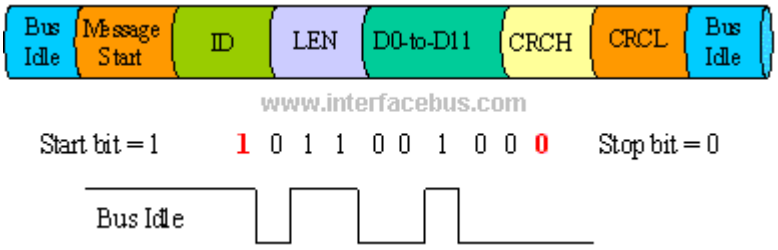
Použitie FlexRay zbernice []:

- Brzdové systémy
- Systém trakčnej kontroly
- Airbag
- Automatizácia a riadenie budov
- Zdravotnícke systémy a atď.

1.4.3 Byteflight

Byteflight zbernica sa zaraďuje medzi najnovšie a najmodernejšie vyvinuté komunikačné protokoly. Bola vytvorená spoločnosťou Motorola z dôvodu narastajúcej zložitosti elektroniky v automobiloch a ďalších komponentov. Protokol je určený na posielanie správ. Využíva niektoré vlastnosti od svojich predchodcov. Od FlexRay prevzala využívanie mixu synchrónnych a asynchrónnych prostriedkov na prenos založených na TDMA (obr. 5), aby sa vyhla nedostatkom spojených s jej predchodcami. Komunikácia je typu Master/Slave. Prenosová rýchlosť zbernice sa pohybuje v 10 Mb/s pri rýchlosti aktualizácie informácie 250 μs [27].

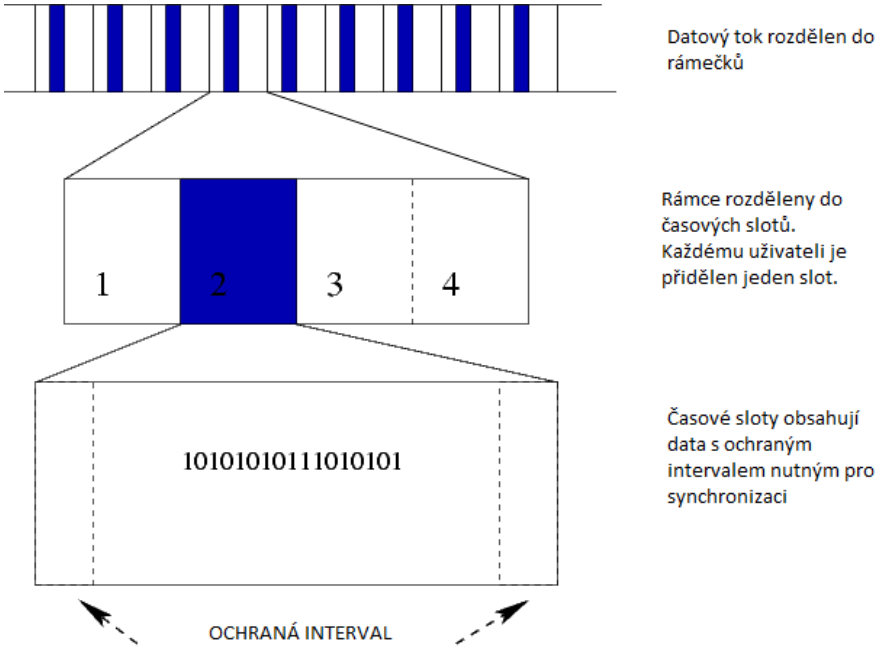
Fyzická vrstva zbernice je riešená opticky, konkrétne jej prenosové médium je plastové optické vlákno, aby sa zmenšilo elektromagnetické rušenie (EMI). Topológiou sa zbernica zaraďuje medzi hviezdu s inteligentným spojovacím blokom [10]. Rámec zbernice ByteFligh je zobrazený na obr. 4.



Obrázok 4 – Rámec zbernice ByteFlight [10]

Time Division Multiple Access (TDMA)

TDMA je viacnásobný prístup do komunikácie s časovým delením (obr. 5).



Obrázok 5 – Time Division Multiple Access, ktorý využíva pre komunikáciu zbernica Byteflight [10]

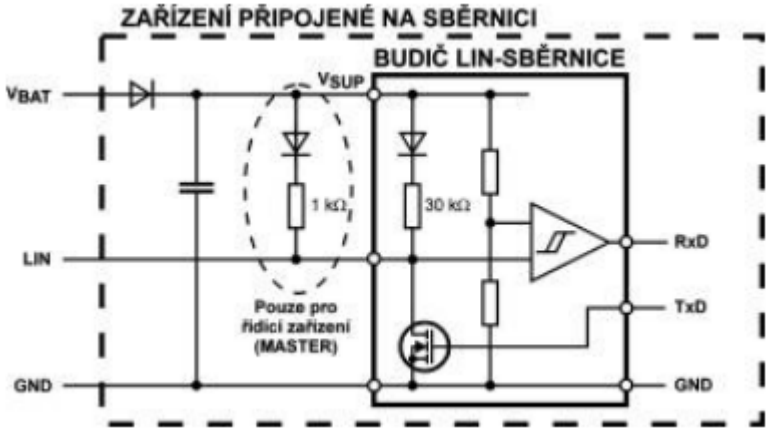
1.4.4 LIN zbernica

LIN (Local Interconnect Network) zbernica patrí do triedy A automobilových zberníc. Zbernica je jednovodičová a patrí medzi obojsmerné komunikačné protokoly. Pracuje na báze sériovej synchrónnej komunikácie UART/RS232. Bola vyvinutá LIN konzorciom.

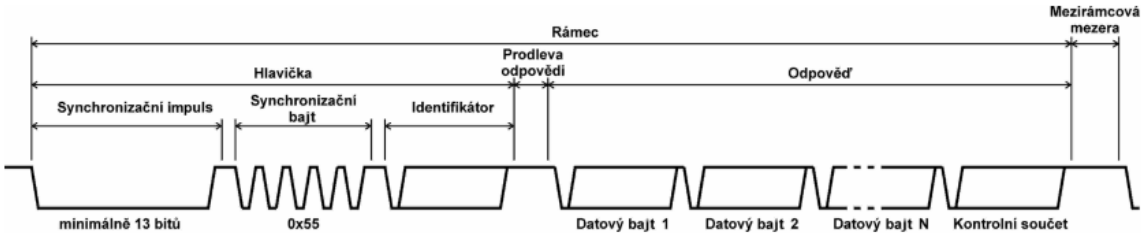
Svoje uplatnenie nachádza v komunikácii senzorov a akčných členov, ktoré pracujú s napätím 12 V. Je jednoduchšia než CAN zbernica. Komunikácia po zbernici je typu Single-Master/Multi-Slave, maximálny počet zariadení je 17 - 1 vysielač (Master) a 16 prijímačov (Slave). Zapojenie je vyobrazené na obrázku 18. Rýchlosť komunikácie po zbernici sa pohybuje od 2400 až do 19200 b/s. Svoje využitie nájde v nenáročných zariadeniach, ktoré nepotrebujú vysokorýchlostný prenos dát [11, 17]. Schéma zbernicového rámca obr. 21.

Využitie LIN zbernice je [11,17]:

- Polohovanie sedadiel, zrkadiel
- Sťahovanie okien
- Ovládanie klimatizácie, stieračov



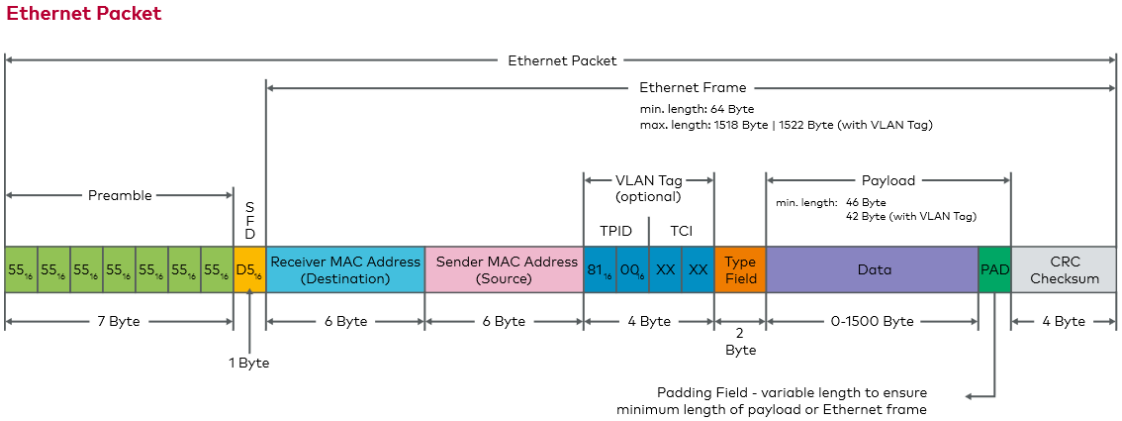
Obrázok 6 – Schéma zapojenia LIN zbernice [18]



Obrázok 7 – Rámec zbernice LIN [21].

1.4.5 Ethernet

V dôsledku zvýšenia dátových tokov v automobiloch je potrebné použiť rýchlejší typ zbernice, preto sa plánuje využiť komunikačný protokol Ethernet v automobilovom priemysle. Eternetový rámec je zobrazený na obr. 8.



Obrázok 8 - Rámec Ethernetu v automobilovom priemysle [18]

1.4.6 CAN zbernica

Zbernica CAN-BUS (Controller Area Network), ktorú vyvinula firma Bosch v roku 1986, je v súčasnej dobe najrozšírenejšou zbernicou, ktorá sa využíva v automobiloch. Zbernica sa radí medzi sériové komunikačné protokoly, podporuje efektívne rozdelenie riadenia v reálnom čase s vysokou úrovňou zabezpečenia. Oblasť jej využitia sa pohybuje od vysokorýchlostných sietí až po nízko nákladové prepojenia. Využíva sa v automobilovej elektronike, v jednotkách ovládajúcich motor, v rôznych senzoroch, v protišmykových systémoch, vo vlakoch, v lodiach. Dátové spojenia s CAN zbernicou môžu dosahovať rýchlosť 1 Mb/s. CAN zbernica je rozdelená do vrstiev [1,6].

Výhody:

- Jednoduchosť komunikačného protokolu
- V časovo kritických aplikáciách poskytuje vysoký výkon
- Činnosť a funkčnosť aj v ťažkých prevádzkových podmienkach (EMI).
- Nízka latencia pre prioritné správy
- Rýchle reakcie umožnené krátku dĺžku dátových segmentov
- Dostupnosť komunikačných obvodov

Porovnanie CAN zbernice s ostatnými zbernicami sa nachádza v Příloze P I.

2 CAN ZBERNICA

Táto kapitola sa zaoberá CAN zbernicou a jej podrobným popisom.

2.1 Všeobecné informácie

Zbernica CAN-BUS (Controller Area Network), ktorú bola vyvinutá na základe spolupráce firiem Bosch GmbH a Mercedes - Benz je v súčasnej dobe najrozšírenejšou zbernicou, ktorá sa využíva v automobiloch. Zbernica sa radí medzi sériové komunikačné protokoly, podporuje efektívne rozdelenie riadenia v reálnom čase s vysokou úrovňou zabezpečenia. Oblasť jej využívania sa pohybuje od vysokorýchlostných sietí až po nízko nákladové prepojenia [1,7].

V roku 1994 bol protokol linkovej vrstvy CAN štandardizovaný normou ISO 11898-1. Zbernica bola navrhnutá pre použitie v automobilovom priemysle. Postupne sa začala rozširovať a nasadzovať aj do iných odvetví a oblastí automatizácie, s rôznymi protokolmi aplikačnej vrstvy [7].

V automobilovej elektronike, v jednotkách slúžiacich na riadenie motora, v rôznych senzoroch, v proti šmykových systémoch, atď. sa využívajú dátové pripojenia s CAN zbernicou až do rýchlosť 1 Mb/s. Vzhľadom k rozdielnym aspektom kompatibility má napr. špecifické elektrické črty a implementácia prispôsobivosti zbernice CAN, môže byť rozdelená do rôznych vrstiev [1, 7]:.

- „(CAN -) objektová vrstva“
- „(CAN -) prenosová vrstva“
- „(CAN -) fyzická vrstva“

Objektová a prenosová vrstva môžu tvoriť tzv. linkovú vrstvu.

2.1.1 Fyzická vrstva CAN

Fyzická vrstva zbernice CAN je popísaná v normách ISO 11898-2, ISO 11898-3, SAE J2411 a ISO 1192. Normy definujú vlastnosti fyzickej vrstvy a spôsob realizácie pre metalické prenosové média ako je napr. tienená alebo netienená krútená dvojlinka. Kódovanie dát prebieha pomocou kódu NRZ viď 2.1.1.1, kde hodnota reprezentujúca jeden bit je úroveň vysokého alebo nízkeho signálu. Topológia je zbernica alebo hviezda. Dĺžka zbernice je závislá na prenosovej rýchlosti. Maximálna dĺžka zbernica je 6000 m, pri prenosovej rýchlosti 10 kb/s.

Tabuľka č.1 uvádza na základe noriem ISO a SAE prenosové rýchlosti zbernice CAN [1,7,8].

Tabuľka 1 - Normy pre fyzickú vrstvu zbernice CAN [7,8,9]

Norma	Max. prenosová rýchlosť	Oblasť použitia
ISO 11898-2	1 Mb/s	Automobilový a iný priemysel
ISO 11898-3(CAN odolný voči poruchám)	125 kb/s	Palubná elektronika automobilu
SAE J2411	33,3 kb/s	Komfortná elektronika automobilu
ISO 1192(dvojbodové spojenie)	125 kb/s	Napr. spojenie ťahača s návesom

Non Return To Zero (NRZ)

NRZ je forma digitálneho prenosu dát, v ktorom sú binárne nízke a vysoké hodnoty reprezentované číslicami 0 a 1 prenášané špecifickým a konštantným jednosmerným napätím [5,28].

V kladnom logickom NRZ je nízky stav reprezentovaný negatívnym alebo menej pozitívnym napätím a vysoký stav je reprezentovaný menej negatívnym alebo pozitívnym napätím [5,28].

