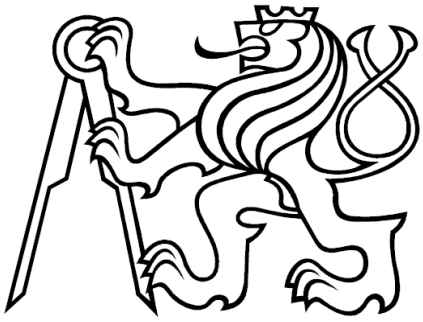
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY



DIPLOMOVÁ PRÁCE

Elektronická řídicí jednotka hydraulického akčního členu

Praha, 2010 Jan Kovář

Vedoucí práce: Ing. Libor Waszniowski, Ph.D.

Studijní program: Elektrotechnika a informatika

Studijní obor: Kybernetika a měření

Zaměření: Řídicí technika

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu. Veškeré použité informační zdroje jsem uvedl v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne Podpis

_____ _____

Poděkování

Na tomto místě bych chtěl poděkovat především vedoucímu mé diplomové práce Ing. Liborovi Waszniowskému, Ph. D. jehož věcné rady a pomoc mě vždy posunovala o krok vpřed. Chtěl bych také poděkovat všem, kdo mi přímo či nepřímo pomáhali při vzniku této práce a poskytli mi tak cennou radu.

V nemalé míře bych chtěl poděkovat také své přítelkyni a rodině, která mi vždy vytvářela vhodné podmínky a klidné zázemí po celou dobu studia a poskytovala mi veškerou podporu hmotnou i duševní.

Abstrakt

Diplomová práce se zabývá návrhem, realizací a testováním elektronické řídicí jednotky pro elektrohydraulický akční člen řídicí plochy letounu. Pro návrh softwarové části jednotky bylo využíváno postupů dle metodiky Model-Based Design. Tato metodika návrhu dovolovala flexibilně reagovat na změny struktury a parametrů řídicího systému a současně průběžně testovat řídicí algoritmus na cílové platformě řídicí jednotky.

Architektura navrhované jednotky vychází jak z požadavků na řízení elektrohydraulického akčního členu, tak zohledňuje i přímou integraci do Fly-By-Wire systému řízení se zachovanou záložní mechanickou vazbou. Cílem bylo navrhnout řídicí jednotku s ohledem na co největší spolehlivost, odolnost a požadovanou redundanci.

Pro komunikaci s ostatními komponentami řídicího systému využívá elektronická řídicí jednotka sběrnici CAN s komunikačním protokolem CANopen.

Pro řízení polohy pístnice elektrohydraulického akčního členu bylo využito algoritmů generovaných z prostředí MATLAB a Simulink. Generované algoritmy respektují použitou procesorovou platformu, především výpočet řídicího algoritmu v pevné řádové čárce.

Součástí práce byl nejen návrh hardwaru a softwaru, ale i výsledná fyzická realizace dvou elektronických řídicích jednotek, sestavení a propojení dvoukanálového řídicího systému pro elektrohydraulický akční člen a následné otestování a měření kompletního řídicího řetězce.

Abstract

This diploma thesis focus on the design, implementation and testing of an electronic control unit of electrohydraulic servo actuator of a plane control surface . Software of the unit was implemented by Model-Based Design methodology. This design methodology enable to react flexibly to changes in the structure and parameters of the control system. Model-Based Design allows control algorithms testing on the target platform controller continuously.

The architecture of the proposed unit is based on control requirements of an electrohydraulic actuator and respect requirements on direct integration into the Fly-By-Wire system. The aim was to design a controller with maximal reliability, durability and required redundancy.

Electronic control unit uses CANopen protocol for communication with other Fly-By-Wire components.

For position control of cylinder rod was used algorithms generated from MATLAB and Simulink. Generated algorithms take into account the processor target, especially real-time control calculations in fixed-point numbers.

Part of this work was not only the hardware and software design but the complete physical implementation of two electronic control units, construction and interconnection of two channel control system for electrohydraulic actuator and the subsequent testing of the complete feedback control system.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Jan Kovář**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Elektronická řídicí jednotka hydraulického akčního členu**

Pokyny pro vypracování:

1. Seznamte se funkcí a parametry dvoukanálového hydraulického akčního členu a s principy metody model-based design a s její podporou v programu Matlab Simulink.
2. Navrhněte architekturu, HW a SW dvoukanálové elektronické řídicí jednotky pro zmíněný akční člen. Návrh SW proveďte pomocí metody model-based design v programu Matlab Simulink. Řídicí jednotka bude komunikovat prostřednictvím sběrnice CANopen.
3. Realizujte navrženou řídicí jednotku a na reálném akčním členu nalaďte regulátor a otestujte SW.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Libor Waszniowski, Ph.D.

Platnost zadání: do konce letního semestru 2010/2011

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 12. 10. 2009

Seznam zkratek a symbolů

HW – Hardware, fyzická část zařízení
SW – Software, programová část zařízení
ECU - Electronic control unit, elektronická řídicí jednotka
FCC – Fly control computer, hlavní letový počítač
CAN – Control area network
EHSA - Electro-hydraulic servo actuator, elektrohydraulický akční člen
FBW – Fly-By-Wire
DSP – Digitální signálový procesor
MBD – Model-Based Design
HDL - Hardware dependent layer, fyzicky závislá vrstva
IHL - Interrupt handling layer, vrstva obsluhy přerušovacího systému
CSL - Communication services layer , vrstva komunikace
SPL - Signal processing layer, vrstva zpracování signálu a regulátoru
BEL - Background execution layer, vrstva programů na pozadí
PCB – Printed circuit board, deska plošných spojů
LVDT – Linear Variable Differential Transformer
ADC – Analog to digital converter, analogově-digitální převodník
PE – Processor Expert
MIL – Model in the loop
HIL – Hardware in the loop
PIL – Processor in the loop
CF – CanFestival
PWM – Pulse width modulation, pulzní šířková modulace
ALU – Arithmetic-logic unit, aritmeticko-logiká jednotka
RTW – Real-Time Workshop
SVN – Systém pro správu a verzování zdrojových kódů
$ - zástupný symbol pro označení čísla okruhu EHSA
# - zástupný symbol pro označení čísla řídicího kanálu dané ECU jednotky

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Hlavním cílem této diplomové práce bylo navrhnout, vyrobit a otestovat dvoukanálovou elektronickou řídicí jednotku (ECU) pro elektrohydraulický akční člen (EHSA) za pomocí metodiky Model-Based design. Při návrhu řídicího algoritmu ECU mělo být využito prostředí Simulinku a MATLABu. Navrhovaná jednotka měla obsahovat potřebné vstupní a výstupní obvody pro kompletní řízení soustavy EHSA. Komunikace s dalšími komponentami řídicího systému měla být zajištěna přes sběrnici CAN s protokolem CANopen. Nedílnou součástí práce bylo propojení navržené ECU do zbytku řídicího systému, především pak odladění komunikace mezi ECU a nadřazeným řídicím systémem Flight Control Computer (FCC).

ECU je jednou ze základních řídicích komponent Fly-By-Wire (FBW) systému řízení letounu. Návrh řídicí jednotky ECU koncepčně vychází z architektury použitého akčního členu EHSA a navrženého FBW systému, který předepisuje redundantní zapojení FBW komponent včetně využití zdvojené komunikační sběrnice. Jako cílová platforma byl zvolen digitální signálový procesor (DSP) 56F8367, který je zároveň sdíleným prvkem pro řídicí, měřicí i komunikační obvody jednotky. Při návrhu periferních a napájecích obvodů byl kladen důraz na galvanické oddělení řídicích a měřících obvodů od řízené soustavy, tak aby byl řídicí systém chráněn proti případným negativním vlivům.

Implementace řídicího algoritmu ECU měla být prováděna v prostředí Simulink a Codewarrior. Model řídicího algoritmu pak navržen v Simulinku a následně generován pomocí nástroje Real-Time Workshop do cílového jazyka.

Součástí řídicího algoritmu měl být i proporcionálně-integrační (PI) regulátor polohy pro EHSA. Ladění PI regulátoru probíhalo jako *Model In the Loop* simulace v prostředí Simulink. K návrhu stavového automatu měl být použit nástroj StateFlow. Samotné testování ECU a měření reálných odezev na EHSA probíhalo jako *Hardware In the Loop* simulace s nadřazeným systémem FCC, jednou ECU a EHSA.

Tato práce využívá výhod metodiky MBD a navazuje a rozšiřuje práci [7], kde byl diskutován a navržen jeden z možných podob řídicího systému ECU. Práce [7] se zabývala návrhem modelu a řídicích algoritmů ECU, avšak již nerealizovala hardware samotné jednotky. Vyvinutý algoritmus řídicího systému vychází z koncepce navržené v práci [7], nicméně jeho architektura byla výrazně rozšířena a modifikována tak, aby ji bylo možné integrovat do dalších vrstev softwaru ECU.

1.1 Motivace k Model-Based Design

V dnešní době je snahovou zrychlit a zefektivnit vývojový cyklus navrhovaného zařízení. Včasné uvedení vyvíjeného produktu na trh navíc výrazně ovlivňuje jeho konkurenceschopnost mezi podobnými produkty.

Vývoj nových typů zařízení včetně řídicích systémů je většinou technicky i finančně náročný, což v praxi znamená prodloužení vývojové cyklu. Proto se hledá vhodný způsob, jak zefektivnit metodiku návrhu vyvíjeného zařízení. Při vývoji řídicího systému existuje hned několik návrhových metodik. V zásadě je lze rozdělit na dva typy. První staví na postupné syntéze řídicího systému od menších částí k větším (klasická metodika). Druhým postupem je systémový náhled na daný problém.

V případě první „klasické“ metodiky se návrh řídicího systému rozpadá na několik vzájemně závislých etap, které je nutno vykonávat striktně za sebou. Budoucí etapy v této metodice vždy, alespoň částečně, závisejí na výsledku předchozí etapy. Příkladem je návrh řídicího algoritmu, který nemůže být vyvíjen dokud není k dispozici cílový hardware. Podobně je tomu u vývoje a ladění regulátoru pro řízenou soustavu, kdy nemůže dojít k návrhu bez samotné řízené soustavy nebo alespoň znalosti jeho modelu. Díky tomu dochází k výrazným časovým ztrátám a neefektivnímu čekání na výsledky předchozí etapy. Velkým nebezpečím u této metodiky je vznik systémových chyb, které jsou odhalitelné až po průchodu celým vývojovým cyklem (chyby způsobené jedním vývojovým týmem se přenášejí i na další vývojovou etapu).

Řešením těchto problémů je použití systémové metodiky resp. systémového pohledu na řešený problém. Ideálním řešením je využití modelů řízené a řídicí soustavy a simulování jejich vzájemného chování ve zpětnovazebních smyčkách. Algoritmy řídicího systému jsou tak zkoušeny z hlediska správné funkčnosti bez ohledu na fázi vývoje nebo dostupnost samotné řízené soustavy a cílového hardwaru. Metodika Model-Based Design využívá všech těchto výhod k zpřesnění a zrychlení vývojového cyklu.

Metodika MBD dovoluje navíc rozdělit syntézu na několik vzájemně kompatibilních a doplňujících se etap. Díky využití simulačního prostředí může dojít hned k několika variantám návrhového procesu, které zohledňují dostupnost jednotlivých částí. Obvykle totiž pro větší časovou úsporu dochází k paralelnímu vývoji, jak hardwarové tak softwarové části projektu. Často se stává, že je potřeba vyvíjet řídicí algoritmus bez ohledu na dostupnost cílového hardwaru nebo řízené soustavy. Pro tyto účely dovoluje MBD využít Model In the Loop (MIL) simulace, kdy lze ladit řídicí systém v simulačním prostředí na modelech. Další možností v MBD je vývoj za pomocí Processor In the Loop (PIL) simulace, kdy je řídicí aplikace verifikována přímo v cílovém procesoru ale akční zásahy či odezvy jsou aplikovány v simulačním prostředí na model soustavy. PIL simulace tak dovoluje zpětnovazební zapojení reálného hardwaru a simulačního modelu. Poslední možností, jak ladit výsledný systém, je zapojení reálného řídicího systému včetně řízené soustavy do smyčky a pozorovat (vizualizovat) reálné odezvy těchto systému v simulačním prostředí (tzv. Hardware In the Loop simulace).

1.2 Fly-By-Wire systém řízení

Tato práce byla součástí projektu vyvíjeného ve spolupráci Katedry řídicí techniky při ČVUT a společnosti Aero Vodochody, a.s. Projekt se zabývá návrhem Fly-By-Wire (FBW) systému řízení letu a systému autopilota pro lehký proudový letoun vyráběný v Aero Vodochody, a.s. Cílem Fly-By-Wire systému je návrh modulového elektronického řídicího systému se zachovanou záložní mechanickou vazbou.

Na obr. 1.1 je znázorněno rozmístění všech komponent ve FBW systému řízení letounu, včetně jejich komunikačních propojení po sběrnici CAN. Zapojení obsahuje také diagnostické komponenty a prostředky pro HIL simulaci celého systému. Toto rozmístění bylo použito i na reálné zkušební soustavě v objektu AERO Vodochody, a.s.

Vstupem navrhovaného FBW systému je aktuální pozice páky letounu (HOTAS). Pozice páky je snímána LVDT snímači a tento signál je přiveden do počítačů řízení letu (FCC). Z tohoto signálu vypočítávají FCC referenční signály pro lokální řídicí jednotky ECU.

Výsledkem tohoto projektu měl být simulační model řídicího systému, hardwarová realizace jednotlivých komponent FBW, následné propojení a otestování celého systému na reálné soustavě v areálu Aero Vodochody, a.s.

1.3 Specifikace požadavků na FBW systém

Pro pochopení základních požadavků a volby architektury ECU je vhodné se seznámit s požadavky na samotný FBW systém, popsaný v [1]. Architektura FBW systému řízení vycházela ze stávajícího stavu hydraulického řízení letounu a z požadavků uvedených v kapitole 5.

Použití FBW systému řízení letounu umožňuje zlepšit jeho letové vlastnosti tím, že letoun může být navržen jako aerodynamicky nestabilní a může mít menší ocasní plochy a tedy i celkovou nižší hmotnost. S úpravou aerodynamiky stávajícího letounu se však v době vývoje FBW systému nepočítalo, a proto je návrh FBW systému zaměřen na dosažení dalších výhod (viz [1]).

Hlavní výhody FBW systému řízení

- Zlepšení řiditelnosti a stranových vlastností
- Zlepšení obratnosti
- Snížení zátěže pilota zavedením tzv. Care-free řízení – tj. omezovače mezních parametrů (úhel náběhu, násobek, atd.)
- Zvýšení odolnosti proti poškození
- Zvýšení efektivity výcviku (cvičný letoun imituje chování jiného letounu)

V dalším odstavci jsou vybrány z [1] základní požadavky normy MIL-STD-882D , Apendix A na řídící systém, které ovlivňují návrh architektury FBW systému.

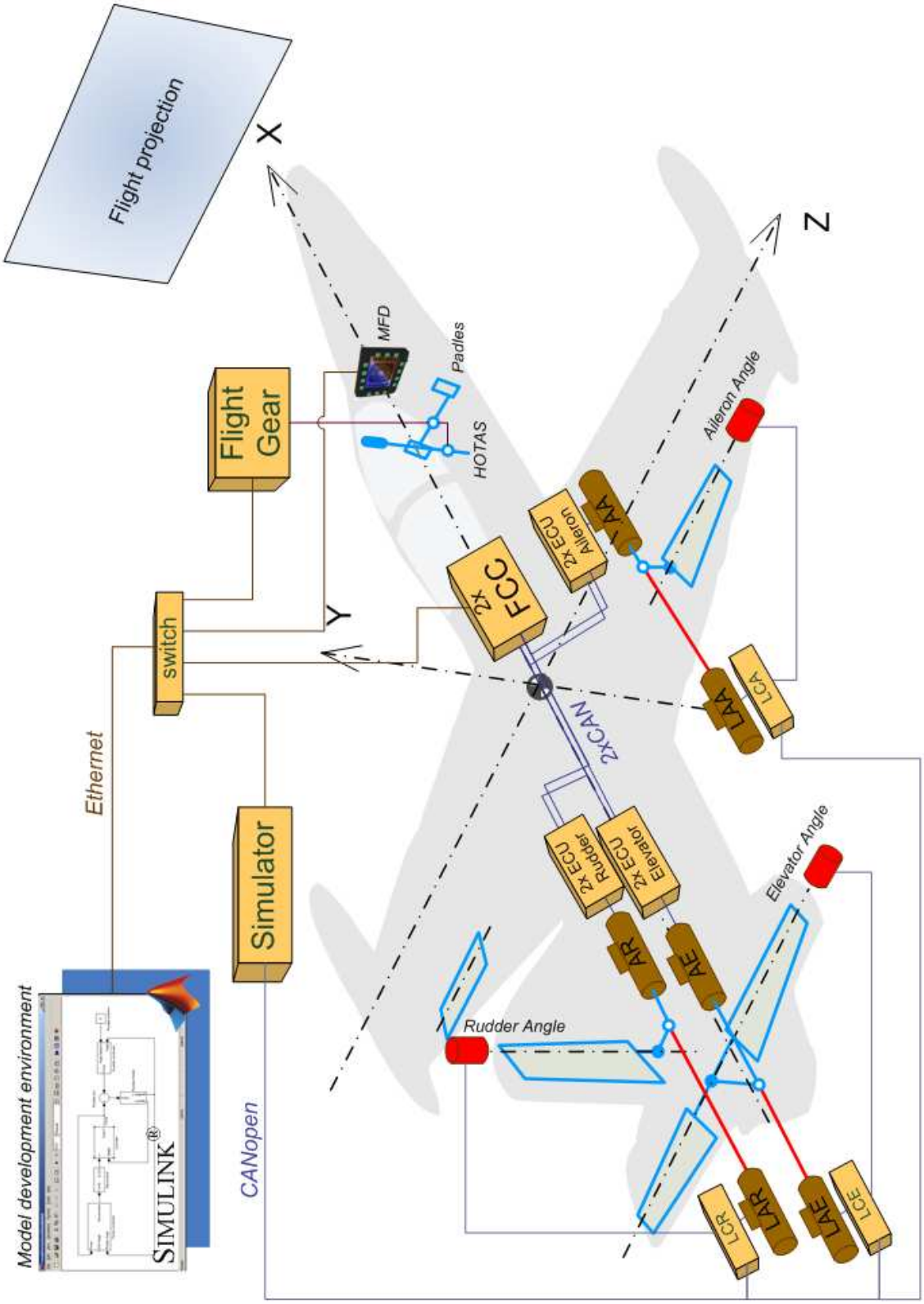
Nepřijatelné podmínky

Dvounásobná porucha nezávislých komponent, dvounásobná lidská chyba a nebo jejich kombinace nesmí přivodit katastrofický nebo kritický následek.

