TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

AUTOMATICKÉ MONITOROVÁNÍ SBĚRNICE CAN V AUTOMOBILECH

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 3906T001 – Mechatronika

Autor práce: **Bc. Tomáš Louč**
Vedoucí práce: Ing. Jan Koprnický, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

AUTOMATIC MONITORING OF CAN-BUS IN CARS

Diploma thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 3906T001 – Mechatronics

Author: **Bc. Tomáš Louč**
Supervisor: Ing. Jan Koprnický, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2013/2014**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš Louč**
Osobní číslo:	**M12000255**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Mechatronika**
Název tématu:	**Automatické monitorování sběrnice CAN v automobilech**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Uveďte účel automatického monitorování sběrnice CAN v automobilech.
2. Rozeberte vhodné monitorovací metody.
3. Pomocí vývojového prostředí CANoe vytvořte skript pro sledování změn na sběrnici CAN.
4. Vytvořený skript odzkoušejte na reálném stavu a zhodnoťte jeho využitelnost.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **40–50 stran**

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

Seznam odborné literatury:

[1] **BOSCH: CAN Specification. Technická zpráva, Robert Bosch GmbH, Stuttgart, 1991.**

[2] **Emaus, Bruce. Programming with CAPL. 1. vydání. Novi, Michigan, Vector CANtech, Inc., 2004.**

[3] **Reif, Konrad. Automobil Elektronik; 2. vydání, Friedrich Vieweg & Sohn Verlag, GWV Fachverlage GmbH, Wiesbaden, 2007.**

Vedoucí diplomové práce: **Ing. Jan Koprnický, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant diplomové práce: **Ing. Václav Urban**
e4t electronics for transportation s.r.o.

Datum zadání diplomové práce: **10. října 2013**

Termín odevzdání diplomové práce: **16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci L.S. - 14 -

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2013

Prohlášení

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

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

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

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Na tomto místě bych chtěl poděkovat vedoucímu diplomové práce Ing. Janu Koprnickému, Ph.D. a konzultantovi práce Ing. Václavu Urbanovi za jejich cenné rady, vedení práce a čas, který mi věnovali. Dále pak Technickému vývoji Škoda Auto, zejména Ing. Petru Stuchlíkovi z oddělení TME/2, za poskytnuté technické zázemí, v němž mohla práce vzniknout.

Poděkování patří také rodičům za veškerou podporu po celou dobu studia, bez které by tato práce nikdy nemohla vzniknout.

Anotace

Diplomová práce se zabývá využitím vývojového a testovacího prostředí Vector CANoe, zejména pak jeho programovacím prostředím CAPL Browser, za účelem realizace programu sloužícího k automatickému monitorování a vyhodnocování změn hodnot signálů na sběrnici CAN pro společnost Škoda Auto. Rešeršní část popisuje sběrnici CAN, účel jejího automatického monitorování a možné monitorovací metody se zaměřením na prostředí CANoe. Praktická část je zaměřena na vývoj programu, popis jeho funkcí a vyhodnocení funkčnosti vytvořeného programu na reálném stavu.

Klíčová slova

CAPL Browser, CAPL program, monitorování sběrnice, sběrnice CAN, Vector CANoe

Annotation

Diploma thesis deals with utilization of development and testing environment Vector CANoe – in particular its programming environment CAPL Browser in order to implement a program which would be used for automatic monitoring and evaluating of value changes of signals on CAN bus for company Škoda Auto. The theoretical part of this thesis is focused on description of CAN bus, purpose of automatic monitoring of CAN bus and possible monitoring methods focused on the environment CANoe. The practical part is focused on development of the program, description of its functions and evaluation of functionality of created program on real state.

Keywords

CAN-BUS, CAPL Browser, CAPL program, monitoring of bus, Vector CANoe

Obsah

Seznam obrázků

Seznam zkratek

Breadboard	testovací tabule (zkušební stav)
CAN	Controller Area Network
CANoe	CAN open environment
CAN-H	CAN-High
CAN-L	CAN-Low
CAPL	Communication Access Programming Language
HS CAN	High-Speed CAN
HW	hardware
LS CAN	Low-Speed CAN
PC	osobní počítač
ŘJ	řídicí jednotka
SW	software
TC Česana	Technologické a vývojové centrum společnosti Škoda Auto
TME/2	oddělení pro zkoušky funkcí a komunikace elektriky/elektroniky automobilu

Úvod

V dnešní době jsou automobily vzhledem k vyšší bezpečnosti a komfortu při jízdě vybaveny stále větším množstvím elektronických systémů (řídicích jednotek), které mezi sebou komunikují. Mezi nejrozšířenější komunikační protokol sloužící k předávání informací právě mezi těmito systémy patří CAN protokol. Data posílaná po sběrnici mohou být monitorována pomocí různých nástrojů. V Technologickém a vývojovém centru společnosti Škoda Auto (TC Česana) se mimo jiné využívá softwarový (SW) nástroj Vector CANoe.

Jako praktikant ve společnosti Škoda Auto jsem si proto jako téma diplomové práce zvolil využití tohoto vývojového a testovacího prostředí pro realizaci programu sloužícího k automatickému monitorování a vyhodnocování změn hodnot signálů na sběrnici CAN. Tento program bude možné využít např. při testování CAN komunikace v oddělení TME/2 (zkoušky funkcí a komunikace elektriky/elektroniky automobilu) TC Česana.

Cílem diplomové práce je vytvořit program v programovacím prostředí CAPL Browser, které je součástí CANoe. Tento program bude v praxi tvořit přídavný síťový uzel, tzv. Network Node, simulačního okna CANoe. V již vytvořené konfiguraci CANoe na reálném stavu bude tento uzel připojený přes určité hardwarové (HW) rozhraní na reálnou sběrnici CAN, po které komunikují i reálné řídicí jednotky (ŘJ) vozidla připojené k téže sběrnici. Následně se budou tímto SW uzlem automaticky vyhodnocovat signály každé zprávy poslané po sběrnici a zjišťovat, zda nějaký signál nepřesáhl svoji minimální nebo maximální povolenou hodnotu.

V rešeršní části této práce je nejprve popsána sběrnice CAN, její vlastnosti, topologie sítě a funkce protokolu CAN. V další kapitole je zmíněn účel automatického monitorování sběrnice CAN. Poslední kapitola rozebírá možné HW a SW nástroje, které v dnešní době umožňují monitorovat sběrnici CAN a analyzovat data, jež se po ní posílají.

V praktické části je uvedeno, za jakým účelem byl program realizován, jaké bude jeho případné využití v TC Česana. Postupně je popsána funkčnost jednotlivých částí programu a v příloze vykreslen jeho vývojový diagram. Na závěr jsou sepsány výsledky z odzkoušení programu v simulačním režimu a z otestování na reálném stavu, kdy byly signály monitorovány na reálné sběrnici testovací tabule (breadboardu) vozidla v TME/2 a nikoliv v simulačním módu v CANoe.

1 Sběrnice CAN (Controller Area Network)

Jedná se o sériový komunikační protokol vyvinutý firmou Bosch určený pro nasazení v automobilech. V dnešní době se však protokol CAN využívá i v jiných průmyslových aplikacích. Je definován normou ISO 11898, která popisuje fyzickou vrstvu a linkovou vrstvu (specifikace datového protokolu) [18], [22]. Fyzická vrstva definuje elektrické, mechanické, funkční parametry síťového uzlu. Linková vrstva definuje pravidla pro přenos dat mezi dvěma síťovými uzly [19]. V současné době jsou v automobilech desítky ŘJ, jež mají své speciální snímače a akční členy. A právě sběrnice CAN umožňuje výměnu informací mezi ŘJ tak, aby byly využity informace od všech snímačů a aby elektrická, elektronická část vozidla zůstala přehledná a nezabírala mnoho místa [20].

1.1 Výhody a nevýhody sběrnice CAN

Mezi pozitivní vlastnosti sběrnice patří nízká cena, snadné nasazení a rozšiřitelnost, vysoká přenosová rychlost až 1 Mbit/s při délce sběrnice do 40 m, rozlišení zpráv pomocí identifikátoru, prioritní přístup, možnost diagnostiky sběrnice, odolnost vůči rušení [18], [22]. Dále pak zjednodušené propojení, úsporu místa díky menším rozměrům ŘJ, snížení počtu poruch soustavnou kontrolou dat posílaných ŘJ, možnost rozšíření protokolu upravením SW, normalizování sběrnice CAN po celém světě [20].

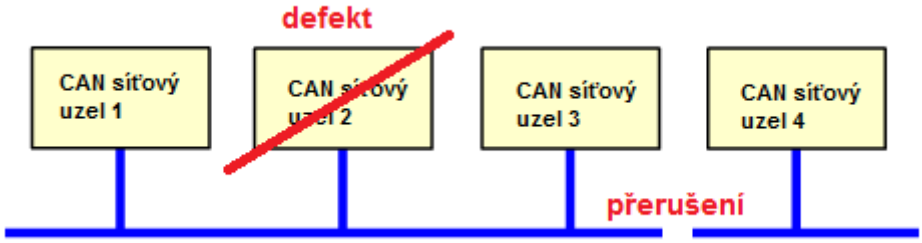
Mezi negativní vlastnosti patří omezený počet dat v rámci jedné zprávy (0 až 8 bytů), náročnost prvotního nastavení registrů sběrnice [22].

1.2 Základní vlastnosti protokolu a princip přenosu dat

Protokol CAN je typu *multi-master*. Každý uzel sběrnice může být *master* a řídit chování ostatních uzlů. Nemusí tedy existovat jeden nadřazený uzel řídicí celou síť. Řízení přístupu k médiu je řešeno náhodným přístupem a kolize je řešena prioritním rozhodováním. Jednotlivé uzly sítě mezi sebou komunikují pomocí zpráv. Zprávy nenesou informaci o cílovém uzlu, ale jsou přijaty všemi uzly v síti. Každá zpráva má však svůj identifikátor udávající její význam a prioritu. Identifikátorem je možné zajistit, aby uzel přijímal jen zprávy, které se ho týkají [18].

1.3 Topologie sběrnice CAN

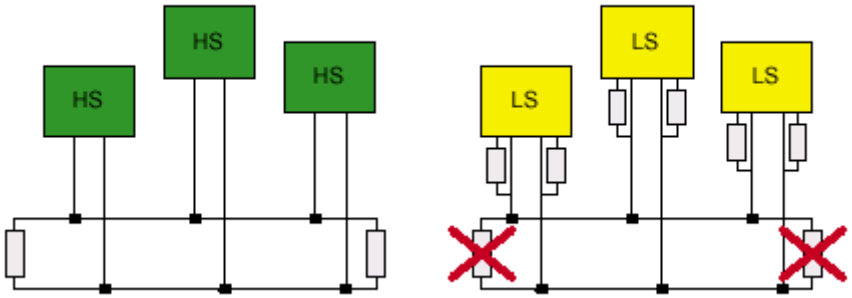
Používá se sběrnicová neboli liniová topologie. Její výhodou je snadné připojení nebo odpojení uzlu sběrnice. Pokud dojde k výpadku některého z uzlů, sběrnicová síť je stále funkční a komunikace mezi ostatními uzly může probíhat dál. V případě, kdy dojde k přerušení sběrnice, vznikají dva samostatně fungující segmenty, ale vzájemná komunikace mezi těmito segmenty je přerušena [12].



Obr. 1: Defekt síťového uzlu a přerušení sběrnice CAN [26]

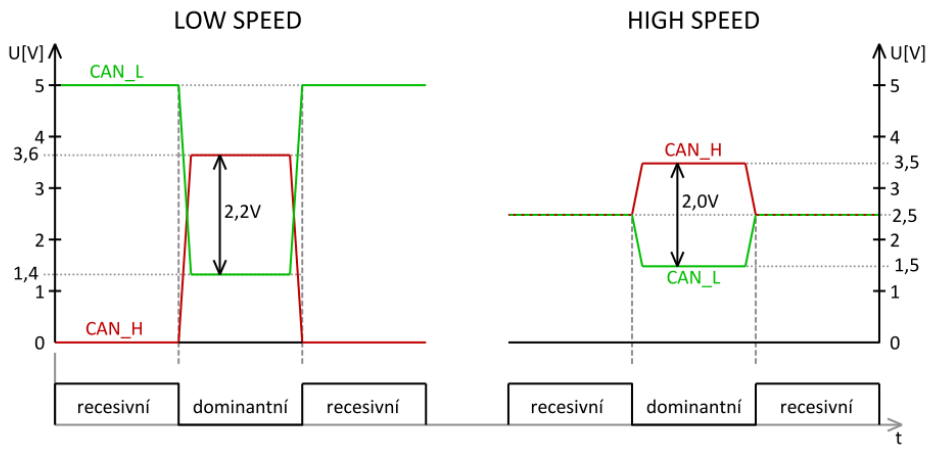
1.4 Fyzická vrstva a médium

Sběrnice CAN je tvořena dvouvodičovým krouceným vedením, tzv. kroucenou dvoulinkou. Signálové vodiče jsou označovány jako CAN-H (CAN-High) a CAN-L (CAN-Low). V případě HS (High-Speed) CAN (viz další kapitola) je vedení na obou stranách zakončeno rezistory s hodnotou 120 Ω, které zabraňují vrácení jednou poslaných dat zpět. Pokud se jedná o LS (Low-Speed) CAN, tak každé připojené zařízení má svoje zakončení. Zařízení připojená na sběrnici nazýváme uzly. Jejich počet může být teoreticky neomezený, v praxi se jich však připojuje maximálně třicet [22], [19], [2].



Obr. 2: Zakončení vedení u HS CAN a LS CAN [3]

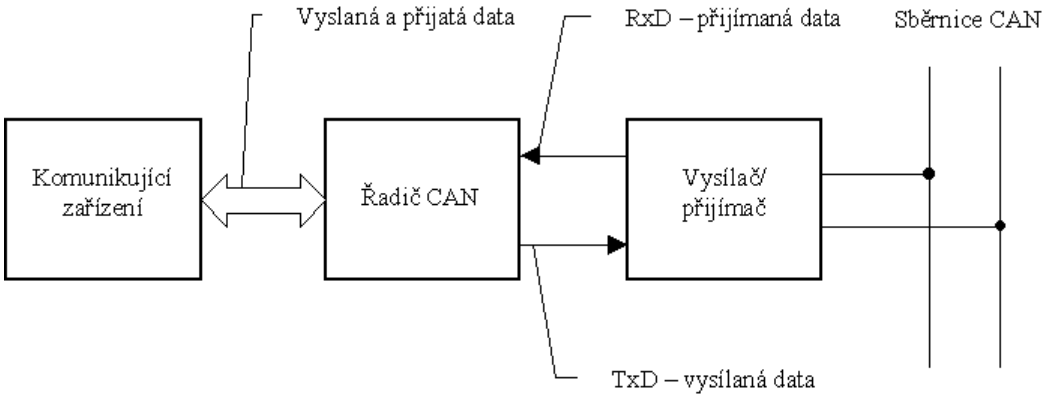
Sběrnice přenáší dva logické stavy, aktivní stav (dominantní) a pasivní stav (recesivní). Aktivní stav představuje logickou 0 a pasivní stav logickou 1. Pokud je alespoň jeden uzel v dominantním stavu, je sběrnice také v dominantním stavu. Naopak v recesivním stavu je sběrnice tehdy, jsou-li všechny uzly v recesivním stavu. V případě HS CAN se jedná o recesivní stav, je-li rozdíl napětí mezi vodiči CAN-H a CAN-L nulový. Nenulový rozdíl odpovídá dominantnímu stavu. Spínače signálových vodičů jsou realizovány tak, aby na vodiči CAN-H bylo napětí 3,5 V až 5 V a na vodiči CAN-L napětí 0 V až 1,5 V v dominantním stavu. V recesivním stavu je na obou vodičích stejné napětí, takže rozdíl je právě 0 V. Maximální rychlost přenosu dat je 1 Mbit/s při délce sběrnice do 40 m. Při délce 130 m rychlost klesá na 500 kbit/s, při 560 m na 125 kbit/s a při 3300 m je rychlost jen 20 kbit/s. Pro delší vzdálenosti sběrnice a menší přenosové rychlosti se používá LS CAN budič, kde maximální přenosová rychlost je 125 kbit/s [22], [1].



Obr. 3: Napěťové úrovně u LS CAN a HS CAN [1]

1.5 Uzel sítě sběrnice CAN

Jednotlivé uzly sítě, v automobilech tedy ŘJ, se skládají z mikropočítače, řadiče a budiče. Budič je vysílač a přijímač v jednom. Využívají se dva typy budičů. Jedná se o HS CAN a LS CAN budič. Vhodnost jejich použití plyne od požadované přenosové rychlosti. Budič mění data od řadiče na elektrické signály, které mají být poslány na sběrnici, nebo elektrické signály ze sběrnice mění na data pro řadič. Úkolem řadiče je dostávat od mikropočítače data, jež mají být poslána. Řadič připravuje a předává data budiči pro odeslání. Současně ale může řadič od budiče data získávat, aby data předal mikropočítači ke zpracování [20], [19], [1].



Obr. 4: Uzel sítě CAN [13]

1.6 Linková vrstva

Linková vrstva je rozdělena na podvrstvy LLC a MAC. LLC (Logical Link Control) slouží pro filtrování přijatých zpráv a hlášení o přetížení. MAC (Medium Access Control) provádí kódování dat, vkládá doplňkové bity do komunikace, řídí přístup uzlů k médiu s rozlišením priorit zpráv, detekuje chyby, potvrzuje správně přijaté zprávy [18].

1.6.1 Řízení přístupu k médiu

Jakmile je sběrnice volná – v klidovém stavu, libovolný uzel může zahájit vysílání. Ostatní můžou vysílat zprávu až po odvysílání zprávy právě vysílajícího uzlu. Do obsazení sběrnice vysílajícím uzlem ostatní uzly jen přijímají. Výjimku tvoří chybové rámce vysílané okamžitě po zjištění chyby kterýmkoliv účastníkem. V případě kolize, kdy zahájí vysílaní více uzlů najednou, má přednost uzel s vyšší prioritou (s nižším identifikátorem) [18].

1.6.2 Zabezpečení, detekce a signalizace chyb

Protokol CAN má několik současně aplikovaných způsobů zabezpečení. Jedná se o [18]:

Monitoring – vysílač porovnává vysílanou hodnotu bitu s úrovní na sběrnici, a pokud jsou obě hodnoty shodné, vysílač pokračuje ve vysílání. Pokud je na sběrnici jiná úroveň než odpovídá vysílanému bitu a probíhá zrovna řízení přístupu k médiu, dojde k přerušení vysílání. Přístup získá uzel s vyšší prioritou. Jestliže je rozdíl vysílané a detekované úrovně zjištěn jinde než v arbitrážním poli a v potvrzení o přijetí zprávy, je generována chyba bitu.

CRC (Cyclic Redundancy Check) kód – jedná se o poslední pole ve vysílané zprávě tvořené patnácti bity. Může se tak generovat ze všech odvysílaných bitů zprávy podle daného polynomu. Dojde-li k detekci chyby CRC libovolným uzlem, je vygenerována chyba CRC.

Vkládání bitu (bit stuffing) – pokud je na sběrnici vysíláno pět po sobě jdoucích bitů stejné úrovně, je do zprávy vložen ještě bit úrovně opačné. To slouží k detekci chyb a ke správné časové synchronizaci přijímačů uzlů.

Kontrola zprávy (message frame check) – jestliže je na nějaké pozici bitu zprávy detekována jiná hodnota, než se uvádí ve specifikaci formátu zprávy, je generována chyba rámce neboli formátu zprávy.

Potvrzení přijetí zprávy (acknowledge) – každý uzel sběrnice musí potvrdit správné přijetí zprávy. Toto platí i pro uzly, které zprávu dále nezpracovávají. Potvrzení se děje změnou bitu v poli ACK.

Každý uzel sběrnice obsahuje dva vnitřní čítače chyb udávající počet chyb při příjmu a při vysílání. Podle jejich obsahů může uzel přecházet mezi třemi stavy. Je-li uzlem generováno velké množství chyb, je automaticky odpojen. Stavy jsou následující [18]:

Aktivní (Error Active) – uzly se aktivně podílejí na komunikaci po sběrnici a jestliže je detekována chyba v přenášené zprávě, vyšle se na sběrnici aktivní příznak chyby. Tento příznak je tvořen šesti po sobě jdoucími bity v dominantní úrovni. Dojde k poškození přenášené zprávy a poruší se tím pravidlo vkládání bitů.