Hodnoty napätia pre logické hodnoty môžu vyzerať napríklad takto:

- Logická hodnota 0 = + 1 V
- Logická hodnota 1 = + 10 V

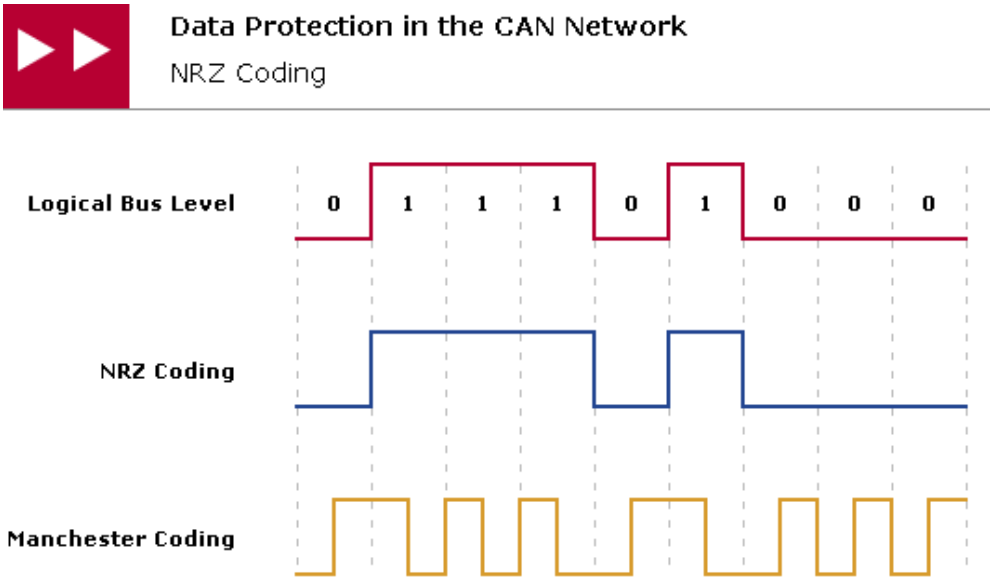
- Logická hodnota 0 = - 5 V
- Logická hodnota 1 = 0 V

V negatívnom logickom NRZ je opačný.

- Logická hodnota 0 = + 10 V
- Logická hodnota 1 = + 1 V

- Logická hodnota 0 = 0 V
- Logická hodnota 1 = - 5 V

V porovnaní NRZ s Machcester Coding nie je použité vlastné časovanie (obr. 9) [5].



Obrázok 9 – NRZ využívané v CAN zbernici [5].

2.1.2 Linková vrstva CAN

Protokol linkovej vrstvy je definovaný normou ISO 11989-1, ktorá formuluje spôsob prenosu dát zbernicou CAN. Prenášané správy sú identifikované na základe identifikátora, ktorý plní funkciu určovania ich priorít. Zbernicou sú prenášané premenné a každá z prenášaných premenných má pridelený identifikátor a tým aj prioritu. Na základe identifikátora správy môžu jednotlivé zariadenia, buď prijať správy alebo odmietnuť. Dĺžka identifikátora správy môže mať 11 bitov (základný formát rámcov) alebo 29 bitov (rozšírený formát rámcov).V jednej sieti je správu dovolené prenášať s obomi druhmi identifikátora. [4,8,9,10]

2.1.3 Objektová a prenosová vrstva CAN

Objektová vrstva a prenosová vrstva obsahujú všetky služby a funkcie vrstvy dátového spojenia definované normou ISO/OSI. Objektová vrstva sa stará o [4,7] :

- „Zistiť, ktoré správy sú na odoslanie“
- „Rozhodnúť, ktoré správy, prijaté prenosovou vrstvou, majú byť použité“
- „Poskytnúť prístup k hardvéru, ktorý súvisí s aplikačnou vrstvou“.

Prenosová vrstva je tvorená hlavne transportným protokolom, t.j. kontrolovanie rámcov, vykonávanie rozhodovania, kontrola chýb, signalizácia chýb a obmedzenie porúch.

Na úrovni prenosovej vrstvy sa rozhoduje, či je zbernica pripravená alebo sa len pripravuje na začatie nového prenosu. Niektoré všeobecné znaky bitového časovania sa považujú za súčasť prenosovej vrstvy. Taktiež pri prenosovej vrstve neexistuje možnosť modifikácií. Na úrovni fyzickej vrstvy pracuje zbernica s reálnym tokom dát medzi uzlami, (pričom hľadí na elektrické vlastnosti). V jednej sieti musí byť fyzická vrstva rovnaká pre všetky uzly. Existuje viacero možností pri výbere fyzickej vrstvy [4,7,27].

2.2 Využitie CAN zbernice mimo automobilový priemysel

Obrázok 10- Využitie CAN zbernice v iných odvetviach [24]

Zbernica CAN si našla svoje využitie aj v iných odvetviach. Pre svoju jednoduchosť, spoľahlivosť sa využíva v priemyselných (CANopen), vojenských (MilCAN), leteckých (CANaerospace) a námorných (SeaCAN) aplikáciách (obr. 10) [7].

- CANopen – je vyšší komunikačný protokol vytvorený na základe zbernice CAN. Bol vyvinutý ako široko konfigurovateľný štandardný protokol pre vstavané riadiace siete pre stroje a zariadenia s riadenými pohyblivými časťami. V súčasnosti je využívaný v rozličných odvetviach priemyslu, v lekárskej technike, automobiloch, námorných systémoch, vo verejnej doprave, pri automatizácií v stavebníctve atď. [16]
- MilCAN – komunikačný protokol bol vytvorený súkromnými spoločnosťami v spolupráci s vládnymi orgánmi pre nasadenie a otestovanie vo vojenských vozidlách [17]

- CANaerospace – protokol odvodený od zbernice CAN bol vyvinutý firmou Stock Flight Systems. Svoje opodstatnenie našiel v inžinierskych simulátoroch, simulačných kokpitoch lietadiel, experimentálnych lietadlách a najmä v Italian field – drones (UAVs) [18]
- SeaCAN – využitie v námorných lodiach

2.3 Základná koncepcia CAN zbernice

Koncept zbernice CAN je založený na [4,27]:

- Uprednostňovanie správ
- Garantované čakacie doby
- Flexibilita konfigurácie
- Príjem multicastu (posielanie dát cez sieť niekoľkým používateľom v danom okamihu) s časovou synchronizáciou
- Úplnosť údajov v systéme
- Multimaster
- Detekcia a signalizácia chýb
- Opätovné automatické poslanie poškodených správ, keď je zbernica v stave nečinná
- Rozlišovanie medzi dočasnými chybami a trvalými poškodeniami uzlov a autonómne vypínanie defektných uzlov

2.3.1 Vrstvová štruktúra uzla CAN zbernice

- Fyzická vrstva definuje spôsob prenosu signálov. V tejto špecifikácií nie je fyzická vrstva definovaná tak, aby mohla umožňovať optimalizáciu prenosového média a implementáciu úrovne signálu pre aplikáciu [4,5,27].
- Prenosová vrstva reprezentuje jadro protokolu CAN. Predstavuje správy, ktoré boli prijaté do tejto vrstvy. Je zodpovedná za časovanie, bitovú synchronizáciu, rámcovanie správ, rozhodovanie, potvrdzovanie, detekciu a signalizáciu chýb a obmedzenia porúch [4,5,27].
- Objektová vrstva sa zaoberá filtrovaním správ, spracovaním a stavov [4,5].

Tabuľka 2 – Štruktúra uzla CAN zbernice [4]

APLIKAČNÁ VRSTVA
OBJEKTOVÁ VRSTVA
-filtrovanie správ -správa a správa stavu
PRENOSOVÁ VRSTVA
-obmedzenie poruchy -detekcia a signalizácia chýb -overovanie správ -potvrdzovací bit -rozhodovanie -rámec správy -prenosová rýchlosť a časovanie
FYZICKÁ VRSTVA
-úroveň signálu a bitové zobrazenie -prenosové médium

Správy

Informácie na zbernici sa posielajú v pevných určených formátoch s rôznymi dĺžkami. V prípade, že je zbernica voľná, môže ju využívať ľubovoľné pripojené uzly k odosielaniu nových správ [4].

Smerovanie informácii (Information Routing)

V zbernici CAN, uzly nevyužijú žiadne informácie o systéme (napr. adresy staníc) [4]:

- Flexibilita systému – uzly je možné pridávať do siete bez potreby akejkoľvek soft-vérovej alebo hardvérovej zmeny aplikačnej vrstvy uzla.
- Smerovanie správ (Message Routing) – k obsah správy je pridelený identifikátor. Identifikátor správ neukazuje na cieľ správy ale popisuje význam dát, aby všetky ostatne uzly v sieti mohli rozhodnúť na základe filtra správ (MESSAGE FILTERING), ktoré dáta búdu spracované a ktoré nie.
- viFILTERING), uzly môžu prijať a súčasne pracovať s tou istou správou
- Konzistencia dát – V sieti je garantované, že správa je súčasne buď prijatá všetkými uzlami alebo žiadnym. Tak je možné dosiahnuť celistvosť a konzistentnosť údajov a vyvarovať sa chybám.

Bitová rýchlosť

Rýchlosť sa môže v rôznych systémoch líšiť. Pre jeden systém, je však rýchlosť pevná a jednotná [1].

Priority

Identifikátor označuje prioritami statické správy pri prístupe k zbernici [1].

Požiadavky vzdialených dát (Remote Data Request)

Odoslaním vzdialeného rámca (Remote Frame) uzol vyžadujúci dáta môže požiadať ďalší uzol k odoslaniu príslušného dátového rámca (Data Frame). Dátový rámec a jemu prislúchajúci vzdialený rámec sú označené rovnakým identifikátorom [1].

Multimaster

V prípade, že je zbernica voľná, každé zariadenie môže začať vysielať správu. Zariadeniu s vyššou prioritou správy bude prednostne udelený prístup k zbernici na poslanie [4,17].

Rozhodovanie (Arbitration)

Vždy, keď je zbernica dostupná, môže každé pripojené zariadenie začať vysielať správu. Ak začnú súčasne vysielať správu dve a viac zariadení, možná kolízia na zbernici je riešená pomocou bitového rozhodovania na základe identifikátora. Mechanizmus rozhodovania zaručuje, že ani informácia ani čas, nie sú stratené. Ak dátový rámec a vzdialený rámec s rovnakým identifikátorom, začínajú naraz, tak vyššiu hodnotu vykonania má dátový rámec nad vzdialeným rámcom. Počas tohto rozhodovania pri dátovom rámci sa porovnáva každý prenos na bitovej úrovni s prenosom na zbernici . Ak sú porovnané hodnoty rovnaké, bit môže byť poslaný. Keď je hodnota recesívna (recessive) môže pokračovať k odoslaniu a dominantná (dominant) hodnota je sledovaná. Bit sa môže stratiť a nastáva zhoda, preto sa musí poslať ešte jeden ďalší bit [4,17].

Bezpečnosť

V každom uzle zbernice sú implementované technológie na detekciu chýb, signalizáciu a sebakontrolu [4,17].

- Na detekciiu sa využíva [4,17]:
    - monitorovanie – porovnávanie prenosu na bitovej úrovni s prenosom prijatým na zbernici
    - cyklická kontrola redundancie
    - bitová časť

- kontrolovanie rámca správy

- **Detekcia chýb**

  Mechanizmus hľadania chýb pozostáva [4,17]:

  - všetky globálne chyby v prenose sú nájdené
  - všetky lokálne chyby sú v prenose nájdené
  - až päť náhodných chýb v prenose môže byť odstránených
  - čisté chyby (burst errors) s dĺžkou aspoň 15 sú v správe nájdené
  - každé chyby nepárnych čísiel sú nájdené (errors of any odd number in a message are detected).

  Všetky ostatné chyby sú nedetekované z dôvodu, že daná správa má menej ako udáva hodnota zo vzorca [4,17,27]

$$message\ error\ rate * 4{,}7 * 10^{-11}$$

Signalizácia chýb a čas potrebný k zotaveniu/obnoveniu

Poškodené správy sú označené uzlami ako chyby. Tieto správy sú automaticky prerobené a opätovne odoslané. Obnovovací čas od odhalenia chyby až po začiatok posielania novej správy je zvyčajne násobok 29 bit time-ov, ak sa v prenášanej správe nevyskytujú ďalšie chyby [4,17,27].

Obmedzovanie porúch

Uzly sú schopné vyhnúť sa trvalému poškodeniu na krátku vzdialenosť. Poškodené uzly sú vypnuté [4,27].

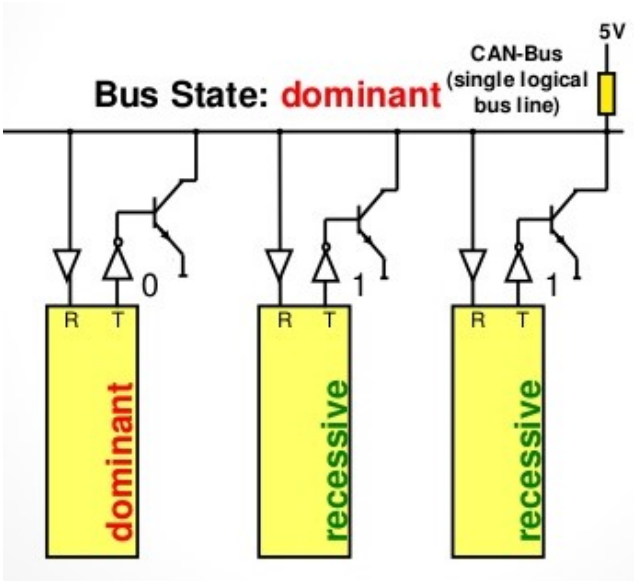
Spojenia

CAN patrí medzi sériové komunikačné zbernicové prepojenia, ktoré môže mať napojené ľubovoľný počet zariadení. Toto je možné len v teoretickej rovine, prakticky je obmedzená časovou odozvou alebo nákladmi potrebnými k spojeniu na linke zbernice [4,17,27].

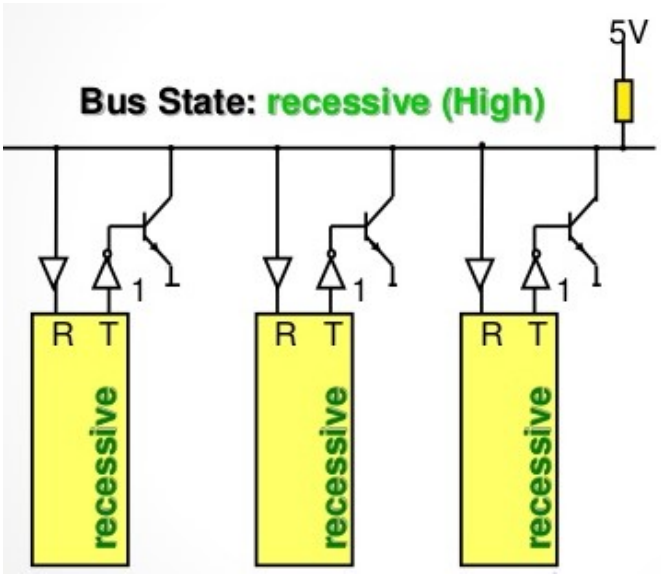
Jednokanálové

Zbernica obsahuje jeden kanál, ktorý sa stará o prenos bitov. Z dát môže byť odvodená opätovná synchronizácia. Spôsob implementácie kanála nie je pevné ustanovený. Napr. jednoduchý vodič s uzemnením, dva vodiče (wires), optické vlákna, atď.[1].