Přijatelné podmínky

a. Pro nekritické (z pohledu bezpečnosti) řízení a ovládání je použit systémový návrh, který vyžaduje pro kritický následek nejméně dvě nezávislé lidské chyby, nebo dvě nebo více poruch, nebo kombinaci poruchy a lidské chyby.
b. Pro kritické funkce řízení a ovládání je použit systémový návrh, který vyžaduje pro kritický následek nejméně tři nezávislé poruchy, nebo tři lidské chyby, nebo kombinaci tří poruchy a lidské chyby (porucha dvou hydraulických systémů není chápána jako dvě nezávislé poruchy ale jako „Common failure“ mód).
c. Systémový návrh, který positivně brání chybě při montáži, instalaci nebo spojení které může vést k nehodě (např. rozlišení konektorů).
d. Systémový návrh který pozitivně brání rozvoji poruchy z jedné komponenty na druhou, nebo který brání pronikání energie dostatečné k rozvoji poruchy (např. galvanické oddělení a pojistky).
e. Omezení systémového návrhu, spolupůsobení, nebo časování, které předchází vzniku poruch (např. izolace komponent při návrhu SW).
f. Systémový návrh, který zajišťuje schválený bezpečnostní koeficient nebo používá přijatelné konstrukční přístupy k zajištění přijatelné úrovně možnosti strukturní poruchy nebo uvolnění energie dostatečné ke způsobení nehody (např. dimenzování pojistných ventilů, dimenzování táhel a hydraulických agregátů).
g. Systémový návrh, který obsahuje zařízení k řízení omezení energie, která by mohla způsobit nehodu (tj. pojistky, pojišťovací ventily nebo ochrana před elektrickým výbuchem).
h. Systémový návrh, kde porucha komponent může být dočasně tolerována, protože je zde zbytková pevnost nebo alternativní funkční cesta, tak že funkce pokračuje s omezenou ale dostatečnou bezpečností (zajištěno redundancí celého systému).
i. Systémový návrh, který pozitivně varuje obsluhující personál o nebezpečné situaci, kde schopnosti reakce obsluhy budou zachovány (detekce poruchy, případně diagnostika stavů indikující blížící se poruchu).

Normy dále stanovují požadavky na konstrukci hydraulického systému, který je pro daný letoun již navržen a nevyžaduje pro instalaci FBW systému koncepční změny.



obr. 1.1 – Celkové uspořádání systému FBW na zkušební soustavě letadla (převzato z [43])

1.3.1 Volba počtu kanálů FBW systému řízení

Z rozboru provedeného v [1] a ze stávajícího stavu systému řízení letounu vyplynula volba dvou kanálů FBW s mechanickou zálohou. Pro systém s mechanickou zálohou platí předpoklad, že katastrofu nezpůsobí žádná porucha FBW systému. Přesto jsou vyžadovány dva kanály FBW, protože přepnutí z Care-free řízení, realizovaného FBW systémem, na mechanické řízení by mohlo být v některých režimech nebezpečné (například když Care-free řízení omezuje pilotem požadovanou výchylku tak, aby nedošlo k překročení letového parametru). Dále je předpokládáno, že vyřazení FBW systému a přepnutí na mechanické řízení vyžaduje ukončení letového úkolu.

Funkce FBW systému je tedy „Misson-critical“, ale v některých režimech letu i „Safety-critical“. Detekce poruchy komponenty FBW a vyřazení vadné komponenty je vždy považováno za „Safety-critical“.

1.4 Propojení komponent FBW systému

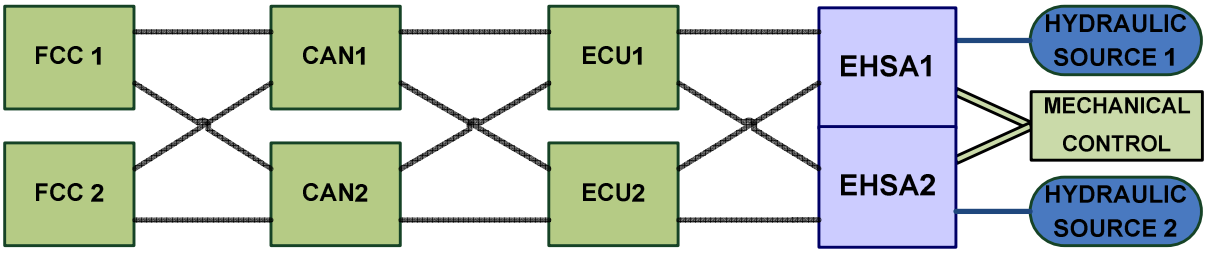
Stávající systém řízení letounu využíval dvouokruhový hydraulický akční člen s mechanicky řízeným ventilem pomocí soustavy táhel a mechanické zpětné vazby od polohy.

V systému FBW byl stávající hydraulický akční člen rozšířen o dvojici elektricky řízených servoventilů (každý pro jeden okruh) a sadu odpojovacích a řídicích ventilů. Tím vznikl nový elektrohydraulický servomechanismus (Electro-Hydraulic Servo Actuator - EHSA) se zachovanou mechanickou zálohou. Detailní popis EHSA je uveden v kapitole 4.

Navrhovaný elektronický řídicí systém je konstruován jako plně redundantní dvoukanálový systém využívající komunikaci po dvou nezávislých sběrnicích CAN. Polohové řízení EHSA má zajišťovat navrhovaná dvoukanálová ECU a to tak, aby každá ECU mohla řídit oba okruhy EHSA samostatně. To je možné díky zdvojení řídicích cívek obou servoventilů. Výsledné otevření servoventilu je ve výsledku dáno průměrem akčních zásahů obou kanálů ECU. Podobně i ovládání odpojovacích ventilů je možné zajistit z obou kanálů ECU jednotek.

Výpočet řídícího algoritmu FBW řízení letounu zajišťuje dvoukanálový letový počítač (Fly Control Computer - FCC). Každý FCC komunikuje s ECU prostřednictvím dvou sběrnic CAN. Oba kanály ECU jednotek jsou křížově připojeny k oběma sběrnicím.

Výše popsaný princip propojení každého kanálu dané komponenty s oběma kanály následující komponenty je znázorněn na obr. 1.2. Tímto způsobem je zajištěno, že porucha dvou různých komponent, byť v odlišných kanálech (například porucha CAN1 a ECU2) nezpůsobí výpadek celého FBW systému. Tento koncept je porušen pouze u hydraulických okruhů, které musí být zcela odděleny (požadavek normy). Norma stanoví, že v hydraulických okruzích nesmí být prvek, jehož porucha by při funkčním čerpadle znemožnila dodávku hydraulické kapaliny. Poruchou hydraulického okruhu se tedy rozumí porucha samotného čerpadla. Pro zajištění dostatečné redundance je proto v jednom okruhu ještě záložní čerpadlo.



obr. 1.2 - Schéma propojení komponent FBW (převzato z [6])

Princip dvoukanálového řízení EHSA je znázorněn na obr. 1.3. Z hlediska EHSA je připojení dvou ECU křížové, tak aby obě řídicí jednotky ECU mohli ovládat oba okruhy EHSA a zároveň byli schopny číst informace o stavech EHSA z obou hydraulických okruhů. Následné připojení dvou ECU na dvě sběrnice CAN 1 a CAN 2 je opět koncipováno kříženě, tak aby při poruše jedné z komunikačních cest nedošlo k úplné neschopnosti řízení EHSA. Každá ECU tak obsahuje dva nezávislé CAN řadiče. Výsledné řízení EHSA je pak plně dvoukanálové (jedna ECU tvoří jeden kanál, druhá ECU druhý kanál).

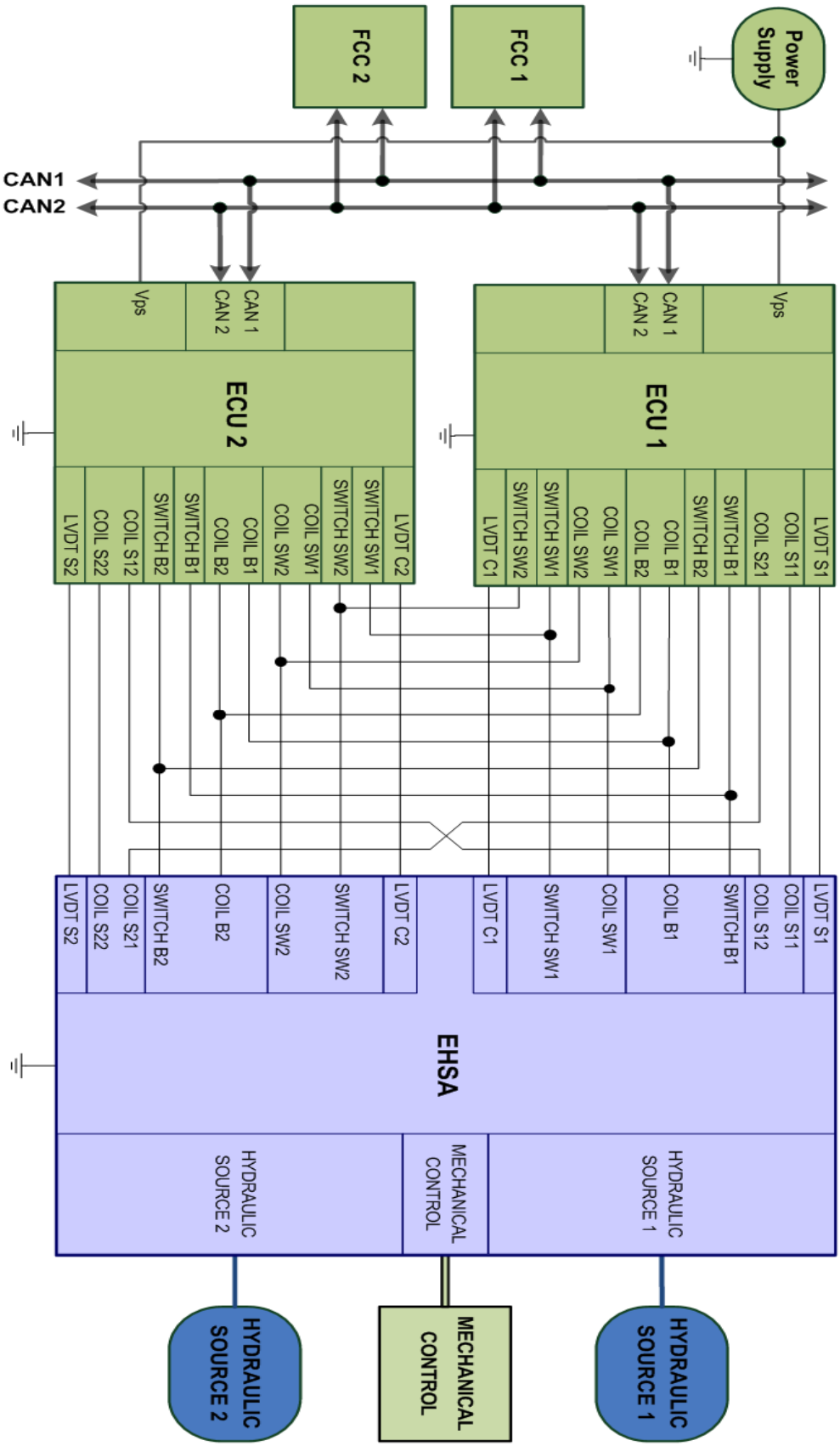
Terminologie

Každé zařízení (FCC, ECU, EHSA) je tvořeno dvěma kanály rozlišovanými čísli 1, 2. Pokud není potřeba rozlišit jednotlivé kanály, je místo čísla kanálu ve jméně zařízení nebo jeho prvků používáno zástupného znaku:

- # pro číslo řídicího kanálu ECU jednotky
- *$* pro číslo EHSA kanálu (hydraulického okruhu)

Například:

Rozlišujeme COIL B1 a COIL B2 pro cívky EHSA kanálu 1 a 2. Pokud není nutné rozlišovat kanály můžeme mluvit obecně o COIL B$. Podobně můžeme zobecnit označení COIL S$# pro cívku EHSA kanálu $ ovládanou z ECU kanálu #.



obr. 1.3 - Detailní propojení ECU a EHSA v systému FBW (modifikováno podle [6])

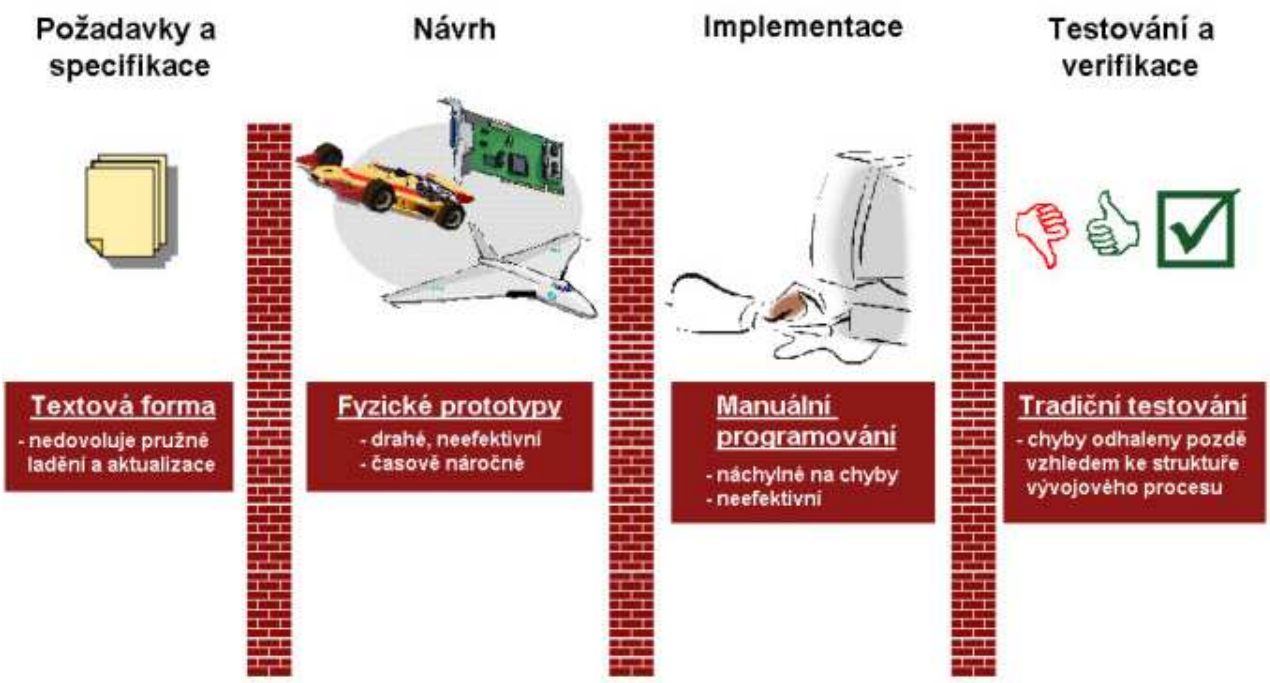
2 Metodika Model-Based Design

Jedním z cílů této diplomové práce bylo navrhnout software pro elektronickou řídicí jednotku za pomoci metodiky Model-Based Design (MBD). Úkolem bylo implementovat řídicí algoritmus ECU v komplexním návrhovém prostředí Simulink. Do tohoto simulačního prostředí jsou integrovány prostředky pro využití postupů z MBD. V této kapitole bude popsán princip a postupy charakteristické pro metodiku MBD a popsána podpora MBD v prostředí MATLAB a Simulink.