Pasivní (Error Pasive) – uzly se rovněž podílejí na komunikaci po sběrnici. V případě signalizace chyby však vysílají jen pasivní příznak chyby. Ten je tvořen šesti po sobě jdoucími bity v recesivní úrovni a nedojde k destrukci vysílané zprávy.

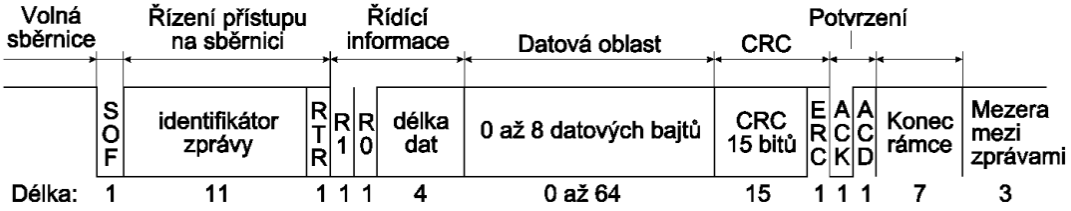
Odpojený (Buss-off) – budiče uzlů jsou vypnuty a uzly tak nemají vliv na komunikaci.

1.6.3 Typy CAN zpráv (datových rámců)

Protokol CAN definuje čtyři typy zpráv. Jedná se o datovou zprávu a zprávu s žádostí o data, ty se týkají přenosu dat. Dále chybovou zprávu a zprávu o přetížení. Tyto dvě zprávy slouží k řízení sběrnice, tedy k signalizaci chyb, eliminaci chybných zpráv a signalizaci o přetížení [18].

a) Datová zpráva (Data Frame)

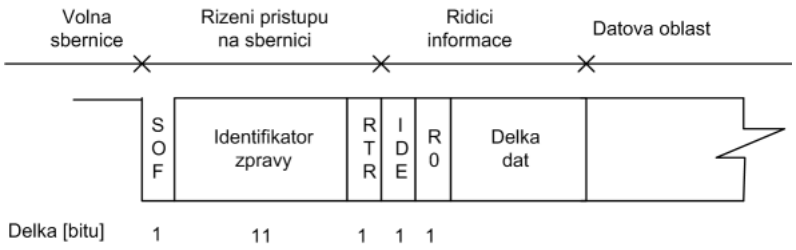
Protokol CAN užívá dva typy datových zpráv lišících se pouze v délce identifikátoru. Prvním typem je standardní formát zprávy s délkou identifikátoru 11 bitů definovaný specifikací 2.0A. Druhým typem je rozšířený formát zprávy s identifikátorem o délce 29 bitů definovaný specifikací 2.0B. Pokud řadič podporuje protokol 2.0B, mohou být na jedné sběrnici použity oba druhy zpráv, se standardním i rozšířeným formátem [18].



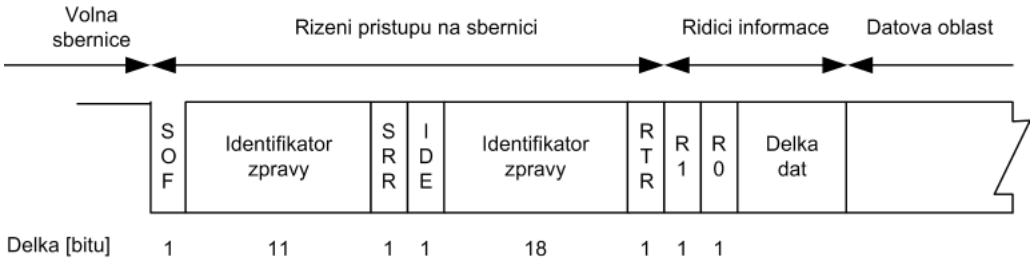
Obr. 5: Datová zpráva formátu CAN 2.0A [19]

Popis jednotlivých částí zprávy formátu CAN 2.0A je následující [19]:

- SOF (Start of frame) – udává začátek zprávy, 1 bit,
- identifikátor zprávy – určuje prioritu a význam přenášené zprávy, 11 bitů,
- RTR (Remote Transmission Request) – udává, zda jde o datovou zprávu (hodnota aktivní) nebo o žádost o data (hodnota pasivní), 1 bit,
- R1, R0 – rezervované bity,
- délka dat – udává počet přenášených datových bytů ve zprávě, 4 bity,
- datová oblast – 0 až 8 datových bytů,
- CRC – cyklický redundantní kód pro kontrolu správnosti přenosu, 15 bitů,
- ERC – ukončení CRC, 1 bit,
- ACK – potvrzení o přijetí zprávy uzlem, 1 bit,
- ACD – oddělovač potvrzení, 1 bit,
- konec rámce – v tuto dobu mohou přijímací uzly informovat vysílací uzel o chybách přenosu, 7 bitů,
- mezera mezi zprávami – 3 bity.



Obr. 6:Začátek datové zprávy (standardní formát) podle specifikace 2.0B [18]



Obr. 7: Začátek datové zprávy (rozšířený formát) podle specifikace 2.0B [18]

Jak je patrné z Obr. 5 a Obr. 6, rozdíl mezi standardním formátem specifikace 2.0A a 2.0B je pouze v bitu R1 (u 2.0A) respektive IDE (u 2.0B). V případě specifikace 2.0A je bit rezervní, ale ve specifikaci 2.0B informuje o tom, zda se jedná o standardní nebo rozšířený formát zprávy. V případě rozšířeného rámce specifikace 2.0B je bit RTR přesunut za druhou část identifikátoru a místo něj je vždy recesivní bit SRR (Substitute Remote Request). Ten zajišťuje, aby při vzájemné kolizi standardního a rozšířeného formátu zprávy (na jedné sběrnici) se shodným jedenáctibitovým identifikátorem získal přednost standardní rámec [18], [19].

b) Žádost o data (Remote Frame)

Formát této zprávy je podobný jako formát datové zprávy. Jen bit RTR není nastaven do dominantní úrovně, ale do recesivní, a není zde datové pole. Uzel žádající o zaslání dat nastaví identifikátor zprávy stejný jako má datová zpráva, od níž chce data poslat. Jestliže ve stejné chvíli jeden uzel žádá o data a jiný se stejným identifikátorem vysílá, přednost má uzel vysílající s dominantním bitem RTR.

c) Chybová zpráva (Error Frame)

Zpráva pro signalizaci chyb na sběrnici. Pokud kterýkoliv uzel detekuje v přenášené zprávě chybu, vygeneruje ihned na sběrnici chybový rámec. Uzel, jenž zjistí chybu, generuje na sběrnici aktivní nebo pasivní příznak chyby podle toho, v jakém stavu pro hlášení chyb se nachází. Při generování aktivního příznaku začnou následně vysílat i ostatní uzly chybové zprávy.

d) Zpráva o přetížení (Overload Frame)

Zpráva sloužící k oddálení vysílání další žádosti o data nebo datové zprávy. Tuto zprávu využívají uzly, které nemohou kvůli vytížení přijímat a zpracovávat další zprávy. Struktura je podobná jako u chybové zprávy [18].

2 Účel automatického monitorování sběrnice CAN

V současné době je snahou co nejvíce procesů automatizovat. Důvodem automatizace je možnost vykonávat potřebnou činnost bez nutnosti přítomnosti nebo zásahu člověka. To umožňuje minimalizovat chyby způsobené lidským faktorem, omezit stereotypní činnost, šetřit čas pro jinou paralelně vykonávanou pracovní náplň, případně úplně nahrazovat činnost lidí a zároveň tím snížit finanční náklady zaměstnavatele.

V TC Česana v oddělení TME/2 se provádí integrační testy zasíťování mezi sebou propojených ŘJ spolu s další elektronikou celého vozu. Důvodem je ověření správné funkčnosti samotné ŘJ, ale také správná funkčnost ŘJ v rámci zasíťování celého automobilu. Testy se provádí na zkušebních stavech (tzv. breadboardech), které souhlasí s reálným stavem budoucího automobilu. Kromě jiného se jedná také o testování CAN komunikace a funkcionality ŘJ. Další testy se provádějí v CAN laboratoři, kde se testují samostatné ŘJ bez závislosti na dalších účastnících na sběrnici. Do oblasti testování patří i verifikace zpráv při testování CAN komunikace vyvíjených jednotek, proto je nutné mít nástroje, které umožňují automatické monitorování sběrnice a následnou analýzu dat v reálném čase.

Obecně automatické monitorování sběrnice (v této práci sběrnice CAN) umožňuje automatický sběr dat, která jsou automaticky vyhodnocována, aniž by se musela určitá osoba na tomto sběru a vyhodnocování dat podílet. Monitorovací systém může generovat výstražná hlášení v případě, kdy se některá hodnota sledované veličiny (v tomto případě hodnoty signálu příchozí zprávy na sběrnici) ocitne mimo přijatelný rozsah. V případě tohoto hlášení je zodpovídající osoba upozorněna a musí danou chybu blíže analyzovat. Monitorovací systém tak včas upozorňuje na chyby, které by jinak nemusely být zjištěny nebo by byly zjištěny příliš pozdě. Analyzovaná data je možné také s minimálním úsilím ukládat po celou dobu automatického monitorování a poté tato data uchovat pro potřeby pozdějšího užití [11].

Data zaznamenaná (tzv. nalogovaná, natraceovaná) do jednoho souboru lze použít kdykoliv později pro zpětné „přehrání“ a analyzovaní dat původně posílaných po sběrnici. K tomu je nutné mít patřičný nástroj (viz třetí kapitola), který tento záznam přehraje. V případě testování CAN komunikace v TC Česana se pak tyto záznamy mohou ukládat k výsledkům testů, aby bylo možné při výskytu nějaké chyby tuto chybu opět prokázat, eventuálně ji podle záznamu znovu reálně navodit.

Monitorovacím zařízením mohou být např. analyzátory. Jedná se o zařízení, které se v libovolném místě a libovolném čase připojí na sběrnici CAN. Není nutné čekat na určitý stav sběrnice před připojením analyzátoru. Analyzátor pak monitoruje veškeré děje a do PC předává

výsledky k dalšímu zpracování. Data musí být předána v takovém formátu, aby šlo zrekonstruovat provoz na sběrnici včetně časových souvislostí. Na vlastnosti analyzátoru jsou kladeny následující požadavky:

- schopnost připojení analyzátoru na sběrnici za plného provozu,
- úplná implementace CAN protokolu podle specifikace CAN 2.0B,
- hodiny reálného času pro generování časové značky k jednotlivým událostem jako jsou chyby, triggery a další důležité informace pro analýzu dat,
- detekce stavových a synchronizačních chyb,
- provoz v reálném čase, kdy jsou data na sběrnici hned přenášena do PC a vyhodnocována,
- režim ukládání dat do vyrovnávací paměti,
- programovatelný výstupní trigger, jenž umožňuje generovat puls pro spuštění např. osciloskopu [15].

3 Příklady nástrojů pro monitorování sběrnice CAN v automobilech

Ve třetí kapitole jsou v jednotlivých podkapitolách uvedeny příklady nástrojů umožňujících monitorovat data na sběrnici CAN, případně i jiných sběrnicích. Jednotlivé nástroje jsou stručně popsány a uvedeny jejich základní vlastnosti. Poslední podkapitola věnovaná CANoe je popsána podrobněji z důvodu: možnosti realizovat tuto diplomovou práci v tomto nástroji, protože dobře splňuje všechny podmínky pro monitorování sběrnice a současně je CANoe hojně používána v automobilovém průmyslu (včetně TC Česana).

3.1 Dataloggery

Pokud je potřeba zaznamenávat komunikaci ŘJ (stavy signálů, průběhy napětí, proudů atd.) bez další obsluhy, je zpravidla možné použít datalogger. Jedná se o zařízení umožňující autonomně nahrávat data ve voze, zachytávat sporadické (náhodné) chyby, vyhodnocovat průběh dat za účelem monitorování chování systémů a funkcí vozu.

Dataloggery lze rozdělit do dvou základních skupin. První skupinou jsou dataloggery, které slouží (nebo jsou nastaveny) pouze jako „zapisovače“ dat. Zařízení nahrává veškerou komunikaci na sběrnicích automobilu, a to od jeho zapnutí až po vypnutí. Takto fungující zařízení však generuje velké množství dat do souboru, čímž nastává problém pro uživatele v případě následné analýzy dat. Např. hodinu trvající záznam průměrně vybavené Octavie III zabere v ASC formátu cca 810 MB. Druhou skupinou jsou dataloggery nastavené pro záznam vybraných dat. Ty jsou vybaveny jistou inteligencí umožňující pomocí odpovídající konfigurace omezovat množství zaznamenaných dat, což ulehčuje uživateli následnou analýzu. Dataloggery umožňují zaznamenávat nejen sběrnici CAN, ale např. i další typy sběrnic LIN, MOST, FlexRay, RS232. Dále také umožňují např. záznam audio/video signálů.

Mezi základní funkce používané převážně u konfigurovatelných dataloggerů patří časově omezený záznam komunikace (tzv. trace), který je omezen velikostí nadefinované paměti. Další funkcí je DriveRec. Jedná se o část uživatelem vyhrazené paměti pro dlouhodobé ukládání vybraných signálů, jež napomáhají při rekonstrukci a analýze záznamu. Některé dataloggery podporují základní diagnostiku např. mazání či vyčtení chybové paměti na podnět uživatele. Další funkcí může být statistické ukládání signálů ve formě histogramu. Mezi rozšířené funkce patří např. dálkový přenos zaznamenaných dat přes Wifi nebo využití HW jako rozhraní pro CANoe [14].

3.2 Převodník USB-CAN a software USB-CAN Adaptor

K propojení a diagnostice sběrnice CAN s PC slouží převodník *USB-CAN*. Převodník je řízen prostřednictvím rozhraní USB z aplikace *USB-CAN Adaptor*. HW řešení převodníku

je založeno mimo jiné na mikroprocesoru Atmel T89C51CC01, který má přímo na čipu integrován řadič CAN, a budiči Philips PCA82C250. Převodník je dodáván s obslužným SW.

Mezi hlavní parametry tohoto monitorovacího nástroje patří: posílání rámců specifikace CAN 2.0A i CAN 2.0B, volba komunikační rychlosti od 10 kbit/s do 1 Mbit/s, příjem a zobrazování CAN zpráv, zobrazení reálného času příjmu zpráv s rozlišením 1 ms a výpočet průměrné periody příjmu, filtrování zpráv podle identifikátoru, možnost generovat až osm zpráv najednou včetně periodického generování s periodou 1 ms až 65,5 s. Dále možnost příjmu zpráv bez potvrzení ACK, vyhledávání v seznamu přijatých zpráv, automatické vkládání popisu zpráv, zobrazování chyb a zatížení sběrnice, záznam seznamu přijatých zpráv, ochrana proti přepětí, napájení z USB konektoru [21].

K zobrazení přijatých zpráv v aplikaci *USB-CAN Adaptor* slouží okamžitý a diagnostický seznam zpráv. Okamžitý seznam zpráv vypisuje pořadí, čas příjmu, identifikátor přijaté zprávy formátu CAN 2.0A nebo CAN 2.0B, data zprávy a její uživatelský popis. Diagnostický seznam vypisuje počet a průměrný čas přijatých zpráv se shodným identifikátorem, identifikátor ve formátu CAN 2.0A nebo CAN 2.0B, data zprávy, uživatelský popis zprávy [24]. Vylepšená verze SW v2.0 přináší možnost zobrazovat i samotné signály včetně jejich fyzikálních veličin, je však nutné předem provést konfiguraci veličin prostřednictvím okna *Signals database configuration* nebo přímo při volbě veličin k zobrazení. Další funkcí je vykreslování až 15 veličin v grafu a také možnost průběžného ukládání přijatých zpráv nebo veličin do textového souboru [23].

3.3 Hardwarové a softwarové nástroje od firmy CANLAB

PP2CAN HW: Jedná se o HW zařízení, které se připojuje k paralelnímu portu PC prostřednictvím 25 pinového konektoru CANON. Slouží jako rozhraní k propojení PC se sběrnicí CAN. Pomocí diagnostického SW lze nastavit komunikační rychlosti od 10 kbit/s do 1 Mbit/s. Existují tři varianty tohoto HW: HS, LS a Single wire [5].

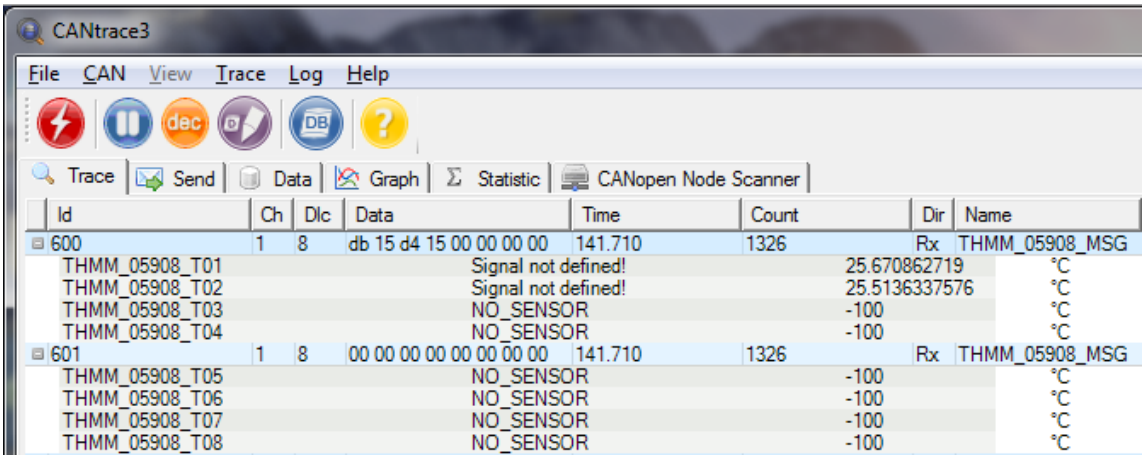
USB2CAN: Tento adaptér slouží pro připojení sběrnice CAN k PC prostřednictvím USB. Existuje ve variantě pro HS a LS CAN. Toto zařízení je inovovanou variantou převodníku *PP2CAN HW*. Pro komunikaci po USB používá obvod firmy FTDI, jako řadič obvod SJA1000. Mezi tyto dva obvody je vložen mikroprocesor PIC řady 18, ten zajišťuje např. obsluhu obou obvodů, provádí transformaci dat. Kromě odesílání, přijímání a filtrace zpráv obsahuje firmware tohoto zařízení 16 speciálních bufferů zpráv sloužících k automatickému generování těchto zpráv na sběrnici CAN. Vyrovnávací paměť pro příjem zpráv je tvořena pamětí mikroprocesoru i pamětí obvodu SJA1000. Přijaté zprávy jsou v převodníku doplněny o časovou značku, jež přesně určuje čas příchodu zprávy. Rozlišení časové značky je 250 µs. Výrobce uvádí, cituji: „Pro nejhorší variantu 29bitového identifikátoru a 8 datových bajtů je možno z PC odeslat 3600 zpráv za sekundu a minimálně stejné množství těchto zpráv paralelně přijímat.“ [5].

Obr. 8: Rozhraní USB2CAN [5]

PP2CAN SW je diagnostický nástroj sběrnice CAN. Obsahuje nástroje pro analyzování přijatých dat, generování dat na sběrnici, záznam komunikace do log souborů a zpětným „přehráním“. Tento SW je společný pro převodník *PP2CAN* i *USB2CAN* [5]. V programu je možné nastavit filtrování zpráv, vytvářet databáze (*.MSG) často používaných zpráv, generovat zprávy (periodicky v intervalu 10 ms až 10 s), graficky zobrazovat počet přijatých a odeslaných zpráv v čase, online sledovat až 10 veličin různých datových typů v různých CAN zprávách. Dále lze vytvářet DLL knihovnu, ve které se implementuje vlastní plugin, který může číst zdroje dat nebo data někam zapisovat apod. Bližší informace lze nalézt v manuálu tohoto SW [6]. Tento monitorovací nástroj však nedokáže vypisovat v jednom okně zprávy s jednotlivými signály zpráv, tak jako např. následující nástroje *CANtrace, CANoe*.