Hodnota zbernice

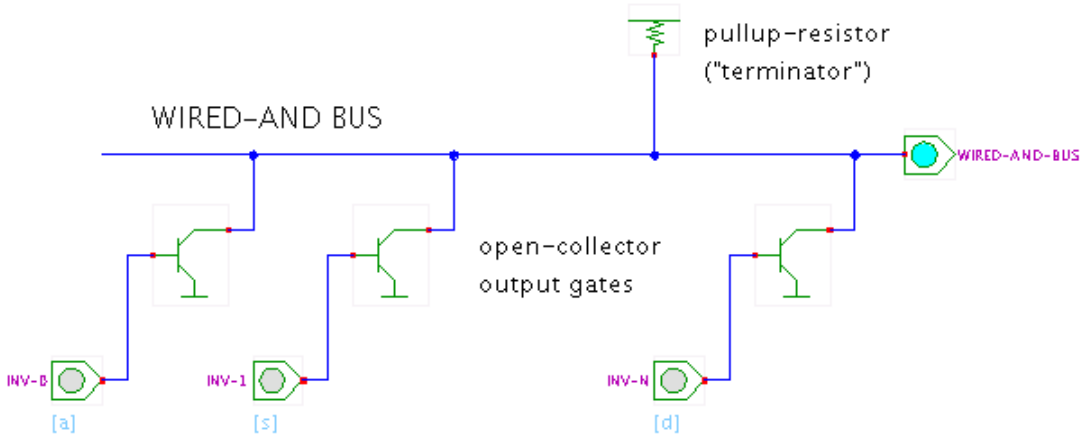


Obrázok 11 – Dominantná hodnota na zbernici [4,17,27]



Obrázok 12 - Recesívna hodnota na zbernici [4,17,27]

Zbernica môže nadobúdať dve logické hodnoty buď dominantnú (dominant) (obr. 11) alebo recesívnu (recessive) (obr. 12). Počas bitového prenosu, dominantných a recesívnych, výsledná hodnota tohto prenosu bude mať hodnotu dominantnú. Napr. v prípade, že je implementovaná zbernica typu wired-AND (obr. 14), dominantná hodnota bude reprezentovaná logickou 0 a recesívna hodnota bude reprezentovaná logickou 1 [4].



Obrázok 13 – zbernica typu wired-AND [29]

Potvrdenie (Acknowledge field)

Všetky prijímače kontrolujú celistvosť správy, ktorá bola prijatá a potvrdia úplnosť a označia neúplnú správu [4,17,27].

Režim spánku a prebudenia

Zariadenia pripojené na CAN zbernicu môžu byť uvedené do režimu spánku kvôli šetreniu elektrickej energie. K prebudeniu dochádza pri jej aktivácií [4].

2.4 Prenos správy

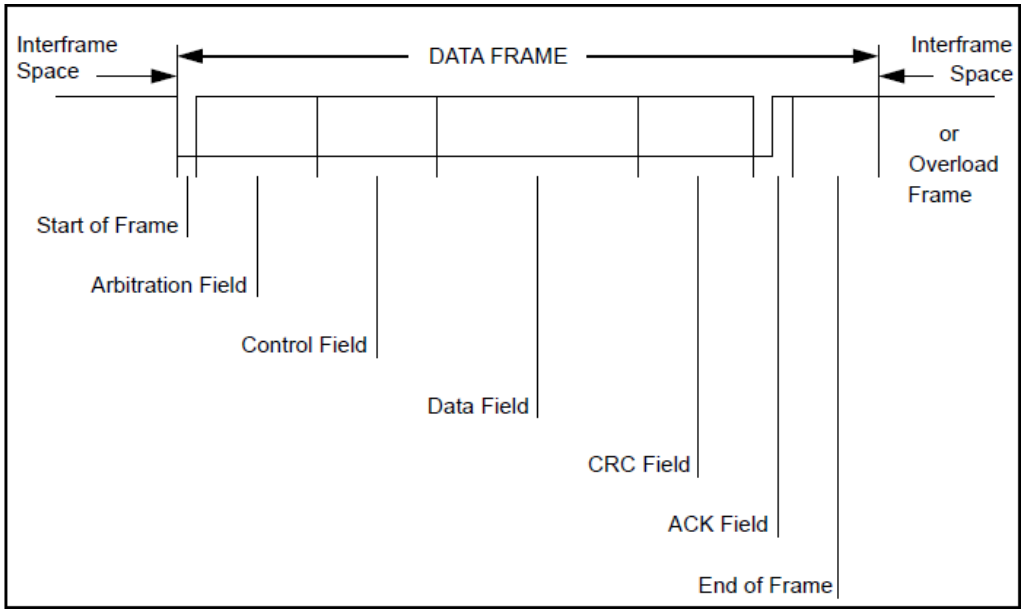
2.4.1 Typy rámcov

Prenos správy sa prejavuje a ovláda pomocou štyroch rôznych rámcov [4,18]:

- Dátový rámec (Data Frame) – stará sa o prenos dát od vysielača k prijímaču
- Vzdialený rámec (Remote Frame) - dátový rámec s rovnakým identifikátorom
- Chybový rámec (Error Frame) – signalizácia chyby na zbernici
- Rámec preťaženia (Overload Frame) – slúži k dosiahnutiu oneskorenia medzi predošlým a nasledujúcim dátovým alebo vzdialeným rámcom. Tento rámec využívajú zariadenia , ktoré nie sú schopné prijímať alebo spracovávať ďalšie rámce, kvôli svojmu vyťaženiu.

Dátový a vzdialený rámec sú oddelené od ostatných rámcov medzi-rámcovým priestorom.

2.4.2 Dátový rámec (Data Frame)



Obrázok 14 – Dátový rámec CAN zbernice [4].

Dátový rámec je zložený zo siedmich bitových polí (obr. 14) [4,18]:

- začiatok rámca
- arbitrážne pole
- riadiace pole
- dátové pole
- kontrolné CRC pole
- potvrdzovacie pole
- koniec rámca

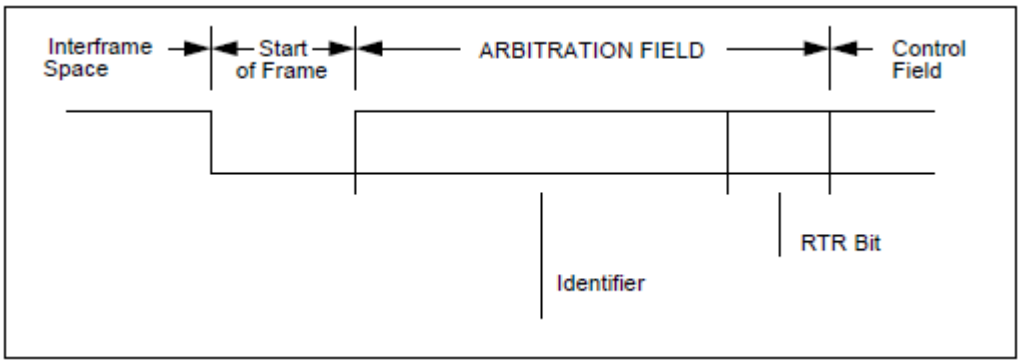
Dátové pole môže mať aj nulovú veľkosť.

Začiatok rámca (Start of Frame)

Označuje začiatok dátového a vzdialeného rámca, je zložené z jedného dominantného bitu [4,18].

Arbitrážne pole(Arbitration Field)

Pole arbitráže, alebo inak rozhodovania, pozostáva z identifikátora a RTB-Bitu (obr. 15) [4].



Obrázok 15 – Arbitrážne pole dátového rámca [4].

- Identifikátor - veľkosť identifikátora je 11 bitov. Tieto bity sú posielané v poradí od ID-10 až po ID-0, kde ID-0 je ten najmenej významným bitom.. Sedem najpodstatnejších bitov (ID-10 – ID-4) nesmú byť všetky recesívne [4,18,27].
- RTR Bit – vzdialený prenos vyžaduje bit. V dátovom rámci je RTR bit dominantný. V rámci vzdialeného rámca je RTR bit recesívny [4,18,27].

Riadiace pole (Control Field)

Je tvorené šiestimi bitmi. Obsahuje dátová dĺžka kódu (data length code) a dva rezervné bity, ktoré majú funkciu rozšírenia. Tieto bity musia byt poslané ako dominantné. Všetky kombinácie dominantných a recesívnych bitov sú prijaté prijímačmi [4,18, 27].

- Dátová dĺžka kódu – počet bitov v dátovej oblasti je určený dátovou dĺžkou kódu. Jeho šírka sú 4 bity a je prenášaný vnútru kontrolnej oblasti [4,18, 27].

Dátové pole(Data Field)

Dátové pole je zložené z dát, ktoré sú prenášané vnútri dátového rámca. Môže obsahovať 0 až 8 bytov a každý jeden obsahuje 8 bitov, ktoré prvé sú prenášané MSB (Most Significant Bite) [4,18, 27].

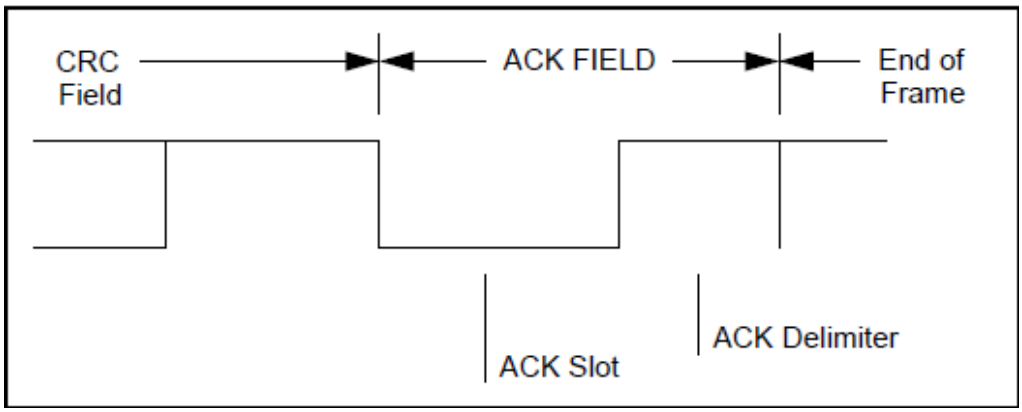
Kontrolné CRC pole (CRC Field)

Je zložená z CRC sekvencie nasledovaná CRC separátorom/oddeľovačom. Slúži k zisťovaniu chýb v prenose [4,18].

- CRC sekvencia – rámec skontrolovaný sekvenciou je odvodený cyklickou redundanciou [4,18].
- CRC separátor/oddeľovač – pozostáva z jedného recesívneho bitu [4,18].

Potvrdzovacie pole (Acknowledge Field)

Potvrdzovacie pole (obr. 16) je veľké dva bity a je tvorená z ACK otvoru (space) a ACK separátorom/oddeľovačom. Vysielač v ACK oblasti vyšle dva recesívne bity. Prijímač, ktorý obdŕžal platnú správu korektne, to ohlási cez ACK otvor vysielaču poslaním dominantného bitu [4,18]



Obrázok 16 – Potvrdzovacie pole dátového rámca [4].

- ACK medzera/otvor – všetky uzly, ktoré obdŕžali zodpovedajúcu CRC sekvenciu oznámia túto skutočnosť v ACK medzere/otvore, tým že popíšu (by superscribing) recesívny bit vysielača dominantným bitom.
- ACK separátor/oddeľovač – je druhý bit ACK oblasti a musí byť recesívny, pretože ACK medzera/otvor je obklopená dvomi recesívnymi bitmi (CRC oddeľovač , ACK oddeľovač)

Koniec rámca

Každý dátový a vzdialený rámec je ohraničený označený sekvenciou (delimited by a flag sequence), ktorá je tvorená siedmimi recesívnymi bitmi [4,18].

2.4.3 Vzdialený rámec (Remote Frame)

Požiadavka na zaslanie vzdialeného rámca má obdobný formát ako dátový rámec. Neobsahuje však dátové pole a bit RTR je recesívny (v prípade dátového rámca je to dominantný). Uzol týmto spôsobom požiada niektorý ďalší uzol o vyslanie dátového rámca s rovnakým identifikátorom aký je v požiadavke o zaslanie [4,11,12,18].

2.4.4 Chybový rámec (Error Frame)

Chybový rámec je tvorený oblasťami ERROR FLAG a ERROR DELIMETER. Uzol, ktorý objaví chybu v reťazci prijímaných bitov, začne vysielať 6 dominantných bitov, čím poruší štruktúru rámca. Ostatné uzly začnú tiež vysielať 6 dominantných bitov. Celková dĺžka ERROR FLAG sa môže pohybovať od 6 až 12 bitov. Za nimi nasleduje pole ERROR DELIMETER, ktoré obsahuje 8 recesívnych bitov [4,12,18].

2.4.5 Rámec preťaženia (Overload Frame)

Rámec preťaženia má podobnú štruktúru ako chybový rámec. Uzol vyšle rámec vtedy, keď potrebuje čas na spracovanie predchádzajúcej správy [4,18,27].

3 POUŽITÉ TECHNOLÓGIE

3.1 Jazyk C

Programovací jazyk C je štandardný programovací jazyk vyvinutý začiatkom sedemdesiatych rokov. Autorom jazyka je Dennis Ritchie. Pôvodne bol určený pre použitie na operačných systémoch UNIX. Odvtedy sa rozšíril na mnohé iné operačné systémy a je jedným z najpoužívanejších programovacích jazykov [2,3].

Jazyk C je všeobecne použiteľný programovací jazyk známy svojou efektivitou, úspornosťou a prenositeľnosťou. Vďaka týmto vlastnostiam je jeho praktické využitie vo všetkých oblastiach programovania. Jazyk je užitočný v systémovom programovaní, svoje využitie má aj pri tvorbe aplikačného software, pretože umožňuje písanie rýchlych a pomerne krátkych programov, ktoré sú ľahko prenositeľné pre iné systémy. Dobre napísaný C program je často rovnako rýchly ako program napísaný v strojovom kóde. Je výborne čitateľnejší a ľahšie udržiavateľný ako iné programovacie jazyky. K jazyku C patrí aj pár jednoduchých pravidiel, pomocou ktorých je možné vytvárať a definovať jednotlivé úseky programov do väčších a väčších celkov [3,4,16].

Bežne sa programovací jazyk využíva pri výuke programovania, aj keď nie je jednoduchý na naučenie pre neprogramátorov.

Jazyk C je implementovaný pre všetky typy procesorov a spolu so systémom UNIX predstavuje dnes jeden z hlavných trendov vo svete. Vznikol tiež celý rad rozšírení jazyka C – napr. Objective C pre objektové programovanie, Concurrent C pre paralelné programovanie, ale najperspektívnejším sa zdá byť objektovo orientované rozšírenie jazyka C++ [3,4,16].

3.1.1 Štruktúra jazyka C

Jeden programový modul, teda jeden zdrojový súbor s príponou .c, sa skladá z deklarácií globálnych premenných, deklarácií funkcií a definícií funkcií. Deklarácia funkcie deklaruje typ funkcie, meno funkcie a zoznam parametrov, definícia obsahuje aj telo. Deklarácie funkcií a premenných, ktoré majú byť dostupné aj z iných modulov, sa zväčša zapisujú do súboru, ktorý nazývame hlavičkový a má s príponou .h. Súbory s príponou .h sa vkladajú do zdrojových súborov iných modulov pomocou direktívy #include. Komentáre v C sa začínajú /* a končia */. Medzery, tabulátory a konce riadkov sú rovnocenné a nie sú zaujímavé (s výnimkou ukončovania mena). Mená typov, premenných, funkcií a podobne začínajú písmenom a

obsahujú písmená, číslice a podčiarkovník. Jazyk C je tzv. case-sensitive, to znamená, že sa dodržiava písanie malých a veľkých písmen. Začiatok a koniec funkcie, tela programu je uzavretý v párových zložených zátvorkách {} [3,4,16].