2.1 Princip a popis MBD

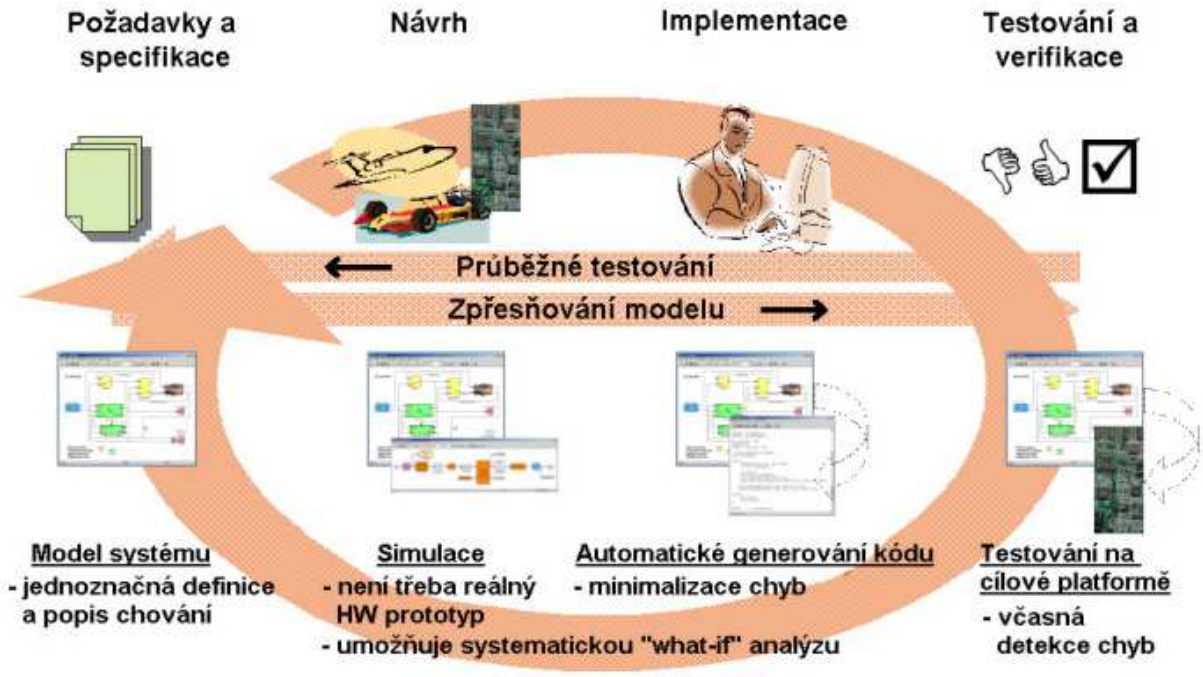
Metodika Model-Based Design je matematická vizualizační metoda založená na modelování řízeného a řídicího systému s možností výsledné generace kódu pro konkrétní procesorovou platformu. MBD v sobě zahrnuje ucelený proces komplexního návrhu řídicího systému od modelování soustav, přes analýzu, simulaci a syntézu vhodných regulačních algoritmů až po možnost postupného testování jednotlivých fází návrhu na konkrétní cílové platformě.

Rozdíl mezi klasickým vývojovým cyklem a metodikou MBD je vidět na obr. 2.1 a obr. 2.2. Klasický vývoj systému se podobá jednosměrnému implementačnímu postupu s malými lokálními cykly, čekajících jeden na druhý. Dá se říci, že tento postup směřuje vždy odspodu nahoru, tedy od menších částí ke kompletnímu systému. Někdy se tato metodiky nazývá také Waterfall.



obr. 2.1 – Tradiční vývojový cyklus a jeho negativní vlastnosti (převzato z [26])

Naproti tomu metodika MBD směřuje vždy shora dolů. Díky tomu je možné vytvořit globální vývojový cyklus. Spojí-li se tato vlastnost s možností generování kódu je možné vyvíjený systém nepřetržitě zpřesňovat a průběžně testovat. Další výhodou této metodiky je možnost použití pouze jednoho vývojového prostředí, všichni vývojáři tak používají společné prostředky pro všechny fáze vývoje. Díky tomu je možné rychle reagovat na změny požadavků na řídicí systém a změny lokálních částí nebo komponent systému.



obr. 2.2 – Vývojový cyklus Model-Based Design a výhody oproti klasickému cyklu (převzato z [26])

Pomocí metod automatického generování kódu na cílovou platformu je navíc možné zajistit přenositelnost řídicího algoritmu na širokou škálu výpočetních prostředků. Metodika MBD tak slučuje všechny potřebné etapy návrhu do jednoho cyklu. V dnešní době metodika MBD zahrnuje hlavně prostředky pro vývoj řídicích algoritmů. Vývojové prostředky pro návrh hardwaru jsou tak od softwarových komponent odděleny a nelze tuto část v současné době do metodiky MBD zahrnout. I tak se dá říci, že výhody MBD výrazně převažují, některé studie ukazují, že i přes vysoké počáteční investice do vybavení pro vývoj systému pomocí MBD se vrátí v časové úspoře vynaložené na vývoj systému (úspora 50% a více).

Díky výše uvedeným vlastnostem metodika MBD výrazně zefektivňuje a zrychluje vývojový postup při návrhu nového systému. Základem MBD je využití a vzájemné propojení čtyř relativně nezávislých činností. Do těchto činností patří modelování a identifikace řízeného systému, analýza a syntéza řídicího algoritmu, simulace řídicího algoritmu na modelu a testování řídicího algoritmu na cílové soustavě. Propojení těchto činností je možné na základě použití společného vývojového prostředí. Algoritmus návrhové metodiky MBD lze shrnout do čtyř základních kroků, kterými jsou

- systémová identifikace,
- analýza řízené soustavy a návrh řídicího algoritmu,
- off-line simulace chování navrženého řídicího algoritmu,
- generování řídicího algoritmu na cílovou platformu.

2.1.1 Systémová identifikace

Základním krokem MBD je správně a co nejpřesněji identifikovat řízenou soustavu. Vytvořením matematického modelu řízené soustavy je možné převést návrh řídicího systému do simulačního či vizualizačního prostředí. Díky tomu je možné provádět opakovaně testování a analýzu řízené soustavy. Simulací chování identifikované soustavy lze pak efektivně zjistit nejen samotný řídicí algoritmus ale také určit kritické body návrhu.

2.1.2 Analýza a syntéza řídicího algoritmu

Na základě modelu soustavy lze určit vhodný řídicí algoritmus, ať už se jedná o spojité, diskrétní nebo dvoustavové řízení. V simulačních prostředích lze také určit rozmezí signálů a rozhodnout, které hodnoty je již možné zanedbat či saturovat. Z hlediska praktického návrhu je tato vlastnost velice důležitá jelikož lze zvolit optimální cestu k výběru cílové platformy. Není tak nutné, již na začátku vývoje, odhadovat na jaké platformě řídicí systém „poběží“. Simulace je také výhodná pro naladění rozsahů výpočtů v pevné či plovoucí řádové čárce.

2.1.3 Simulace

Po návrhu řídicího algoritmu dovoluje metodika vyzkoušet odezvy celého systému a to jak, přímovazebního, tak zpětnovazebního obvodu. Díky tomu lze ještě před provedením finančně náročných testů na skutečné soustavě zjistit chyby v řídicím algoritmu. Může tak dojít k včasným úpravám řídicího systému a reálným odhadům dopadu navrženého systému na řízenou soustavu.

2.1.4 Generování kódu

Důležitým bodem při postupu dle MBD je využití automatického přechodu z oblasti modelů na cílovou platformu. V dnešní době je tento krok realizován generátorem kódu do cílové platformy. Tento přechod je v metodice MBD zásadním, protože určuje výslednou kvalitu řídicího systému. Generátory kódu musí totiž nejen zajistit správnost algoritmu ale také efektivní přechod do cílového jazyka (paměťová a výpočetní náročnost). V dnešní době je nejrozšířenějším nástrojem pro automatické generování kódu Real-time Workshop od společnosti Mathworks. Tento nástroj je integrován do prostředí Simulink a spolu se Simulinkem tvoří komplexní prostředí pro použití metodiky MBD. Přesnější popis tohoto nástroje je uveden v kapitole 2.3.

2.2 Postup MBD v programu MATLAB a Simulink

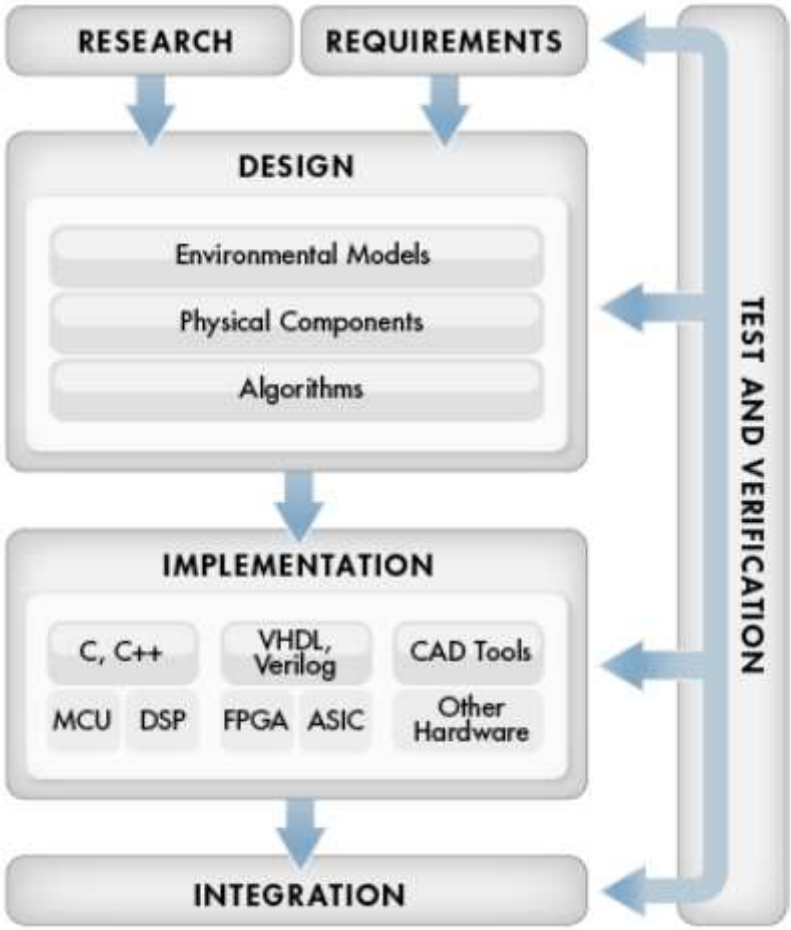
Pro praktické využití metodiky MBD je nutné zvolit vhodné návrhové prostředí, které musí obsahovat všechny výše zmíněné nástroje. Důležitá je tedy komplexnost vývojových prostředků. Z tohoto pohledu vyhovuje jen několik málo produktů. Pro návrh řídicí jednotky byl zvolen jako vývojový nástroj program Simulink od společnosti Mathworks.

Simulink je grafická nadstavba programu MATLAB a využívá algoritmy MATLABu pro numerické řešení lineárních a nelineárních diferenciálních rovnic. Simulink obsahuje simulační prostředí, prostředky pro identifikaci soustav, prostředky pro syntézu stavových automatů i spojitých řídicích algoritmů a nástroj na automatické generování kódu pro cílovou platformu. Obsahuje tedy vše potřebné pro využití metodiky MBD.

Na obr. 2.3 jsou zobrazeny komponenty MBD v prostředí Simulink. Lze pozorovat, že návrh řídicího systému v Simulinku se rozpadá na několik vzájemně provázaných etap. Jádrem celého návrhu je model řídicí a řízené soustavy. Tyto modely se posléze používají ve všech fázích návrhu, od etapy vstupních požadavků, přes implementační část až po finální testování.

První etapou v návrhovém cyklu je specifikace řídicího systému. V ní jsou definovány požadavky na vstupy, výstupy a chování řídicího systému. Zde se poprvé objevuje vnější model řídicího sytému. Při specifikaci musí být jednoznačně určen cíl a kritéria pro finální hodnocení návrhu. Je potřeba zde také zvážit proveditelnost a náročnost návrhu (jak finanční tak technickou). V technické fázi je potřeba zvážit volbu cílové platformy, je nutné se zamyslet jaký druh výpočetní jednotky bude zvolen, především pak možnost využití výpočtu v plovoucí nebo pevné řádové čárce. Tento krok úzce souvisí s finančním aspektem projektu, protože volba výpočetní jednotky ve většině

případů určuje i celkovou strukturu a tedy i cenu zařízení. Pro návrh ECU byla již od počátku zvolena platforma 56F8367, která nepodporuje operace v plovoucí řádové čárce, proto bylo zvoleno řešení využít výpočet řídicího algoritmu v pevné řádové čárce. To však obnáší některé negativní důsledky a omezení. Jejich vliv na výsledný řídicí systém bude diskutován v kapitole 2.4.



obr. 2.3– Vývojový cyklus Model-Based Design v prostředí Simulink (převzato z [26])

Do specifikační fáze lze zahrnout také výsledky z výzkumu a vývoje, protože většina těchto výsledků je produkována ve formě matematických modelů. V programu Simulink se pro identifikaci soustavy dá použít nástroj System identification Toolbox, který z naměřených dat vytvoří lineární nebo nelineární dynamické modely.

V případě, že je identifikován model řízené soustavy a jsou stanoveny vstupní požadavky na řídicí systém, může dojít k etapě návrhu řídicího algoritmu. Modely systému jsou v této fázi analyzovány a zpřesňovány, tak aby co nejvíce odpovídali reálným systémům. Ze zjištěných dat se následně určí nejvhodnější metoda návrhu řídicího algoritmu. Simulink obsahuje hned několik nástrojů pro návrh a analýzu řídicích systémů. Pro případ návrhu spojitého (diferenciálního) nebo diskretizovaného (diferenčního) modelu lze použít nástroje Control System Toolbox a Simulink Control Design. Ty dovolují efektivní ladění a syntézu PID včetně nastavení fázové a amplitudové bezpečnosti, či doby náběhu a ustálení výstupu soustavy. Pro případ syntézy stavových automatů je možné použít nástroj Stateflow. Všechny výše zmíněné nástroje dokážou generovat vzájemně kompatibilní modely simulovatelné v prostředí Simulink. Z obr. 2.3 je vidět, že již ve fázi simulací lze testovat navržený algoritmus na splnění vstupních požadavků, tím je možné včasně odhalit zásadní chyby v řídicím algoritmu. Verifikace je v této fázi důležitá také z hlediska splnitelnosti některých vstupních požadavků. Při návrhu se totiž mohou objevit kritické nebo špatně řešitelné úkoly, které ve většině případů znamenají časovou prodlevu v celkovém návrhu.

V okamžiku splnění vstupních požadavků na řídicí systém (časové odezvy, dynamika systému, omezení signálů, apod.) je možné přejít z fáze simulační (modelové) do fáze implementační (reálné). Tento přechod je zásadním krokem v návrhu řídicího systému, protože je nutné zvolit správnou cílovou platformu. Ne vždy je jasné jestli bude zvolený hardware dostatečně výkonný pro implementovaný algoritmus. Z tohoto pohledu je ideální využít automatické generování kódu z modelového prostředí do konkrétního hardwarově závislého jazyka. Zásadním parametrem je zde především časová úspora tohoto přechodu.

V Simulinku existuje pro tuto možnost nástroj Real-Time Workshop (RTW). Ten umí přegenerovat konkrétní modelovou strukturu do uživatelem zvoleného cílového jazyka. Navíc umožňuje generovaný kód spustit na cílové platformě a současně vizualizovat reálná data v prostředí Simulinku. Tím lze testovat navržený řídicí systém v jednom prostředí bez nutnosti dalších nástrojů. Nástroj RTW bude dále diskutován v kapitole 2.3.

Posledním krokem v návrhu je verifikační fáze. Obrovskou výhodou metodiky MBD je postupná a kontinuálně prováděná verifikace. Díky tomu nedochází k fatálním důsledkům až na samotném konci návrhu a lze tak předejít závažným systémovým chybám.

2.3 Proces generování kódu a nástroj RTW

Pro efektivní využití metodiky MBD je nutno použít řadu progresivních postupů, jedním ze základů MBD je proces automatického generování kódu do cílového jazyka. Program MATLAB ve své nadstavbě Simulink právě takovýto nástroj obsahuje.

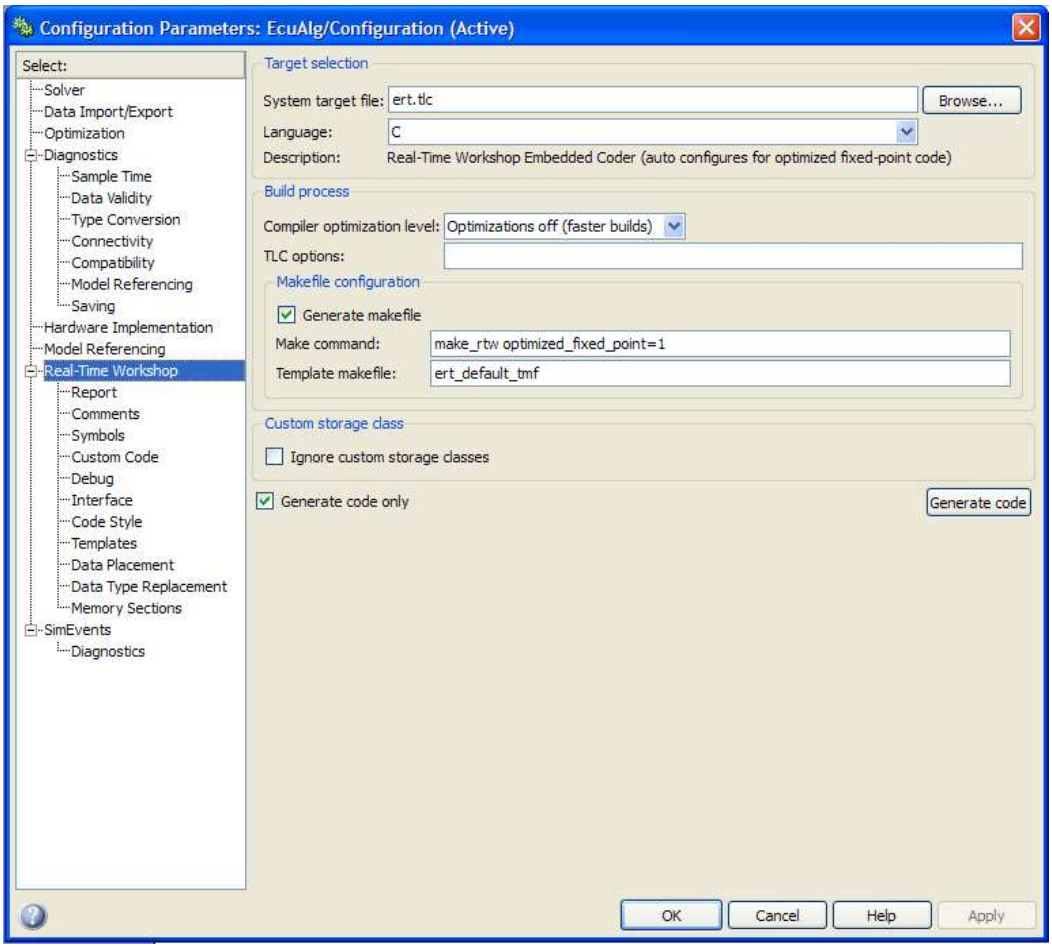
Nástroj RTW umí ze simulačního prostředí Simulink vygenerovat a spouštět samostatné soubory v jazyku C. Výsledný kód může být následně využit v jednovláknových, vícevláknových nebo real-time aplikacích. Navíc RTW obsahuje rozhranní pro externí běh části kódu s možností hardwarové akcelerace Simulinkového modelu. Vygenerovaný kód splňuje normu ANSI/ISO a je přeložitelný pro různé mikroprocesorové platformy nebo operační systémy.