3.4 Analyzátor TKE CANtrace

Společnost TKE poskytuje analyzátor sběrnice CAN s názvem *CANtrace*. Tento nástroj umožňuje monitorovat a vypisovat CAN zprávy a signály posílané po sběrnici CAN, a to v reálném čase. Dále lze zaznamenávat (logovat) komunikaci do souboru s příponou ASC, který lze později „přehrát“ v offline režimu, a generovat zprávy na sběrnici. Analyzátor vypisuje identifikátor zpráv (ID), jejich délku (DLC), číslo CAN kanálu, samotná data zpráv, časové značky jednotlivých zpráv, zda jsou data vysílána nebo přijímána (Tx/Rx). CANtrace podporuje standardní i rozšířený identifikátor. Díky připojené databázi (*.DBC) lze dekódovat probíhající komunikaci a nezobrazovat tak pro uživatele nic neříkající surová data, ale konkrétní názvy posílaných zpráv, signálů a jejich veličin. Databázi je možné připojit pro každý CAN kanál zvlášť. Komunikační rychlost lze nastavit od 5 kbit/s do 1 Mbit/s [7].



Obr. 9: Ukázka výpisu zpráv z analyzátoru CANtrace [7]

V *CANtrace* lze zobrazovat také průběhy signálů v grafu. Grafy signálů mohou být vykreslovány i z nalogovaných souborů formátu ASC. Analyzátor *CANtrace* pracuje s HW rozhraním od společností Kvaser, Vector nebo PeakSystem. Tím je možné ušetřit finance za případné dokupování nového rozhraní pro tento SW. Připojit lze až devět rozhraní [7].

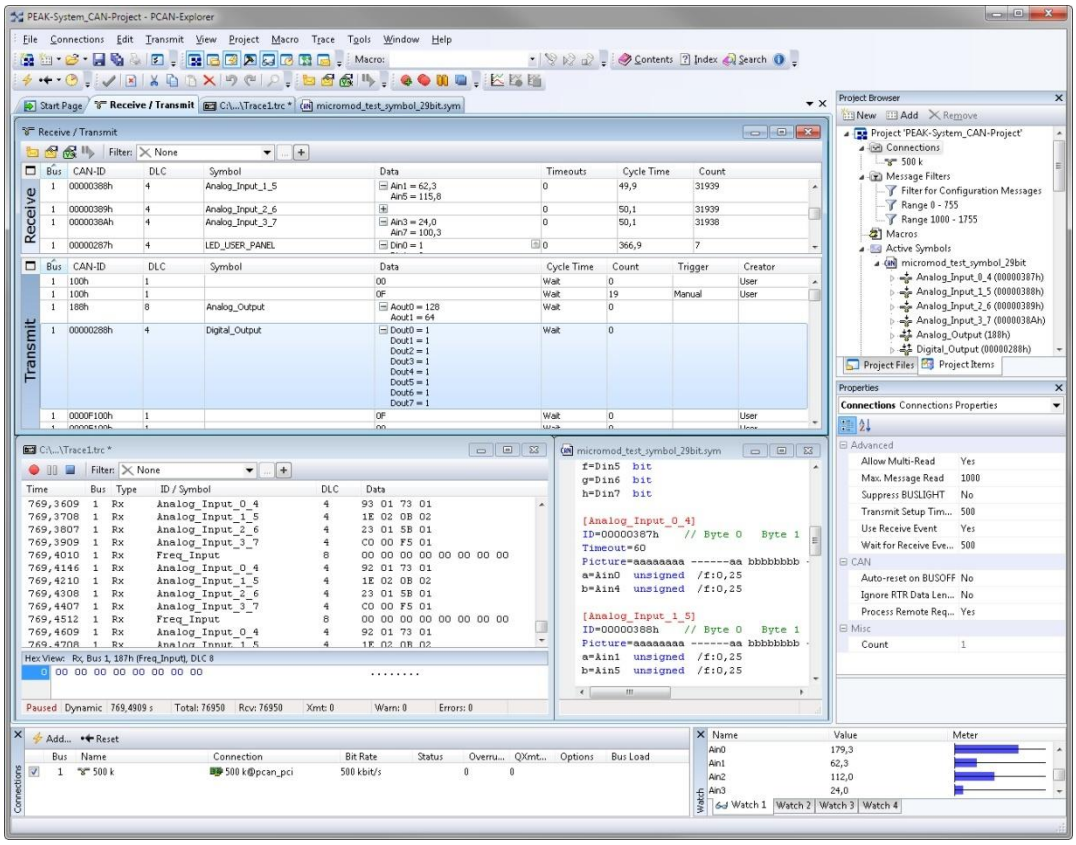
3.5 Hardwarové a softwarové nástroje od firmy PEAK System

Společnost PEAK System nabízí poměrně rozsáhlou škálu rozhraní pro propojení PC se sběrnicí CAN. Příkladem může být rozhraní *PCAN-USB* napájené přes USB, umožňující komunikaci s rychlostí od 5 kbit/s až 1 Mbit/s, s rozlišením časové značky 42 µs, s podporou identifikátoru specifikace CAN 2.0A (11 bitů) i CAN 2.0B (29 bitů), řadičem SJA1000, teplotním rozsahem −40 až 85 °C. V případě optické verze poskytuje galvanické oddělení až do 500 V mezi PC a stranou CAN. Další možností připojení na sběrnici CAN je přes *PCAN-PCI* rozhraní, které má stejné parametry, ale je k PC připojeno přes PCI slot [17].

Obr. 10: Fotografie rozhraní PCAN-USB a PCAN-PCI [17]

Jedním ze SW nástrojů pro monitorování dat na sběrnici CAN od firmy PEAK System je *PCAN-Explorer 5*. Jeho integrovaný datalogger umožňuje monitorovat, analyzovat i ukládat data do souboru (např. *.CSV). *PCAN-Explorer 5* podporuje současné připojení až 16 rozhraní CAN stejného HW, lze nastavit filtrování zpráv, odděleně monitorovat zprávy odeslané

a přijaté. U zpráv je vypisováno číslo CAN kanálu (Bus), identifikátor (CAN-ID), délka (DLC), symbolické jméno (Symbol), data (Data), čas opakování (Cycle Time) a počet výskytů (Count) zprávy. Mezi další vlastnosti patří vypisování chyb sběrnice, možnost generovat zprávy na stisk tlačítka nebo periodicky (od 1 ms), lze vytvářet makra pomocí vlastního jazyka VBScript v integrovaném textovém editoru. K tomuto SW existují ještě další doplňky (upgrady). Jedná se o *J1939 Add-in.* SAE J1939 je síťový protokol popisující komunikaci po sběrnici CAN v automobilech, obsahuje kompletní definici rozšířeného rámce CAN 2.0B. Druhým doplňkem je *Instruments Panel Add-in 3*, který umožňuje vytvářet grafické rozhraní (panely) pro grafickou reprezentaci digitálních a analogových signálů příchozích zpráv pomocí různých zobrazovacích nástrojů. Umožňuje vkládat přepínače, potenciometry ke generování zpráv apod. Další doplněk je *CANdb Import Add-in 3* pro vkládání databází (*.DBC) sloužících k zobrazování symbolických názvů. Posledním doplňkem je *Plotter Add-in 2*, který slouží k záznamu a grafické reprezentaci průběhu signálů příchozích nebo odchozích zpráv [17].



Obr. 11: Ukázka prostředí PCAN-Explorer 5 [17]

3.6 Hardwarová rozhraní od firmy KVASER

Firma KVASER nabízí kromě jiných HW výrobků také sběrnicová rozhraní pro sběrnici CAN. HW je kompatibilní s určitými SW nástroji uvedenými na webu firmy. Škála rozhraní pro sběrnici CAN zahrnuje např. *Kvaser USBcan Rugged HS*. Jedná se o rozhraní s USB 2.0 a ochranou IP65. Existuje ve variantě pro jeden nebo dva HS CAN kanály. Podporuje identifi-

kátory CAN 2.0A (11 bitů) i CAN 2.0B (29 bitů). Rozlišení časové značky zpráv je 10 µs, komunikační rychlost od 50 kbit/s do 1 Mbit/s, napájení ze strany sběrnice CAN nebo z USB. Nemá galvanické oddělení, lze číst počet chyb, generovat chybové rámce, jeho teplotní rozsah je −40 až 85 °C. Lze doručit až 12 000 zpráv/s a 6 000 zpráv/s odeslat. Dalším rozhraní je např. *Kvaser PCIcanx HS* pro vysokorychlostní CAN s galvanickým oddělením proti napěťovým špičkám, řadičem SJA1000. Do PC je připojen přes PCI slot. Jeho parametry jsou: podpora rámců specifikace CAN 2.0A i 2.0B, teplotní rozsah −40 až 85 °C, až 14 000 přijatých zpráv/s, až 18 000 odeslaných zpráv/s, čtení počtu chyb, neumožňuje generovat chybové rámce. Dalším rozhraním je bezdrátový *Kvaser BlackBird SemiPro HS* komunikující přes WLAN. Mezi jeho parametry patří: rychlost 20 až 1000 kbit/s, teplotní rozsah −30 až 85 °C, galvanické oddělení, až 15 000 přijatých a 15 000 odeslaných zpráv/s, lze číst počet chyb, generovat chybové rámce. Více informací je k nalezení v produktech firmy Kvaser [16].

Obr. 12: Rozhraní Kvaser USBcan Rugged a Kvaser BlackBird SemiPro [16]

3.7 CAN4t od firmy Electronics for transportation

Toto zařízení je dodáváno jako součást diagnostického nástroje DIAG4t. Umožňuje zjišťovat závady, které nelze detekovat běžnou diagnostikou, pracuje v reálném čase.

CAN4t je universální přenosný nástroj pro záznam dat ze sběrnice CAN, vysílání dat na sběrnici CAN a monitorování sběrnice včetně analýzy identifikátorů. Vysílání lze provádět generováním periodických zpráv nebo ze zaznamenaných dat. Monitorovat lze celý rámec zprávy nebo hodnotu fyzikální veličiny. Lze také provádět maskování identifikátorů. Komunikační rychlost může být 62,5 kbit/s až 1 Mbit/s. Mezi výhody tohoto nástroje patří bezdrátová komunikace pomocí Bluetooth s dosahem do deseti metrů (existuje však i varianta s kabelem), rozhraní HS CAN i LS CAN, ukládání dat na SD kartu až do 32 GB, záznam dat ve formátu ASC (což umožňuje i pozdější analýzu dat na PC) a podpora CAN databází (*.DBC) pro dekódování obsahu CAN rámců. Rozhraní CAN komunikuje se zařízením PDA, na kterém je nainstalovaný příslušný SW, prostřednictvím Bluetooth nebo RS-232. Pro připojení rozhraní ke sběrnici CAN slouží diagnostický konektor J1962 nebo standardní konektor Sub-D [8], [10].

3.8 Vector CANoe

CANoe (CAN open environment) je univerzální nástroj pro vývoj, testování a analýzu systémů založených na sběrnici CAN. Jedná se o používaný nástroj v TC Česana vhodný pro řešení této práce, a proto bude na následujících stránkách popsán podrobněji. Mezi základní funkce CANoe patří: simulování kompletních systémů nebo zbylých částí sběrnice, analyzování dat na sběrnici, testování celých sběrnicových sítí a/nebo jednotlivých ŘJ, diagnostika (jako s diagnostickým přístrojem), používání databází k popisu dat na sběrnici, programovací jazyk CAPL jako podpora simulování, analyzování a testování, vytváření uživatelských rozhraní pro ovládání simulací a testů nebo k zobrazování analyzovaných dat. Mezi podporované sběrnice patří např. CAN, LIN, MOST, FlexRay [31], [30].

CANoe existuje ve třech variantách, přičemž varianta FULL poskytuje plnou funkcionalitu. Dále existují varianty RUN a PEX, které už jsou funkčně ořezané. Rozdíly mezi jednotlivými verzemi lze nalézt v [28].

3.8.1 Hardwarová rozhraní pro CANoe

Jako nejlepší HW rozhraní pro nástroje od firmy Vector, jako je i CANoe a CANalyzer, lze použít produktovou řadu *VN1600*. Rozhraní *VN1610* obsahuje dva HS CAN vysílače pro dva kanály, ale jeden D-Sub9 konektor. Proto je potřeba ještě použít *CANcable 2Y* pro získání dvou samostatných kanálů (2x D-Sub9 konektor). Mezi technické specifikace patří: komunikační rychlost až 2 Mbit/s, provozní teplota −40 až 70 °C, přesnost časové značky 1 μs (pro jedno zařízení) a 30 μs (mezi více zařízeními), průměrná reakční doba 300 μs, přesné generování, detekce chybových rámců a rámců žádosti o data [25], [29].

Dalším možným rozhraním je *CANboardXL*, jež se do PC zapojuje pomocí rozhraní PCI. Technické vlastnosti jsou: dva nezávislé CAN kanály, rozlišení standardního i rozšířeného identifikátoru, provozní teplota −20 až 65 °C, detekování a generování chybových rámců a rámců žádosti o data, maximální komunikační rychlost 1 Mbit/s, přesnost časové značky 1 μs, průměrná doba odezvy 100 μs [4].

3.8.2 Fáze vývojového procesu

Vývojový proces systému je založen na modelu, který má tři vývojové fáze [31]:

1. fáze – analýza požadavků a návrh síťového systému

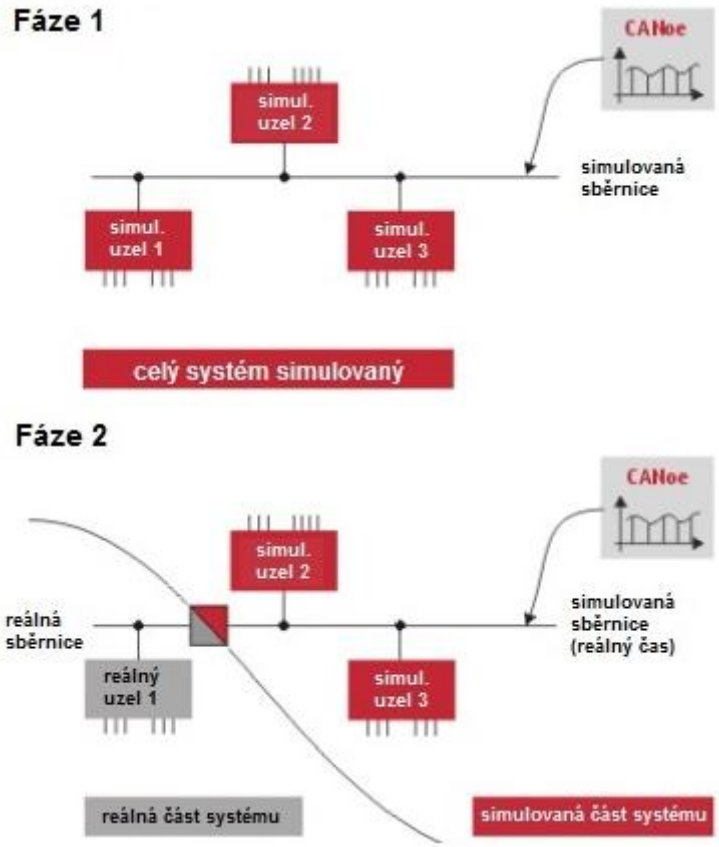
Návrh funkcionality celého systému je rozdělen mezi jednotlivé síťové uzly. Jsou definovány zprávy, které se budou po sběrnici přenášet, výběr přenosové rychlosti sběrnice. Nakonec musí být specifikováno chování jednotlivých uzlů sítě např. cyklický čas pro posílání zprávy. Pro přesnější studii celého systému je vytvořen dynamický funkční model. Ten určuje, jak se síťové uzly chovají s ohledem k vstupním, výstupním proměnným a k přijímaným, odesílaným zprávám.

2. fáze – implementace komponent a simulace zbývající části sběrnice

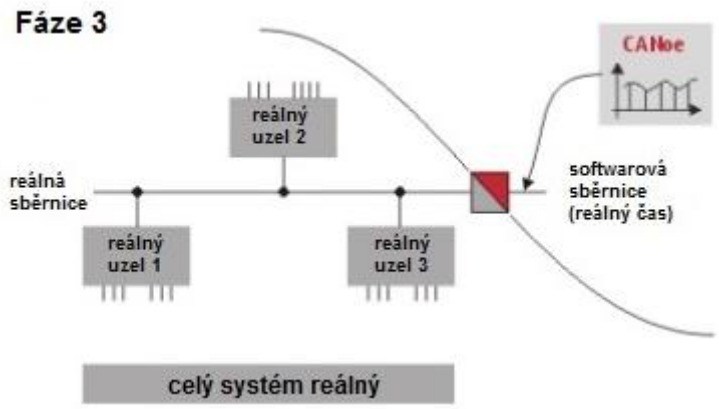
Poté, co je dokončena první fáze, se navrhují a vyvíjí jednotlivé síťové uzly samostatně, nezávisle na ostatních. Ostatní modely uzlů lze použít k doplnění simulace zbytku sběrnice. Simulace musí probíhat v reálném čase.

3. fáze – integrace celého systému

V poslední vývojové fázi jsou postupně připojeny všechny reálné síťové uzly. Proto musí být odpojeny všechny modely v simulované sběrnici. CANoe pak umožňuje sledovat zprávy mezi síťovými uzly a porovnat výsledky se specifikovanými požadavky.



Obr. 13: Fáze 1 a fáze 2 vývojového procesu [31]



Obr. 14: Fáze 3 vývojového procesu [31]

3.8.3 Seznámení s pracovním prostředí CANoe

CANoe obsahuje několik vyhodnocovacích oken (Trace, Data, Graphics, Statistics a Bus Statistics) a také okna Measurement Setup, Simulation Setup pro zobrazení toku dat a současně ke konfiguraci CANoe. V následujících odstavcích jsou popsány tyto základní okna CANoe [30], [31].

Okno **Simulation Setup** slouží pro grafické zobrazení celého systému včetně všech síťových uzlů. Simulovaná část sběrnice je reprezentována červenou čárou, reálná část sběrnice pak černou čárou. Ty jsou spojeny v jednu sběrnici přes PC kartu. Jednotlivé uzly lze aktivovat či deaktivovat podle potřeby užití.

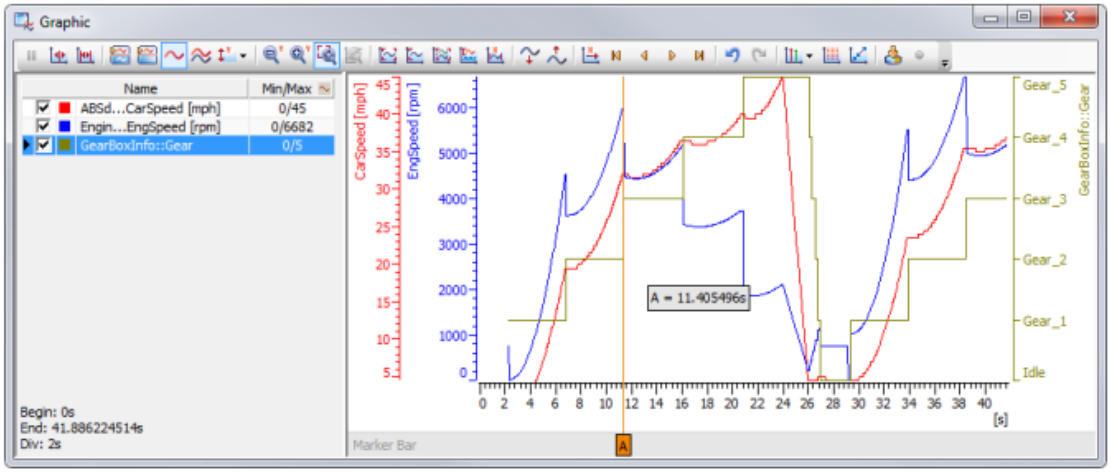
Measurement Setup se používá k analýze datového toku. Symbol „»“ naznačuje připojení k Simulation Setup a tok dat je směrem doprava ovlivňován různými funkčními bloky např. filtry, generátorem, programovými bloky CAPL. Dále zde lze definovat zdroj dat (online nebo offline ze záznamu), vkládat okna pro analýzu dat, nastavit logování dat do souboru.