Štruktúra programu v programovacom jazyku C má všeobecný tvar:

- Skupina hlavičkových súborov štandardných funkcií
- Definície užívateľských funkcií
- Deklarácie globálnych premenných
- Funkcia main
- Ostatné užívateľské funkcie

3.2 POSIX

POSIX (The Portable Operating System Interface) je prenosné rozhranie pre operačné systémy, štandardizované ako norma IEEE 1003 a ISO/IEC 9945. Vychádza zo systému UNIX, a určuje, ako majú POSIX-konformné systémy vyzerať, čo majú vedieť, čo sa ako spracováva apod. POSIX zahrňuje rôzne aspekty operačných systémov, napr. správu procesov, prácu so súbormi, medziprocesorovú komunikáciu, základné programy (ed, awk, Korn Shell apod.), sieťové záležitosti atď. [22].

Celkovo sa jedná o 15 dokumentov. GNU/Linux je od základu navrhnutý podľa normy POSIX, a zaisťuje teda dobrú prenositeľnosť zo systému a na iné systémy splňujúci tento štandard [22].

3.2.1 POSIX jazyka C

POSIX jazyka C využíva knižnice C POSIX, čo je nezávislá knižnica, ktorá využíva C konvencie volania. Pridáva funkcie, ktoré sú špecifické práve pre systémy, čo splňujú POSIX štandard. Špecifikuje množstvo vecí, ktoré sú nad rámec tých, ktoré sú definované v štandardných knísaniach jazyka C. Vyvíjaná bola spolu so štandardom ANSI C, existujú niektoré funkcie, ktoré neboli zaradené z POSIXu do tohto štandardu. Knižnica C POSIXu obsahuje 31 knižníc [23,24].

3.3 Graphical User Interface (GUI)

Grafické používateľské rozhranie (Graphical User Interface) skrátene GUI, umožňuje ovládať elektronické zariadenia pomocou súboru interaktívnych obrazových prvkov. Tie spúšťajú príkazy a umožňujú priamu interakciu so zariadením [17].

Iný variant je CLI (Command Line interface) – ovládanie pomocou príkazového riadku.

II. PRAKTICKÁ ČÁST

Cieľom praktickej časti je opísanie prevodníku Vector VN1610, API ovládačov firiem Vector a Eberspächer a ich nadstavieb, vytvorenie konzolových aplikácií pre odosielanie a prijímanie na zbernici CAN a ich GUI rozhranie.

4 CAN PREVODNÍK

V tejto kapitole je uvedený prevodník, na ktorom bola vyskúšaná funkčnosť utilít pre CAN zbernicu. V praxi sa používajú prevodníky, ktoré splňujú normu ISO 11898-2 [35].

4.1 Vector

Nemecká firma sa špecializuje na vývoj automobilovej techniky a softvéru [41].

- Vytvára testovacie nástroje pre ECU (Engine control unit)
- ECU kalibrovanie.
- Embedded softvér a systémy
- Meracie technológie pre analýzy dát a ich zaznamenávanie
- Diagnostické nástroje
- Distribučné systémy.

Okrem prevodníkov pre CAN zbernicu vyrába prevodníky aj pre zbernice FlexRay, LIN, Ethernet [].

4.1.1 Vector VN1600 séria

Prevodníky série VN1600 Vector poskytujú flexibilný a rýchly prístup k zberniciam CAN a LIN. VN1610/VN1611 poskytujú dva kanály, pre komunikáciu po zbernici [35].

4.2 VECTOR VN1610

Zariadenie pochádza z rodiny VN1600 od firmy Vector (obr. 17). Rozhranie využíva sieťové rozhranie, nástroje ako CANoe, CANalyzer, CANape, Indigo, vFlash a mnoho ďalších aplikácií. Oblasť využitia tohto zariadenia je od jednoduchej zbernicovej analýzu až po zložitú zbernicovú simuláciu, diagnostiku, kalibráciu a úloh pre flash programovanie. Zariadenie podporuje paralelný prístup k viacerým aplikáciám na rovnakom kanály zariadenia. Kóduje signál do NRZ automaticky. [35]

Hlavné rysy:

- 2x CAN vysokorýchlostné 1051cap vysielač (kapacitne oddelené) [35]
- Softwarová synchronizácia [35]

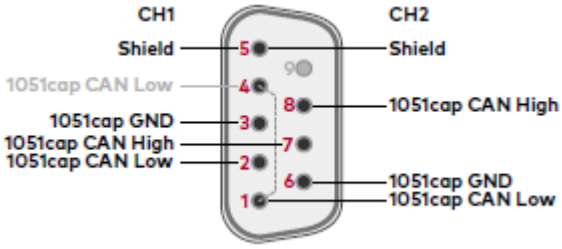
Konektory:

- D-SUB9 (CH1/2) – prevodník Vector VN1610 obsahuje D-SUB9 konektor s dvoma kanálmi CAN zbernice [35]
- USB – spojenie s pc je realizované pomocou zbernice USB [35]

Obrázok 17– Prevodník VN1610 s CAN rozhraním [35]

4.2.1 Piny D-SUB9 konektora CH1 a CH2

Konektor prevodníku VN1610 D-SUB9 (obr.18) [35]



Obrázok 18 – *D-SUB9* konektor prevodníka Vector VN1610 – rozloženie pinov [35]

4.2.2 Technické špecifikácie zariadenia VN1610

Tabuľka 3 – Technická špecifikácia Vector VN1610 [35]

CAN kanály	1x CAN high-speed 1051cap, CAN rýchlosť 2 Mbit/s CAN FD: rýchlosť 8 Mbit/s
LIN kanály	1x LIN7269cap, rýchlosť 330 Kbit/s
K-LINE kanál	1
Teplotný rozsah [°C]	-40...+70, -40...+85
Rozmery (VxSxH)	65 mm x 42 mm x 20 mm
Váha	80g
OS vyžadovaný	Windows 7 SP1,Windows 8.1, Windows 10

5 API OVLADAČOV CAN PREVODNÍKOV

V tejto stati sú opísané API (application programming interface) pre Eberspächer a Vector pre CAN zbernicu. API je zbierka knižníc, funkcií a tried. Bližšie opísané budú len tie, ktoré sú využívané v utilitách, viď kapitola 6.0. API sa nachádza v Příloze P II, v elektronickej forme na priloženom disku.

Pre implementáciu API je možné použiť jeden z dvoch spôsobov implementovania.

1. Využitie vo Windows pomocou Dynamic Linking Library (.dll) [32]
   Ekvivalent v Unixových systémoch je Shared Libraries (.so) [31]
2. Využitie priamo linkeru v C jazyku s cestou ku knižniciam s príponou „.lib“

V prvom prípade pre jednoduchšiu prácu s prevodníkom sú používane už vyvinuté API z jeho knižnicami, pretože sú univerzálnejšie pre použitie oboch druhov prevodníkov. Pod pojmom univerzálnejšie je myslené využitie tabuľky ukazateľov (pointrov), kde sa nachádzajú jednotlivé funkcie pôvodného API. Slúži na to Windows funkcia *GetProcAddress ()* [33], ktorá prideľuje jednotlivé .dll knižnice ku slovným názvom funkcií a vracia ich adresu. V Unixových systémoch je možne využiť funkciu *dlopen()* [34]. Samotné spájanie knižníc s funkciami je využite v *layer_vector_Init()* a *layer_ebel_Init ()*.

5.1 API Vector

Kompletné API od firmy Vector sa nachádza v knižnici *vxlapi.h*.

API funkcie sú definované *DECL_STDXL_FUNC*, typ oznamujúci linkeru, že sa jedná o dynamickú knižnicu.

Hlavné funkcie a funkcionality sú deklarované v knižniciach *layer_vector.h* a *libhw_vector.h*.

Využívané funkcie API v utilitách

- „layer_vector_Init ()“[37]
- „vector_open_can ()“[37]
- „vector_CanTransmit ()“[36]
- „vector_Receive ()“[36]
- „vector_store_canmsg ()“[37]
- „vector_close ()“[37]

Layer Vector Init

Najdôležitejšia súčasť vyvinutého API, kde prebieha samotná načítanie hardvéru a ovládačov spojených s ním so API od Vectoru. Využívajú sú vyššie spomenuté .dll, ktorým sú cyklicky prideľované ukazatele na funkcie, ktoré sú využívané v jednotlivých funkciách (obr. 19) [36, 37].

```
for (i = 0; i < IDX_layer_vector_MAX; i++)
{
    layer_vector_exports[i] = GetProcAddress(hMod, layer_vector_names[i]);
    if (NULL != layer_vector_exports[i])
    {
        nThunks++;
    }
    else
    {
        layer_vector_exports[i] = (FARPROC)layer_vector_NoImp;
    }
}
```

Obrázok 19 – Inicializovanie prevodníkov a ovládačov Vector [37]

Použite funkcie v utilitách využívajú vyvinuté API, ktoré implementuje pôvodné. Pre lepší prehlaď sú použite ekvivalenty oboch verzií.

Vector Open CAN

Otvorenie komunikácie pomocou funkcie *vector_open_can ()* prebieha nasledovne [36, 37].

- *vector_OpenDriver () – xlOpenDriver ()*, povoľuje prístup k ovládačom pre prevodník [36, 37]
- *vector_GetDriverConfig () – xlGetDriverConfig ()*, načíta konfiguráciu o zariadení [36, 37]
- *vector_OpenPort () – xlOpenPort()*, otvára komunikačný port, aktivuje dostupné kanály [36, 37]
- *vector_CanSetChannelBitrate () – xlCanSetChannelBitrate ()*, určuje rýchlosť komunikácie na kanáloch [36, 37]
- *vector_SetNotification () – xlSetNotification ()*, vytvorí udalosť, ktorá informuje, či sa na kanály nenachádza správa v portoch určených na príjem [36, 37]
- *vector_ActivateChannel () – xlActivateChannel ()*, nastavuje, ktorý kanál má byť aktívny k použitiu [36, 37]

Vector CAN Transmit

Poslanie správy je realizované funkciou *vector_CanTransmit ()*, ktorá prenáša vytvorenú CAN správu po aktívnom kanále pre odosielanie. Vstupné parametre funkcie sú štyri (obr. 20). Na výstupe funkcia vracia počet prenesených správ.

```
DECL_STDXL_FUNC ( xlCanTransmit, XLCANTRANSMIT,
                  XLportHandle  portHandle,
                  XLaccess      accessMask,
                  unsigned int  *pEventCount,
                  void          *pEvents)
                  );
```

Obrázok 20 – Funkcia xlCanTransmit () pre poslanie CAN správy po zbernici [36]

Vector Receive

Funkcia *vector_Receive ()* slúži na prijímanie vytvorenej CAN správy cez aktívnym kanál pre prijímanie. Vstupné parametre funkcie sú tri (obr. 20), porthandler, počet prijatých správ, ktorú sú prečítané v jednom prijímacom cykle a typ eventu. Na výstupe funkcia vracia informáciu o fronte správ či je prázdna.

```
DECL_STDXL_FUNC ( xlReceive, XLRECEIVE, (
                  XLportHandle  portHandle,
                  unsigned int  *pEventCount,
                  XLevent       *pEvents)
                  );
```

Obrázok 21 – Funkcia xlReceive () pre obdržanie CAN správy po zbernici [36]

Vector Store CAN Message

Úložisko pre ukladanie prijatých správ, ktoré majú rozdielne identifikačné číslo. V prípade, že je rovnaké ako už bolo prijate v predchádzajúcich iteráciách, je nová správa neuložená.

Funkcia je nadstavbou pôvodného API, keďže to neobsahuje priamo funkciu na ukladanie prijatých správ. Vstup do funkcie *vector_store_canmsg (CanMsg_t *pMsg, XLevent *xlEvent)* je ukazateľ na prijatú správu CAN a typ eventu. Vracia v prípade úspešného uloženia nulovú hodnotu.

Vector Close

Zatvorenie komunikácie zavolaním funkcie *vector_close (hw_vector *pData)* prebieha nasledovne [36, 37].

- *vector_ClosePort () – xlClosePort ()*, uzatvára komunikačný port a tým sa kanály stavajú neaktívne [36, 37]
- *vector_CloseDriver () – xlGetDriverConfig ()*, deaktivuje prístup k ovládačom prevodníka [36, 37]

Funkcia nedokáže úplne uzavrieť komunikáciu v uzatváraním *closeHandle ()*, pretože je jej predaná nulová hodnota. [39]

API štruktúra XLevent

Štruktúra ukladá správu, ktorá ma byť poslaná po zbernici (obr.22).

```
struct s_xl_event {
    XLeventTag          tag;
    unsigned char       chanIndex;
    unsigned short      transId;
    unsigned short      portHandle;
    unsigned short      reserved;
    XLuint64            timeStamp;
    union s_xl_tag_data tagData;
};
typedef struct s_xl_event XLevent;
```

Obrázok 22 – API štruktúra XLevent [36]

5.2 API Eberspächer

Vyvinuté API vychádzajúce z Eberspächer (ebel) API podporuje viac typov zberníc. API je len v krátkosti popísane, keďže bol použitý hardvér od firmy Vector. Kompletné API pre CAN prevodníky Eberspächer sa nachádza v knižniciach *fcBaseCAN*, *fcBaseTypesCAN.h* a *fcBase.h* [37, 38].

Hlavné funkcie a funkcionality sú deklarované v knižniciach *layer_ebel.h* a *libhw_ebel.h*. Používa funkcie, ktoré sú všeobecnejšie zamerané na použitie rôznych zberníc ako sú CAN, CAN-FD [37,38]

- „layer_ebel_Init ()“[37]
- „ebel_open_can ()“[37]

- „ebel_CANSetMessageBuffer ()“[38]
- „ebel_CANTransmit ()“[38]
- „ebel_Receive ()“[38]
- „ebel_store_canmsg ()“[37]
- „ebel_close_can ()“[37]

Ebel Layer Init

Inicializovanie prebieha rovnako ako pri prevodníkoch od firmy Vector. Najprv sú načítané všetky funkcie a k nim sú pomocou funkcie *GetProcAddress()* [33], pridelené knižnice .dll (obr. 23)

```
for (i = 0; i < IDX_layer_ebel_MAX; i++)
{
    layer_ebel_exports[i] = GetProcAddress(hMod, layer_ebel_names[i]);
    if (NULL != layer_ebel_exports[i])
    {
        nThunks++;
    }
    else
    {
        layer_ebel_exports[i] = (FARPROC)layer_ebel_NoImp;
    }
```

Obrázok 23 – Inicializovanie prevodníkov a ovládačov Eberspächer [37]

Ebel Open CAN

Otvára komunikáciu podobne ako v prípade Vectoru.