Nástroj RTW je integrován přímo do prostředí Simulink. Ovládání RTW probíhá přes uživatelské rozhranní Model Explorer. Model Explorer zprostředkovává:

- generování kódu ze Simulinku,
- nastavení RTW pro cílovou platformu (target),
- optimalizaci generovaného kódu.

V nástroji RTW lze přesně nakonfigurovat výslednou podobu generovaného kódu. Pro snadnější a bezproblémovou generaci jsou navíc k dispozici předdefinované vzory, tak aby finální podoba kódu bylo časově i paměťově optimální. Základní sada těchto vzorů obsahuje také platformu 56F8000 od společnosti Freescale, která byla použita také pro ECU.

Parametry generátoru lze nastavovat v dialogovém okně konkrétního modelu, na obr. 2.4 je finální nastavení pro platformu 56F8367. Detaily ohledně nastavení RTW lze nalézt v [35] a [36]. Podrobné nastavení RTW pro navrhovanou elektronickou řídicí jednotku lze nalézt také na přiloženém CD ve složce *Software_ECU*.



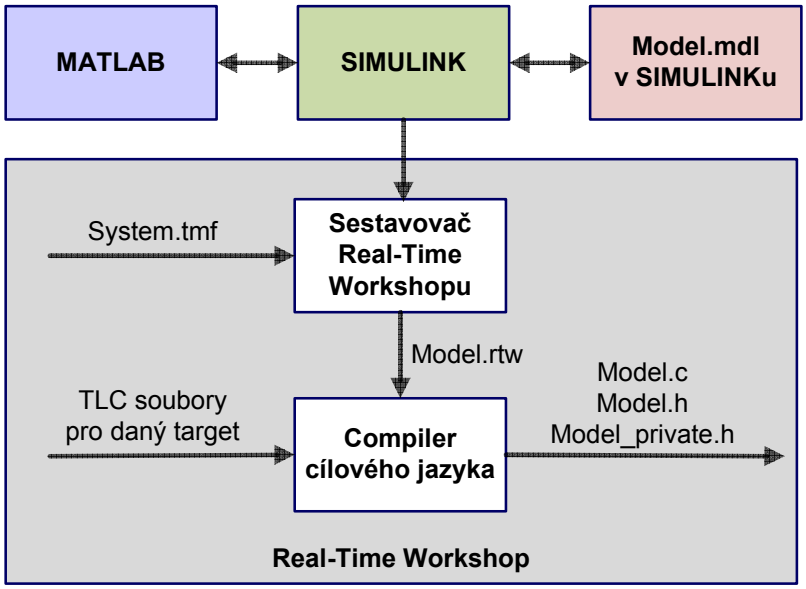
obr. 2.4– Konfigurace nástroje Real-Time Workshop

2.3.1 Target a postup generování kódu

Při konfiguraci RTW se používá pojem target. Na obr. 2.4 je např. nastaven v nabídce *Target selection* soubor *ert.tlc* a jazyk C. Target pro RTW představuje souborový objekt reprezentující cílový procesor nebo mikrokontrolér. V targetu je pak definován typ použitého CPU, jazyk výsledného generovaného kódu, knihovní komponenty, atd.

Prostřednictvím TLC souborů se určuje jakým způsobem se bude generovat výsledný kód. V případě, že má například dojít ke generování do jazyka HDL musí být zajisté použito jiných principů než případě generátoru kódu pro platformu 56F8000. TLC tedy definuje co a jakým způsobem se bude přegenerovávat do cílového kódu.

Průběh generování kódu v nástroji RTW je ukázán na obr. 2.5. Základem je prostředí Simulink, ve kterém je vytvořen model řídicího algoritmu (Model.mdl). Po odladění tohoto řídicího algoritmu na modelu řízené soustavy lze spustit nástroj RTW, který zajistí přegenerování modelu do cílového jazyka. Spuštění generování se provádí přes nabídku *Tools* v Simulinku nebo pomocí zkratky *CTRL+B*. Nástroj RTW nejdříve přeloží model z grafické podoby do popisné formy, která definuje propojení jednotlivých bloků (soubor Model.rtw). Následně je tento soubor přeložen do cílového jazyka pomocí programu Target Language compiler . Způsob jakým má Target Language compiler přetransformovat význam prvků v souboru Model.rtw je určen v targetu tedy v souborech *tlc* (v našem případě *ert.tlc*). Výsledný kód je pak optimalizován pro cílový procesor a využívá všech dostupných knihoven pro danou platformu.



obr. 2.5– Generování kódu pomocí Real-Time Workshop

2.4 Výpočty v pevné řádové čárce

Tato podkapitola popisuje a diskutuje princip výpočtů v pevné řádové čárce (fixed-point). Při simulaci a syntéze modelů se obvykle používá aritmetika v plovoucí řádové čárce (floating-point). Výsledné modely a algoritmy je však obvykle nutné spustit na platformě neobsahující floating-point jednotku (matematický koprocesor). To může mít z hlediska reálného použití vyvinutého algoritmu fatální důsledek. Nejen že bude algoritmus na cílové platformě nespustitelný (neexistují potřebné knihovní floating-point funkce) ale zároveň (v tom lepším případě, když existují knihovny) dojde k pozdnímu dokončení výpočtu. V řadě aplikací je včasné dokončení kritickou podmínkou stability celého systému, proto se v mnoha případech přechází z floating-point na fixed-point aritmetiku. Nezanedbatelná je i cena použitého hardwaru, jednotky s hardwarovou podporou výpočtu v plovoucí řádové čárce jsou výrazně dražší než platformy bez této jednotky.

S přechodem na fixed-point aritmetiku však dochází k několika zásadním odlišnostem a obtížím, s kterými je nutné se vypořádat. Některým problémům se lze již na začátku vývoje vyhnout použijeme-li vhodnou platformu (z hlediska šířky datové sběrnice a registrů). V mnoha případech si ulehčíme práci využijeme-li ladících prostředků jako je Fixed-point Toolbox v programu Simulink.

Jádrem navrhované ECU je 16-ti bitová platforma 56F8367, která neobsahuje koprocesor pro výpočet v plovoucí řádové čárce, proto bylo již od počátku vývoje počítáno s přechodem simulačního modelu na model v pevné řádové čárce.

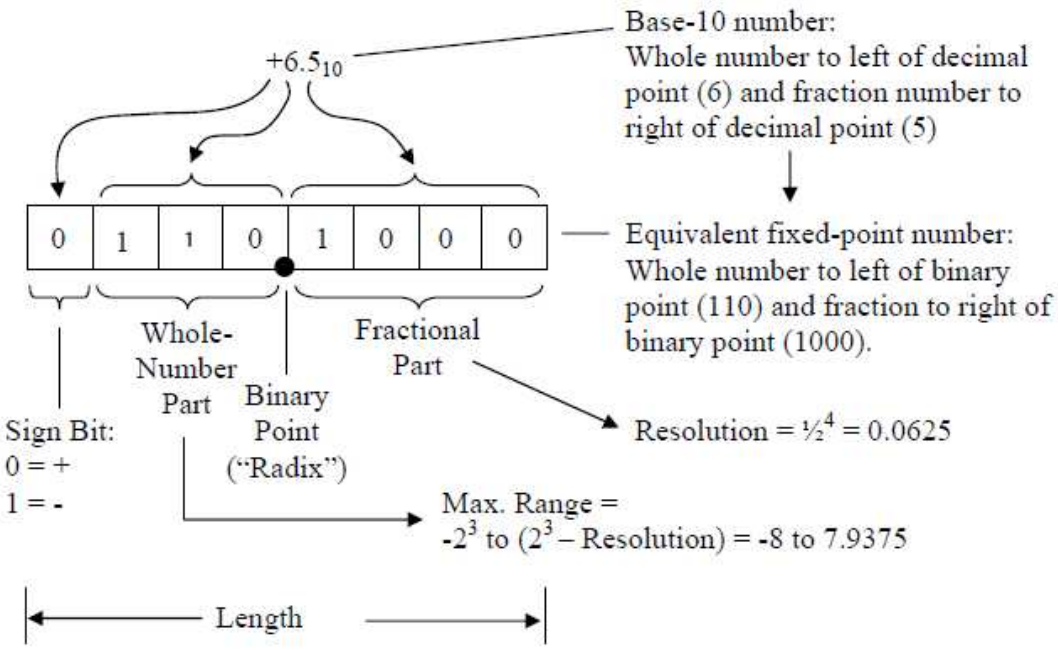
Binární floating-point reprezentace popisuje každé reálné číslo *V* jako:

$$V = \pm M.2^{\pm X}$$

, kde *M* je mantisa, *X* je exponent. Obvykle jsou tyto dvě části spolu s informacemi o znaménkách uloženy v jednom *n*-bitovém slově.

Díky exponenciálnímu zápisu čísla je možno efektivně zapisovat extrémně malá i velká čísla. Problém však nastává v případech aritmetických operací dvou či více výrazně odlišných čísel. Pak totiž dochází k výrazným zaokrouhlovacím chybám, což v případě delších iteračních algoritmů vede na numericky nestabilní systém (špatný výsledek).

Na rozdíl od floating-point formátu čísel je fixed-point reprezentace čísla charakterizována jedním *n*-bitovým celým číslem rozděleným na tři části (viz obr. 2.6). První částí (zleva, MSB) je jednobitová znaménková část reprezentující kladnost a zápornost čísla. Druhou částí je celočíselná část, která popisuje hodnotu čísla před desetinnou tečkou. Poslední částí je desetinná část, která určuje hodnotu čísla za desetinnou tečkou. Z hlediska matematického popisu tedy stačí udávat polohu desetinné tečky, tím je totiž přesně daná jak velikost celočíselné tak desetinné části.



obr. 2.6– Reprezentace desetinného čísla ve fixed-point formátu (převzato z [26])

Převod desetinného čísla z desítkové soustavy do binární fixed-point reprezentace se skládá ze dvou úkonů. Nejdříve je nutné převést celočíselnou část, to je z hlediska postupu stejné jako převod z desítkového do binárního kódu. U desetinné části je převod závislý na bitové délce této části. Desetinná část je charakterizována maximálním rozlišením. Rozlišení Δ je desetinné číslo v desítkové soustavě:

$$\Delta = \left(\frac{1}{2}\right)^{E}$$

, kde *E* je počet bitů desetinné části. Desetinnou část ve fixed-point (FP) formátu dostaneme tak, že zlomkovou část, kterou chceme převést do FP podělíme Δ a výsledek převedeme do binární reprezentace. Rozlišení Δ je pak číslo, kterým můžeme ve FP reprezentovat nejmenší nenulovou hodnotu. V prostředí Simulink je rozlišení Δ nazýváno přesnost FP, nebo též precision). Tabulka tab. 2.1 ukazuje vlastnosti a charakter čísel v závislosti na volbě číselné reprezentace.

tab. 2.1 – Porovnání fixed-point a floating-point reprezentace

Zohledněné zdroje	Aritmetika v pevné řádové čárce (fixed-point)	Aritmetika v plovoucí řádové čárce (floating-point)
Paměťová spotřeba RAM a ROM	Malá	Velká
Rychlost výpočtu	Krátká doba výpočtu	Delší doby výpočtu
Vývojový čas a implementace	Dlouhý, náročná	Krátký, obecně použitelné knihovny
Determinismus výsledků a zaokrouhlování	Předem známý rozsah a přesnost, deterministické zaokrouhlování	Dynamické nastavení rozsahu při výpočtech, časté chyby díky zaokrouhlování
Runtime chyby	Náchylný na přetečení a podtečení	Odolný proti přetečení - dynamický rozsah
Energetická spotřeba výpočetního hardwaru	Nízká	Vysoká

Pro úplnost je vhodné poznamenat, že zápisová konvence čísel do FP reprezentace není jednotná, v některých prostředích se objevuje kromě FP čísla ve formátu s umístěním desetinné čárky a celkové délky slova s uvedeným znaménkovým bitem (jako na obr. 2.6), také zápis FP čísla pomocí dvou parametrů *Slope* a *Bias*. Pomocí tohoto zápisu je desetinné číslo V zapsáno jako:

$$V = S.W + B$$

, kde *S* charakterizuje přírůstek a odpovídá *Δ*, *W* je binární hodnota *n*-bitového slova (binární celočíselná reprezentace FP slova) a *B* je stejnosměrná (aditivní) složka signálu.

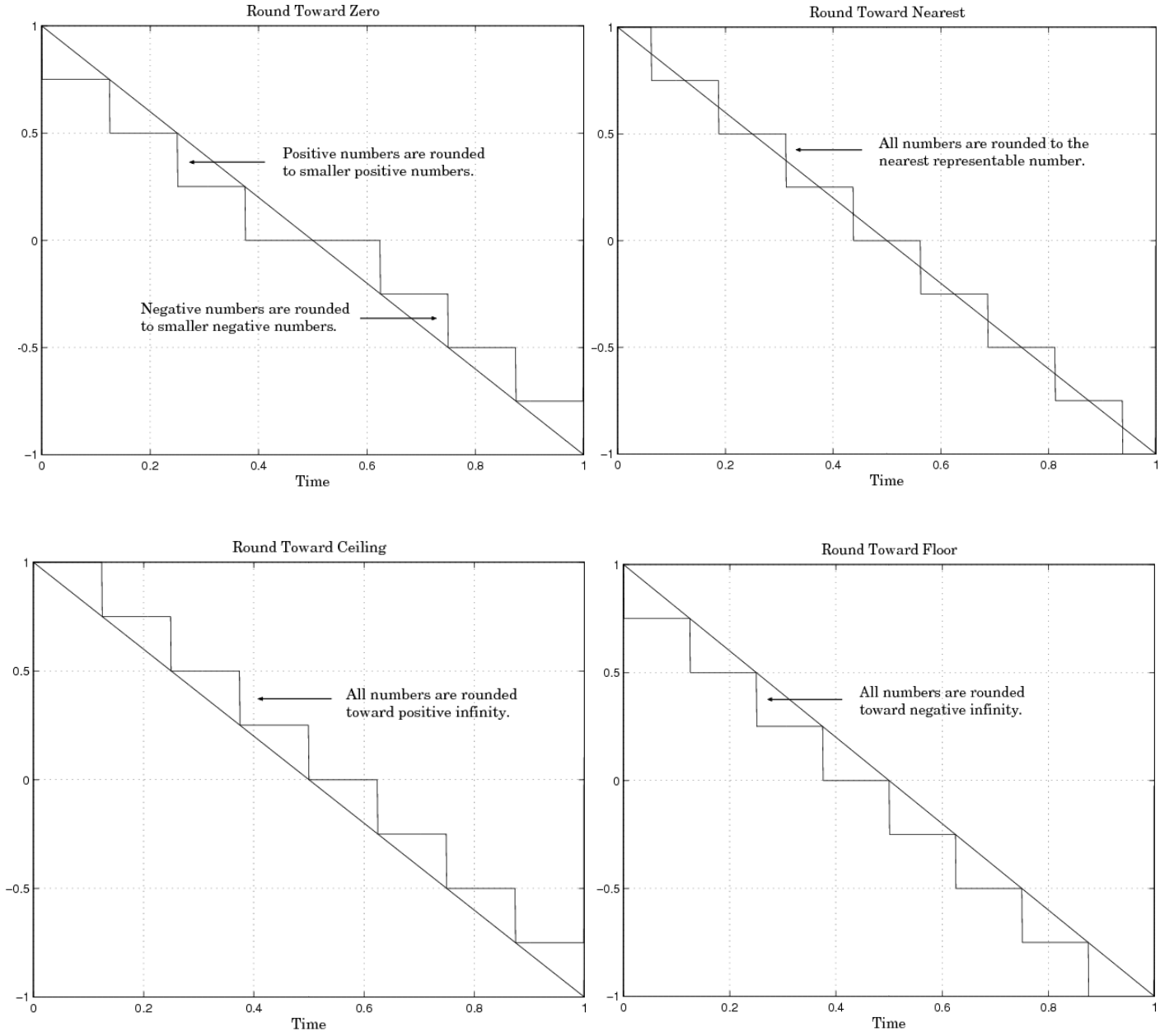
2.4.1 Zaokrouhlování a saturace

Jedním ze základních problému u FP aritmetiky je volba zaokrouhlovací metody. Jak již bylo uvedeno výše, reprezentace čísla ve FP je charakterizována především rozlišením *Δ*, které udává nejmenší nenulový přírůstek. Rozlišení *Δ* je dáno jak počtem bitů používaného slova, tak umístěním řádové čárky. V případě, že je potřeba FP číslo rozšířit na vyšší přesnost (rozlišení) je potřeba desetinnou část rozšířit o příslušný počet bitů (připojení *n*-bitů k LSB). Opačný problém nastává, jestliže chceme FP číslo s vyšší přesností konvertovat na číslo s nižší přesností. Zde je potřeba vybrat metodu jakou bude provedena zaokrouhlovací konverze.