Pro sledování požadovaných dat, která mohou být vyfiltrována (viz okno Measurement Setup), je určeno okno **Trace**. V okně je zobrazován čas příchozí zprávy (Time), číslo CAN kanálu (Chn), identifikátor zprávy (ID), název zprávy (Name), je-li vložena patřičná databáze, označení, zda jde o odchozí zprávu (Tx) nebo příchozí zprávu (Rx), počet datových bytů (DLC) a data zprávy. Je možné zobrazit i jednotlivé signály zpráv – jejich názvy, fyzikální hodnoty a veličiny. Okno Trace má také svůj panel nástrojů s tlačítky např. pro absolutní nebo relativní časové zobrazení zpráv, stacionární nebo rolovací vypisování zpráv.

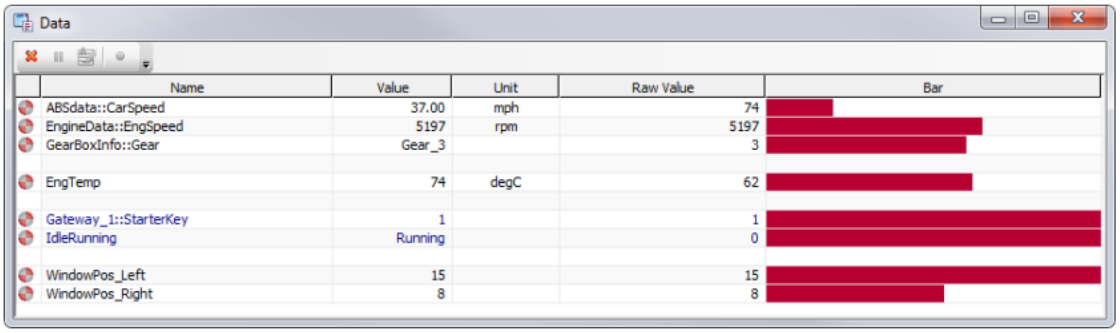
Okno **Write** vypisuje za prvé informace o zahájení nebo ukončení měření, nastavené přenosové rychlosti a další systémové informace. Za druhé vypisuje zprávy z CAPL programů.

Průběh signálů v závislosti na čase je možné sledovat v okně **Graphics**. Je-li použita patřičná databáze, je možné sledovat signály přímo ve fyzikálních veličinách, např. otáčky motoru v jednotce *rpm*, teplotu v *°C*. Okno zobrazuje legendu, lze používat kurzory pro odečet hodnot. Pro zobrazování hodnot signálů ve fyzikálních veličinách slouží i okno **Data**. Kromě názvu

(Name), hodnoty a jednotky signálu (Value, Unit) se zobrazuje surová hodnota signálu (Raw Value) a v jakém rozsahu hodnota je (Bar).



Obr. 15: Ukázka okna Graphic [31]



Obr. 16: Ukázka okna Data [31]

Okno **Statistics** zobrazuje statistické údaje o činnosti sběrnice v průběhu měření. Jedná se o histogram, kde na svislé ose je průměrný čas mezi odesláním zpráv (secs/mes) nebo počet zpráv za sekundu (msg/s) a na vodorovné ose identifikátor zpráv. Další statistickým oknem je **Bus Statistics**, které informuje o přenosu dat. Zobrazuje frekvenci dat, chybné, přetečené rámce, vytížení sběrnice, apod.

3.8.4 Databázový editor CANdb++

Pro velký počet zpráv a ještě větší počet signálů obsažených ve zprávách, přenášených po sběrnici CAN, by bylo velmi obtížné sledovat komunikaci podle identifikátorů a umístění bitů ve zprávě. Proto je vhodné použít databázi. S tímto editorem lze vytvářet a upravovat databáze (*.DBC) obsahující symbolické informace pro CANoe. Jedná se o síťové uzly a symbolické názvy zpráv, signálů a proměnných prostředí (Environment Variables). Díky použití databáze lze jednoduše sledovat komunikaci po sběrnici a stejné symbolické informace použít i v programovacím prostředí CAPL (viz další kapitola). Databázový editor umožňuje definovat:

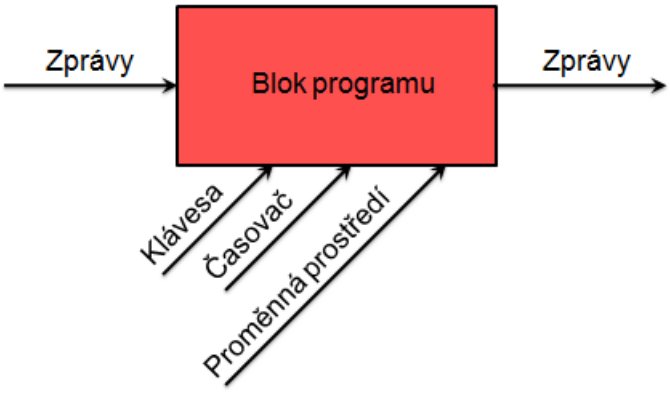
- síťové uzly, zprávy, umístění signálů v rámci zpráv,

- proměnné prostředí pro propojení s panely,
- fyzikální jednotky signálů,
- symbolické hodnoty signálů.

Zprávy jsou přenášeny po sběrnici v datových rámcích a jejich data se v CANoe zobrazují hexadecimálně nebo dekadicky. Pro zobrazení názvu zpráv a datového pole musí být zpráva definována v databázi. Data signálů jsou kódována do datového pole zpráv. Pomocí databáze jsou tzv. surová data signálů převáděna do fyzikálních veličin s příslušnými fyzikálními jednotkami. Proměnné prostředí jsou globální proměnné k propojení funkcí grafických rozhraní vytvořených v Panel Designeru a CAPL programů [9], [31].

3.8.5 Programovací jazyk CAPL a prostředí CAPL Browser

CAPL (Communication Access Programming Language) je vlastní programovací jazyk CANoe obdobný jazyku C. Umožňuje naprogramovat individuální uživatelské aplikace, např. při vývoji síťových uzlů může být uzel simulován prostřednictvím CAPL programu. CAPL uzly jsou vkládány jako funkční bloky. CAPL lze použít také pro analýzu datového přenosu nebo k naprogramování rozhraní (gateway) pro výměnu dat mezi více sběrnicemi CAN. Tento jazyk obsahuje velké množství implementovaných funkcí včetně procedur spouštěných na vnější události (Event procedures). CAPL programy mají vstup, jenž se prostřednictvím zprávy předá do bloku programu. Na výstupu jsou zprávy, které jsou programem předány dále nebo jsou poslány na sběrnici. Programový blok může reagovat na stisknutí klávesy (Key), časové události (Timer) a změny proměnných prostředí (envVar). Proto lze právě CAPL program využít i k monitorování dat na sběrnici a následné analýze dat [9], [31].



Obr. 17: Chování CAPL programu [31]

Soubory CAPL programů jsou formátu ASCII, takže je lze upravovat prostřednictvím klasického textového editoru. CANoe má však své vlastní programovací prostředí nazvané CAPL Browser. To kombinuje textový editor, kompilátor a uživatelské rozhraní. Browser zobrazuje ve strukturované formě jednotlivé proměnné, funkce a procedury událostí. Pokud je

v CANoe vložena databáze, lze v programu používat symbolické názvy vyskytující se v databázi. Pokud je při zkompilování programu detekována chyba, je kurzor umístěn na tuto chybu [9], [31].

3.8.6 Panel Designer

Tímto nástrojem lze vytvářet grafická uživatelská rozhraní (panely) umožňující měnit a zobrazovat hodnoty diskrétních i spojitých proměnných prostředí během spuštěné simulace [31]. Je tak možné např. stiskem tlačítka vyvolat poslání zprávy nebo graficky zobrazit hodnotu signálu. Lze vkládat různé typy nastavovacích prvků (např. tlačítko, přepínač, Combo Box, Check Box, Numeric Up/Down, Track Bar) a zobrazovacích prvků (např. Progress Bar, analogový ukazatel, digitální displej, textové okno). Nastavené signály musí být však vždy poslány v rámci celé zprávy a nelze měnit a posílat samotný signál.

3.9 Stručné porovnání prostředí Vector CANoe s Vector CANalyzer

CANalyzer a *CANoe* jsou univerzální multi-sběrnicové nástroje, tj. nástroje pro analyzování, simulování i několika rozdílných sběrnic současně, a to v jedné konfiguraci. Umožňují analýzu dat v online režimu, případně i ze záznamu pořízeného již dříve (v režimu offline). Oba nástroje jsou si podobné svými funkcemi. CANoe se používá zejména k vývoji. Pro samotné testování komunikace na sběrnici stačí využívat CANalyzer [28], [30].

Následně je uvedeno několik rozdílů mezi nimi:

- CANalyzer neobsahuje okno State Tracker pro grafické zobrazování systémových stavů, diskrétních hodnot a CAN rámců,
- CANalyzer neobsahuje okno Simulation Setup pro symbolické zobrazování a upravování sběrnicové sítě,
- CANalyzer neumožňuje definovat proměnné prostředí,
- spustit CAPL program v CANalyzeru lze pouze v nejvyšší PRO verzi,
- „debugging“ programu v CAPLu lze provádět jen ve FULL verzi CANoe,
- CANalyzer neumožňuje diagnostickou simulaci (Diagnostic Simulation),
- implementace vlastní knihovny DLL lze u CANanalyzeru pouze v PRO verzi,
- CANalyzer neumožňuje vkládat Simulink modely z Matlabu.

Více informací o rozdílech těchto dvou nástrojů (verze 8.2) lze nalézt v [28].

4 Postup řešení praktické úlohy – monitorování sběrnice CAN s využitím CANoe

Tato kapitola popisuje postup při řešení praktické úlohy, jejíž cíl je program reprezentovaný síťovým uzlem CANoe sloužící pro automatické online (eventuálně offline) monitorování a analyzování signálů zpráv u automobilových sběrnic CAN.

Účelem vytvoření tohoto monitorovacího programu (uzlu) je možnost automaticky sledovat komunikaci na sběrnici a vyhodnocovat hodnoty signálů, zda jsou či nejsou ve vymezeném rozsahu stanoveném v příslušné databázi zpráv. Pokud by hodnota některého ze signálů nebyla v daném rozsahu, uživatel bude přes grafické rozhraní (panel) upozorněn. Tento program by tak případně mohl být využíván v rámci testování v oddělení TME/2 TC Česana, kde by mohl sloužit jako nástroj upozorňující na možné chyby v komunikaci či zasíťování elektronických komponent.

Než bylo možné naprogramovat samotný síťový uzel, který bude schopen monitorovat jednotlivé signály zpráv posílaných po sběrnici, bylo nutné provést několik kroků popsaných v následujících podkapitolách.

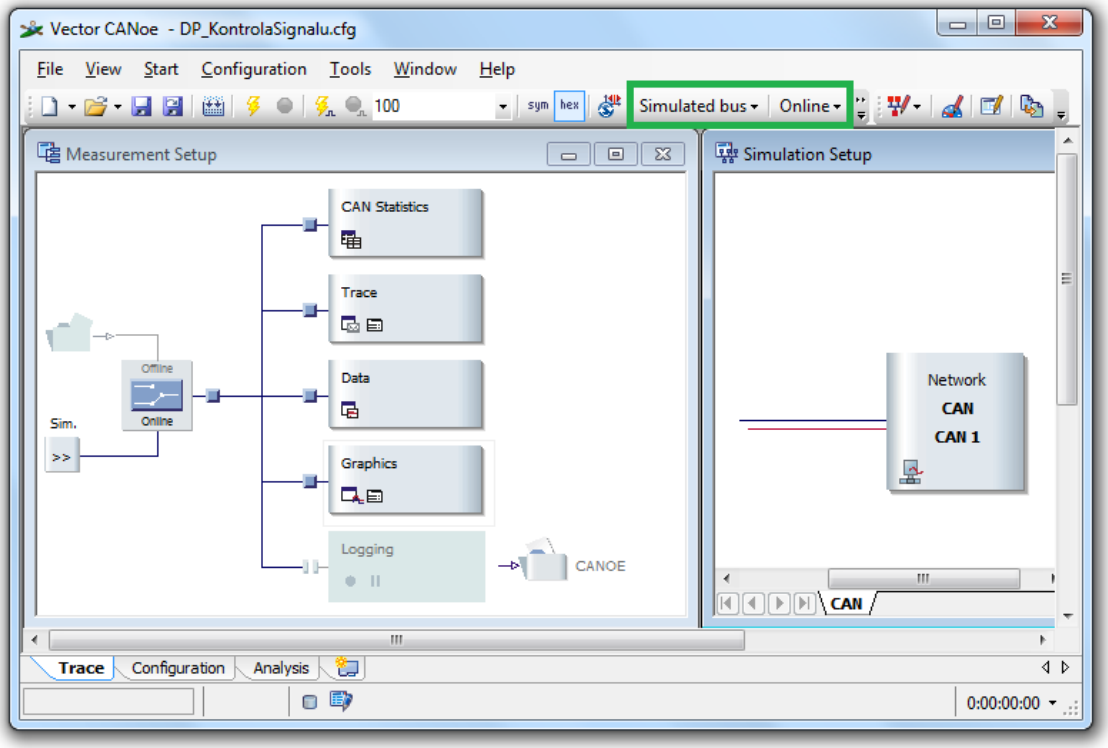
4.1 Vytvoření nové konfigurace v CANoe

Po spuštění CANoe verze 8.1 byla založena nová konfigurace obsahující standardní okna Simulation Setup, Measurement Setup, Trace, Write, a to v simulačním online režimu. To je naznačeno zeleným rámečkem na Obr. 18.

Simulační režim byl vybrán z několika důvodů. Jednak nebyla možnost neustále pracovat na reálném stavu (s připojením k reálné sběrnici). Za druhé demo verze CANoe (pro domácí použití) neumožňuje připojení k reálné sběrnici. A za třetí lze pomocí interaktivního generátoru generovat jakékoliv zprávy z přiložené databáze zpráv. Takže lze navodit i hodnoty signálů, které by na reálném stavu nemuselo jít navodit. V případě reálného režimu tvoří CANoe další síťový uzel na reálné sběrnici. Je tedy uzlem sběrnice stejně jako reálné ŘJ.

Online a offline mód lze zvolit nejen v simulačním režimu, ale i v reálném – pokud existuje reálné připojení ke sběrnici. Online mód slouží v případě, kdy uživatel CANoe chce pracovat se zprávami momentálně dostupnými na sběrnici, ať už na reálné nebo simulované. Offline mód se používá, pokud je potřeba „přehrát“ záznam (z log souboru *.ASC) komunikace na sběrnici.

Cílem této konfigurace bylo vytvořit testovací prostředí pro postupně vyvíjený monitorovací program v programovacím prostředí CAPL Browser a pro vyvíjení uživatelského panelu (grafického rozhraní) v Panel Designeru.



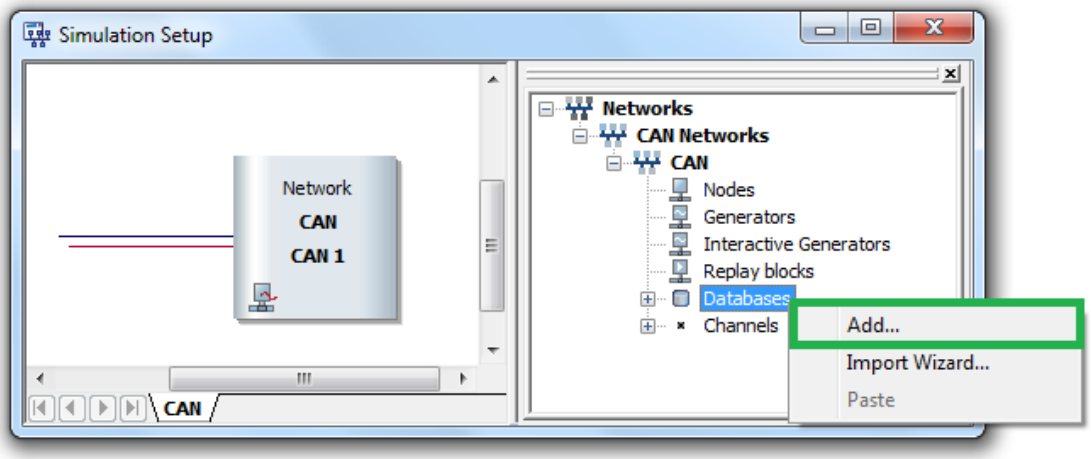
Obr. 18: Založení nové konfigurace v simulačním online režimu

4.2 Importování databáze zpráv

Dalším krokem bylo vložení databáze zpráv formátu DBC, aby bylo možné zobrazovat symbolické názvy zpráv, signálů, hodnoty signálů ve fyzikálních veličinách. Výběr vložené databáze souvisí s typem automobilové sběrnice CAN. Jestliže byla např. simulována sběrnice hnacího ústrojí (Antrieb CAN), bylo nutné vložit patřičnou databázi pro hnací ústrojí. Pokud by CANoe byla připojena k reálné sběrnici hnacího ústrojí (reálný režim) s databází např. pro komfortní CAN, tak by v CANoe nemohlo dojít k dekódování zpráv hnacího ústrojí a nebyly by zobrazovány symbolické názvy.

Jelikož obsah databází je neveřejný, je majetkem koncernu VW, nejsou v této práci zobrazovány konkrétní hodnoty identifikátorů, názvů zpráv, signálů apod.

Databáze byla vložena v okně Simulation Setup přes kontextové menu položky *Databases* → *Add*, jak je možné vidět na Obr. 19.