- *ebel_GetEnumFlexCardsV3 () - fcbGetEnumFlexCardsV3 ()*, alokuje pamäť pre používanie [38]
- *ebel_FreeMemory () – fcFreeMemory ()*, uvoľňuje pamäť pridelenú funkciám v API. [38]
- *ebel_Open () – fcbOpen ()*, otvára spojenie s prevodníkom a vracia jeho handler [38]

Ebel CAN Set Message Buffer

Konfiguruje vyrovnaciu pamäť správy pre poslanie *ebel_CANSetMessageBuffer ()* [38].

Ebel CAN Transmit

Prenáša správy pomocou funkcie *ebel_CANTransmit ()* po kanáloch zbernice CAN [38].

V prípade, že prenos prebehol v poriadku je vrátená nula.

Ebel Receive

Príma CAN správy *ebel_Receive ()* obdržané po kanály zbernice [38].

Ebel Store CAN Message

Funkcionalita *ebel_store_canmsg ()* je identická s *vector_store_canmsg ()* viď vyššie. Rovnako ako pri Vector API, ani túto funkciu priamo neobsahuje Eberspächer API [37].

Ebel Close CAN

Zatvorenie komunikácie pomocou funkcie *ebel_open_can ()* [37]. Funkcia volá API funkciu.

- *ebel_Close () – fcbClose ()*, uzatvára spojenie s prevodníkom [38]

6 UTILITY PRE POSIELANIE A PRÍJIMANIE SPRÁV CAN ZBERNICOU

Utility pre posielanie a prijímanie správ po CAN zbernici sú vytvorené programovacím jazykom C, pri dodržaný štandardu POSIX a volaním API funkcií pre CAN prevodník od firmy Vector vo vývojovom prostredí Microsoft Visual Studio 2017 (VS).

Utility sa nachádzajú v Příloze P II, v elektronickej forme na priloženom disku.

Komunikácia po zbernici sa delí do dvoch častí a to:

- Odosielacia časť – slúži na odosielanie správy po kanále zbernice.
- Prijímacia časť – slúži na prijatie poslanej správy po kanále zbernice.

Knižnice, ktorých funkcie, premenné, štruktúry sa v programe využívajú sú zadefinované v základných knižniciach od firmy Vector. Aby sme dodržali POSIX, použijeme knižnice, ktoré spĺňajú tento štandard.

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "stdafx.h"
#include "vxlapi.h"
#include "lib/libhw_vector.h"
#include "lib/layer_vector.h"
```

Obrázok 24 - [Zdroj: Vlastný]

- Knižnica „<stdio.h>'' – základná knižnica v jazyku C v ktorej sú deklarované základne funkcie [3]
- Knižnica „<stdlib.h>'' – knižnica v jazyku C v ktorej sú deklarované rozširujúce funkcie (atoi), makrá [3]
- Knižnica „stdafx.h“ – špeciálna knižnica pre Visual Studio zahŕňa súbory ktoré fungujú ako štandardný systémové vkladania alebo špecifické vkladacie súbory, ktoré sú často využívané ale nemenné [39]
- Knižnica „vxlapi.h“ – API knižnica Vector. Táto knižnica obsahuje celé API, ktoré je poskytované firmou Vector pre prevodníky s CAN [36]
- Knižnica „libhw_vector.h “ – vytvorená špeciálne pre prevodníky firmy Vector. Obsahuje štruktúry, ktoré deklarujú prevodník, ďalej sú v nej štruktúry využívane pri komunikáciu po zbernici CAN, ako sú otvorenie samotnej komunikácie medzi uzlami a aj jej zatvorenie [37]

- Knižnica „layer_vector“ – obsahujú funkcie, pre inicializovanie hardvéru s ovládačmi [36]

Utilita obsahuje aj iné knižnice, ktoré sú licencované firmou Vector. Pre zariadenia od firmy Eberspächer sú obdobné knižnice ako pre zariadenia od Vector.

Jednotlivé štruktúry, definujúce správu určenú na odoslanie alebo prijatie:

```
typedef struct tagCanTxRec_t
{
    DWORD dwTime;
    DWORD dwID;
    int nLen;
    BYTE byData[CAN_MAX_PAYLOAD];
} CanTxRec_t;
```

Obrázok 25 – Štruktúra správy CanTxRec_T [Zdroj: Vlastný]

Štruktúra „CanTxRec_t“ slúži na uloženie správy do pamäte aby bolo možné s údajmi pracovať ďalej. Pozostáva zo štyroch prvkov, ktoré budú popísané nižšie:

- „DWORD dwTime“
  - interval čakania správy, za ktorý bude správa odoslaná alebo prijatá
- „DWORD dwID“
  - špecifický identifikátor správy
- „int nLen“
  - uchováva informáciu o dĺžke dát, ktoré bude správa obsahovať
- „BYTE byDATA [CAN_MAX_PAYLOAD]“
  - maximálna veľkosť poľa dát, je obmedzená na 8 bajtov. Z dôvodu obmedzenia zo strany zbernice, nie je možné previesť viac ako práve 8 bajtov dát.

Štruktúra „PrgData_t“, posiela alebo prijíma správy po kanáloch zbernici CAN. Odosielacia štruktúra obsahuje štyri (obr. 26), prijímacia šesť členov (obr. 27), obe budú vysvetlené nižšie:

- „int nMsg“
  - počet správ, ktoré majú byť odoslané alebo prijaté.

- „CanTxRec_t Msg“
  - obsahuje štruktúru jednej správy. Vnorená štruktúra, je použitá z dôvodu zjednodušovania ďalšieho používania

- „hw_vector_t vector“

  - štruktúru, ktorá definuje prevodník

- „char *HwName“

  - ukazateľ na char. Tento ukazateľ ukazuje na fyzickú adresu prevodník. Pomocou prvého argumentu pri oboch utilitách je ho možne meniť.

- „long CountRecvMsg“

  - tento člen štruktúry obsahuje len prijímacia utilita. Celkový počet prijatých správ z odosielacej utility po zbernici CAN.

- „long AcceptedRecvMsg“

  - podobne ako „CountRecvMsg“, i tento to člen je špecifický pre prijímanie správy. Uchováva údaj o počte akceptovaných prijatých správ (obr. 27).

```
typedef struct tagPrgData_t
{
    int nMsg;
    CanTxRec_t Msg;
    hw_vector_t vector;
    char *HwName;
} PrgData_t;
```

Obrázok 26 – Štruktúra „PrgData_t“ pre odosielanie správ [Zdroj: Vlastný]

```
typedef struct tagPrgData_t
{
    int nMsg;
    CanTxRec_t Msg;
    hw_vector_t vector;
    char *HwName;
    long CountRecvMsg;
    long AcceptedRecvMsg;
} PrgData_t;
```

Obrázok 27 –Štruktúra „PrgData_t“ pre prijímaciu utilitu [Zdroj: Vlastný]

Nadefinovanie komunikačného zariadenia:

- „int retVal“ – slúži na návrat dát
- „PrgData_t pPrgData“ – definovanie „pPrgdata“, ktorá je určená štruktúrou „PrgData_t“
- „XLstatus s“ – slúži na identifikáciu statusu ovládača. Ak premenná *s* je nulová, prenos prebehne
- „XLevent xlEvent“ – štruktúra API Vector-u, do ktoré je uložená alebo načítaná CAN správa zo zbernice. Viac o štruktúre je v stati 5.1 Vector API.

```
int retVal = 0;
PrgData_t pPrgData;
XLstatus s = XL_SUCCESS;
XLevent xlEvent;
XLdriverConfig  xlDrvConfig;
```

Obrázok 28 - Definovanie základných premenných pre beh oboch programov. [Zdroj: Vlastný]

Základná hierarchia v utilitách:

Programy sú tvorený dvoma hlavnými funkciami:

a) „Main ()“ – hlavná funkcia, volaná ako prvá.
b) „Process_tran ()“ – funkcia odosielania správ na zbernici CAN
c) „Process_rec ()“ – funkcia prijímanie správy, ktoré boli odoslané

6.1.1 Main()

Hlavná funkcia obsahuje buď iba prázdne vstupné parametre (*void*) alebo môže obsahovať parametre *Argc* a *Arvg[]* (pointer na pole argumentov, pole argumentov).

Utility sú ovládane pomocou parametrov z funkcie *Main ()*. Tie sú spracúvané a uložené do premenných, s ktorými sa pracuje. Kvôli lepšej čitateľnosti sú argumenty zadávané v desiatkovej sústave.

Ďalšie hlavné funkcie budú vysvetlené v príslušných pasážach pre odosielanie v kapitolu 6.2, a prijímanie v kapitole 6.3.

6.2 Odosielacia utilita

Konzolová aplikácia, je vytvorená v C štandardu POSIX s API funkciami od Vectoru. ktoré sú používané pre vytvorenie a odoslanie správy po zbernici CAN. Vysvetlenie funkcionalít a funkcií API od Vectoru sa nachádza v kapitole 5.

V odosielacej časti majú vstupné argumenty do funkcie *main (int argc, char * argv [])* tvar „48829,0 1 1000 3 032180111 8“, ktoré sú uložené do pamäti pomocou premenných (obr. 27).

- Prvý argument - „48829,0“ značí špecifický identifikátor pre prevodník s použitím kanálom pre odosielanie.

- Druhý argument – „1“ je identifikátor správy, ktorú chceme vytvoriť. Z dôvodu obmedzenia, ktoré obsahuje CAN zbernica, je možné poslať 256 správ naraz z rôznym ID.
- Tretí argument – „1000“ oneskorenie odoslania správy v milisekundách [ms]. Oneskorenie nesmie byť záporné.
- Štvrtý argument – „3“ dĺžka dát, ktoré obsahuje správa. V prípade nezhody veľkosti dát s reálnymi dátami, správu nie je možné spracovať.
- Piaty argument – „032180111“ dáta správy na odoslanie.
- Šiesty argument – „8“ počet správ na odoslanie.

Tie sú spracované do nasledujúcich premenných uvedených na obrázku 29. Identifikátor zariadenia je vo forme char, ostatné sú pomocou funkcie atoi [3] prevedené do integer.

```
char *DeviceName = argv[1];
int MessID = atoi(argv[2]);
int delay_msg = atoi(argv[3]);
int CANlen = atoi(argv[4]);
int nMesgs = atoi(argv[6]);
```

Obrázok 29 - Vkladanie vstupných argumentov do pamäte pomocou premenných [Zdroj: Vlastný]

Meno zariadenia s aktívnym kanálom je vložené do štruktúry odosielanie *PrgData*, ktorá je vidieť na obrázku 27, do člena *HwName*. Identifikátor správy je pridelený do štruktúry *Msg* do člena *dwID* (obr. 25), ktorá je súčasťou v hlavnej štruktúre odosielanie *PrgData*. Časové oneskorenie je vložené do člena *PrgData.Msg.dwTime*. K spracovaniu časového oneskorenia správy je vytvorená funkcia *wait (DWORD interval)* (obr. 30), ktorá prijíma na vstupe *PrgData.dwTime*, ktorá sa zavolá po úspešnom prenesení správy.

```
void wait(DWORD interval)
{
    DWORD startTime = GetTickCount();
    while (GetTickCount() < (startTime + interval))
    {
        Sleep(0);
    }
}
```

Obrázok 30 – Funkcia oneskorenia posielania správy [Zdroj: Vlastný]

Dáta sú spracované po bajtoch. Maximálna veľkosť dát je 8 bajtov. Jeden bajt je tvorený hodnotami z rozmedzia od 0 do 255, čo reprezentujú hodnoty v ASCII tabuľke (obr. 37, obr. 38). Do poľa štruktúry *PrgData.Msg.byData[]* sú spracovávané cyklicky. Veľkosť dát určuje premenná *CANlen* (obsahuje štvrtý argument). Posledný parameter je priradený do člena *nMsg*.

Štruktúra *PrgData* je pre daná funkcii *process_tran ()*,ako jej vstupný argument.

6.2.1 Process_tran

Funkcia obsahuje hlavnú logiku a funkčnosť pre odoslanie správy pomocou zbernici CAN. Využíva funkcie opísané v API Vector (viď kapitola 5).

Najprv je potrebné vo funkcii *process_tran ()* inicializovať prevodník a ovládače (obr.19). K naplneným členom štruktúry *pPrgData* (odosielania *PrgData*) vo funkcii *main ()*, sú pridané členy vnorenej štruktúry *vector* (obr.31), ktoré sú potrebné pre otvorenie prenosu na zbernici.

```
pPrgData.vector.nSize = sizeof(pPrgData.vector);
pPrgData.vector.nCnt = 1;
pPrgData.vector.pAppName = (LPCSTR)&AppName;
pPrgData.vector.pHwName = (LPCSTR)pPrgData.HwName;
pPrgData.vector.dwWaitForEvent = VECTOR DEFAULT WAIT FOR EVENT;
```

Obrázok 31 – Napĺňanie členov štruktúry *pPrgData.vector* [Zdroj: Vlastný]

Členy obsiahnuté v štruktúre *pPrgData* sú predaná štruktúre *xlEvent*, ktorý sa stará o prenos po zbernici. Vložené členy štruktúry *pPrgData* sú nasledovne (obr. 32).

- identifikátor správy *pPrgData.Msg.dwID*
- dĺžka dát *pPrgData.Msg.nLen*
- dáta správy *pPrg.Data.Msg.byData* , ktoré boli naplnené v *main()* pomocou argumentov

```
memset(&xlEvent, 0, sizeof(xlEvent));
xlEvent.tag = XL_TRANSMIT_MSG;
xlEvent.tagData.msg.id = pPrgData.Msg.dwID;
xlEvent.tagData.msg.dlc = pPrgData.Msg.nLen;
memcpy(&xlEvent.tagData.msg.data, &pPrgData.Msg.byData, xlEvent.tagData.msg.dlc);
```

Obrázok 32 – Do štrukrúry xlEvent sú vložené členy funkcie *pPrgData*.

Po naplnení celej štruktúry *pPrgData* a štruktúry *xlEvent*, je otvorená komunikácia na zbernici API funkciou *vector_open_can ()* (viď kapitola 5.1.). V prípade, že nastane chybné otvorenie komunikácie, nie je možné pokračovať. Je potrebné znova skontrolovať, či je

správne naplnená štruktúra *vector*. Úspešným otvorením komunikácie je vytvorené textový súbor *SentCANMsgs*, do ktorého búdu zapísané odoslané dáta, pre možnosť skontrolovania korektnosti dát.

Odosielanie správy je realizované funkciou *vector_CanTransmit ()*, ktorej je predaný port, prístupová maska zo štruktúry *vector*, počet správ a samotná správa obsiahnutá v štruktúre *xlEvent* (obr. 33). Odoslaná môže byť maximálne jedná správa za je jednu iteráciu cyklu.