Typickým příkladem, kde je nutné toto zaokrouhlení udělat, je typová konverze z většího FP čísla (vícebitové číslo) na FP číslo s nižším rozlišením (menší počet bitů). Dalším příkladem je mezivýpočet při složitějším vícetypovém sčítání či násobení. Díky zaokrouhlení dochází k úmyslnému vnesení výpočetní chyby, někdy též nazývané kvantizační šum. V Simulinku je možno využít hned několika typů zaokrouhlovacích metod:

- Zero – zaokrouhluje číslo směrem k nejbližší (v absolutní hodnotě) menší dostupné kvantizované hodnotě,
- Nearest – zaokrouhluje číslo směrem k nejbližší dostupné kvantizované hodnotě (symetrické),
- Ceiling - zaokrouhluje číslo směrem k nejbližší větší kvantizované hodnotě,
- Floor - zaokrouhluje číslo směrem k nejbližší nižší kvantizované hodnotě.

Grafické interpretace těchto metod jsou znázorněny na obr. 2.7.



obr. 2.7– Zaokrouhlovací metody při fixed-point aritmetice (převzato z [26])

Druhou zásadní úlohou při výpočtu v pevné řádové čárce je problém přetečení (overflow) či podtečení datového typu. K přetečení resp. podtečení datového typu dochází vždy, když je výsledek jakékoliv aritmetické operace větší resp. menší než maximální resp. minimální velikost datového typu.

Řešením tohoto problému může být saturace výsledku na maximální nebo minimální hodnotě datového typu. To však vnáší do výpočtu další chybu. Zároveň proces saturace obnáší několik matematických operací navíc (porovnávání čísel) a tak je potřeba při použití saturace počítat s vyšší výpočetní náročností algoritmu. V některých případech není nutné problém přetečení hlídat, protože přetečení je z hlediska funkce žádoucí (sčítačka bez přenosu, atd.). V případě, že jsou výsledky aritmetických operací v mezích datového typu může být povoleno přetečení. Je však nutné zajistit při každém opakovaní výpočtu stejné podmínky (vstupní signál nesmí růst nad povolenou mez, čímž by zapříčinil výsledné přetečení). Vhodným prostředkem pro tyto případy je použití lokálního omezení hodnot na všech nebo kritických vstupech.

3 Sběrnice CAN a protokol CANopen

Základním komunikačním prostředkem navrhované elektronické řídicí jednotky s dalšími komponentami systému Fly-By-Wire (hlavní řídicí počítač, sesterská řídicí jednotka, senzorové jednotky, atd.) je komunikační standard CAN.

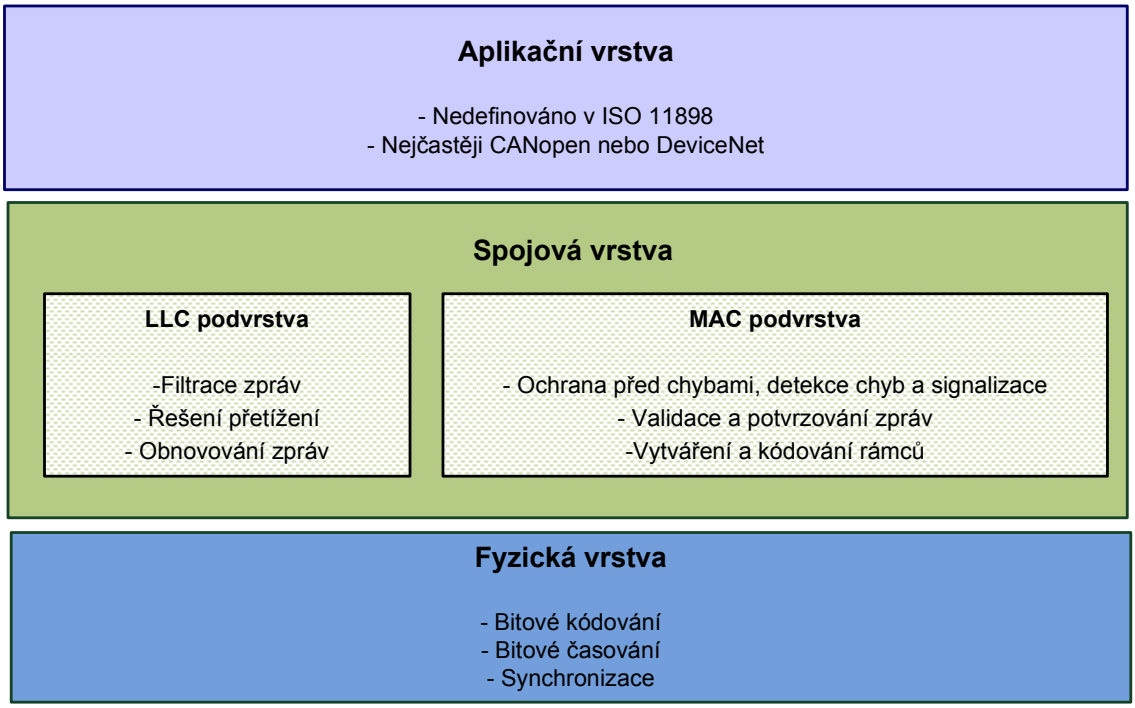
Každá ECU jednotka obsahuje dva nezávislé řadiče CAN. Díky těmto dvěma nezávislým CAN řadičům je možné koncipovat komunikaci řídicího systému EHSA dvoukanálově, tak aby bylo zaručeno, že při poruše jedné z komunikačních cest nedojde k úplnému přerušení toku informací mezi komponentami FBW systému. Každý komunikační kanál řídicího systému je navíc zdvojen a připojen na obě sběrnice CAN1 a CAN2, tím vznikne křížové propojení (viz obr. 1.2).

Při návrhu komunikační vrstvy ECU se bylo nutné seznámit jak s fyzickou a spojovou vrstvou CAN, tak s vyšším protokolem CANopen. V následujících podkapitolách je proto uveden přehled a základní popis standardu CAN a protokolu CANopen, který byl nezbytný pro návrh hardwaru a softwaru komunikační části řídicí jednotky ECU. Další podrobnosti ohledně protokolu CAN a aplikační vrstvy CANopen lze nalézt v [30].

3.1 Popis sběrnice a fyzické vrstvy CAN

Controller Area Network neboli CAN je sériový komunikační protokol určený pro distribuované řídicí systémy. Je navrhnut tak, aby ho bylo možné použít v systémech reálného času. Navíc poskytuje vysoký stupeň ochranných a bezpečnostních mechanismů.

Primárním uplatněním CAN sběrnice je sice automobilový průmysl, nicméně jeho relativní jednoduchost a spolehlivost jí předurčili také k širokému uplatnění v leteckých dopravních systémech. V dnešní době existují dvě standardizované, vzájemně kompatibilní verze protokolu CAN a to verze *1.2* a *2.0*. Návrh ECU jednotky vyžadoval CAN *2.0* a proto se následující kapitoly zabývají pouze touto verzí.



obr. 3.1- Vrstvový model protokolu CAN

Z hlediska architektury lze CAN rozdělit do několika vrstev:

- fyzická vrstva (Physical layer),
- spojová vrstva (Data Link layer),
- aplikační vrstva (Application layer).

Spojová vrstva se skládá z dvou podvrstev:

- Logické spojové vrstvy (Logical Link Control layer) - LLC,
- Přístupové vrstvy (Medium Access Control layer) - MAC.

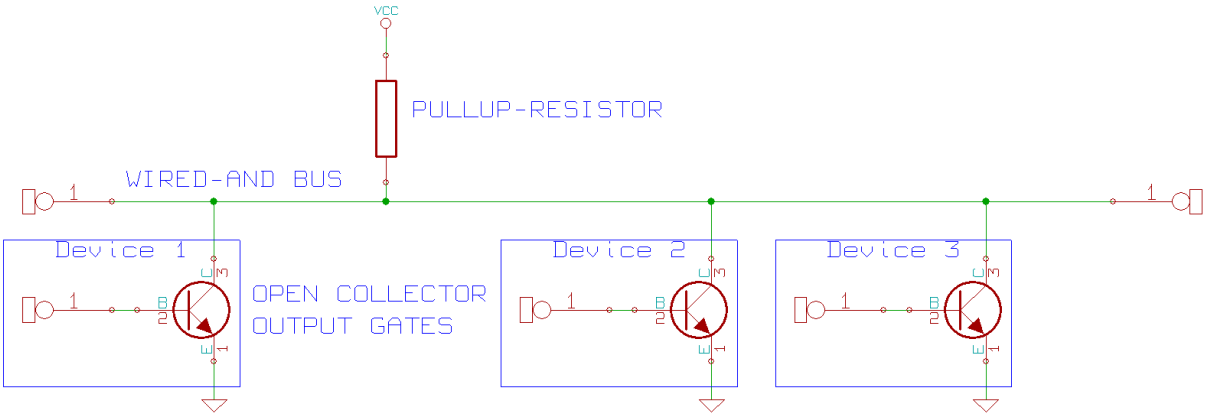
Funkce LLC a MAC vrstvy jsou popsány v kapitole 3.2. Aplikační vrstva není, jako jediná ze všech vrstev protokolu CAN, definovaná a lze ji pro danou aplikaci implementovat podle požadavků na vyvíjený systém. Standard CAN (ISO 11898), tak předepisuje jen přesnou podobu fyzické a spojové vrstvy. Jednotlivé členění vrstev CAN i s přehledem nejdůležitějších funkcí lze nalézt na obr. 3.1.

Protokol CAN zaručuje a definuje následující vlastnosti komunikačního systému:

- prioritním řazením zpráv,
- garanci doby zpoždění,
- přenos typu *multicast* s možností synchronizace,
- síť typu *multimaster*,
- detekci a signalizaci chyb,
- automatické znovuvyslání porušené zprávy,
- automatické odpojení porušeného uzlu ze sítě a určení druhu chyb.

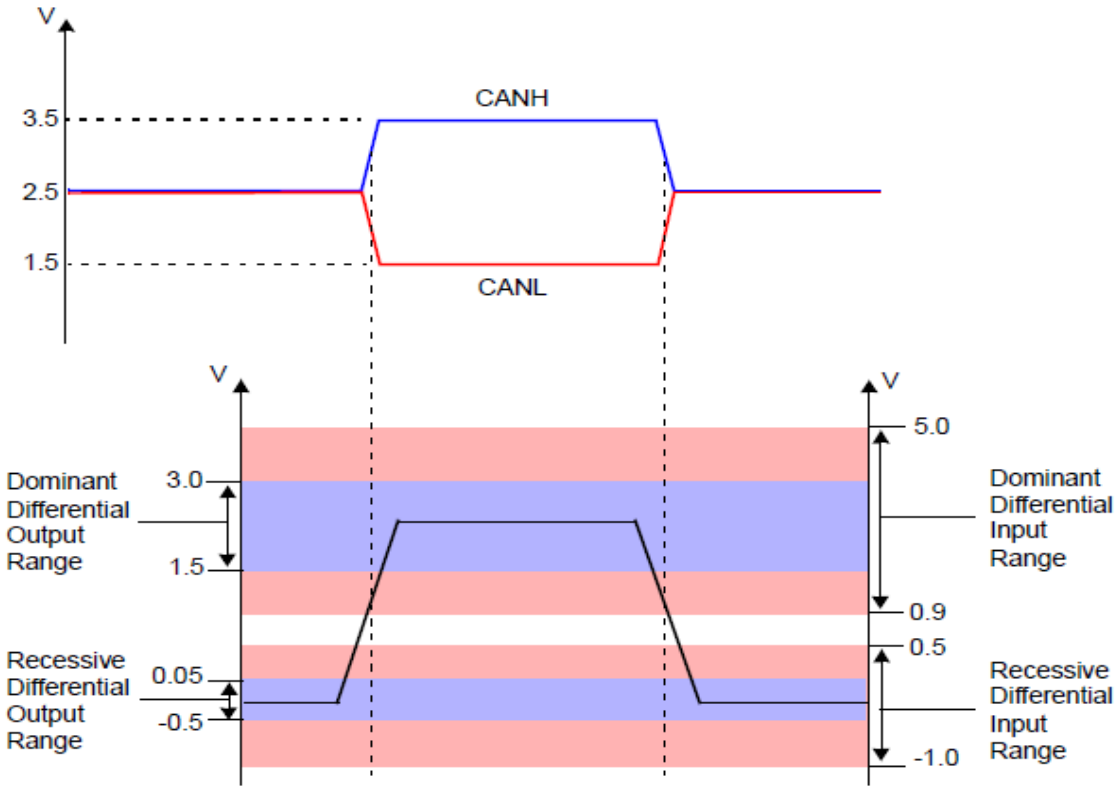
Fyzická vrstva definuje způsob a formu, jakou mají být signály přes médium posílány. Standard CAN definuje bitové kódování, časování a synchronizaci. Neurčuje však přesnou velikost napěťových signálů ani druh přenosového média. Tyto prvky mohou být různé podle typu aplikace. Standard ISO 11898 předepisuje několik typů fyzických konfigurací protokolu CAN, nejpoužívanější jsou standardy ISO 11898-2, ISO 11898-3, SAE J2411 či ISO 11992.

Základním požadavkem na fyzickou vrstvu CAN je zajištění funkce logického součinu *Wired-And*. Ten je základem jak pro samotný přenos zprávy, tak pro prioritní arbitraci při vysílání. Princip *Wired-And* komunikace je vidět na obr. 3.2. Na sběrnici se vždy vyskytují dvě logické úrovně (dominantní a recesivní). Napěťové hodnoty recesivní a dominantní úrovně jsou pak závislé na konkrétním typu budícího obvodu. Díky funkci logického součinu jsou pravidla pro určení aktuálního stavu (úrovně) na sběrnici jasně definovatelná, v případě že vysílá alespoň jeden uzel na síti dominantní úroveň je na sběrnici dominantní úroveň, v případě že všechny uzly vysílají na sběrnici recesivní bit je na sběrnici recesivní úroveň.



obr. 3.2- Principiální schéma logického součinu na CAN sběrnici

Vstupním požadavkem na ECU je schopnost komunikace v konfiguraci ISO 11898-2. Tento standard vyžaduje aby vyvíjené zařízení používalo vysílač s funkcí *High-speed* do přenosové rychlosti *1 Mbit/s*, dvouvodičovou diferenciální sběrnici se zpožděním signálu maximálně *5ns/m* (pro *1 Mbit/s* je pak maximální délka *40 m)*, nominální impedanci vedení *120 Ω*, napěťové úrovně na vodičích *CAN_H = +3,5 V* a *CAN_L = 1,5 V* s napěťovou odolností vysílačů od *-3 V* ač *32 V*. Na obr. 3.3 ze vidět doporučený průběh a hodnoty signálů *CAN_H* a *CAN_L* v konfigurací ISO 11898-2. V recesivní úrovni jsou potenciály vodičů *CAN_H* a *CAN_L* stejné nebo je rozdíl potenciálů menší než *0,5 V*. Dominantní úroveň nastává při rozdílu *CAN_H – CAN_L > 0,9 V*.



obr. 3.3- Dynamické vlastnosti CAN sběrnice (převzato z [30])

3.2 Spojová vrstva Data link a její podvrstvy

CAN definuje kromě části fyzické vrstvy také spojovou vrstvu Data link. Spojová vrstva se stará o kompletní spojeni mezi dvěma sousedními CAN systémy, uspořádává datové zprávy z fyzické vrstvy do rámců a zajišťuje arbitraci.

Z hlediska funkce spojové vrstvy ji lze rozdělit na dvě podvrstvy MAC a LLC. Podvrstva MAC je jádrem celého standardu CAN ISO 11898. Vrstva MAC zajišťuje zabalování datových zpráv do rámců, kódování zpráv, řízení přístupu na sdílené médium, stará se o detekci a signalizaci chyb a obsahuje potvrzovací mechanismus. Zároveň MAC vrstva obsahuje mechanismus pro automatické opravy při vzniku chyby (Fault Confinement mechanism) díky němuž dokáže komunikační systém CAN rozlišit mezi krátkodobou a dlouhodobou poruchou.

Podvrstva LLC pak zabezpečuje proces obnovení a znovuzasílání poškozené zprávy. Zároveň LLC poskytuje službu sledování propustnosti sítě a filtraci přijímaných zpráv.

3.2.1 Arbitráž, zprávy a rámce

Datové zprávy posílané na sběrnici mají vždy stejný formát a strukturu, jediné v čem se zprávy od sebe mohou lišit je jejich celková délka. Z hlediska přístupu na médium a mechanismu zasílání zprávy na sdílené médium se používá metoda CSMA/CR (Carrier Sense Multiple Access/ Collision Resolution), někdy též nazývané CSMA/BA (Carrier sense multiple access with bit arbitration). Díky bitové nedestruktivní arbitráži je tak možné i v náhodné přístupové metodě, jako je CSMA, zaručit některé důležité rysy deterministické komunikace a definovat maximální zpoždění zpráv při kritických úkolech.