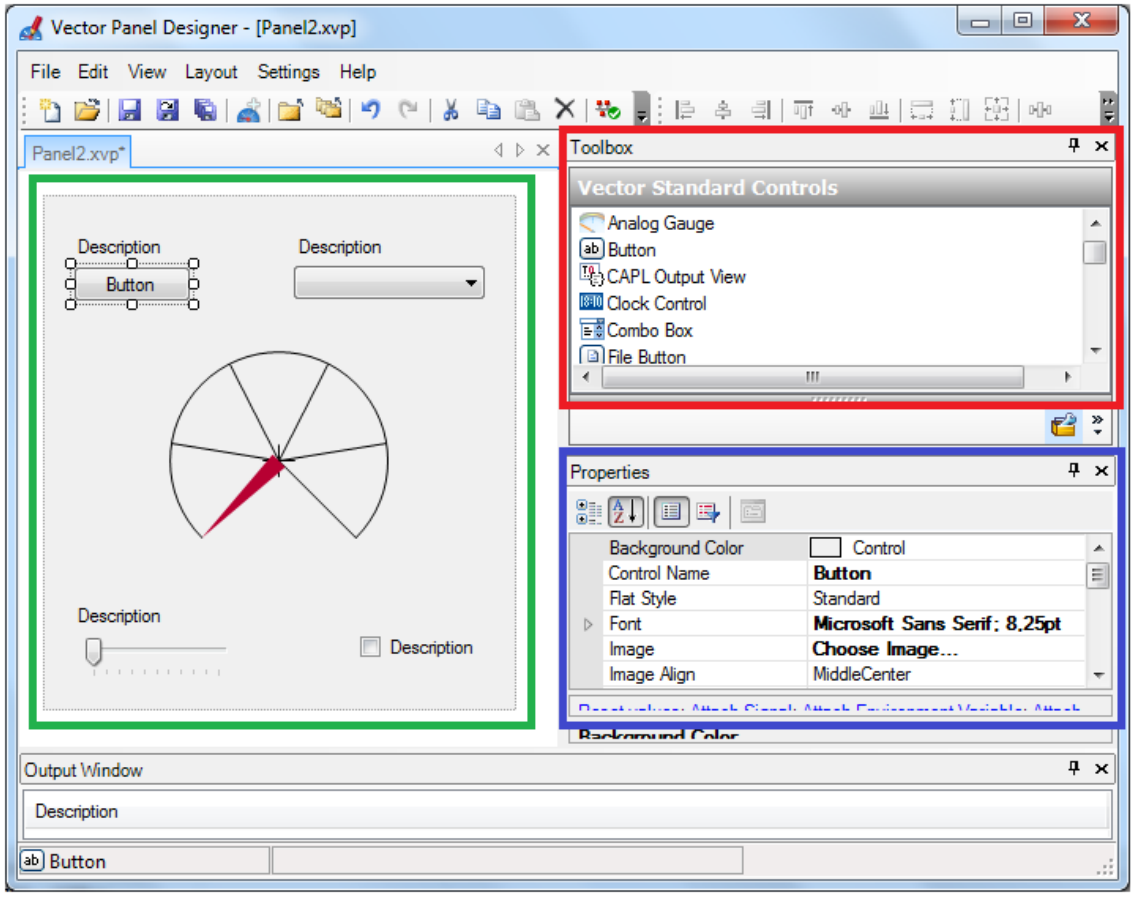
Obr. 19: Vložení databáze do konfigurace

4.3 Vytvoření panelu a databáze proměnných prostředí

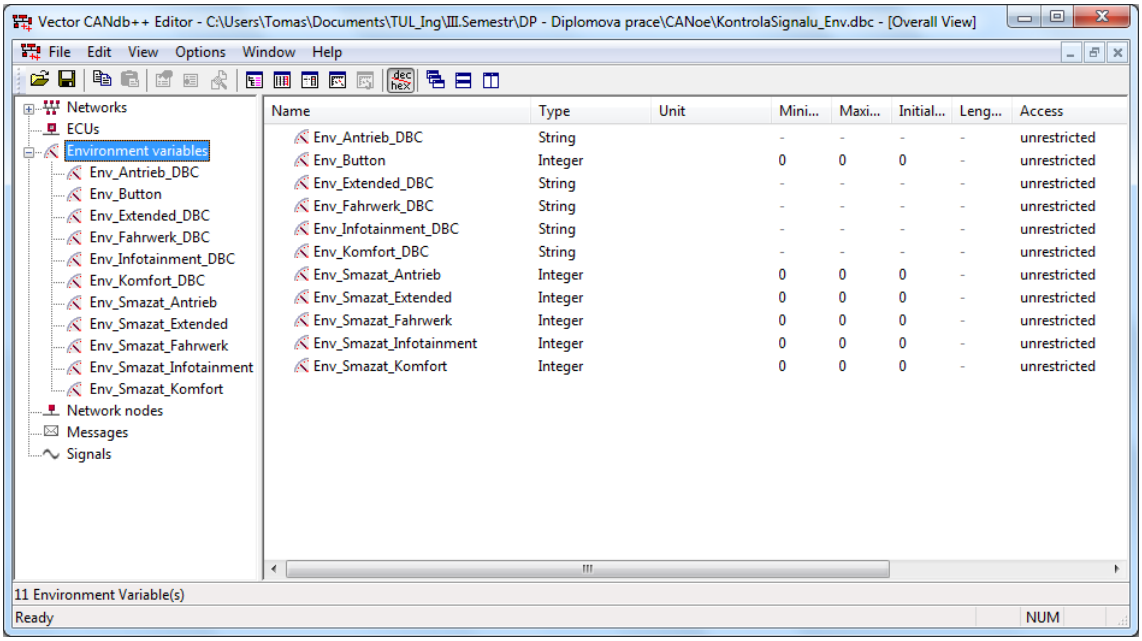
V průběhu vyvíjení programu pro automatické monitorování sběrnice (kapitola 4.4) bylo také vyvíjeno uživatelské grafické rozhraní tzv. panel. Pro vytváření panelů má CANoe svůj vlastní nástroj Panel Designer, o kterém bylo zmíněno v kapitole 3.8.6.

Panel Designer byl spuštěn přes ikonu nacházející se přímo na liště hlavního okna CANoe. Na Obr. 20 je na ukázku zobrazeno prostředí Panel Designer s několika náhodně vloženými prvky. Červený rámeček označuje okno s prvky, ze kterých bylo při tvorbě panelu vybíráno. Jedná se o vstupní (ovládací) i výstupní (zobrazovací) prvky. Modrý rámeček označuje okno, jež bylo využito při nastavování vlastností jednotlivých prvků panelu, např. velikosti, názvu, ale třeba i propojení prvku s proměnnou prostředí z databáze. To jsou globální proměnné uložené ve vlastní databázi formátu DBC a slouží jako prostřední „článek“ mezi panelem a programem. Zelený rámeček označuje uživatelem vytvářený panel.

V souvislosti s vyvíjeným panelem byla založena databáze proměnných prostředí. Přes tlačítko s ikonou byl spuštěn CANdb++ editor pro vytváření a editování databází, jak bylo uvedeno v kapitole 3.8.4. V něm byla založena nová databáze a průběžně podle potřeby definovány proměnné. Databáze byla uložena pod názvem *KontrolaSignalu_Env.dbc* a opět vložena jako předchozí databáze do konfigurace CANoe. Na Obr. 21 je zobrazena výsledná databáze. V ní se nachází proměnné sloužící pro uložení cesty ke zvolené databázi zpráv, pro sledování stavu tlačítek – mazání cesty k databázi, spouštění procesu monitorování. Využití proměnných prostředí v programu je popsáno v kapitole 4.4.1.



Obr. 20: Ukázka prostředí Panel Designer pro vytváření uživatelských panelů



Obr. 21: Vytvořená databáze proměnných prostředí

Do vytvářeného panelu byly postupně přidávány nové prvky. Pokud bylo nutné, aby monitorovací program mohl s těmito prvky pracovat, musela být danému prvku přiřazena patřičná proměnná prostředí určitého datového typu. Díky tomu mohl program přes proměnnou reagovat například na stav tlačítka v panelu.

4.3.1 Popis prvků panelu a jejich funkcionalita

Panel vytvořený v této práci slouží pro částečné ovládání programu, kdy jsou prostřednictvím určitých ovládacích prvků panelu nastavovány vstupní parametry programu přes proměnné prostředí. Zároveň vytvořené rozhraní funguje jako zobrazovací okno pro chybné signály, které nabývají hodnot mimo stanovený rozsah. Tím je uživatel upozorňován na případné problémy.

Do panelu byly vloženy následující prvky:

Tlačítka START a STOP (1) – tato tlačítka slouží pro spuštění a vypnutí celého procesu měření v CANoe. Nebylo potřeba je propojovat s konkrétní proměnnou prostředí. Stačilo je vložit jako jeden prvek s názvem *Start Stop Control* z nabídky Toolboxu. Nahrazují klasická tlačítka Start a Stop na liště hlavního okna CANoe. Byla vložena z důvodu, aby uživatel panelu nemusel přepínat mezi panelem a hlavním oknem CANoe.

Názvy databází DBC (2) – jedná se pouze o popisný text (*Static Text*) informující uživatele do jakého *Path Dialogu* mají zadávat cestu ke konkrétní databázi.

Path Dialogy (3) – tyto prvky slouží pro určení cesty k databázi, která má být použita při monitorování sběrnice. *Path Dialogy* byly nastaveny pouze pro otevírání souborů formátu DBC. Aby monitorovací program mohl zjistit, s jakou databází má pracovat, musela být ještě každému *Path Dialogu* přiřazena jedna proměnná prostředí s názvem *Env_názevCANu_DBC*. Při spuštění programu se pak zvolená cesta k databázi uloží do této proměnné, s níž program může pracovat.

Tlačítka Smazat (4) – pokud se uživatel rozhodne zvolenou cestu k databázi smazat, nikoliv přepsat, musí využít toto tlačítko, jemuž musela být také přiřazena proměnná prostředí (*Env_Smazat_názevCANu*). Tím se zajistila provázanost tlačítka panelu s programem.

Tlačítko načtení DBC a spuštění kontroly signálů (5) – toto tlačítko bylo vloženo pro povel k načtení databáze, ke které (kterým) byla vožena cesta v *Path Dialogu* a následně k zahájení kontroly hodnot signálů zpráv. Je provázáno s proměnnou prostředí *Env_Button*. Pokud uživatel nevloží do panelu cestu k databázi a stiskne tlačítko, nic se nevykoná a CANoe je pořád spuštěna.

Na Obr. 22 je vyříznuta ovládací část panelu. Zeleně jsou očíslovány jednotlivé skupiny prvků popsané výše.

Obr. 22: Ovládací část vytvořeného panelu

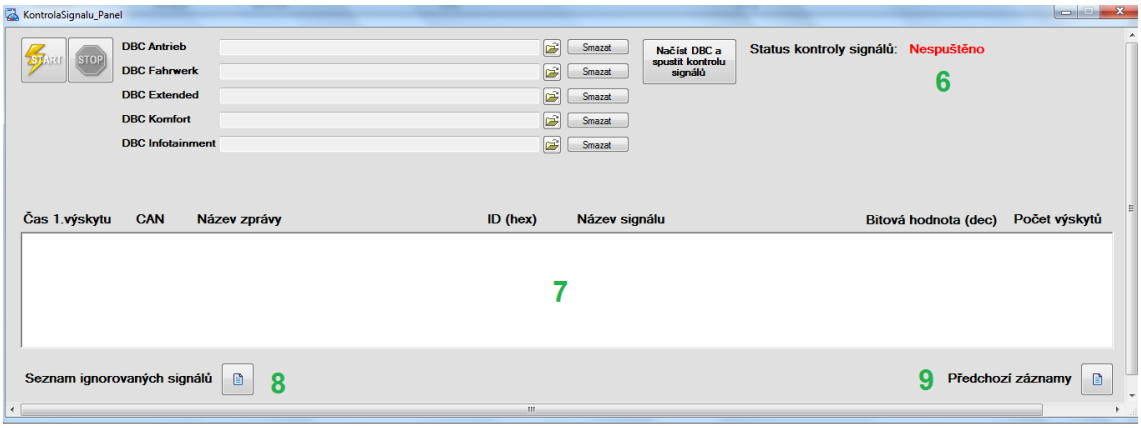
Informační text (6) – aby byl uživatel informován o stavu programu, zobrazuje se při daných podmínkách informační text, např. o načtení databáze, status kontroly signálů. Jedná se o klasický prvek *Static Text*, který je pouze podle podmínky určené programem zviditelněn či skryt ve spuštěném panelu. V Obr. 23 se jedná o text „Nespuštěno“, který je zobrazen, pokud ještě nedošlo k načtení databáze a spuštění kontroly signálů.

Zobrazovací okno (7) – pro zobrazování chybných signálů, které nabývají hodnot mimo rozsah uvedený v databázi, slouží prvek *CAPL Output View*. Pro každý signál se vypisuje sedm parametrů:

- čas prvního výskytu určitého chybného signálu,
- CAN kanál,
- název zprávy, ve které se signál vyskytuje,
- identifikátor zprávy v hexadecimálním tvaru,
- název signálu,
- bitová (surová) hodnota signálu v decimálním tvaru,
- počet výskytů daného signálu za dobu spuštění CANoe.

Seznam ignorovaných signálů (8) – aby nemusely být vyhodnocovány signály, které mají sice chybnou hodnotu, ale uživatel o nich ví a nechce je schválně vypisovat, byl vytvořen textový soubor nazvaný *Ignorovane_signaly.txt*. Na tento soubor odkazuje tlačítko panelu *File Button* s ikonou  . Pokud nebude chtít uživatel zobrazovat nějaký signál, stačí jeho jméno odpovídající jménu z databáze zpráv zapsat do tohoto souboru.

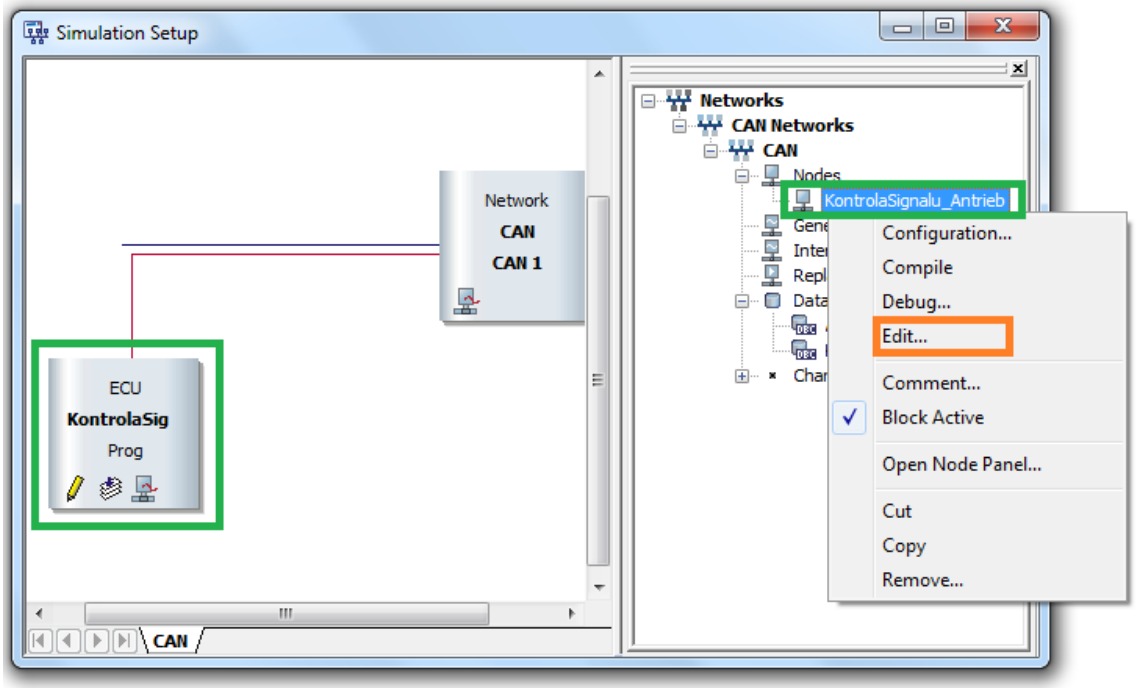
Předchozí záznamy (9) – pro uchování předchozích záznamů z monitorování sběrnice je automaticky vytvořen textový soubor s názvem *Vypis_chyb.txt*. Do něj jsou ukládány stejné údaje, jako se vypisují do panelu. Přidává se navíc pouze datum, kdy bylo měření provedeno.



Obr. 23: Panel pro monitorování chybných hodnot signálů

4.4 Naprogramování síťového uzlu pro monitorování sběrnice CAN

Pro programování síťových uzlů (Network Nodes) má CANoe svůj vlastní programovací jazyk CAPL a programovací prostředí CAPL Browser viz kapitola 3.8.5. Nejprve musel být tento síťový uzel vložen do okna Simulation Setup (na sběrnici) přes kontextové menu položky *Nodes → Insert Network Node*. Přes další kontextové menu, tentokrát samotného uzlu sítě, byla zvolena volba Edit – viz oranžový rámeček na Obr. 24. Došlo ke spuštění CAPL Browseru, ale ještě předtím má uživatel možnost pojmenovat síťový uzel a zvolit místo jeho uložení v PC. CAPL Browser lze také spustit přes ikonu  na liště hlavního okna CANoe.



Obr. 24: Vložený síťový uzel a jeho možnosti

4.4.1 Popis programu pro automatické monitorování a vyhodnocování změn hodnot signálů na sběrnici CAN

V této podkapitole jsou popsány jednotlivé funkce programu síťového uzlu. Samotný program je k nalezení na přiloženém CD. Na konci práce, Příloha B – vývojový diagram CAPL programu, je vykreslen vývojový diagram programu.

a) Definování proměnných

V první části samotného programu síťového uzlu jsou v sekci pod označením *variables* nadefinovány proměnné používané programem. Jedná se zejména o číselné proměnné, jednorozměrná i vícerozměrná pole určitých datových typů. Využití hlavních proměnných bude popsáno v další části popisu programu.

b) Funkce (event handler) *on Start*

Část programu obsažená v této funkci se provede hned po spuštění měření pomocí tlačítka START. Je to tedy první blok programu, který se vykoná.

Funkce *on Start* zobrazí v panelu status oznamující, že kontrola signálů zpráv není spuštěna. Uloží do proměnné *systemTime* aktuální systémový čas, který se poté vypisuje do textového souboru o předchozích záznamech měření (*Vypis_chyb.txt*). Dojde k vynulování několika dvourozměrných polí. Ze souboru *Nazvy_DBC.txt* se vyčte cesta k naposledy použité databázi pro danou sběrnici CAN. V tomto již existujícím textovém souboru jsou ukládány cesty k naposledy otevřeným databázím pro jednotlivé typy sběrnic CAN. To v konečném důsledku umožňuje uživateli, že nemusí po každém zapnutí nastavovat před spuštěním kontroly signálů cestu k databázi – pokud tedy nechce zvolit cestu k jiné databázi pro stejnou sběrnici. V další části této funkce je podmínka, že pokud existuje v souboru *Nazvy_DBC.txt* zapsána cesta k určité databázi, tak se tato cesta vypíše do *Path Dialogu* v panelu prostřednictvím proměnné prostředí *Env_názevCANu_DBC*. Tak je uživatel informován o existenci výchozí nastavené cesty. Nakonec je spuštěn časovač (funkce *on timer casovac*).

c) Funkce (event handler) *on timer casovac*

Tato funkce je vykonávána každých 50 ms od spuštění, dokud není časovač zastaven. V této funkci se nejprve zjišťuje cesta k databázi, konkrétně z proměnné prostředí *Env_názevCANu_DBC* do proměnné *jmenoDBC* v programu.

Pokud je následně stisknuto tlačítko panelu *Načíst DBC a spustit kontrolu signálů* (proměnná prostředí *Env_Button* = 1) a zároveň cesta k databázi obsahuje určitý počet znaků, zastaví se časovač a zavolají se funkce *LoadDBC* a *LoadIgnoredSig*.

Jestliže předchozí podmínka neplatí, vyhodnocuje se další – zda není stisknuto tlačítko *Smazat* (proměnná prostředí *Env_Smazat_názevCANu* = 1). Pokud ano, obsah proměnné prostředí *Env_názevCANu_DBC* se smaže. Tudíž v panelu se smaže také obsah pole *Path Dialogu*

(cesty k databázi) pro danou sběrnici. Dále je smazána tato cesta k databázi v souboru *Nazvy_DBC.txt*. Nakonec je resetován časovač.

Pokud ani předchozí dvě podmínky neplatí, je resetován časovač.

```
on timer casovac
{
  getValue(Env_Antrieb_DBC, jmenoDBC);
  if(getValue(Env_Button) == 1 && strlen(jmenoDBC) > 5)
  {
    cancelTimer(casovac);
    LoadDBC();
    LoadIgnoredSig();
  }
  else if(getValue(Env_Smazat_Antrieb) == 1)
  {
    putValue(Env_Antrieb_DBC, "\0");
    writeProfileString("Antrieb", "DBC Antrieb", "\0", "Nazvy_DBC.txt");
    cancelTimer(casovac);
    setTimer(casovac,50);
  }
  else
  {
    cancelTimer(casovac);
    setTimer(casovac,50);
  }
}
```

Zdrojový kód 1: Ukázka kódu funkce časovače

d) Funkce *LoadDBC*

Celá tato funkce slouží k načtení potřebných parametrů z databáze zpráv, ke které byla definována cesta pomocí *Path Dialogu* v panelu. Díky načteným parametrům z databáze může následující část programu porovnávat skutečné hodnoty signálů zpráv aktuálně posílaných po sběrnici s minimálními a maximálními hodnotami signálů definovaných v načtené databázi.

Než mohla být naprogramována tato funkce, bylo potřeba nastudovat strukturu databází (*.DBC) ve formátu TXT. Bylo totiž nutné zjistit, na jakých pozicích se v textové databázi nachází údaje, které je potřeba načíst pomocí programu z databáze do proměnných v programu. Tyto informace se daly zjistit přepsáním přípony databáze *.DBC na příponu *.TXT a otevřením přes textový editor. Další velmi užitečný materiál k prostudování byl přímo od Vectoru viz [27]. Při prvních pokusech testování programu byla načítána nejprve databáze ve formátu *.TXT. Poté bylo ale zjištěno, že pokud je vložena databáze v klasickém formátu *.DBC, tak to na funkci programu nic nemění. Vyhledávání parametrů je funkční u obou formátů.