```
vector_CanTransmit(pPrgData.vector.xlPortHandle, pPrgData.vector.xlAccessMask, &c, &xlEvent);
```

Obrázok 33 – Odosielanie správy funkciou *vector_CanTransmit ()*

Úspešne odoslanie správy je oznámené na konzole správou „Successfully transmited“ a dáta sú zapísane do textového súboru *SentCANMsgs*.

Na konci odosielania je ukončený zápis do textového súboru *SentCANMsgs* a ukončená komunikácia na zbernici funkciou *vector_close ()*. Z dôvodu nefunkčnosti *CloseHandle ()* funkcie *vector_close* v API, nie je možné úplne zavrieť komunikačný kanál. Kanál odosielania ostáva aktívny.

6.3 Prijímacia utilita

Konzolová aplikácia, ktorá je vytvorená tiež dodržaním POSIXU jazyka C a API funkcií pre prijímanie správy po zbernici CAN od firmy Vector. Utilita je vytvorená bez čakacej fronty pomocou funkcie *vector_Receive()*, ktorá podporuje prijímanie jednej správy počas jedného prijímacieho cyklu programu.

V prípade prijímacej časti má vstupný argument do funkcie *main (int argc, char * argv[])* tvar „48829,1“, identifikátor prevodníka s požadovaným kanálom, je vložený do pamäte premennej *char* DeviceName*.

Do premenenej *retValue* je vložená funkcia *process_rec (PrgData)*, ktorá spracováva prijaté správy z odosielacej utility.

6.3.1 Process_rec ()

Opäť je potrebne inicializovať hardvér s ovládačmi ako v prípade odosielania (obr. 19)

V prijímacej utilite sa zaznamenáva počet akceptovaných prijatých správ a celkový počet obdržaných správ do členov štruktúry *pPrgData (PrgData)*, *AccpetedRecvMsg* a *CountRecvMsg* (obr.34).

```
pPrgData.vector.nSize = sizeof(pPrgData.vector);
pPrgData.vector.nCnt = 1;
pPrgData.vector.pAppName = (LPCSTR)&AppName;
pPrgData.vector.pHwName = (LPCSTR)pPrgData.HwName;
pPrgData.vector.dwWaitForEvent = VECTOR_DEFAULT_WAIT_FOR_EVENT;
pPrgData.AcceptedRecvMsg = 0;
pPrgData.CountRecvMsg = 0;
pPrgData.nMsg = 0;
```

Obrázok 34 - Naplnenie členov štruktúry *pPrgData.vector* a inicializácia členov *pPrgData* [Zdroj: Vlastný]

Otvorenie komunikačného rozhrania prebieha obdobne ako v prípade odosielacieho programu, funkciou *vector_open_can ()*. Ak je otvorenie úspešne, vytvorí sa textový súbor *ReceiveCANMsgs*, do ktorého búdu prijaté dáta zapísané.

Čakacia fronta nebola vytvorená, pretože API umožňuje prijať len jednu správu. Počas prijímacieho cyklu sa prijme len jedna štruktúra *xlEvent*, ktorý obsahuje CAN správu (obr. 35).

```
xlEvent.tag = XL_RECEIVE_MSG;
pPrgData.Msg.dwID = xlEvent.tagData.msg.id;
pPrgData.Msg.nLen = xlEvent.tagData.msg.dlc;
memcpy(&pPrgData.Msg.byData, &xlEvent.tagData.msg.data, xlEvent.tagData.msg.dlc);
```

Obrázok 35 – Naplnenie členov štruktúry *pPrgData.Msg* prijatým štruktúrou xlEvent [Zdroj: Vlastný]

Prijatá správa je vložená do úložiska *vector_store_canmsg ()*.

Úspešne prijatá správa je zobrazená konzolou správou „Succesfully receive message“ a dáta sú zapísane do textového súboru *ReceiveCANMsgs*.

Ukončenie zápisu do textového súboru *ReceiveCANMsgs* nastáva na konci prijímacieho cyklu. Následne je ukončená komunikácia na zbernici funkciou *vector_close ()*. Z dôvodu nefunkčnosti *CloseHandle ()* funkcie *vector_close* v API, nie je možné úplne zavrieť komunikačný kanál. Kanál prijímania ostáva aktívny.

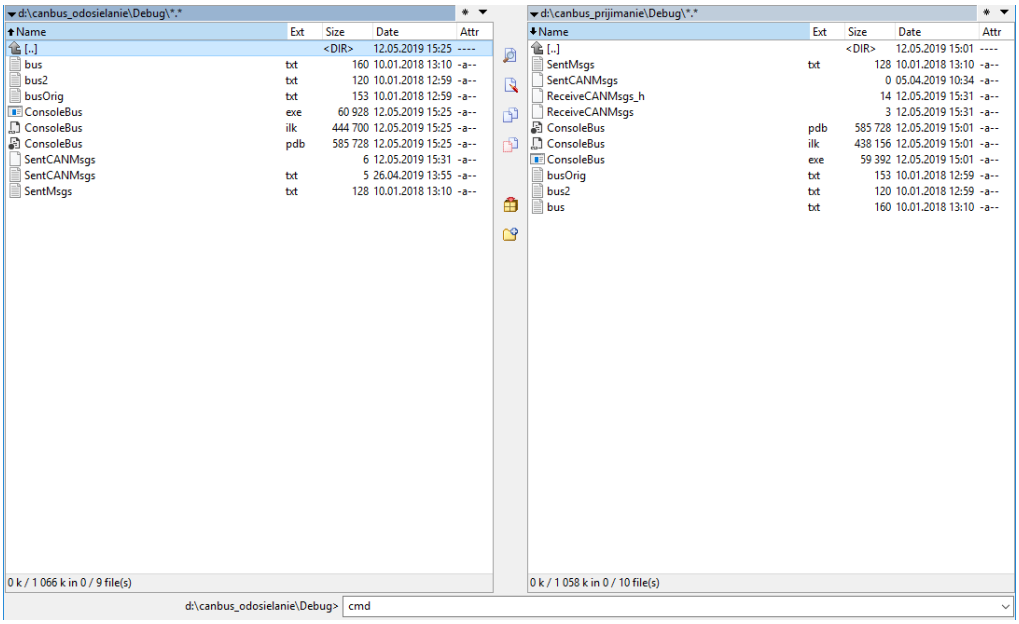
7 OVERENIE FUNKČNOSTI UTILÍT

Funkčnosť utilít bola overovaná na prevodníku od firmy Vector, konkrétne na zariadení Vector VN1610, viď kapitola 4. Zariadenie od Vector bolo poskytnuté firmou z dôvodu, že obsahuje dva kanály zbernice CAN, takže bolo možné overiť odosielanie i prijímanie na jednom prevodníku.

K prevodníku Vector VN1610, je potrebné nainštalovať všetky potrebné ovládače pre správnu komunikáciu počítača s prevodníkom a následne ho pripojiť k počítaču aby bolo pripravené na otestovanie funkčnosti utilít.

Na prevodníku bolo spravené premostenie, aby bolo možne testovať utility na oboch kanáloch. Jeden kanál (48829,0) je využitý na odosielanie vytvorenej správy a druhý kanál prevodníka (48829,1) na prijímanie.

Samotné testovanie bolo uskutočnené pomocou programu Total Commander. Spustenie konzolových aplikácií príkazovým riadkom v priečinku Debug odosielacej utility a prijímacej utility (obr. 36). Potrebné je mať prepojený prevodník VN 1610.



Obrázok 36 – Zavolanie príkazového riadku pomocou programu Total Commander pre overenie funkčnosti utilít [Zdroj: Vlastný]

7.1 Testovanie odosielania správy po zbernici CAN

Správu je možné odoslať vo forme dekadického zápisu. Po úspešnom). Za každým po úspešnom odoslaní správy je v konzolovej aplikácii zobrazovaná správa „Successfully transmited.“(úspešne prenesenie). Dáta sú uložené do textové dokumentu v hexadecimálnom zápise, pre overenie správnosti je možné použiť ASCII tabuľku (obr. 37, obr. 38)

Utilita podporuje veľkosť dát až do 8 bajtov pričom je možné ich zadávať bez oddeľovania medzerou a tabulátorom. Bolo by to brané ako ďalšie argumenty, preto je potrebné v prípade jednociferných čísiel doplniť dve nuly pred danú cifru alebo v prípade keď je číslo dvojciferné, tak len jednu nulu pred dané cifry. Znak nulu považuje utilita pri načítaní dát z argumentov ako oddeľovací znak medzi jednotlivými bajtami dát.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Obrázok 37 – ASCII tabuľka so znakmi od 0 do 127 [30].

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	|	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	169	⌐	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	▉	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

Obrázok 38 – Rozšírenie ASCII tabuľky so špecifickými znakmi pre dalšie jazyky [30].

Testovanie odosielania správ po zbernici CAN bolo uskutočnene za bežných podmienok. Boli otestované aj prípady, keď argumenty sa nezhodovali s požadovanou formou.

1. V konzole bola vytvorená správa pozostávajúca z ID zariadenia s aktívnym kanálom *48829,0*, identifikátorom správy *1*, s oneskorením posielania *1000* ms, ktorá obsahovala dáta o veľkosti troch bajtov s celkovým počtom poslaní tejto správy jeden krát. Prípad úspešného otestovane v praxi na konzolovej aplikácii je vidieť na obrázku 39.

```
d:\canbus_odosielanie\Debug>ConsoleBus.exe 48829,0 1 1000 3 032180111 1
Program name ConsoleBus.exe
Device name with active channel 48829,0
ID spravy: 1
Interval: 1000
Velkost dat: 3
Data: 032180111
Pocet sprav: 1
Successfully transmited.
```

Obrázok 39 – Úspešné odoslanie CAN správy na prevodníku Vector VN1610. [Zdroj: Vlastný]

2. V konzole bola vytvorená správa ktorá je prázdna, neobsahuje žiadne argumenty, len zavolanie spustiteľného *ConsoleBus.exe* súboru. (obr. 40).

```
d:\canbus_odosielanie\Debug>ConsoleBus.exe
Bad arguments!
Enter exe name with 6 arguments only!
For instance: ConsoleBus.exe 48829,0 2 300 3 002123250 5
```

Obrázok 40 – Spustenie odosielania bez zadania argumentov pre vytvorenie správy [Zdroj: Vlastný]

3. V konzole bola vytvorená správa, ktorej identifikátor bol menší ako 1 alebo väčší ako 256 , obmedzenie je dané maximálnym možným počtom 256 zaslaní s unikátnym ID (obr. 41).

```
d:\canbus_odosielanie\Debug>ConsoleBus.exe 48829,0 555 300 3 002123250 1
Program name ConsoleBus.exe
Device name with active channel 48829,0
ID spravy: 555
Interval: 300
Velkost dat: 3
Data: 002123250
Pocet sprav: 1
Incorrect message ID value!
```

Obrázok 41 – Odoslanie správy z väčším ID ako je 256 [Zdroj: Vlastný]

4. V konzole bola vytvorená správa, ktorej čas oneskorenia bolo záporný.

```
d:\canbus_odosielanie\Debug>ConsoleBus.exe 48829,0 1 -15 3 002123250 1
Program name ConsoleBus.exe
Device name with active channel 48829,0
ID spravy: 1
Interval: -15
Velkost dat: 3
Data: 002123250
Pocet sprav: 1
Delay value is too low.
```

Obrázok 42 – Odosielanie správy so záporným časom poslania [Zdroj: Vlastný]

5. V konzole bola vytvorené správa, ktorej veľkosť dát sa nezhodoval s reálnou hodnotou (obr. 43).

```
d:\canbus_odosielanie\Debug>ConsoleBus.exe 48829,0 1 1000 5 032180111 1
Program name ConsoleBus.exe
Device name with active channel 48829,0
ID spravy: 1
Interval: 1000
Velkost dat: 5
Data: 032180111
Pocet sprav: 1
Message date length missmatch!
```

Obrázok 43 – Odosielanie správy z rozdielnou veľkosťou dát a reálnou veľkosťou dát [Zdroj: Vlastný]

6. V konzole bola vytvorená správa zodpovedá definovanej forme ale prevodník nebol pripojený (obr. 44).

```
d:\canbus_odosielanie\Debug>ConsoleBus.exe 48829,0 1 1000 3 032180111 1
Program name ConsoleBus.exe
Device name with active channel 48829,0
ID spravy: 1
Interval: 1000
Velkost dat: 3
Data: 032180111
Pocet sprav: 1
vector_open_can failed!
```

Obrázok 44 – Odosielanie korektnej správy s nepripojením prevodníkom [Zdroj: Vlastný]

7.2 Testovanie prijímania správy po zbernici CAN

Otestovanie prijímania správy prebiehalo rovnako ako pri odosielaní správ. Pri úspešnom akceptovaní správy je v konzolovom výstupe zobrazená informatívna správa „Succesfully receive message..“ (úspešne prijatá správa). Keďže je možné prijať len jednu správu je potrebné príjem znova aktivovať.

Testovanie prijímania správy po zbernici CAN bolo uskutočnene za bežných podmienok.

Do konzolovej aplikácie pre príjem bol vložený argument identifikátora prevodníka s aktívnym kanálom.