Metoda přístupu komunikačního uzlu na médium lze popsat následujícím algoritmem. Chce-li nějaký komunikační uzel vysílat na sběrnici musí nejdříve zjistit aktuální stav sběrnice. V případě, že je sběrnice volná začne vysílat data. V případě že v daném okamžiku začnou vysílat dvě a více jednotek najednou uplatní se proces bitové arbitrace (nejdříve se posílá identifikátor zprávy, na kterém se vykoná bitová arbitrace). Bitová arbitrace se provádí na základě hardwarového logického součinu *Wired-And*, díky němuž dochází k postupné prioritní selekci komunikačního uzlu. Výhodou tohoto arbitrážního principu je výrazná úspora času při rozhodování o pořadí vysílaných zpráv. Nevýhodou je pak nutnost použití delších datových zpráv (dlouhý identifikátor).

Arbitrace při vysílání dvou a více uzlů je následující. Každý komunikační uzel si postupně při posílání své zprávy zpětně čte bitovou hodnotu na sběrnici a porovnává ji ze svou poslanou hodnotou. V případě, že se tyto hodnoty shodují, jednotka pokračuje v přenosu zprávy na sběrnici, v případě že je na sběrnici jiná hodnota než jednotka zaslala, ukončí přenos svého datového rámce a přepne se do naslouchacího módu.

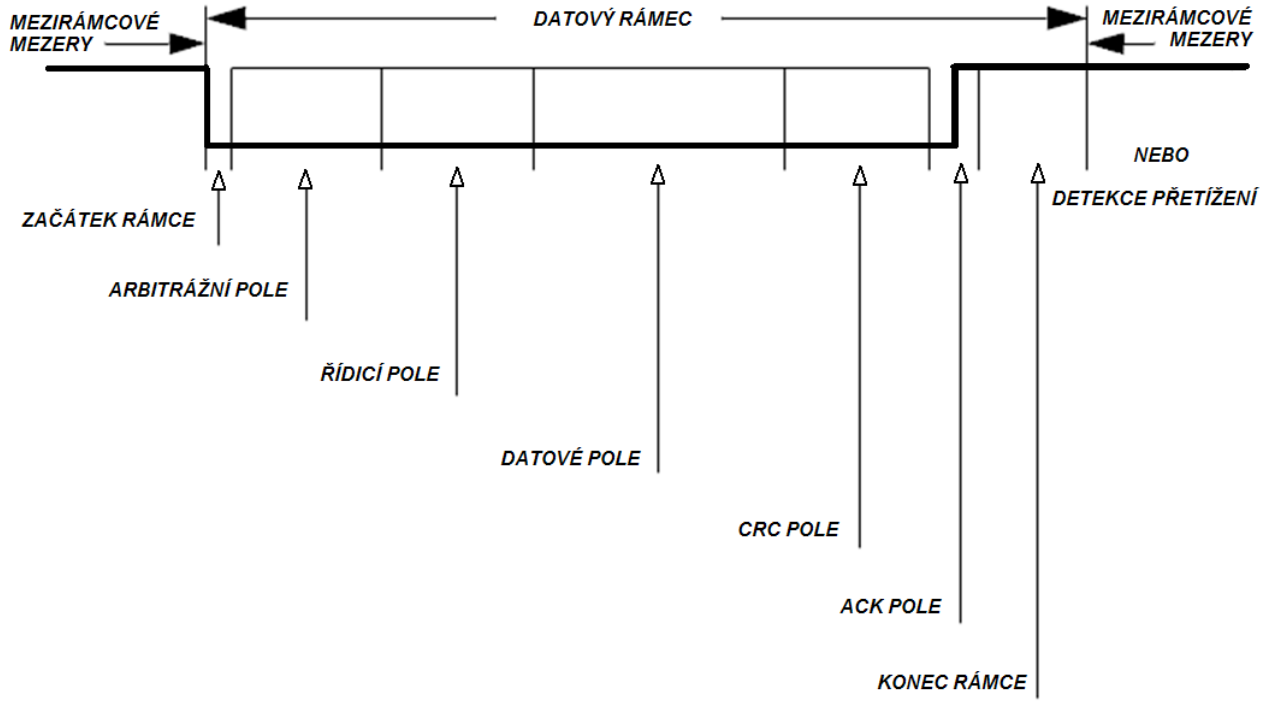
Z hlediska druhů zasílaných zpráv lze rámce rozdělit na dva typy:

- Standardní rámec s 11 bitovým identifikátorem,
- Rozšířený rámec s 29 bitovým identifikátorem.

Formát obou výše zmíněných rámců se liší podle účelu a druhu přenášených informací. Protokol CAN definuje 4 druhy rámců:

- Datový rámec (Data frame) –přenos dat od odesilatele k příjemci,
- Vzdálený rámec (Remote frame) – žádost o datový rámec s daným identifikátorem,
- Chybový rámec (Error frame) – vysílání signalizačních zpráv při vzniku chyby,
- Ochranný rámec proti přetížení (Overload frame) – speciální zpožďovací rámce.

Formát datových rámců je strukturován do sedmi částí (obr. 3.4). První jednobitová část SOF (Start of Frame) má vždy dominantní úroveň a slouží k inicializaci komunikace na CAN sběrnici. Druhá část obsahuje arbitrážní informace (Arbitration field). Arbitratrážní část se dělí na dvě podčásti identifikátor zprávy a bit RTR (Remote Transmission Request). Délka identifikátoru je závislá na typu rámce (rozšířený nebo standardní rámec). V případě standardního rámce je identifikátor dlouhý 11 bitů, v případě rozšířeného rámce pak 29 bitů. RTR bit slouží k rozlišení druhu vysílaného rámce. V případě dominantní úrovně se jedná o datový rámec (Data frame) v opačném případě RTR bit signalizuje vzdálený rámec (Remote frame).



obr. 3.4 – Formát datového rámce

Třetí částí datového rámce je šestibitové řídicí pole (Control field). To obsahuje informaci o počtu přenášených datových bytů. Protokol CAN definuje maximální počet datových bytů, pro případy větších objemnějších zpráv je nutné zajistit v aplikační vrstvě zřetězení více rámců dohromady.

Čtvrtým polem uvnitř CAN rámce je datové pole (Data field). To slouží k přenosu datových informací mezi odesilatelem a příjemcem. Délka tohoto pole může být 0 až 8 bytů. Pořadí jednotlivých bitů při vysílání je nejdříve od MSB bitu až po LSB bit. Následujícím polem je 15-ti bitové kontrolní pole CRC (Cyclic redundancy code). Toto pole slouží k detekci a ochraně přenášených dat před nahodilou chybou. Díky použití polynomu řádu 15 zaručuje CAN že, pravděpodobnost příjmu nedetekované chybové zprávy je menší než $4{,}7.10^{-7}$.

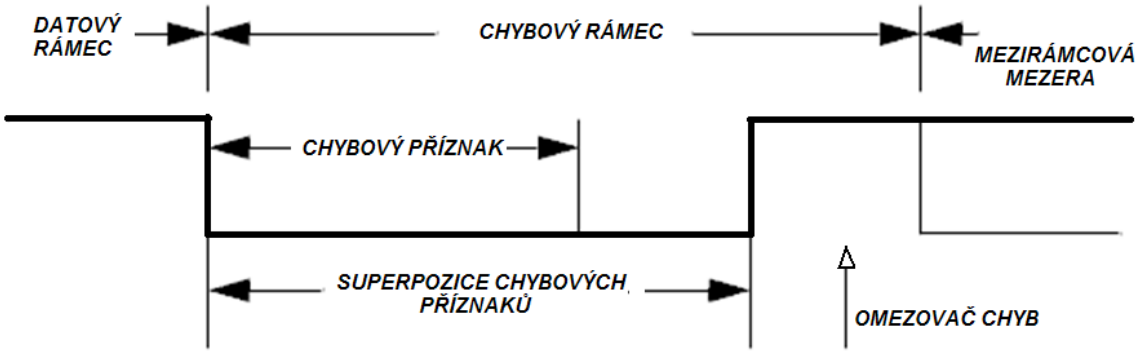
Předposledním polem v CAN rámci je potvrzovací pole ACK. Toto pole je dlouhé 2 bity. Vysílač musí zajistit aby tyto bity měli při vysílání zprávy recesivní úroveň. Naopak příjemce zprávy při správnosti přijatého rámce vyšle dominantní úroveň, čímž potvrdí příjem vyslané zprávy.

Posledním povinným polem v rámci je zakončovací pole EOF (End of Frame). Toto pole ukončuje bitovou sekvenci zasílaného rámce. Konec rámce je vytvořen posloupností sedmi po sobě jsoucích recesivních bitů.

V případě vzdáleného rámce (Remote frame) je mechanismus posílání zpráv postaven na principu žádosti o zaslání určitých dat. Jednotka, která vystupuje jako přijímač posílá data až tehdy, když je o ně požádána určitým vysílačem. Žádost o určitá data se provádí pomocí vzdáleného rámce.

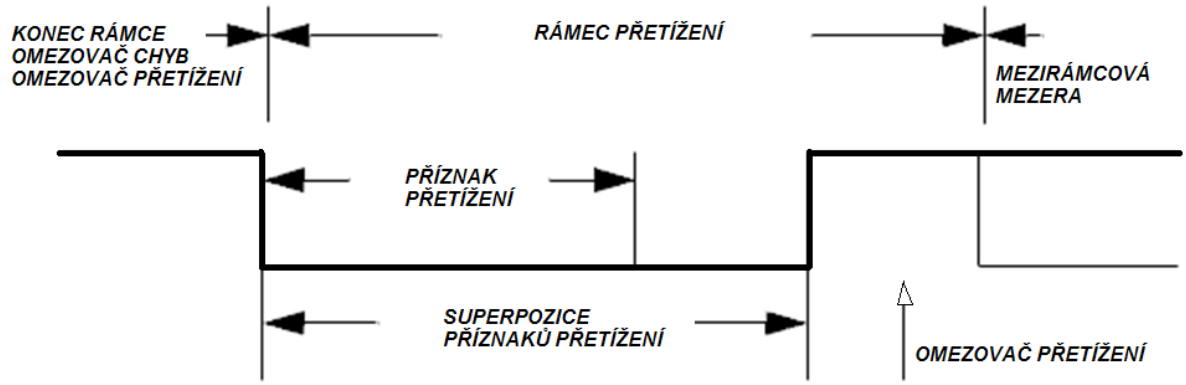
Vzdálený rámec má podobnou strukturu jako datový rámec ale na rozdíl od něho neobsahuje datovou oblast. Druhou odlišností je nastavení RTR bitu na recesivní úroveň.

Mimo datový a vzdálený rámec se používá chybový rámec (Error frame). Ten je určen výhradně k důležitým signalizačním účelům. Struktura chybového rámce se skládá ze dvou částí (obr. 3.5). První část vznikne superpozicí chybových příznaků jednotlivých stanic na síti, druhá část je pak tzv. zakončovací, skládající se z 8-bitového recesivního pole. O vzniklé chybě daného uzlu se pak ostatní jednotky dozvědí pomocí mechanismu řízené destrukce polí ACK, EOF, SOF či narušením bitového kódování (vznik nepravidelnosti v bit stuffingu).



obr. 3.5– Formát chybového rámce

Posledním typem rámce je ochranný rámec při přetížení (Overload frame).Tento typ rámce zajišťuje žádané zpoždění mezi vysíláním a příjmem dalšího rámce na CAN. Struktura tohoto rámce je opět složena ze dvou nezávislých částí. První část je složená z příznaků přetížení jednotlivých stanic a částí zakončovací. Formát tohoto rámce je nakreslen na obr. 3.6. Principiálně je mechanismus ohlašování přetížení podobný jako u ohlašování chyb. Opět zde dochází k řízené destrukci. Jednotlivé uzly na síti jsou informováné o vzniku přetížení pomocí destrukce části bitového pole mezirámcového klidového místa (Interframe space).



obr. 3.6– Formát rámce typu přetížení

Společnou vlastností výše zmíněných rámců, jak již bylo uvedeno, je pole identifikátor zprávy. Tento identifikátor slouží pro adresaci příjemce zprávy resp. prioritní arbitrážní proces během vysílání a příjmu. Protokol CAN dovoluje, aby jednotlivé uzly na síti mohli během přijímání využít filtrace jednotlivých zpráv (filtr na identifikátor). Tato filtrace dovoluje každému komunikačnímu uzlu ignorovat buďto určitou skupinu nebo konkrétní typ přijímané zprávy. Díky hardwarové implementaci filtračního mechanismu navíc nedochází k časovým prodlevám a je možno dělat prioritní rozhodování a filtraci v jeden časový okamžik.

3.3 Aplikační vrstva protokolu CAN

Většina dnešních průmyslových komunikačních standardů, jakou je i komunikace po CAN sběrnici, definuje jen některé ze 7 doporučených vrstev v referenčním modelu ISO/OSI. Standardy většinou specifikují jen první fyzickou vrstvu nebo její podstatnou část, dále druhou spojovou vrstvu a v některých případech i sedmou aplikační vrstvu. Vrstvy mezi druhou a sedmou vrstvou nejsou v průmyslových sítích ve většině případů potřeba, protože se skládají z jednoho komunikačního segmentu nebo používají jednotný formát dat a adresování.

Podobně je tomu i u standardu CAN. Ten navíc specifikuje jen druhou spojovou a část první fyzické vrstvy. V případě prvních dvou vrstev je implementace CAN protokolu do vyvíjeného zařízení poměrně jednoduchá, protože existuje široké spektrum integrovaných obvodů implementujících všechny prostředky spojové a fyzické vrstvy. Tyto CAN řadiče dovolují velice snadno zvolit nastavení přenosových parametrů jako je rychlost, časování nebo vzorkování.

Pro většinu praktických aplikací je ale nutné implementovat ještě sedmou vrstvu, která by zajišťovala rozhranní pro komunikaci s finální aplikací a zprostředkovávala předávání a vysílání dat na komunikační síť. Z pohledu návrháře systému využívajícího CAN je tak nutné vymyslet sedmou aplikační vrstvu nebo využít jednu z několika dostupných typů již implementovaných aplikačních protokolů. Nejběžnější typy aplikačních vrstev používaných v oblasti dopravních systémů a vestavných zařízení jsou protokoly CANopen a DeviceNET. V prvním případě navíc existuje řada volně šířených verzí tohoto standardu.

Hlavním požadavkem na komunikační vlastnosti ECU byla schopnost komunikace pomocí protokolu CANopen. Pro implementaci protokolu CANopen je vhodné se seznámit se základními rysy tohoto standardu.

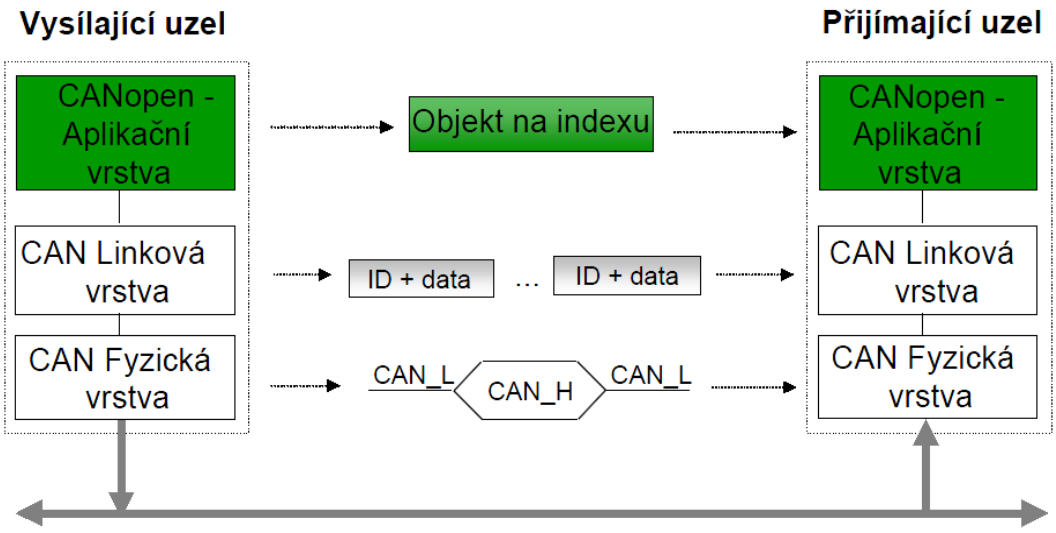
3.3.1 Protokol CAL a vrstva CANopen

V dnešní době existuje hned několik typů aplikačních vrstev standardu CAN. Pro oblast vestavěných systému je nejpoužívanějším typem protokol CAL (CAN Application Layer). Účelem aplikační vrstvy CAL je v prvé řadě specifikovat význam jednotlivých zpráv (rámců), tj. definovat význam 11-ti bitového identifikátoru rámce a význam 8 bytů přenášených dat. Navíc je potřeba tímto protokolem zajistit vzájemnou kompatibilitu a zaměnitelnost komunikačního systému na aplikační úrovni. Na obr. 3.7 je znázorněn princip komunikace dvou CANopen zařízení. Vzájemná kompatibilita dvou zařízení od různých výrobců je zajištěna na úrovni linkové a fyzické vrstvy standardem CAN, na aplikační úrovni je pak zajištěna protokolem CANopen. Rozhranní mezi koncovou aplikací a spodními vrstvami CAN komunikace zajišťuje objektový slovník (viz níže).

Neméně důležitou roli aplikační vrstvy je zprostředkovat uživatelské aplikaci administraci komunikačního řetězce (adresy uzlů, deklaraci rámců, definici a význam dat). Protokol CANopen zabezpečuje všechny výše zmíněné služby. Pro snadnější přehlednost protokol CAL definuje tzv. komunikační (communication) a přístrojové (device) profily.