Funkce *LoadDBC* začíná načtením obsahu proměnné prostředí *Env_názevCANu_DBC* do proměnné *jmenoDBC* v programu a následným zapsáním této proměnné do textového souboru *Nazvy_DBC.txt*. Tím je zajištěno, že uživatelem zvolená cesta k databázi v *Path Dialogu* bude uložena a při příštím spuštění programu nastavena jako výchozí.

V další fázi funkce je otevřen (v pozadí) ke čtení soubor databáze a prochází se řádek po řádku. Hledají se známá klíčová slova – nejprve pro označení zprávy. Pokud je na řádku nalezeno hledané klíčové slovo, zjistí se jeho pozice na řádku. Jestli na řádku není, pokračuje se na dalším řádku. Za tímto známým slovem se o určitý počet znaků nachází konkrétní identifikátor a název zprávy, které jsou uloženy do proměnných v programu. Podobným způsobem bylo naprogramováno další hledání klíčových slov, podle kterých se dohledají a uloží neznámé parametry do proměnných v programu. Pro všechny signály zpráv obsažené ve zvolené databázi byly do proměnných ukládány následující údaje:

- identifikátor zprávy,
- název zprávy,
- název signálu,
- multiplexor signálu,
- start bit signálu,
- délka signálu v bitech,
- faktor signálu,
- offset signálu,
- minimální hodnota signálu,
- maximální hodnota signálu.

Start bit definuje počáteční bit signálu ve zprávě a délka signálu definuje počet bitů signálu. Multiplexor je parametr, který může i nemusí obsahovat nějaký signál. V jedné zprávě může být jeden signál obsahující multiplexor „M“. Hodnota tohoto signálu pak udává, čemu se tento „M“ rovná. Podle hodnoty „M“ se posílají v rámci zprávy jiné signály závislé na tomto parametru. Tímto lze dosáhnout definice např. dvou signálů se stejným start bitem a délkou signálu, ale přitom se jedná o dva jiné signály s jinými hodnotami. Faktor a offset slouží k výpočtu hodnot signálů v příslušných veličinách ze surové hodnoty signálů.

Na konci funkce *LoadDBC* se zavře soubor databáze a zobrazí se informační texty v panelu. Konkrétně status o načtení databáze a spuštěné kontrole signálů.

e) Funkce *LoadIgnoredSig*

V této funkci se pouze otevře (na pozadí) ke čtení textový soubor *Ignorovane_signaly.txt.* Pokud není již založen, tak se tento soubor automaticky vytvoří. Následně jsou jednotlivé řádky souboru postupně zkopírovány do pole o více řádcích v programu – pokud je v souboru něco zapsané. Každý řádek totiž obsahuje uživatelem vepsaný název signálu, který nemá být vyhodnocován programem. Na konci funkce opět dojde k zavření textového souboru.

Důvod vytvoření souboru *Ignorovane_signaly.txt* byl zmíněn již v kapitole 4.3.1 – *Seznam ignorovaných signálů.*

f) Funkce (event handler) *on message **

Tato funkce programu reaguje na všechny zprávy, které jsou posílány po sběrnici. Nejprve se v této funkci nastaví do proměnné *pocetSig* (počet signálů v příchozí zprávě) nulová hodnota. Dále se do definovaných proměnných uloží číslo CAN kanálu a hodnota identifikátoru přijaté zprávy. Následuje *FOR* cyklus, ve kterém se postupně prochází jednotlivé řádky polí s uloženými parametry signálů (načtených z přiložené databáze pomocí funkce *LoadDBC*). Pokud platí podmínka, že identifikátor příchozí zprávy je roven identifikátoru v daném poli, načtou se do pole *intbyte* hodnoty všech osmi datových bytů přijaté zprávy.

Poté je zavolána funkce ***bytes_to_bin***, která slouží k převodu jednotlivých datových bytů zprávy na binární hodnotu seřazenou od 0. bitu (nultého bytu) do 63. bitu (sedmého bytu). Parametr funkce je číslo bytu, tedy postupně nula až sedm. Funkce využívá početní operaci modulo dvěma a převádí jednotlivé byty do binární formy. Bity ukládá do jednoho jednorozměrného pole *bin*.

```
void bytes_to_bin(int n)
{
  int t;
  for(t = 8*n; t < (8*n + 8); t++)
  {
    if (intbyte[n] == 0)
      bin[t] = '0';
    else if (intbyte[n] != 0 && intbyte[n] % 2 == 1)
      bin[t] = '1';
    else bin[t] = '0';
    intbyte[n] = intbyte[n] / 2;
  }
}
```

Zdrojový kód 2: Ukázka kódu funkce pro převod bytů zprávy do binární hodnoty

Po převodu se spočítá počet signálů obsažených ve zprávě. Tento počet je využit při hledání signálu s hodnotou multiplexoru „M“, pokud takový v dané zprávě existuje. Pokud ano, zavolá se funkce *find_signal*, aby se zjistila hodnota „M“ (odpovídá hodnotě signálu). Funkce *find_signal* je však volána při procházení každého řádku pole, pokud se jedná o signál z příchozí zprávy. Proto při načítání databáze byly k parametrům signálů ukládány i identifikátory zpráv, aby program mohl vyhodnotit, zda signál patří nebo nepatří do dané zprávy. Pokud by se jednalo o řádek s parametry signálu, který není obsažen v příchozí zprávě, bylo by zbytečné volat tuto funkci.

g) Funkce *find_signal*

Tato funkce je vždy volána pro každý signál příchozí zprávy, aby se vyhodnotilo, zda konkrétní signál nabývá hodnoty odpovídající definovanému rozsahu z databáze.

Nejprve se do proměnných *Startbit, Length, MinBit* a *MaxBit* načítají parametry signálu z načtené databáze: start bit, délka signálu, minimální a maximální hodnota signálu. Potom se podle start bitu a délky signálu nalezne v poli *bin* (binární hodnota celé zprávy) binární hodnota tohoto jednoho signálu a uloží do pole *charBinSignal*. Následuje převod binární hodnoty signálu do dekadického čísla ukládaného do proměnné *longDecSignal*. Při převodu se využívá funkce *_pow* pro výpočet mocniny.

```
// prevod surove hodnoty konkretniho signalu z binarniho cisla na
dekadicke
  r = 0;
  longDecSignal = 0;
  while (charBinSignal[r] != '\0')
  {
    if(charBinSignal[r] == '1')
      longDecSignal = longDecSignal + _pow(2,r);
    else if (charBinSignal[r] == '0')
      longDecSignal = longDecSignal + 0;
    r++;
  }
```

Zdrojový kód 3: Ukázka kódu funkce pro převod binární hodnoty signálu do dekadické

Další část funkce *find_signal* provádí kontrolu, jestli daný signál obsahuje multiplexor „M“ nebo jestli se nejedná o signál ovlivňovaný signálem s parametrem „M“. V druhém případě se jedná o signály, jež mají místo parametru „M“ parametr např. „m0“. To znamená, že při hodnotě signálu s „M“ rovnému nule bude posílán v rámci zprávy signál s parametrem „m0“ a nikoliv s „m1“, „m2“ apod. Pokud je splněna první podmínka, uloží se hodnota signálu do proměnné *Mul*. Jestliže platí druhá podmínka, zjišťuje se, jestli vyhodnocovaný signál má správný parametr „mX“, kde *X* značí číslo.

Hlavní část funkce *find_signal* začíná podmínkou, zda vyhodnocovaný signál přijaté zprávy není v povoleném rozsahu (tzn. mimo rozsah hodnot *MinBit* až *MaxBit*) a zároveň zda má správný parametr „mX“ nebo tento parametr vůbec nemá. Pokud není splněna podmínka, program se vrací do funkce *on message**. Pokud je podmínka splněna, pomocí *FOR* cyklu se projde seznam ignorovaných signálů a zjistí se, jestli vyhodnocovaný signál je/není v tomto seznamu. Pokud je, kontrola signálu se přeruší (konec funkce *find_signal*). Pokud není, v panelu se zobrazí status o výskytu chybných signálů a funkce pokračuje dál. V dalším vnořeném *FOR* cyklu se prochází pole s parametry signálů, které jsou již vyhodnoceny jako chybné. Pokud v daném poli chybných signálů ještě takový signál neexistuje – vyskytl se poprvé, je tento signál zapsán na konec tohoto pole s údaji uvedenými v kapitole 4.3.1 – *Zobrazovací okno*. Pokud je zjištěno, že chybný signál už se minimálně jednou vyskytl, je pouze inkrementován čítač výskytu tohoto signálu. V obou případech se pak zavolá funkce *write_to_PanelDesigner1*. Po jejím vykonání se program vrátí do funkce *find_signal*, kde následuje příkaz *break* pro návrat

programu do funkce *on message**. Tam se přejde na další signál stejné zprávy a opět se pak volá funkce *find_signal*. Takto se musí analyzovat všechny signály příchozí zprávy. Při dalším signálu zprávy se však už nemusí znovu načítat osm datových bytů zprávy, převádět data do binární formy a hledat „M“. Pouze se zavolá funkce *find_signal*.

h) Funkce *write_to_PanelDesigner1* a funkce *write_to_PanelDesigner2*

Nejprve je vymazáno pole *VypisChyb*, do kterého se ukládají chybné signály a další údaje vypisované do panelu (viz kapitola 4.3.1 – *Zobrazovací okno*).

Následně se pomocí *FOR* cyklu postupně prochází všechny chybné signály. Počet chybných signálů je uložen v proměnné *count*. Číselné parametry signálů: čas prvního výskytu signálu, číslo CAN kanálu, identifikátor zprávy, hodnota signálu a počet výskytů signálu se převádí funkcemi CAPLu *ltoa* (popř. *_gcvt*) z číselných vícerozměrných polí do jednorozměrných polí typu *char*. Název signálu a zprávy se zkopíruje s využitím funkce CAPLu *strncpy* z vícerozměrných polí typu *char* také do jednorozměrných polí typu *char*. Díky tomu může být volána funkce *write_to_PanelDesigner2*. Do té se jako parametr funkce ukládá právě jeden z vypisovaných údajů (parametr *str[ ]*) a také počet znaků (parametr *pocetznaku*), které má tvořit vypsaný údaj společně s mezerou.

```
void write_to_PanelDesigner1()
{
  int q;
  for(q = 0; q < strlen(VypisChyb); q++){
    VypisChyb[q] = 0;
  }
  for(q = 0; q < count; q++)
  {
    _gcvt(floatCas[q][0], 10, charCas); // cas
    write_to_PanelDesigner2(20, charCas);
    ltoa(intCAN[q][0], charCAN, 10); // CAN kanal
    write_to_PanelDesigner2(10, charCAN);
    strncpy(Zprava_, Zpravy[q], elCount(Zpravy[q])); // zprava
    write_to_PanelDesigner2(50, Zprava_);
    ltoa(longIDpole[q][0], charID, 16); // ID
    write_to_PanelDesigner2(15, charID);
    strncpy(Signal_, Signaly[q], elCount(Signaly[q])); // signal
    write_to_PanelDesigner2(50, Signal_);
    ltoa(longHodnoty[q][0], charHodnota, 10); // hodnota sig
    write_to_PanelDesigner2(25, charHodnota);
    ltoa(citac[q][0], charCitac, 10); // citac
    strncat(VypisChyb, charCitac, 500000);
    strncat (VypisChyb, "\n", 500000);
  }
  putValueToControl("KontrolaSignalu_Panel","ControlOutput", Vy-
pisChyb);
}
```

Zdrojový kód 4: Ukázka kódu funkce ***write_to_PanelDesigner1***

Ve funkci *write_to_PanelDesigner2* se díky CAPL funkci *strncat* postupně ukládají do pole *VypisChyb* všechny údaje potřebné k vypsání do panelu. Nejprve se do pole zapíše daný údaj a za něj takový počet mezer, aby celkový počet znaků i s údajem odpovídal parametru *pocetznaku*. Vypočtený počet mezer se ukládá do proměnné *mezera*. Takto se postupně vypisují do pole všechny údaje všech chybných signálů.

Na konci funkce *write_to_PanelDesigner1* se pomocí CAPL funkce *putValueToControl* vypíše pole *VypisChyb* do grafického rozhraní (panelu) – do zobrazovacího okna. Tím jsou v jeden okamžik zapsány do panelu všechny, v ten moment aktuální, chybné signály.

```
void write_to_PanelDesigner2(int pocetznaku, char str[])
{
  int mezera, i;
  strncat (VypisChyb, str, 500000);
  mezera  = pocetznaku - strlen(str);
  for (i = 0; i < mezera; ++i)
    strncat (VypisChyb, space, 500000);
}
```

Zdrojový kód 5: Ukázka kódu funkce ***write_to_PanelDesigner2***

i) Funkce (event handler) *on stopMeasurement*

Pokud dojde k zastavení měření pomocí tlačítka STOP na panelu, spustí se tato funkce. Status kontroly signálů se změní na „Nespuštěno“. Ostatní informační texty na panelu jsou skryty. Dále je otevřen (na pozadí) pro zápis textový soubor *Vypis_chyb.txt*. Do něj je pomocí CAPL funkce *filePutString* zapsán z proměnné *systemTime* systémový čas. Na další řádek textového souboru jsou s využitím vytvořené funkce *write_to_txtfile* vypsány nadpisy jako např. „Čas 1.výskytu [s]“ nebo „Zpráva“ atd. Následuje *FOR* cyklus podobný, jako byl použit u funkce *write_to_PanelDesigner1*. Pouze míso volání funkce *write_to_PanelDesigner2* je volána funkce *write_to_txtfile*. Předávané parametry funkce jsou však stejné. Po vypsání chybných signálů do souboru *Vypis_chyb.txt* je CAPL funkcí *fileClose* soubor zavřen.

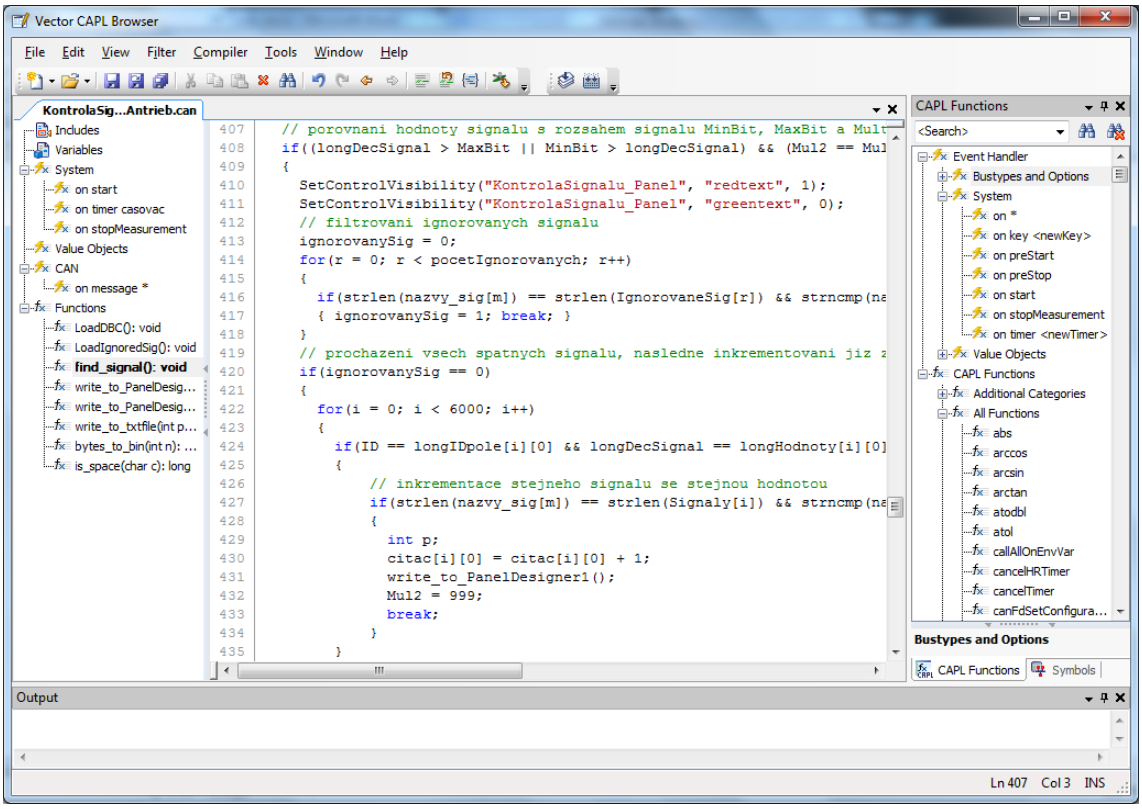
j) Funkce *write_to_txtfile*

```
void write_to_txtfile(int pocetznaku, char str[])
{
  int mezera, i;
  filePutString (str, elCount(str),writeHandle);
  mezera  = pocetznaku - strlen(str);
  for (i = 0; i < mezera; ++i)
    filePutString (space, elCount(space),writeHandle);
}
```

Zdrojový kód 6: Ukázka kódu funkce ***write_to_txtfile***

Zdrojový kód 6 zobrazuje kód funkce *write_to_txtfile*. Rozdíl oproti funkci *write_to_PanelDesigner2* je ve vypisování do textového souboru pomocí CAPL funkce *filePutString*, nikoliv vypisování do proměnné typu pole (funkce *strncat*).

Na Obr. 25 je na ukázku zobrazeno prostředí CAPL Browser s vytvořeným programem. V levé části okna jsou zobrazeny funkce popisované v této podkapitole, uprostřed část kódu programu, vpravo nabídka všech CAPL funkcí a úplně dole okno *Output* pro vypisování zpráv např. o zkompilování programu.

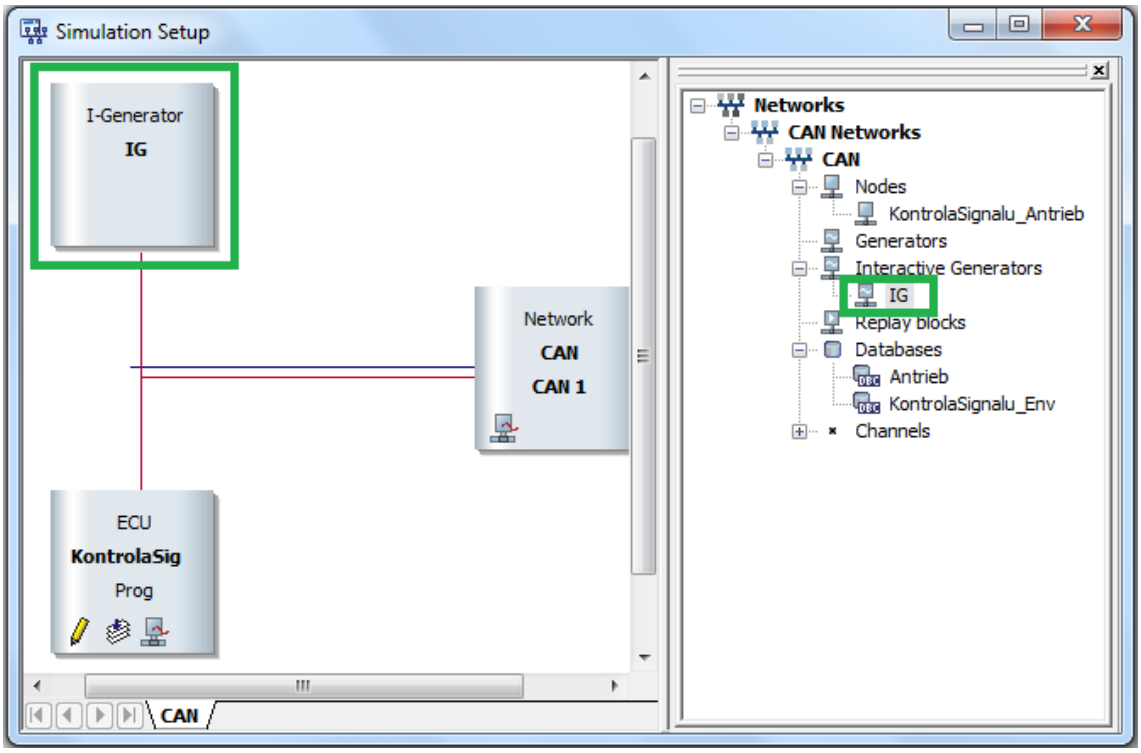


Obr. 25: Ukázka prostředí CAPL Browser s částí vytvořeného programu

4.5 Vložení interaktivního generátoru zpráv

Když byl program ve fázi, kdy dovedl načítat databázi zpráv formátu DBC a měl automaticky monitorovat signály zpráv na sběrnici, bylo nutné vložit do simulačního schéma (Simulation Setup) interaktivní generátor zpráv označovaný IG.