```
d:\canbus_prijimanie\Debug>ConsoleBus.exe 48829,1
Device name with active channel 48829,1
Succesfully receive message.
ID: 1
Data[0]: 32 Data[1]: 180 Data[2]: 111
```

Obrázok 45 – Úspešné prijatie správy po zbernici CAN [Zdroj: Vlastný]

Prijatá správa mala identifikátor (ID) *1*, čo zhoduje s identifikátorom odoslanej správy, dáta o veľkosti troch bajtov, čo sa tiež zhoduje s veľkosťou posielaných dát z odosielacej utility. Posledný údaj, samotné dáta, ktoré sú rovnaké ako v prípade odosielaných dát . Úspešne prijatá správa z odosielacej utility (obr.45).

Viac prípadov nemôže nastať, keďže odosielacia utilita odošle správu v už požadovanom tvare.

8 GUI NADSTAVBA PRE UTILITY

Kapitola sa zaoberá vytvorením grafického užívateľského rozhrania pre vytvorené utility odosielania a prijímania na zbernici CAN.

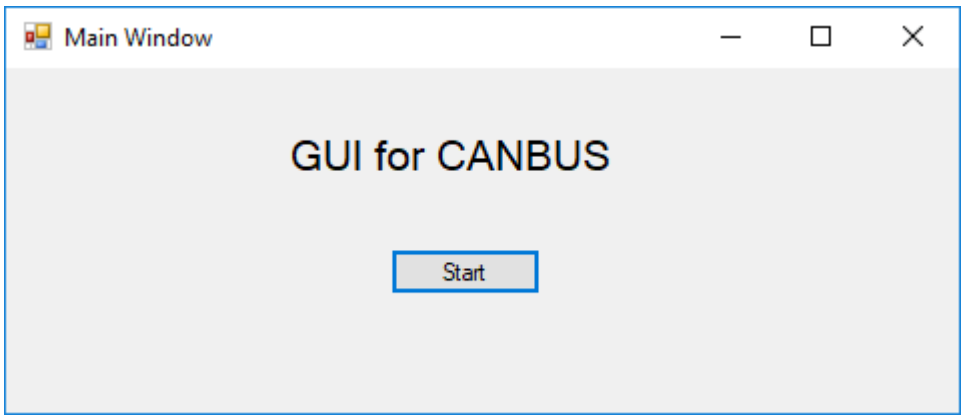
GUI je tvorené vo Windows Form (WF) postavanom na technológií .NET vo vývojovom prostredí Microsoft Visual Studio 2017 (VS). GUI sa nachádza v Příloze P II, v elektronickej forme na priloženom disku.

Grafické používateľské rozhranie je tvorené troma *Form*-s.

- Main Window
- Send Window
- Receive Window

8.1 Main Window

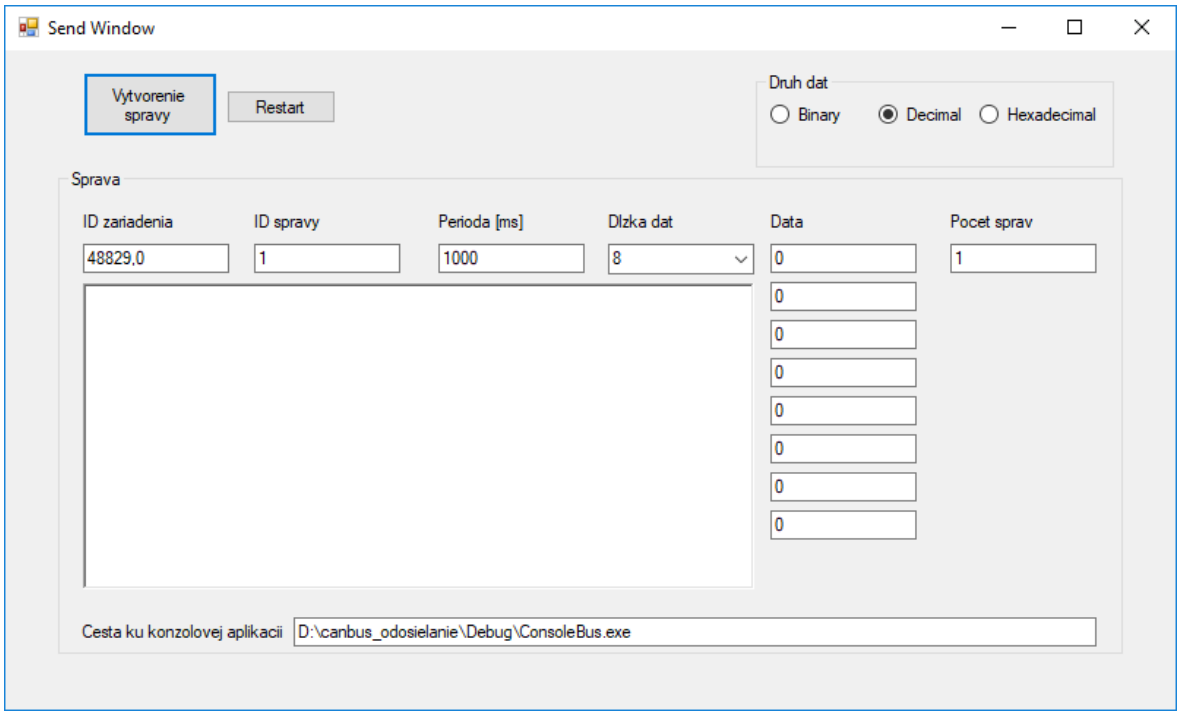
Informuje užívateľa o spustení GUI aplikácie (obr. 46). Tlačidlom *Start* sa vyvolajú ovládacie nadstavby pre odosielaciu utilitu *Send Window* a prijímaciu utilitu *Receive Window*.



Obrázok 46 – Hlavné spúšťacie okno GUI [Zdroj: Vlastný]

8.2 Send Window

Nadstavba *Send Window* slúži na ovládanie utility pre odosielanie správy po kanály CAN prevodníka (obr. 47).

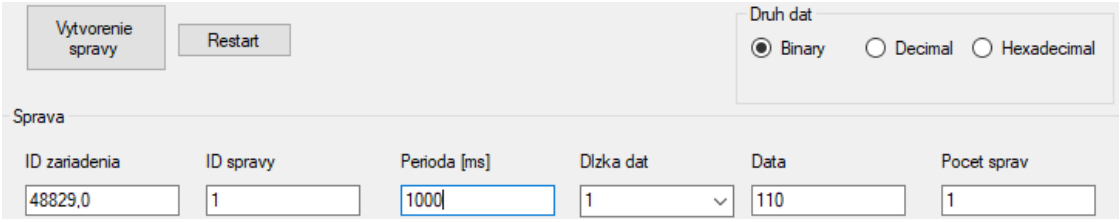


Obrázok 47 – Grafická nadstavba odosielacej utility [Zdroj: Vlastný]

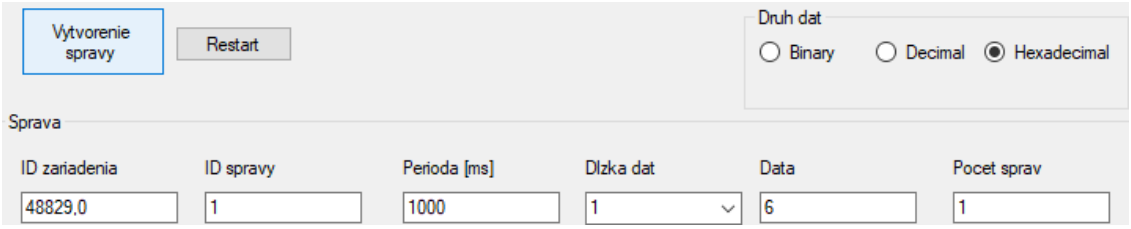
Send Window obsahuje funkcie na vytvorenie správy, ktoré sú:

- *private void SetInputType ()* – nastavuje, ktorý radionbutton je zopnutý
- *private int GetInputType ()* – vracia ktorý radiobutton bol aktivný
- *private void PovolPrislusneZnakyNaKeyPress ()* – povoľuje maximálnu veľkosť zadaných cifier do dátových textbox-ov pri jednotlivých radiobuttonoch. V každej sústave je počet cifier daný maximálnou veľkosť jedného bajtu v danej sústave, v binárnej 8 znakov, v decimálnej 3 znaky a v hexadecimálnej sú to 2 znaky.
- *private void PovolBinaryNaKeyPress ()* – obmedzuje povolené znaky len na nulu a jednotku, prípadne na mazacie klávesy
- *private void PovolHexaNaKeyPress ()* – povoľuje zadávanie hexadecimálnych znakov, čo sú čísla a písmená A – F a využitie mazacích tlačidiel na klávesnici
- *private void PovolDecimalNaKeyPress ()* – limituje akceptované znaky len na čísla do hodnoty 255 a využitie mazacích tlačidiel na klávesnici
- *private void Dlzka_sprav_SelectedIndexChange ()* – Combobox-om ovláda počet aktívnych dátových textbox-ov, ktoré je možné využiť.

- *private void ZmenStustavy_CheckedChanged ()* – prevádza vstupné hodnoty v aktívnych dátových textbox-och z jednej sústavy do ďalšej. Napríklad z binárnej (obr. 48) do hexadecimálnej (obr. 49).
- *private void PrevodDoDecimal ()* – prevádza hodnoty dát z hexadecimálnej a binárnej na desiatkovú aby bolo možné dáta prijať odosielaciu utilitou.

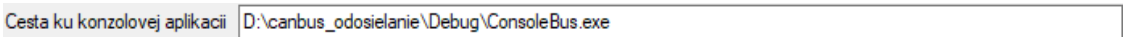


Obrázok 48 – Číslo 110 (6) zadaná v binárnej podobe [Zdroj: Vlastný]



Obrázok 49 –Prevedené číslo do hexadecimálneho čísla 6 [Zdroj: Vlastný]

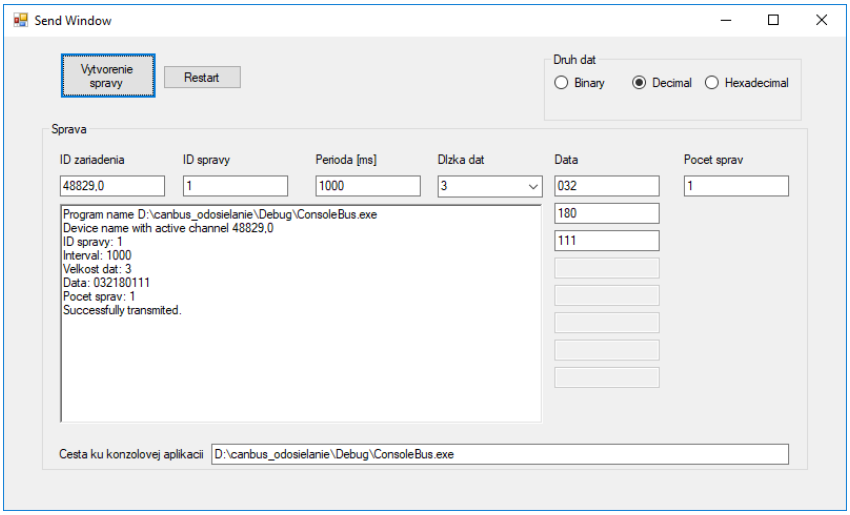
Odovzdávanie vytvorenej správy je realizované po stlačení tlačidla *Vytvorenie správy*. Pomocou tohto tlačidla sú konzolovej aplikácie pre odosielanie zaslané argumenty z textbox-ov, ktoré tvoria vytvorenú správu. Na zavolanie konzolovej aplikácie je potrebné zadať cestu kde sa nachádza (obr. 50).



Obrázok 50 - Cesta k odosielacej utilite zadaná do GUI nadstavby pre odosielanie [Zdroj: Vlastný]

Na zavolanie konzoly v .NET bola použitá trieda (class) *Process*. Tá najprv presmeruje output (výstup) z konzoly do richtextbox-u a následne jej predá správu vo forme vstupných argumentov pre funkciu *main ()* v utilite. Pre čítanie výstupu z konzole bolo použité asynchrónne čítanie.

Tlačidlom *Restart* je možné reštartovať rozhranie do predefinovaného stavu.



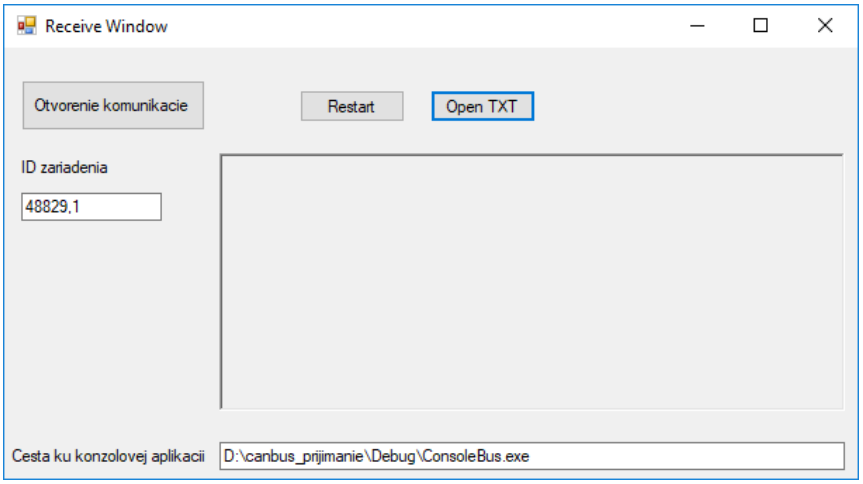
Obrázok 51 – Úspešne odoslanie správy nadstavbou pre odosielajúcu utilitu [Zdroj: Vlastný]

8.3 Receive Window

Receive Window slúži na ovládanie utility pre prijímanie správy po kanály CAN prevodníka (obr. 52).

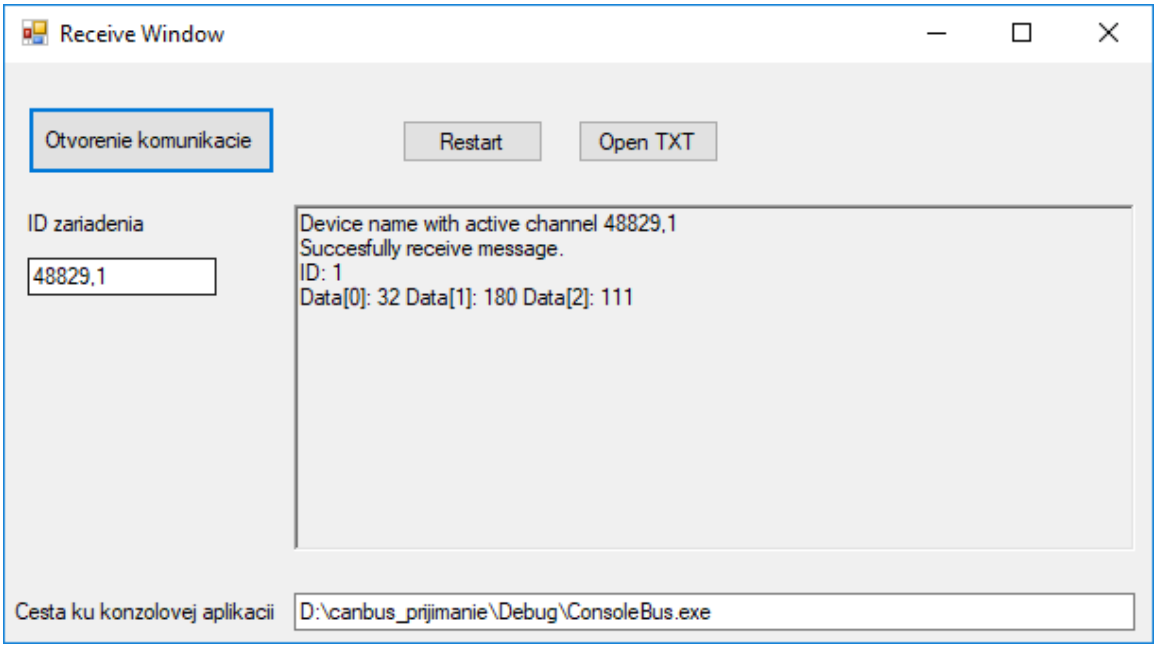
Na zavolanie konzolovej aplikácie je potrebné zadať cestu kde sa nachádza

Do konzolovej aplikácie stlačením tlačidla *Otvorenie komunikacie* zasiela parameter o mene zariadenia a kanál, na ktorom bude očakávať príjem CAN správy. Presmeruje sa výstup z konzolovej aplikácie rovnako ako v prípade odosielania. (obr. 53).



Obrázok 52 - Grafické nadstavba prijímacej utility s predefinovaným vyplnením [Zdroj: Vlastný]

V prípade úspešného prijatia je v richtextbox-e zobrazený výstup z konzoly. Pri prijatý správy je konzolou generovaný textový dokument, z ktorého je možné overiť správnosť prijatých dát. Po stlačený tlačidla *Open TXT* je zobrazené Windows okno, kde je možné zadaním cesty k vygenerovanému textovému súboru *ReceiveCANMsgs* zobraziť jeho obsah na richtextbox-e.



Obrázok 53 – Úspešné prijatie odoslanej správy grafickou nadstavbou pre prijímaciu utilitu [Zdroj: Vlastný].

ZÁVĚR

Cieľom tejto práce bolo vytvoriť utility, ktorý umožňujú poslať a prijať správu po zbernici CAN. K dosiahnutiu cieľa boli použité knižnice a funkcie APU Vector-u, ktoré obsahovali definície používania štruktúry a funkcie.

V teoretickej časti sú opísané základné rozdelenia zberníc a následne aj za kategorizovanie zberníc využívaných v automobilovom priemysle, pričom hlavné zameranie teoretickej časti bola zbernica CAN. Rozdelená je podľa noriem ISO/OSI, ktoré definujú štandard fyzickej a linkovej vrstvy. Na týchto vrstvách je tvorená v aplikačnej vrstve, správa, ktorá je posielaná po kanáloch zbernice. Správa je posielaná pomocou rámcov, v prvom rade pomocou dátového rámca, ktorý je detailne rozobratý na fragmenty z ktorých pozostáva. Fragmenty dátového rámca sú opísané a vysvetlené v teoretickej časti. Ďalej boli zadefinované dominantné a recesívne bity z ktorých pozostavujú rámce a pomocou nich sa rozhoduje o správe, kam a kde má byť po zbernici zaslaná vo forme dát.

V praktickej časti bol detailne charakterizovaný používaný prevodník, pomocou ktorého je na rozhraní zbernice CAN realizované odosielanie a prijímanie správ. Ďalej boli popísané podrobne API ovládačov Vector-u a okrajovo API Eberspächer-u pre CAN zbernicu. Samotné programovanie utilít je spracované v jazyku C s dodržaním normy POSIX pre ľahšiu prenositeľnosť medzi ďalšími platformami.

Na prevodníku je overená funkčnosť odosielania a prijímania správ. Testovanie prebiehalo na prevodníku od firmy Vector.

Overenie funkčnosti utility, prebehlo bez väčších komplikácií. V odosielacej utilite bola vytvorená správa, ktorá bola prijatá na prijímacou utilitou, kde boli v konzolovom výstupe zobrazene prijaté správy. Nad utilitami bolo vytvorené grafické ovládacie prostredie.

Výsledky práce uvedené v teoretickej i v praktickej časti môžu poslúžiť na ďalší rozvoj CAN zbernice ako aj jej využitie v iných smeroch.

Na tejto práci bola vykonávaná spolupráca s externou firmou.

SEZNAM POUŽITÉ LITERATURY

[1] MESSMER, Hans-Peter a Klaus DEMBOWSKI. Velká kniha hardware. Brno: CP Books, 2005. ISBN 80-251-0416-8.

[2] PRATA, Stephen. Mistrovství v C++. Praha: Computer Press, 2001. Všechny cesty k informacím. ISBN 80-7226-339-0.

[3] KERNIGHAN, Brian W. a Dennis M. RITCHIE. Programovací jazyk C. Dotisk 1.vydání. Brne: ALBATROS MEDIA, 2013. ISBN 978-80-251-0897-0.

[4] BOSCH. CAN Specification: Version 2.0 [online]. 1991, , 73 [cit. 2017-11-24]. Dostupné z: http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf

[5] JAVAID, Tariq. New Network Technologies & Challenges for the Future - FlexRay, CAN FD, IP [online]. 2013, , 45 [cit. 2017-11-24]. Dostupné z: https://vector.com/portal/medien/cmc/events/commercial_events/VU_Conference/VUC13/VeCoUK13/VeCoUK13_03_Multi-Network-Challenges_Javaid/VeCoUK13_03_Multi-Network-Challenges_Tariq-Javaid_Presentation.pdf

[6] CAN in Automation (CiA). CAN FD - The basic idea [online]. CAN in Automation, 2017 [cit. 2017-11-24]. Dostupné z: https://www.can-cia.org/can-knowledge/can/can-fd/

[7] CAN bus Communication. Inventure FMS Gateway [online]. Budapest, Hungury: INVENTURE AUTOMOTIVE ELECTRONICS RESEARCH & DEVELOPMENT, 2018 [cit. 2018-05-15]. Dostupné z: http://fmsgateway.com/glossary/can-bus-communication