Komunikační profil (communication profile) určuje druh konfigurace komunikačního řetězce. Nastavením stejného komunikačního profilu u jednotlivých zařízení na síti je zajištěno vzájemnému porozumění o obsahu přenášených dat a porozumění jak jsou data mezi uzly předávána.

Přístrojový profil (device profile) určuje chování daného uzlu vzhledem k jeho specifické funkci. Ve své podstatě se jedná o označení daného zařízení do jaké skupiny či třídy systémů patří. Protokol CANopen zná hned několik druhů přístrojových profilů, nejčastějšími skupinami jsou profily digitálních a analogových vstupů a výstupů, enkodéry a snímače nebo třída pohybových kontrolérů (motion controller).



obr. 3.7 – Princip komunikace mezi dvěma CANopen uzly (převzato z [34])

Protokol CAL poskytuje čtyři hlavní komunikační služby:

1. Specifikaci zpráv CMS (CAN-based Message Specification) – specifikuje význam, typ a velikost přenášených dat a definuje komunikační objekty a proměnné.
2. Řízení komunikace na síti NMT (Network Management) – služby pro inicializaci, spuštění, ukončení komunikace komunikačních uzlů, poskytuje služby pro detekci závad daného zařízení.
3. Distribuci adres (DistriBuTor) – služby dynamického nastavení hodnot identifikátorů zpráv.
4. Řízení komunikačních parametrů zařízení (Layer Management) – umožňuje nastavit některé parametry komunikace, jako je časování či přenosová rychlost.

tab. 3.1 – Mapování objektů protokolu CAL na COB-ID zprávy

CAN aplikační vrstva (CAL)	
COB-ID	**Použití**
0	NMT Start/Stop služby
1 - 220	CMS objekt priority 0
221 - 440	CMS objekt priority 1
441 - 660	CMS objekt priority 2
661 - 880	CMS objekt priority 3
881 - 1100	CMS objekt priority 4
1101 - 1320	CMS objekt priority 5
1321 - 1540	CMS objekt priority 6
1541 - 1760	CMS objekt priority 7
1761 - 2015	NMT Node Guarding
2016 - 2031	NMT, LMT, DBT služby

Protokol CAL sice specifikuje jak jsou jednotlivé objekty (zprávy) mapovány na základě jejich hodnoty identifikátoru COB-ID (tab. 3.1), čímž určuje jejich význam, avšak již nespecifikuje jaká data nebo proměnné budou v dané zprávě uloženy, tedy neurčuje obsah.

Pro specifikaci obsahu komunikačních objektů se používá protokol CANopen, někdy též nazývaný vrchní vrstva protokolu CAL. Tento standard je z hlediska referenčního modelu OSI na nejvyšší pozici a zaštiťuje komunikační systém na úrovni aplikace.

Protokol CANopen používá určitou množinu služeb CAL, definuje datové typy, kódování dat, význam zpráv a mapování datových objektů aplikace do komunikačních objektů CAL. Pro flexibilní mapování dat aplikace do objektů CAL se používá mechanismus tzv. objektového slovníku (Object Dictionary).

3.3.2 Datové typy, objekty a dekódování

Pro správnou interpretaci komunikačních dat mezi příjemcem a vysílačem po síti CAN je nutné definovat jejich význam a bitovou strukturu, tak aby při dekódování jednotlivých bitů přijatých dat nedošlo k chybnému sestavení původní informace. Protokol CANopen definuje několik základních a složitějších datových typů a předepisuje, jak mají být vysílané a přijímané datová bity dekódovány.

Každá posílaná informace či hodnota se skládá z bitové sekvence. Tato sekvence se následně dělí vždy na osmice bitů nazvané byte. Pro stejnou interpretaci těchto bytů na přijímací a vysílací části se používá dekódování *Little endien*. Formát *Little endien* předepisuje, že data jsou rozdělena po bytech (8 bitech) a pořadí uložení těchto bytů je vždy takové, že na paměťové místo s nejnižší adresou se uloží (řádově) nejméně významný byte. Stejného principu se užívá, také pro pořadí vysílání jednotlivých datových bitů, nejdříve se vysílá nejméně významný byte a poté více významné byty. Jednotlivé bity v každém bytu se vysílají vždy od nejvyššího k nejnižšímu. Princip bitové sekvence při posílání po sběrnici CAN s protokolem CANopen je ukázán v tab. 3.2.

Datové typy definují vztah mezi hodnotou a použitým kódováním. CANopen předepisuje formát hned několika datových typů. Datové typy v CANopen se dělí na:

- Základní
  - BOOLEAN
  - VOID
  - UNSIGNED INTEGER
  - SIGNED INTEGER
  - FLOATING-POINT NUMBERS

- Složené (ze základních datových typů)
  - STRUCT OF
  - ARRAY

- Rozšířené (ze základních a složených datových typů)
  - OCTET STRING
  - VISIBLE STRING
  - UNICODE STRING
  - TIME OF DAY
  - TIME DIFFERENCE
  - DOMAIN

Detaily ohledně datových typů a principu dekódování lze nalézt v [34].

tab. 3.2 – Pořadí datových bitů a bytů při uložení a posílání po CANopen

Číslo oktetu	*1.*	*2.*	*k*-tý
Pořadí bitů	$b_7 \ldots b_0$	$b_{15} \ldots b_8$	$b_{8k-1} \ldots b_{8k-8}$

3.3.3 Komunikační objekty

Objektový slovník definuje základní sadu komunikačních objektů (zpráv), tato sada se nazývá komunikační model protokolu CANopen. Komunikační objekty jsou dělené podle jejich významu a funkčnosti a jejich rozdělení kopíruje služby aplikační vrstvy CAL. Na sběrnici CAN se tak mohou objevit následující typy zpráv:

- Administrativní zprávy – zprávy pro přenos informací o síti jako takové, zprávy řízení sítě (NMT), zprávy přidělování adres (DBT), zprávy konfigurace uzlu (LMT).
- Servisní zprávy – zprávy pro nastavení a inicializaci položek v objektovém slovníku daného uzlu (SDO).
- Procesní zprávy – zprávy pro přenos *real-time* dat (PDO). Přenos může být i typu *multicast* k více příjemcům.
- Speciální zprávy – zprávy pro synchronizaci (SYNC), zprávy pro signalizaci času (Time Stamp), zprávy stavu nouze (Emergency), zprávy pro signalizaci činnosti a živosti uzlu (Node/ Life Guarding), oznamovací zprávy (Boot-up)

Komunikace pomocí PDO zpráv navíc umožňuje určit vysílací režim. Tabulka tab. 3.3 shrnuje přehled všech možných vysílacích režimů PDO komunikace. Synchronní režim je režim, ve kterém jsou PDO zprávy zasílány pokaždé, když jednotka detekuje příchod speciální zprávy SYNC. Synchronní režim může pracovat ve dvou módech, a to acyklicky a cyklicky. Rozdíl v těchto módech je v okamžiku vysílání PDO. V případě acyklického módu dochází k neperiodickému vysílání PDO podle výskytu určité události, v případě cyklického režimu je vysílání PDO periodické, vždy po určitém počtu zpráv SYNC. Při asynchronním režimu dochází k vysílání PDO jen tehdy, nastane-li nějaká vnější nebo vnitřní událost. Vnější událostí se rozumí příchod zprávy RTR, vnitřní událostí pak změna hodnoty nějakého objektu v OD.

tab. 3.3 – Přehled vysílacích režimů PDO zpráv protokolu CANopen

Typ vysílání	Podmínka pro spuštění PDO (B = nutné obě, O = alespoň jedna)			Typ vyslaného PDO
	SYNC objekt	RTR zpráva	Vznik události	
0	B	_	B	Synchronní acyklické
1-240	O	_	_	Synchronní cyklické
241-251	_	_	_	Rezervováno
252	B	B	_	Synchronní vyslané po RTR zprávě
253	_	O	_	Asynchronní vyslané po RTR zprávě
254	_	O	O	Asynchronní při změně továrních dat
255	_	O	O	Asynchronní při změně profilových dat

Pro navrhovanou elektronickou řídicí jednotku bylo požadováno, aby byla schopna přenášet jak data přes PDO zprávy v synchronním režimu, tak reagovala na administrativní a speciální zprávy. Nevyžadovalo se však, aby uměla konfiguraci přes SDO zprávy. Detailnější specifikaci a popis zpráv lze nalézt v [34].

3.3.4 Objektový slovník

Objektový slovník (OD) je soubor seřazených objektů. Protokol CANopen požaduje, aby každý uzel na síti obsahoval svůj OD. V objektovém slovníku jsou pak nejen uloženy deklarace datových typů a mapování aplikačních proměnných do zpráv, ale také veškeré komunikační parametry a chování zařízení.

Protokol požaduje aby, byli v OD některé specifické položky vyplněny. Těchto položek je relativně málo a závisí na zvoleném profilu. Ostatní položky pak mohou zůstat nevyplněné. Tento fakt je důležitý zejména z hlediska implementace, protože není nutné navrhovat celou strukturu OD ale je potřeba se zaměřit jen na některé položky.

Položky, které je nutné vyplnit závisejí na typu zvoleného komunikačního profilu [31] a přístrojového profilu, např. podle [32], [33]. Nevyhovuje-li žádný dostupný standardizovaný *Device* profil, je možné zvolit libovolnou vlastní konfiguraci.

Pro vzájemnou kompatibilitu, zápis a čtení do objektových slovníku od různých výrobců bylo nutné standardizovat syntaxi a strukturu souboru, pomocí kterého dochází k předávání informací o OD daného zařízení. CANopen tento ASCII soubor nazývá EDS (Electronic Data Sheet) a předepisuje jeho striktní podobu a formát.

Popis a řazení položek v objektovém slovníku

Každý objekt zapsaný v OD má přidělenu svou adresu, 16-ti bitový index. Pro adresaci jednotlivých elementů ve složitějších objektech se navíc používá 8-mi bitový subindex. V tab. 3.4 je vidět přesná podoba OD a řazení objektů podle jejich významu (hodnoty indexů jsou uvedeny v hexadecimální soustavě).

První skupinu indexů tvoří rozsah *0001 – 001F*, kde se nalézají definice standardních datových typů, od *0020 – 0023* pak definice složitých datových typů. Objekty od indexu *0024* do *003F* jsou určeny k budoucímu použití a nejsou v této době rezervovány.

Druhou velkou skupinu indexů tvoří rozsah od *0040* do *0FFF*, kde jsou uloženy základní a složené datové typy pro přístrojový profil zařízení.

Třetí skupinu indexů tvoří indexy od čísla *1000* do hodnoty *1FFF*. V této oblasti se nalézají objekty k danému komunikačnímu profilu zařízení. Tyto objekty se nazývají komunikační položky (communication entries) a jsou obvykle stejné u většiny zařízení. Hodnoty v těchto komunikačních položkách (objektech) charakterizují parametry samotné CANopen komunikace, určují některé parametry přenosu zpráv a chybového vysílání.

Čtvrtou a zároveň nejdůležitější skupinou indexů je rozsah *2000 – 5FFF*. Do této oblasti se ukládají výrobcem používané datové objekty. Obvykle se zde mapují všechny aplikační proměnné a data určené pro přenos informací. Objekty s indexem v tomto rozsahu se následně mapují do komunikačních objektů jako např. PDO. Ze softwarového hlediska tyto objekty tvoří interface mezi aplikací a komunikačním kanálem.

Pátou skupinou jsou objekty s indexy od *6000* do *9FFF*, kde jsou uloženy objekty zvoleného přístrojového profilu. V posledním rozsahu *A000* až *FFFF* se nalézají nedefinované indexy pro budoucí použití.

tab. 3.4 – Objektový slovník protokolu CANopen

Objektový slovník protokolu CANopen	
Hodnota indexu [hex]	**Popis objektu**
0000	Nepoužívá se
0001 - 001F	Statické datové typy (Standardní typy, boolean, atd.)
0020 - 003F	Složené datové typy (předdefinované struktury, atd.)
0040 - 005F	Tovární složené datové typy
0060 - 007F	Profilové statické datové typy
0080- 009F	Profilové složené datové typy
00A0 - 0FFF	Rezervováno
1000 - 1FFF	Oblast komunikačních profilů
2000 - 5FFF	Oblast továrních proměnných
6000 - 9FFF	Oblast standardizovaných profilů
A000 - FFFF	Rezervováno

Struktura objektového slovníku

Jak již bylo výše uvedeno, OD je speciální datová struktura s předepsaným formátem. Standard CANopen předepisuje přesnou podobu hlavičky jednotlivých položek uložených v OD. Každá zapisovaná položka v OD představuje jeden datový (základní, složený nebo rozšířený) objekt nebo komunikační objekt. Každá zapsaná položka v OD je charakterizována šesti údaji. V tab. 3.5 je ukázán předepsaný tvar hlavičky jedné z položek uložené v OD.

Každý datový nebo komunikační objekt zapisovaný do OD musí mít vyplněny tyto přívlastky:

1) Index – globální očíslování dané položky v OD, jinak též pozice objektu v OD. V případě složitých nebo rozšířených objektů, jako je ARRAY, se navíc používá subindex. Index a subindex tak tvoří pár, kterým se adresují konkrétní prvky ve složitějších datových typech.

2) Object – definuje druh objektu. Každý druh je charakterizován buďto textovým popisem nebo číselným kódem (Object code).

3) Name – jednoduchý textový popis charakterizující funkci daného objektu. Standard nepředepisuje minimální ani maximální délku popisu, není proto nutné toto pole vyplňovat.

4) Type – povinné pole definující jakého datového či komunikačního typu je daný objekt.

5) Attribute – určuje přístupová práva k danému objektu. Tyto práva se vztahují vždy k zařízení žádající o daný objekt, tedy externí žádost přicházející ze sběrnice. Mezi základní typy práv patří volný přístup k objektu (Read/Write access), omezené čtení (Write only access), omezený zápis (Read only access) a zakázaný zápis (Read only access, value is constant).

6) M/O – určuje, zda-li je daný objekt povinný (Mandatory) nebo volitelný (Optional), jinak řečeno je-li nutné implementovat tento objekt v zařízení. Tato vlastnost je důležitá zejména pro vzájemnou kompatibilitu a zaměnitelnost určité skupiny zařízení od různých výrobců.

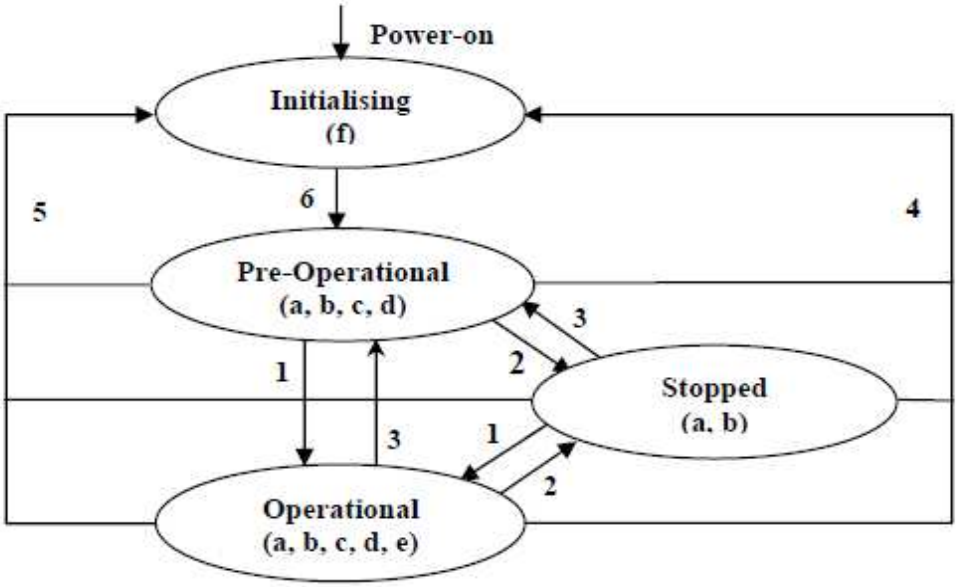
tab. 3.5 – Formát hlavičky jednotlivých položek zapsaných v objektovém slovníku

Index	Druh objektu (Object)	Popis objektu (Name)	Datový typ (Type)	Atributy (Attribute)	Povinný/nepovinný (M/O)

3.3.5 Stavový automat CANopen zařízení

Protokol CANopen, kromě definice OD, specifikuje vnitřní chování komunikačního uzlu. Standard předepisuje v jakých stavech se zařízení může nacházet při komunikaci. Stanovuje také jak, a za jakých událostí může dojít k přechodu do určitého stavu a co musí zařízení splnit při přechodu do nového stavu. Stavový automat CANopen zařízení je nakreslen na obr. 3.8. Stavy jsou nakresleny elipsami, přechody mezi stavy pak lomenými čárami s šipkou. Čísla u těchto čar popisují činnost uzlu při přechodu do jiného stavu. Každé CANopen zařízení se může nacházet v jednom s následujících stavů:

- Inicializace (Initialising)– nestabilní stav, při kterém dochází k vyslání zprávy Boot-up.
- Předprovozní (Pre-operational) – stabilní stav, uzel čeká na spuštění do provozního nebo zastaveného stavu. CANopen komunikace neprobíhá.
- Provozní (Operational) – Stabilní stav, při kterém je uzel schopen odpovídat na CANopen komunikaci. Možný přechod do dalších stavů.
- Zastavený (Stopped) – Stabilní stav, při kterém uzel není schopen odpovídat na CANopen komunikaci ale pasivně naslouchá na CAN sběrnici. Čeká na opětovné spuštění.



a. NMT, **b.** Node Guard, **c.** SDO, **d.** Emergency, **e.** PDO, **f.** Boot-up

State transitions (1-5: initiated by *NMT* services) and the NMT command specifier

1: *Start_Remote_Node (0x01)*
2: *Stop_Remote_Node (0x02)*
3: *Enter_Pre-Operational_State (0x80)*
4: *Reset_Node (0x81)*
5: *Reset_Communication (0x82)*
6: Device initialisation finished,
enter *Pre-Operational* state automatically, send *Boot-up* message

obr. 3.8 – Stavový diagram CANopen komunikačního uzlu (převzato z [34])

3.4 Projekt CanFestival

Z hlediska vývojáře protokolu CANopen existuje hned několik možností, jak aplikační vrstvu (CANopen stack) implementovat. První variantou je vyvinout aplikační vrstvu od začátku samostatně a s vlastními zdroji. Druhou cestou je použít a integrovat komerčně dodávané CANopen stacky pro danou platformu. Třetí možností je použít některý ze stávajících *opensource* projektů. V posledním případě je obvykle nutné počítat s nutností doplnit či upravit stávající kód vyvíjeného CANopen stacku o drivery k námi používané procesorové platformě.