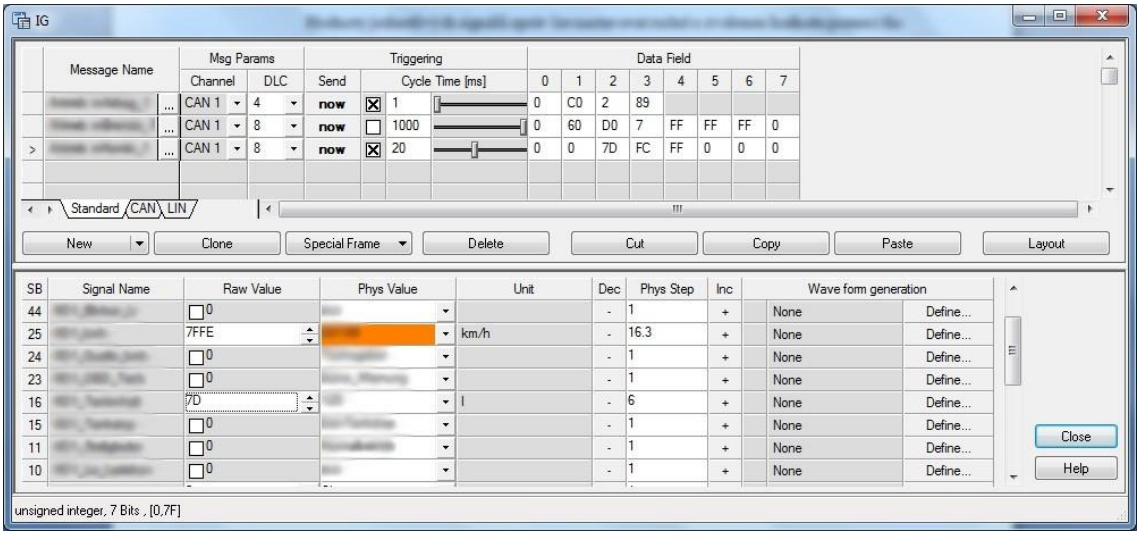
Vložit IG lze v okně Simulation Setup přes kontextové menu položky *Interactive Generators* → *Insert Interactive Generator Block*. Poté se v simulačním schéma objeví blok IG.



Obr. 26: Vložený interaktivní generátor

IG umožňuje na simulovanou sběrnici generovat libovolný počet nastavených zpráv z databáze. Tyto zprávy jsou pak analyzovány vytvořeným monitorovacím programem. IG lze použít i v případě generování zpráv na reálnou sběrnici např. při simulování ŘJ pomocí CANoe. Hodnoty jednotlivých signálů zpráv lze nastavovat ručně o zvolenou hodnotu pomocí tlačítek na inkrementování nebo dekrementování. Nebo lze hodnoty signálů měnit automaticky, např. sinusově, náhodně, lineárně. Hodnoty signálů jsou zobrazovány v surových hodnotách i ve fyzikálních veličinách. Jednotlivé zprávy lze na sběrnici posílat po stisku příslušného tlačítka *Now* nebo automaticky se zvolenou periodou v rozmezí 1 ms až 1 s.

Při testování monitorovacího programu bylo nutné vyzkoušet, zda hodnoty signálů nastavené mimo rozsah definovaný v databázi (viz oranžové políčko v Obr. 27), budou programem zachyceny a následně bude na tyto signály upozorněno prostřednictvím panelu. A protože IG umožňuje snadno nastavit a generovat signály mimo rozsah, bylo užitečné ho využít pro prvotní testování vytvořeného programu.



Obr. 27: Interaktivní generátor s hodnotou signálu mimo definovaný rozsah

5 Výsledky z otestování monitorovacího síťového uzlu

Při vyvíjení programu pro automatické monitorování hodnot signálů zpráv na sběrnici CAN byla využívána simulovaná sběrnice (simulační režim CANoe). K testování jeho funkčnosti byl pak používán interaktivní generátor zpráv (IG). Po dokončení programu bylo nutné otestovat jeho funkčnost i při reálné komunikaci na reálném stavu, tedy s připojením k reálným automobilovým sběrnicím. V obou případech byly získány výsledky popsané v dalších dvou podkapitolách.

5.1 Testování v simulačním režimu

Když byl program síťového uzlu schopen sledovat dění na sběrnici, byl pro jeho otestování začleněn do Simulation Setup okna IG, jak bylo uvedeno v kapitole 4.5. Pomocí IG byly na simulovanou sběrnici generovány zprávy s manuálně nastavenými hodnotami signálů a různou periodou generování (1 ms až 1 s). Tak, jak byl program vyvíjen, byly testovány i jeho nové funkce.

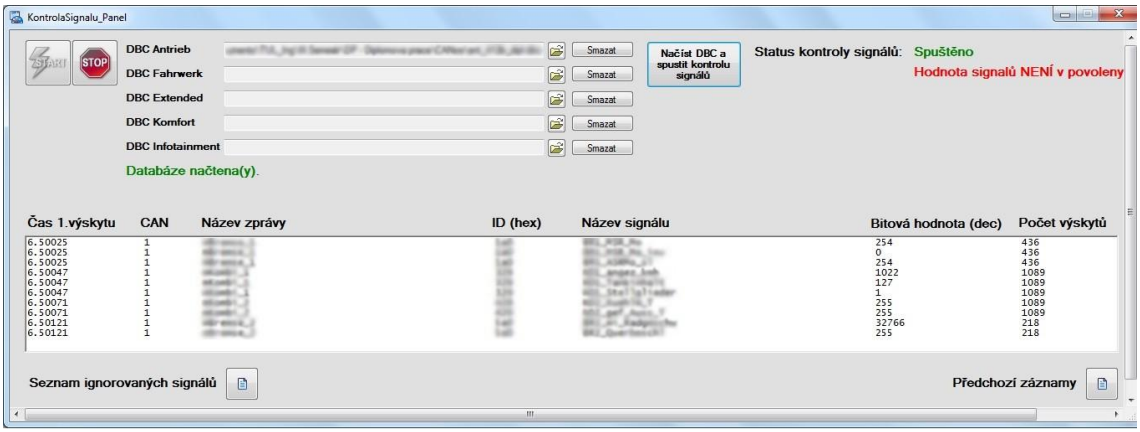
Při analyzování signálů zpráv se za prvé testovala základní funkce – vyhodnocovat hodnoty signálů a zjistit, zda jsou v rozsahu určeném databází zpráv.

Za druhé bylo testováno správné čítání chybných signálů, aby se stejný signál se stejnou chybnou hodnotou nevyskytoval ve výpisu chybných signálů vícekrát.

Za třetí byla zkoušena schopnost rozlišit signály určené multiplexorem.

Za čtvrté – poslední testovanou funkcí pro analyzování signálů byla schopnost programu ignorovat signály zapsané uživatelem do textového souboru *Ignorovane_signaly.txt*, a to i v případě, že se jedná o chybné signály.

Nakonec byl program při testování na simulované sběrnici odladěn a byl připraven k otestování na reálném stavu. Na Obr. 28 je zobrazen výpis chybných signálů při testování v simulačním módu.



Obr. 28: Ukázka panelu při spuštěném programu

5.2 Testování na reálném stavu

Testování programu na reálném stavu probíhalo na testovací tabuli (breadboardu) v oddělení TME/2, kde je nástroj CANoe také využíván. Síťový uzel tvořený monitorovacím programem, databáze proměnných prostředí a panel byly vloženy do konfigurace CANoe, která je na breadboardu používána. Konfigurace je zde nastavena v reálném módu, aby byla součástí reálných sběrnic automobilu.

Po spuštění kontroly signálů programem došlo však k „zamrazení“ CANoe, kdy do panelu nebyly vypisovány signály. CANoe byla přetížena. Až po zastavení měření tlačítkem STOP byly vypsány chybné signály. Proto byl program upraven a místo funkce *on message* * byla použita funkce *on message CAN1.** (pro sběrnici hnacího ústrojí – Antrieb CAN). Tím bylo zajištěno, že programem budou monitorovány pouze zprávy ze sběrnice Antrieb, nikoliv ostatních sběrnic Fahrwerk, Extended, Komfort nebo Infotainment. Po následném spuštění již nedošlo k „zamrazení“ CANoe a vše fungovalo správně jako v simulačním režimu. Stejně tak pro sběrnici Infotainment (*on message CAN6.**) byl program funkční. Problémy však opět nastaly při monitorování sběrnice Komfort nebo více sběrnic najednou (Antrieb + Infotainment), kdy pro každou sběrnici byl použit jeden monitorovací program. Docházelo vždy buď k „zamrazení“ nebo k takovému stavu CANoe, kdy nestíhá monitorovat všechny zprávy na sběrnici (status *lost messages*). Bylo vysledováno, že tyto dvě situace nastávají, pokud má být vypisováno do panelu více chybných signálů – při testování jich bylo 40. Pokud je vypisován menší počet chybných signálů, vše funguje správně. Funkčnost programu nezávisí na vytížení sběrnice. Při testování bylo důležité vkládat aktuální databáze zpráv. Jinak se mohou i správné signály vyhodnocovat jako chybné. Což se projevilo při testování. Kromě signálů které byly opravdu chybné v důsledku nekompletní výbavy na breadboardu, se do panelu vypisovaly i signály kvůli vložené starší verzi databáze zpráv.

Reálné použití monitorovacího programu na breadboardu je tedy možné, jestliže nebude existovat větší počet chybných signálů vypisovaných do panelu nebo pokud jejich počet bude omezen pomocí souboru *Ignorovane_signaly.txt*. Bude nutné se ještě zaměřit na optimalizování programu a eventuálně na požadavky HW PC vzhledem k funkčnosti programu – pro vypisování neomezeného počtu signálů.

Závěr

Cílem této diplomové práce bylo vytvořit program v programovacím prostředí CAPL Browser, které je součástí vývojového a testovacího nástroje CANoe, pro automatické monitorování a analyzování signálů zpráv posílaných po automobilové sběrnici CAN mezi jednotlivými ŘJ, konkrétně na breadboardu v oddělení TME/2 TC Česana.

První část byla věnována teoretickému rozboru dané problematiky, zejména sběrnici CAN a různým nástrojům pro monitorování této sběrnice se zaměřením právě na prostředí CANoe využívané také v TC Česana.

Praktická část práce byla věnována realizaci programu, tvořícího síťový uzel (Network Node) v konfiguraci CANoe, v prostředí CAPL Browser a grafického uživatelského rozhraní v Panel Designeru. Než bylo možné věnovat se samotnému programování síťového uzlu, bylo nutné vytvořit nový projekt (konfiguraci) v CANoe. Tato konfigurace, jejíž součástí byl i vytvářený program, byla nastavena do simulačního módu a sloužila jako „testovací konfigurace“ do té doby, než byl program dokončen, aby mohl být samostatně vyzkoušen na reálném stavu v již vytvořené konfiguraci. Bylo vhodné prostudovat funkce programovacího prostředí, jež by se daly při tvorbě programu použít. Při vytváření programu bylo nutné pracovat s databází obsahující zprávy a signály posílané po sběrnici CAN a jejich parametry. Proto byla nastudována struktura databáze v textovém formátu. Nakonec byl realizován samotný univerzální program umožňující monitorovat signály zpráv a zjišťovat, zda signály dosahují hodnot ve vymezeném rozsahu daném databází zpráv či nikoliv. Aby uživatel mohl online sledovat chybné signály, bylo vytvořeno v Panel Designeru grafické rozhraní (panel), ve kterém se tyto signály vypisují. Celkový výpis chybných signálů je také automaticky generován do textového souboru.

Program byl nejprve vyzkoušen a odladěn v „testovací konfiguraci“ a poté otestován v konfiguraci CANoe používané na reálném stavu. Zde byl program také funkční, a je tedy možné ho využívat jako přídavný síťový uzel CANoe v rámci testování na reálném stavu v oddělení TME/2. Jediné omezení zatím nastává při vypisování většího počtu chybných signálů do panelu, jak je uvedeno v kapitole 5.2. Na počet chybných signálů má vliv také správná databáze zpráv a odpojené elektronické komponenty na testovací tabuli.

Přínosem této práce pro mne z praktického hlediska bylo prohloubení znalostí při práci v prostředí CANoe (zejména v CAPL Browseru), které se značně využívá v automobilovém odvětví. Dále práce s databázemi zpráv a vlastní analýza problémů vyskytnutých při řešení práce. Byly tak splněny jednotlivé cíle práce. V budoucnu bude vhodné optimalizovat vytvořený program a případně se zaměřit na nároky HW PC pro tento program.

Seznam použité literatury

[1] BECK, Tomáš. *Měřicí zařízení s rozhraním CAN*. Plzeň, 2013. Diplomová práce. Západočeská univerzita v Plzni. Vedoucí práce Kamil Kosturik.

[2] CAN Physical Layer Standards: High-Speed vs. Low-Speed/Fault-Tolerant CAN. *National Instruments* [online]. 2002, 2012 [cit. 2014-05-02]. Dostupné z: http://digital.ni.com/public.nsf/allkb/84210794086E9C0886256C1C006BE6AE

[3] CAN (Controller Area Network). *CARBUSSYSTEMS* [online]. [2002] [cit. 2014-05-02]. Dostupné z: http://www.carbussystems.com/CAN.html

[4] CANboardXL. VECTOR INFORMATIK GMBH. *Vector* [online]. 2014 [cit. 2014-05-02]. Dostupné z: http://vector.com/vi_canboardxl_en.html

[5] CANLAB S.R.O. *CAN-LAB* [online]. [cit. 2014-05-02]. Dostupné z: http://rs.canlab.cz/?q=cs

[6] CANLAB S.R.O. *Diagnostický SW PP2CAN: Uživatelský manuál v1.10 CZ*. 2013.

[7] CANtrace - CAN Bus Analyzer. *TKE* [online]. [2012] [cit. 2014-05-02]. Dostupné z: http://www.tke.fi/can-analyzing-tools/cantrace

[8] CAN4t. *Electronics for transportation* [online]. 2010 [cit. 2014-05-02]. Dostupné z: http://www.e4t.cz/Vyrobky/CAN4t.aspx

[9] DUŠEK, Pavel. *Simulace kombiinstrumentu automobilu Škoda Superb*. Liberec, 2008. Diplomová práce. Technická univerzita v Liberci. Vedoucí práce Jan Koprnický.

[10] ELECTRONICS FOR TRANSPORTATION S.R.O. *Product Information: CAN4t*. Praha. Dostupné z: http://www.e4t.cz/Download/can4t/CAN4tProduct_info_S2_EN.pdf

[11] HIGHTON, Roger a Cyrus KELLY. Význam diagnostiky průmyslové sběrnice pro operátora. *Automa: časopis pro automatizační techniku* [online]. Praha: FCC Public, 2006, č. 07 [cit. 2014-05-02]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=31234

[12] CHÁRA, Tomáš. *Elektricky polohovatelná sedačka vozidla Škoda Superb*. Liberec, 2012. Semestrální projekt. Technická univerzita v Liberci. Vedoucí práce Jan Koprnický.

[13] JANDA, Petr. Měřicí a řídicí systémy se sběrnicí CAN. In: *Počítače v měření, diagnostice a řízení: sborník XXI. semináře ASŘ '98, Ostrava 22. květen 1998* [online]. Ostrava:

Vysoká škola báňská - Technická univerzita Ostrava, Strojní fakulta, 1998 [cit. 2014-05-02]. ISBN 80-7078-559-4. Dostupné z: http://akce.fs.vsb.cz/1998/asr98/Sbornik/janda/janda.htm

[14] JAREŠ, Tomáš. *Představení funkcí Datenloggerů*. Mladá Boleslav, 2013.

[15] KNÍŽEK, Roman. *Analyzátor sběrnice CAN*. Praha, 2002. Diplomová práce. České vysoké učení technické.

[16] KVASER INC. *KVASER: Our Products* [online]. 2014 [cit. 2014-05-02]. Dostupné z: http://www.kvaser.com/products-services/our-products/

[17] PEAK-SYSTEM TECHNIK GMBH. *PEAK System: PRODUCTS* [online]. 2014 [cit. 2014-05-02]. Dostupné z: http://www.peak-system.com/Products.57.0.html?&L=1

[18] POLÁK, Karel. Sběrnice CAN. *Elektrorevue* [online]. 2003 [cit. 2014-05-01]. Dostupné z: http://www.elektrorevue.cz/clanky/03021/index.html

[19] SEMENEC, Pavel a Lubomír NOVÁK. Funkce protokolu CAN. *Simple CAN Node* [online]. 2002 [cit. 2014-05-01]. Dostupné z: http://simple_can_node.sweb.cz/funkce.htm

[20] ŠKODA AUTO A.S. *Dílenská učební pomůcka 24: OCTAVIA, Datová sběrnice CAN-BUS*. Mladá Boleslav.

[21] TARABA, Radek. Adaptér USB-CAN. *HW.cz* [online]. 2004 [cit. 2014-05-02]. Dostupné z: http://www.hw.cz/produkty/adapter-usb-can.html

[22] TARABA, Radek. Aplikování sběrnice CAN. *HW.cz* [online]. 2004 [cit. 2014-05-01]. Dostupné z: http://www.hw.cz/navrh-obvodu/rozhrani/aplikovani-sbernice-can.html

[23] TARABA, Radek. Novinky v USB-CAN V2.0. *HW.cz* [online]. 2005 [cit. 2014-05-02]. Dostupné z: http://www.hw.cz/produkty/novinky-v-usb-can-v20.html

[24] TARABA, Radek. USB–CAN adaptor V1.2. *HW.cz* [online]. 2004 [cit. 2014-05-02]. Dostupné z: http://www.hw.cz/navrh-obvodu/rozhrani/usb/usb-can-adaptor-v12.html

[25] Technical Data VN1600. VECTOR INFORMATIK GMBH. *Vector* [online]. 2014 [cit. 2014-05-02]. Dostupné z: http://vector.com/vi_vn1600_en.html#!vi_vn1600_portal_tdata_iframe_en.html

[26] Topologie des CAN-Busses. *Original Marken Partner* [online]. [2013] [cit. 2014-05-02]. Dostupné z: http://www.original-marken-partner.de/?page_id=319

[27] VECTOR INFORMATIK GMBH. *DBC File Format Documentation*. 2007.