[8] BAŽANT, Ladislav. DIGITÁLNÍ SNÍMÁNÍ OBRAZU A JEHO PŘENOS POMOCÍ SÍTĚ FLEXRAY. Brno, 2012. Bakalárska práca. VUT Brne, Fakulta elektrotechniky a komunikačných technológií. Vedoucí práce Ing. Soběslav Valch.

[9] MIKULIČ, Martin. Program pro sběr dat z vozidla s využitím diagnostického rozhraní. Praha, 2015. Diplomová práca. ČVUT, Fakulta strojná. Vedoucí práce Ing. Vojtěch Klír, Ph.D.

[10] Automotive byteflight Bus: byteflight (SI-Bus) Description[online]. Internet: Internet, 2015 [cit. 2018-05-3]. Dostupné z: http://www.interface-bus.com/byteflight_Interface.html

[11] CANopen – vyšší komunikační protokol pro vestavné sítě. Automa [online]. Praha: Automa, 2004 [cit. 2018-05-15]. Dostupné z: http://automa.cz/cz/casopis-clanky/canopen-vyssi-komunikacni-protokol-pro-vestavne-site-2004_04_32279_854/

[12] CAN-based protocols in Avionics [online]. Internet: Vector, 2012 [cit. 2018-05-15]. Dostupné z: https://vector.com/portal/medien/cmc/application_notes/AN-ION-1-0104_CAN-based_protocols_in_Avionics.pdf

[13] Použití komunikace FlexRay a integrované řadiče Freescale. Vyvoj.hw.cz [online]. Internet: HW server, 2007 [cit. 2018-05-15]. Dostupné z: https://vyvoj.hw.cz/soucastky/pouziti-komunikace-flexray-a-integrovane-radice-freescale.html

[14] Zbernica. Zbernica [online]. Bratislava: STU, 2008 [cit. 2019-05-12]. Dostupné z: http://www.dnp.fmph.uniba.sk/~kollar/pc_hw_sw/pc3.htm

[15] Principiální schéma sběrnice CAN. In: Vyvoj.hw.cz [online]. Internet: Redakcia HW serveru, 2004 [cit. 2018-05-15]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/rozhrani/aplikovani-sbernice-can.html

[16] Development of CAN, PWM, ADC and Dio Drivers under µCOS-II for The LPC2294 Target [online]. Internet: LinkedIn Corporation, 2018 [cit. 2018-05-15]. Dostupné z: https://www.slideshare.net/AymenAbdelhakim/development-of-can-pwm-adc-and

[17] CAN bus. In: Slideshare [online]. Internet: LinkedIn Corporation, 2018 [cit. 2018-05-15]. Dostupné z: https://www.slideshare.net/ssuser92b33b/can-bus-46397046?next_slideshow=1

[18] Ethernet Frame. Vector [online]. Germany: Vector, 2018 [cit. 2019-05-10]. Dostupné z: https://elearning.vector.com/mod/page/view.php?id=157

[19] POSIX [online]. Internet: Internet, 2005 [cit. 2018-05-15]. Dostupné z: http://www.abclinuxu.cz/slovnik/posix

[20] Programming Language C [online]. USA: Internet, 1988 [cit. 2018-05-23]. Dostupné z: https://web.archive.org/web/20161223125339/http://flash-gordon.me.uk:80/ansi.c.txt

[21] C Programming/POSIX Reference [online]. Internet: Internet, 2015 [cit. 2018-05-15]. Dostupné z: https://en.wikibooks.org/wiki/C_Programming/POSIX_Reference

[22] Media Oriented Systems Transport (MOST) [online]. Internet: Vector, 2015 [cit. 2018-05-19]. Dostupné z: https://vector.com/vi_most_en.html

[23] GRZEMBA, Andreas, ed. MOST: The Automotive Multimedia Network [online]. Second Edition. Berlin: Franzis, 2008 [cit. 2018-05-23]. ISBN 978-3-645-65061-8. Dostupné z: http://www2.ciando.com/img/books/extract/3645250611_lp.pdf

[24] KAŇOVSKÝ, Andrej. Informačná sieť vozidla. Žilina, 2006. Diplomová práca. Žilinská Univerzita v Žiline, Elektrotechnická fakulta. Vedoucí práce Doc. Ing Martin Vaculík, PhD.

[25] NRZ (non-return-to-zero) [online]. Internet: Tech Target, 2011 [cit. 2018-05-23]. Dostupné z: https://whatis.techtarget.com/definition/NRZ-non-return-to-zero

[26] PRIEMYSELNÉ KOMUNIKAČNÉ ZBERNICE [online]. Internet: neznáme, 2010 [cit. 2018-05-20]. Dostupné z: https://elearning.mechatronika.cool/wp-content/uploads/2017/05/kapitola_4.pdf

[27] ISO 11898-2:2016. ISO 11898-2:2016: Road vehicles -- Controller area network (CAN) -- Part 2: High-speed medium access unit. 2016. ISO: ISO, 2016.

[28] ISO 11898-5:2007. ISO 11898-5:2007: Road vehicles -- Controller area network (CAN) -- Part 5: High-speed medium access unit with low-power mode. 2007. ISO: ISO, 2007.

[29] Wired-AND bus demonstration. In: Hades simulation framework [online]. internet: internet, 2015 [cit. 2018-04-01]. Dostupné z: https://tams.informatik.uni-hamburg.de/applets/hades/webdemos/00-intro/03-stdlogic/wired-and.html

[30] ASCII tabuľka a jej praktické využitie. XTechnik [online]. Slovenská republika: XTechnik, 2007 [cit. 2019-05-09]. Dostupné z: http://www.xtechnik.szm.com/Files/Main/ascii.html

[31] MATTHEW, Neil a Richard STONES. Beginning Linux®Programming, 4th Edition. 4th Edition. USA: Wiley Publishing, 2008. ISBN 978-0-470-14762-7.

[32] Dynamic-Link Libraries. Microsoft [online]. USA: Microsoft, 2018 [cit. 2019-05-11]. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/Dlls/dynamic-link-libraries

[33] GetProcAddress function. Microsoft [online]. USA: Microsoft, 2018 [cit. 2019-05-11]. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/api/libloaderapi/nf-libloaderapi-getprocaddress

[34] Dlopen(3) - Linux man page. Die.net [online]. USA: die.net, 2019 [cit. 2019-05-11]. Dostupné z: https://linux.die.net/man/3/dlopen

[35] VN1600 Interface Family: Manual. Vector [online]. Stuttgart: Vector Informatik, 2017 [cit. 2019-05-12]. Dostupné z: https://assets.vector.com/cms/content/products/VN16xx/docs/VN1600_Interface_Family_Manual_EN.pdf

[36] API Vector - Driver Interface Prototypes - customer version, Vector Informatik GmbH [CD]. Gernany: Vector Informatik GmbH, 2008 [cit. 2019-05-12] Dostupné z : Přílohy II na priloženom disku

[37] API knižnice, NAVCON s. r. o. [CD]. Czech Republic: Navcon s.r.o., 2016 [cit. 2019-05-12] Dostupné z: Přílohy II na priloženom disku.

[38] API Eberspächer - Eberspächer Electronics GmbH [CD]. Germany: Eberspächer Electronics GmbH, 2016 [cit. 2019-05-12]. Dostupné z: Přílohy II na priloženom disku.

[39] CloseHandle function. Microsoft [online]. USA: Microsoft, 2019 [cit. 2019-05-12]. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/api/handleapi/nf-handleapi-closehandle

[40] Stdafx.h. Cplusplus.com [online]. Internet: Internet, 2013 [cit. 2019-05-12]. Dostupné z: http://www.cplusplus.com/articles/1TUq5Di1/

[41] Vector. Vector [online]. Germany: Vector, 2019 [cit. 2019-05-12]. Dostupné z: https://www.vector.com/int/en/

[42] Zbernice. Prostriedky na prenos dát - zbernice a rozhrania [online]. Piešťany: SPŠE, 2009 [cit. 2019-05-12]. Dostupné z: http://www.portal.strednaskola.eu/doc/zbernice.pdf

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application programmig interface
PC	Personal computer
ECU	Engine control unit
CAN-Bus	Control Area Network
LIN-Bus	Local Interconnect Network
MOST	Media Oriented Systems Transport
UAV	unmanned aerial vehicle
m	meter
Mb/s	Megabit za sekundu
kb/s	Kilobit za sekundu
V	Volt
TDMA	Time Division Multiple Access
NRZ	Non Return To Zero
MSB	Most Significant Bite
Int	integer
DWORD	unsigned long int
HPT	High Performance Timer
Char	character
s	sekunda
Ms	milisekunda
ebel	Eberspächer
ISO	International Organization for Standardization
OSI	Open Systems Interconnection Reference Model
SAE	Society of Automotive Engineers

POSIX	Portable Operating System Interface
DLL	Dynamic Link Library
SO	Shared Libraries
Msg	message
MHz	Megahertz
kHz	Kilohertz
mm	Milimeter
VxSxH	Výška, šírka, hĺbka
g	gram
C	Programovací jazyk C
VS	Visual Studio
GUI	Graphical User Interface
WF	Windows Form
t.j	To jest
atď.	A tak ďalej
viď	vidieť

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: TABUĽKA POROVANIE ZBERNÍC

Príloha obsahuje programové časti utility pre komunikáciu po zbernici CAN [Zdroj: Vlastný]

Vlastnosti	CAN	Byteflight	LIN	FlexRay	MOST	Ethernet
prenos správy	asynchrónny	asynchrónny a synchrónny	synchrónny	asynchrónny a synchrónny	asynchrónny a synchrónny	-
identifikátor správy	Identifikátor správy	identifikátor správy	identifikátor správy	časový záznam	nepodporované	IPv6
rýchlosť dátového prenosu	1 Mb/s	10 Mb/s	20 kb/s	10 Mb/s	24 Mb/s	Gb/s
bitové kódovanie	NRZ s bitovým vyplňovaním	NRZ s bitmi štart/stop	neznáme	neznáme	neznáme	neudáva sa
fyzická vrstva	dvojitá linka	dvojitá linka/optická linka	jedno-linka	dvojitá linka/optická linka	dvojitá linka na báze optickej linky	jedno/dvojitá linka
časová nestabilita (Latency jitter)	závislá na zaťaženosti zbernice	konštanta pre prioritné správy t_cyc	konštantná	konštantná	prúd dát	nulová
kontrola	Multi-master	Multi-master	Single-master	Multi-master	Timing-master	
Časová kompozičnosť	nepodporovaná	podporované pre správy s vysokou prioritou	-	neznáma	neznáma	neznáma
obsahová chyba (fyzická vrstva)	čiastočne	poskytované optickým vláknom a vysielačom	-	neznáma	neznáma	neznáma
rozšíriteľnosť	výborná	rozšírenie možné pre správy s vysokou prioritou s vplyvom na asynchrónnu šírku pásma	-	výborná	-	výborná
prispôsobivosť	flexibilná šírka pásma pre každý uzol	flexibilná šírka pásma pre každý uzol	Malá, z dôvodu pomalého dátového prenosu	vysoká úroveň	-	

PŘÍLOHA P II: UTILITA

Príloha obsahuje programové časti utility pre komunikáciu po zbernici CAN spolu s GUI nadstavbou.