[28] VECTOR INFORMATIK GMBH. *Feature Matrix CANoe 8.2 and CANalyzer 8.2*. Německo, 2014. Dostupné z: http://vector.com/portal/medien/cmc/datasheets/CANoe_CANalyzer_FeatureMatrix_DataSheet_EN.pdf

[29] VECTOR INFORMATIK GMBH. *Manual VN1600 Interface Family: Version 1.7*. Německo, 2012. Dostupné z: http://www.vector.com/portal/medien/cmc/manuals/VN1600_Interface_Family_Manual_EN.pdf

[30] VECTOR INFORMATIK GMBH. *Product Information CANoe*. Německo, 2013. Dostupné z: http://vector.com/portal/medien/cmc/info/CANoe_ProductInformation_EN.pdf

[31] VECTOR INFORMATIK GMBH. *User Manual CANoe: Version 7.5*. Německo, 2010. Dostupné z: http://www.vector.com/portal/medien/cmc/manuals/CANoe75_Manual_EN.pdf

Příloha A – obsah přiloženého CD

Text diplomové práce

- DP_2014_Tomas_Louc.pdf
- DP_kopie_zadani_2014_Tomas_Louc.pdf

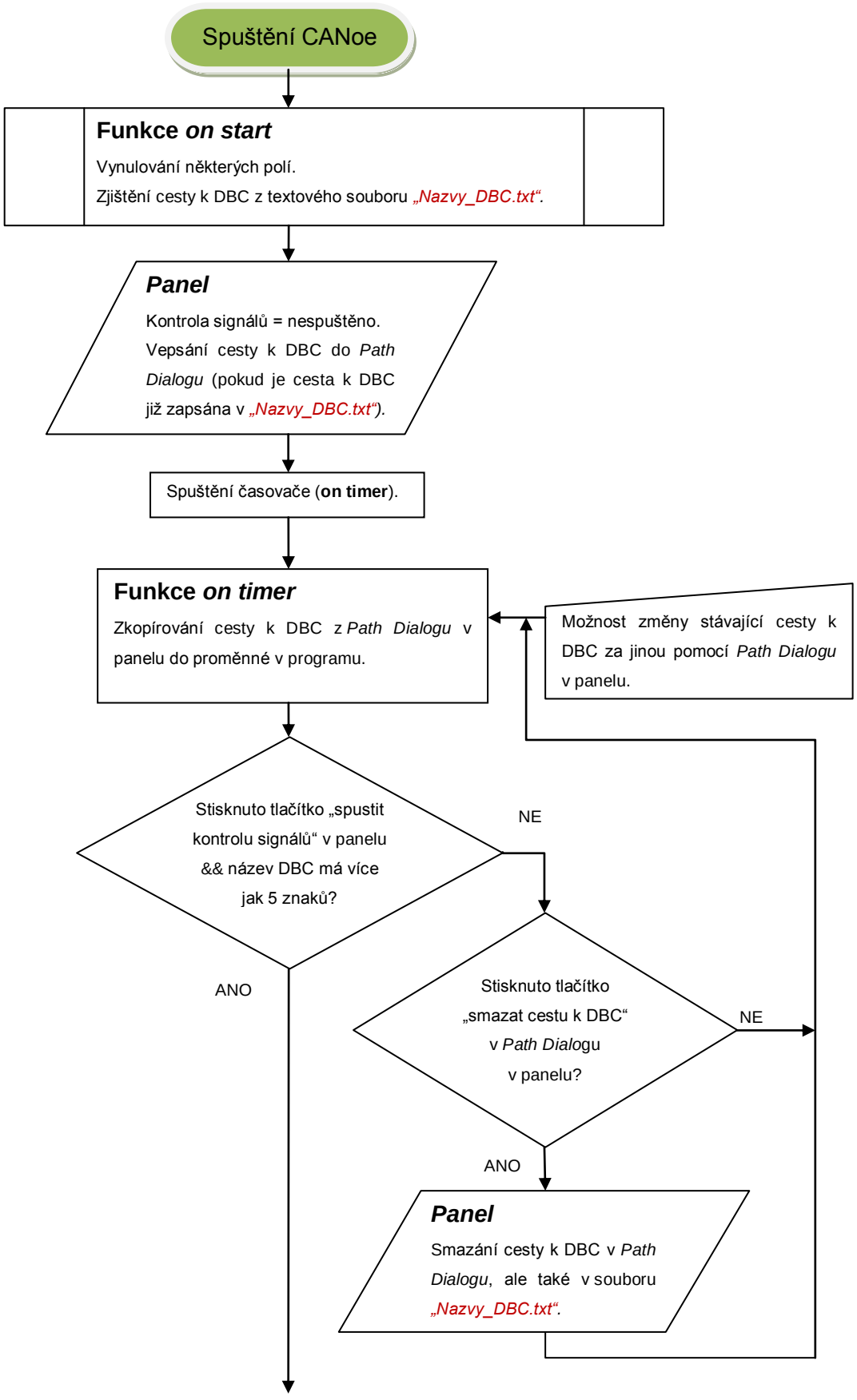
Soubory CANoe

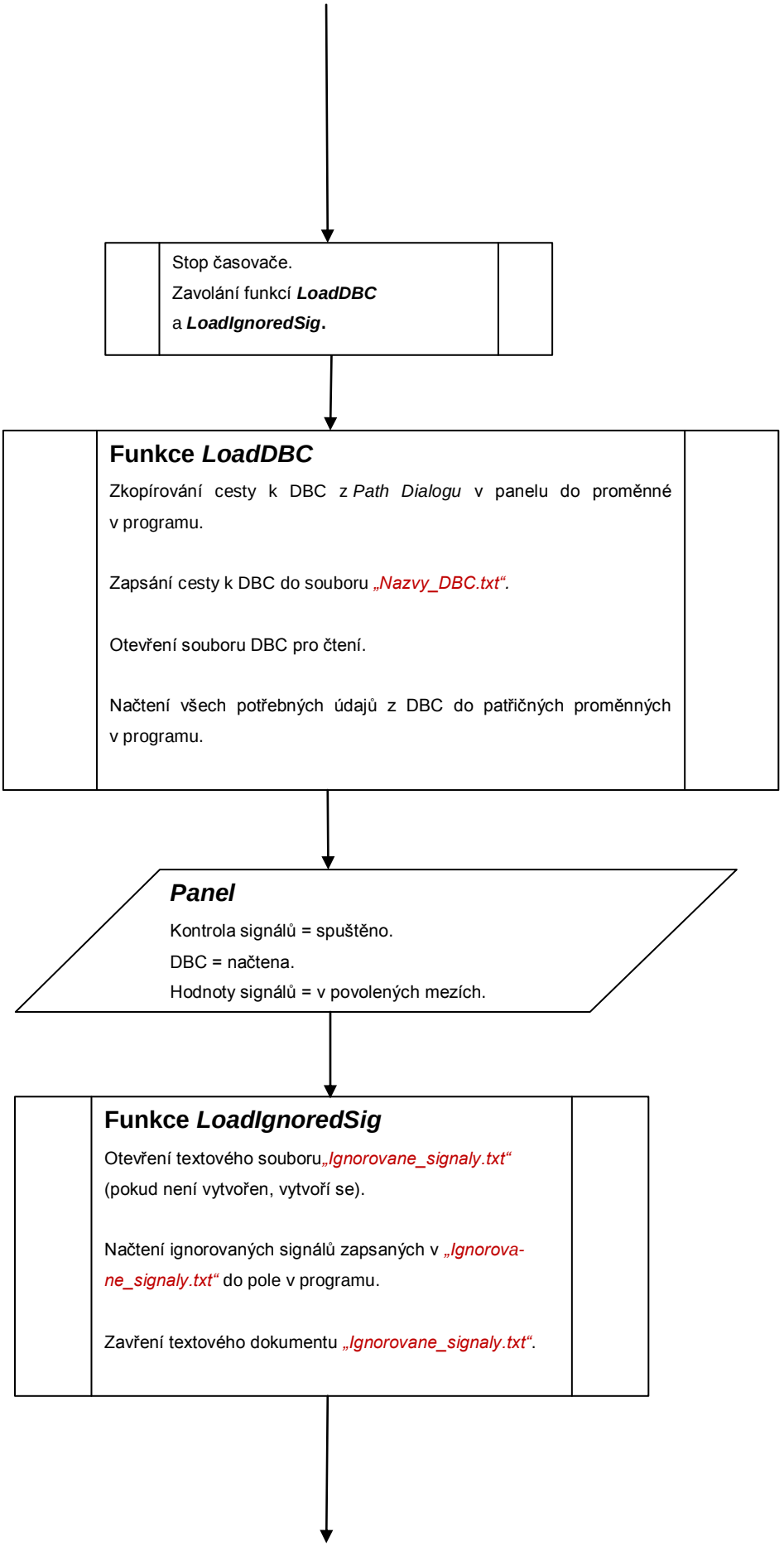
- KontrolaSignalu_Antrieb.can (program v programovacím jazyce CAPL)
- KontrolaSignalu_Panel.xvp (grafické rozhraní pro uživatele)
- KontrolaSignalu_Env.dbc (databáze proměnných prostředí)

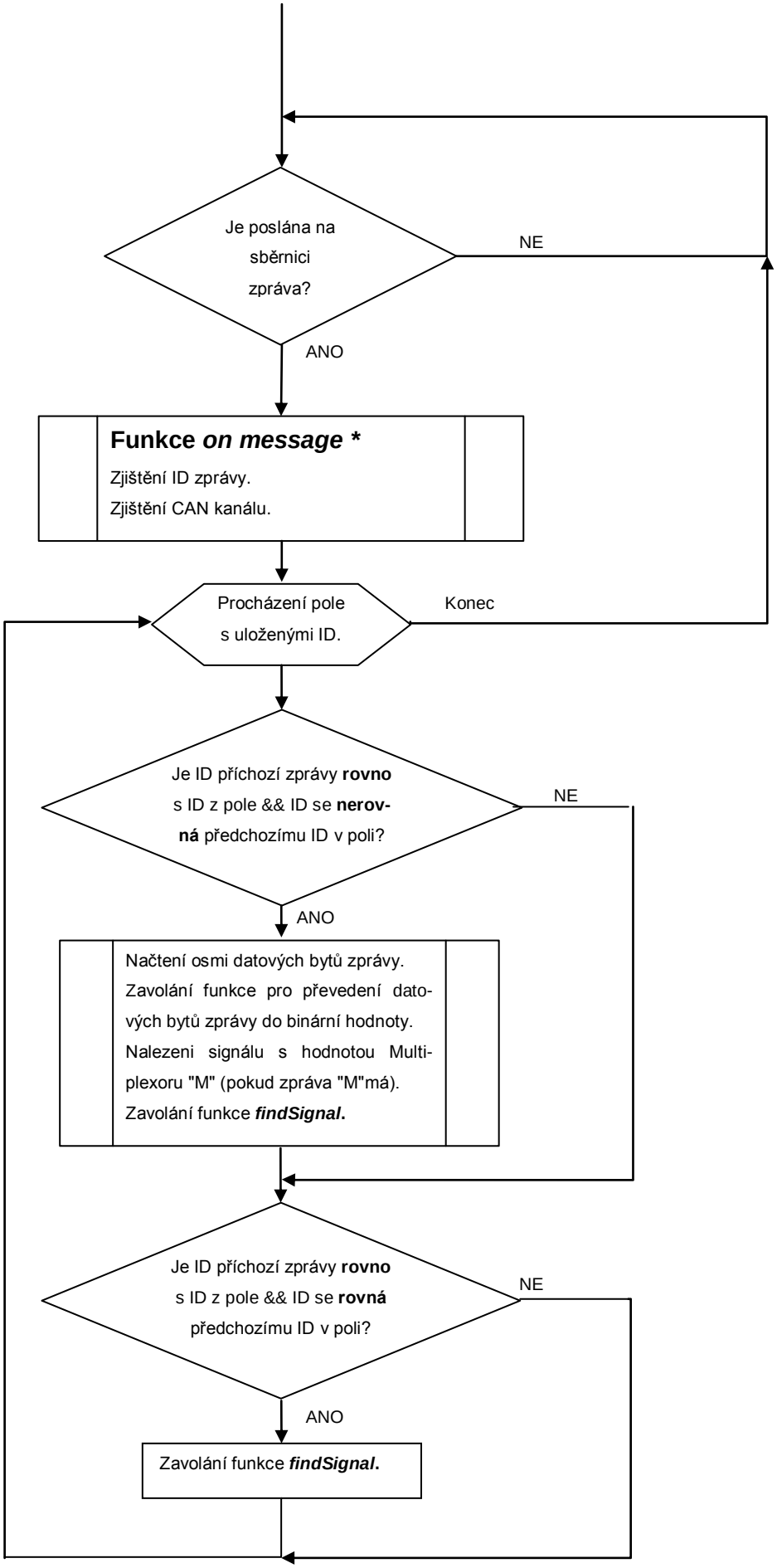
Vývojový diagram programu CAPL

- Vyvojovy_diagram_CAPL.pdf

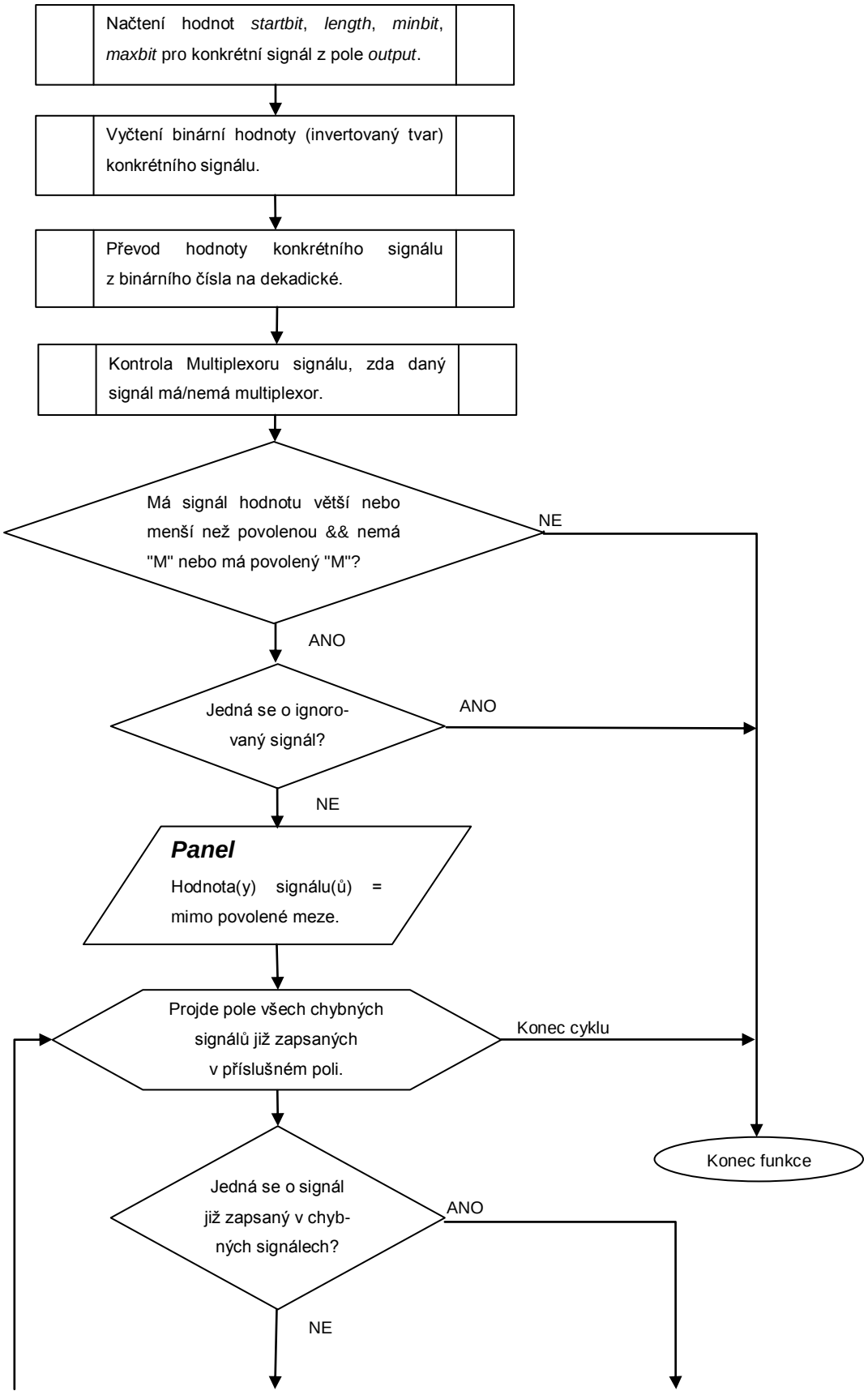
Příloha B – vývojový diagram CAPL programu

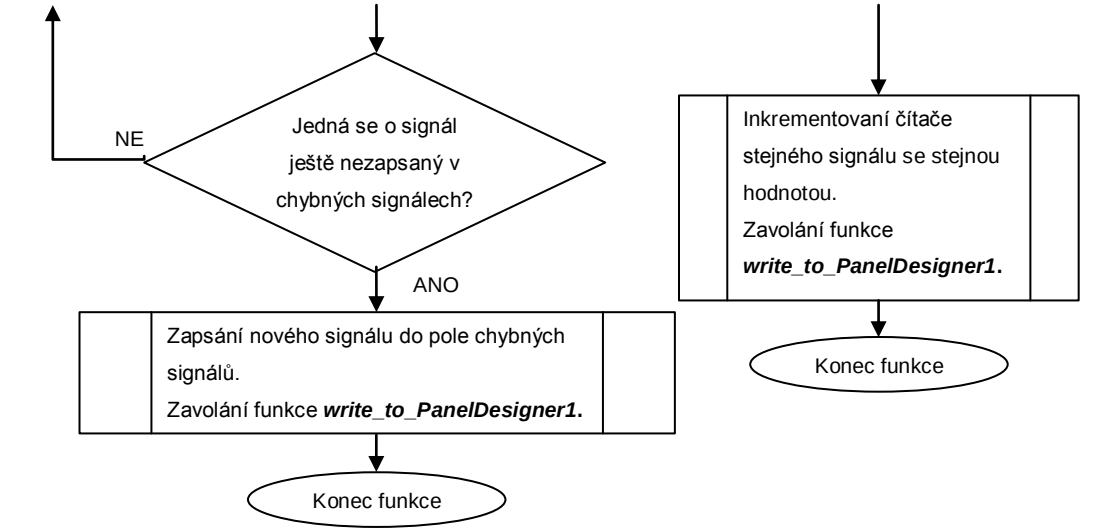



Stop časovače.
Zavolání funkcí **LoadDBC**
a **LoadIgnoredSig**.
Funkce *LoadDBC*
Zkopírování cesty k DBC z *Path Dialogu* v panelu do proměnné v programu.
Zapsání cesty k DBC do souboru *„Nazvy_DBC.txt“*.
Otevření souboru DBC pro čtení.
Načtení všech potřebných údajů z DBC do patřičných proměnných v programu.
Panel
Kontrola signálů = spuštěno.
DBC = načtena.
Hodnoty signálů = v povolených mezích.
Funkce *LoadIgnoredSig*
Otevření textového souboru*„Ignorovane_signaly.txt“* (pokud není vytvořen, vytvoří se).
Načtení ignorovaných signálů zapsaných v *„Ignorovane_signaly.txt“* do pole v programu.
Zavření textového dokumentu *„Ignorovane_signaly.txt“*.

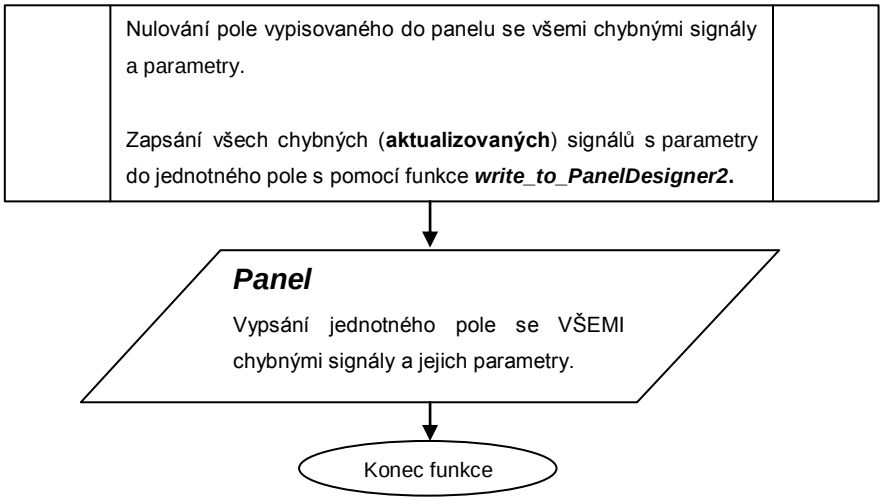

Je poslána na sběrnici zpráva?
NE
ANO
Funkce *on message* *
Zjištění ID zprávy.
Zjištění CAN kanálu.
Procházení pole s uloženými ID.
Konec
Je ID příchozí zprávy **rovno** s ID z pole && ID se **nerovná** předchozímu ID v poli?
NE
ANO
Načtení osmi datových bytů zprávy.
Zavolání funkce pro převedení datových bytů zprávy do binární hodnoty.
Nalezeni signálu s hodnotou Multiplexoru "M" (pokud zpráva "M"má).
Zavolání funkce ***findSignal*.**
Je ID příchozí zprávy **rovno** s ID z pole && ID se **rovná** předchozímu ID v poli?
NE
ANO
Zavolání funkce ***findSignal*.**

Funkce *findSignal*





Funkce *write_to_PanelDesigner1 + write_to_PanelDesigner2*



Funkce *on stopMeasurement + write_to_txtfile*