Jedním z takovýchto *opensource* projektů je CanFestival (CF). CF je otevřený projekt implementující protokol CANopen. CF tvoří kompletní framework pro návrh CANopen komunikace do vyvíjeného zařízení. CF je šířen pod licencí LGPL (GPL) a dává k dispozici kompletní balík zdrojových kódů CANopen vrstvy včetně pomocných návrhových programů. Veškeré zdrojové kódy aplikační vrstvy jsou psány v ANSI C a jsou přeložitelné na libovolnou cílovou platformu (pokud existuje příslušný kompilační nástroj). Aktuální verzi CF lze stáhnout z repositáře z [37].

Projekt CF se skládá z několika částí:

1) Pomocné návrhové nástroje – adresář *./objdictgen*
   a. Editor a generátor objektového slovníku
   b. Konfigurační skript pro nastavení kompilačních parametrů

2) Zdrojové kódy aplikační vrstvy CANopen, nazývané též CanFestival library
   a. Hlavičkové soubory - adresář *./include*
   b. Zdrojové soubory - adresář *./src*

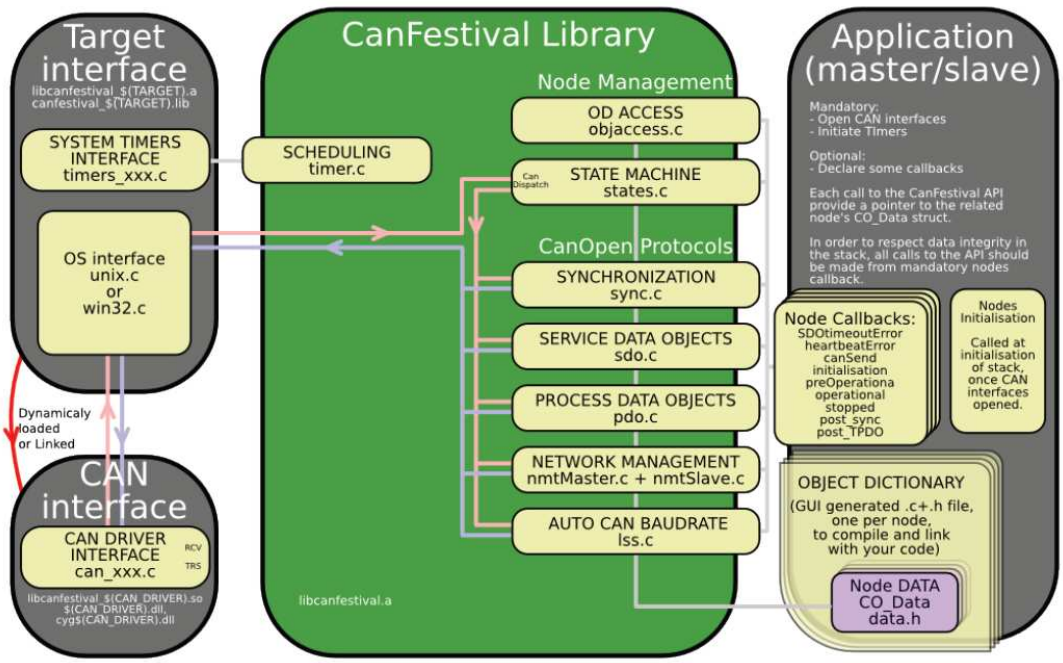
3) Zdrojové kódy ovladačů pro cílovou platformu - adresář *./drivers*

4) Dokumentace k projektu včetně HTML dokumentace zdrojových kódů vytvořená pomocí nástroje Doxygen – adresář *./doc*

5) Příklady a demonstrační programy – adresář *./examples*

Pomocí CF je možné implementaci celého CANopen protokolu do cílového zařízení omezit na implementaci platformě závislých driverů a jejich navázání na již hotové zdrojové kódy CF. Implementace objektového slovníku je totiž zajištěna z nástroje *Objdictgen*, který dokáže vygenerovat zdrojové kódy OD do jazyka ANSI C. Spojením těchto tří částí vznikne kompletní balík zdrojových kódů implementující cílový CANopen stack.

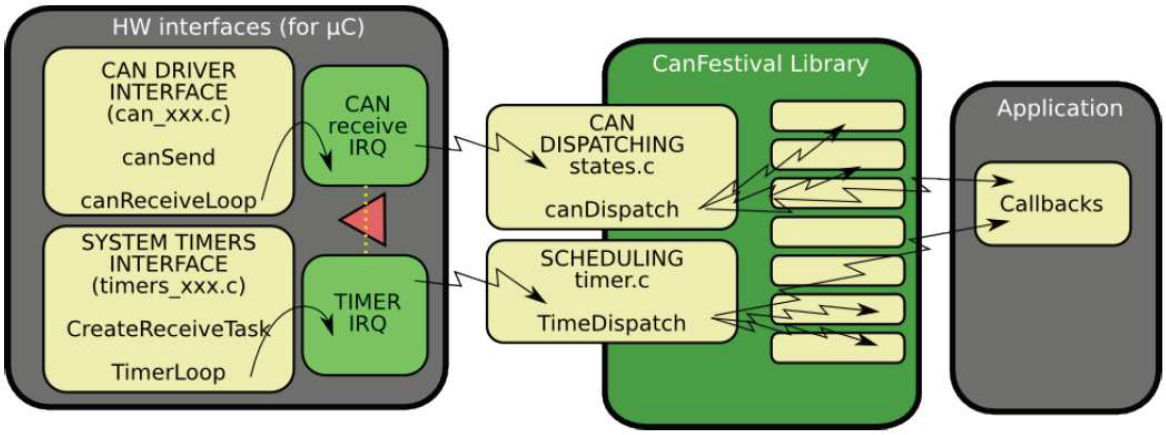
Blokové schéma a propojení jednotlivých komponent pro implementaci CANopen protokolu na cílovém zařízení je zobrazeno na obr. 3.9. Cílová aplikace nebo obslužný program komunikuje s aplikační vrstvou CANopen prostřednictvím zápisu do objektového slovníku, obslužných a řídicích funkcí knihovny CanFestival nebo prostřednictvím tzv. *callback* funkcí. Objektový slovník tvoří zároveň rozhranní mezi cílovou aplikací a softwarovým balíkem CanFestival Library.



obr. 3.9 – Blokové schéma projektu CanFestival (převzato z [25])

Knihovna CanFestival library implementuje dvě nutné komponenty pro CANopen komunikaci. První částí je implementace CANopen stavového automatu, druhá část pak implementuje samotný CANopen protokol.

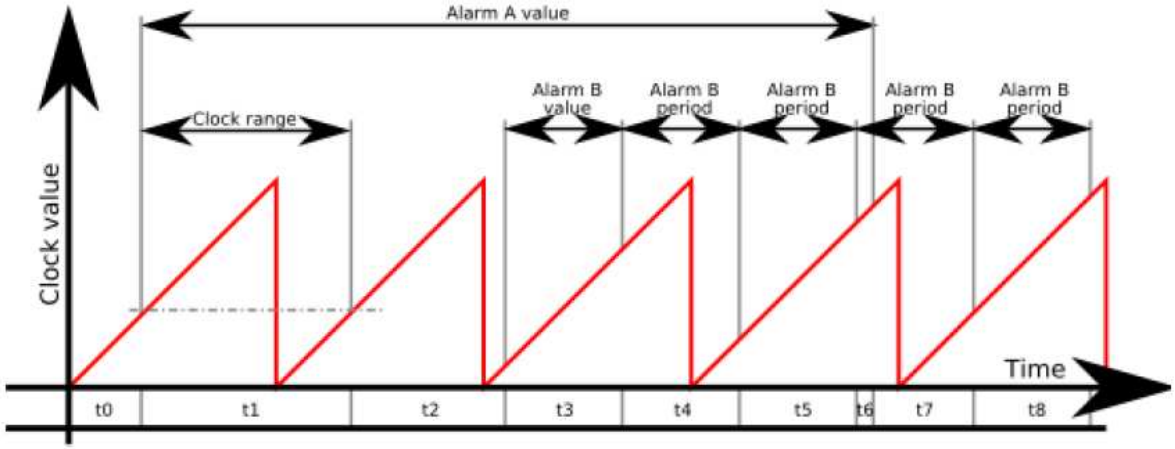
Rozhranní mezi ovladači cílového hardwaru zajišťují funkce z části *Scheduling* a *CAN Dispatching*, které zajišťují reakci aplikační vrstvy na vstupně-výstupní požadavky CAN řadiče cílové platformy. Na obr. 3.10 je znázorněn princip navázání CanFestivalu na ovladače cílového hardwaru. Použitý hardware (CAN řadič) je ovládán drivery (SW ovladači). Při příchodu nebo vysílání zpráv z CAN sběrnice zabezpečují tyto drivery čtení a posílání dat do vyšší vrstvy SW, v tomto případě do aplikační vrstvy CANopen implementované pomocí knihovny CanFestival. Rozhranní mezi drivery a aplikační vrstvou zabezpečují funkce *canSend(), canDispatch()* a *TimeDispatch()*. Funkce *canSend()* je volána z CanFestivalu při jakémkoliv požadavku aplikace na vyslání zprávy na síť. Funkci je proto potřeba implementovat a umístit do ovladače cílového hardwaru. Funkce *canDispatch()* volá ovladač cílového hardwaru vždy, když je potřeba zajistit reakci aplikační vrstvy CANopen na některou z příchozích zpráv nebo událost na CAN sběrnici. Funkce *TimeDispatch()* zajišťuje navázání CanFestivalu na časovače hardwaru, který poskytuje časování pro plánovač (Micro-scheduler) CanFestivalu.



obr. 3.10 – Princip navázání CanFestivalu na ovladače cílového hardwaru (převzato z [25])

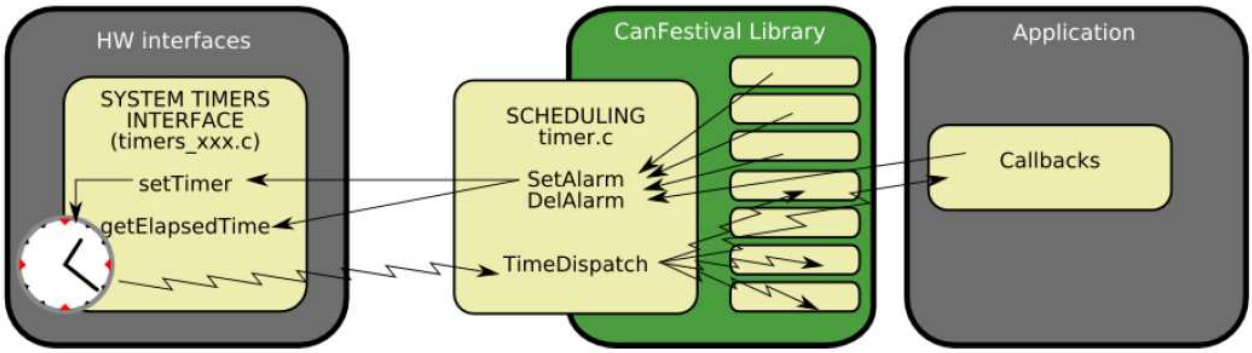
Micro-scheduler zabezpečuje časování veškerých alarmů pro CANopen funkce. Tento plánovač je implementován tak, že vytváří z jednoho dostatečně rychlého a přesného hardwarového časovače několikanásobné softwarové časování. Navíc hlídá a segmentuje dlouhé časy pro alarmy s velkou periodou. Principiálně je segmentace a spouštění alarmů ukázán na chronogramu na obr. 3.11.

Dlouhé alarmy (Alarm A) musí být obvykle segmentovány do více period hardwarového časovače, protože maximální perioda HW časovače je omezena. Například pro 16-ti bitový časovač s hodinami *4 us* je maximální perioda jen *0,26 s,* což pro dlouhé alarmy nedostačuje. U krátkých alarmů (Alarm B), které se „vejdou“ do jedné periody HW časovače, není potřeba alarm segmentovat do více period. Je však nutné hlídat vznik alarmu na rozmezí staré a nové periody (Alarm B v čase t4 na obr. 3.11). Zde totiž dochází k přetečení registru, a tím i ke skokové změně hodnoty časování (červená čára na obr. 3.11). Všechny tyto informace o segmentaci, počtu přetečení HW časovače a začátku a konci alarmu si udržuje plánovač CanFestivalu v tzv. tabulce alarmů.



obr. 3.11 – Chronogram spouštění jednotlivých alarmů v CanFestivalu (převzato z [25])

Plánovač CanFestivalu si hlídá segmentaci, tím že pravidelně čte aktuální hodnotu v registru HW časovače a počítá kolikrát došlo k přetečení. Informace o aktuálním stavu HW časovače se předává pomocí funkce *getElapsedTime()* , nastavování periody HW časovače pak pomocí funkce *setTimer()*. Na obr. 3.12 je vidět princip volání a nastavování HW časovače a plánovače CanFestivalu. Výsledná aplikace se o spuštění jednotlivých alarmů dozví pomocí příslušného zavolání *callback* funkce. U platforem s operační systémem je tento mechanismus odlišný, zde se však tímto případem nebudeme zabývat, podrobné informace o časování lze nalézt v [25].

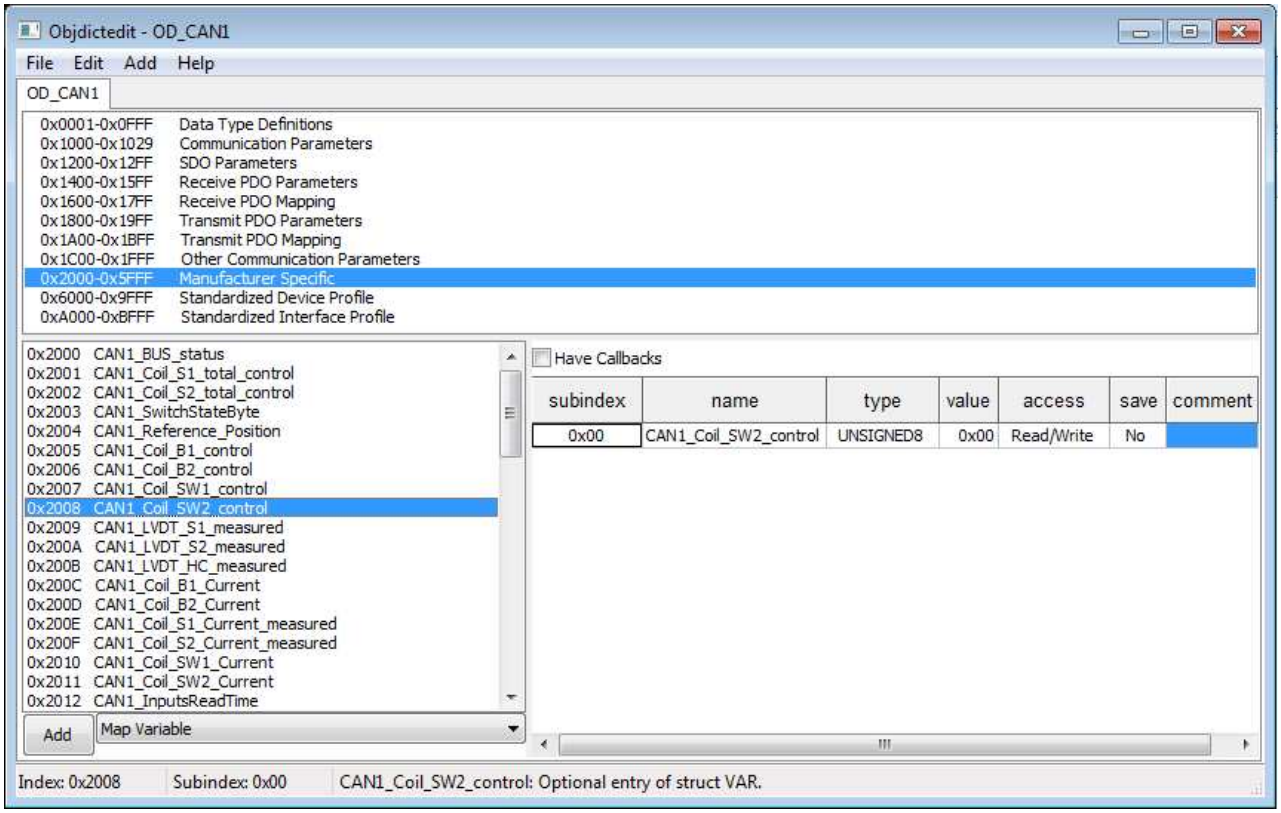


obr. 3.12 – Spouštění a propojení HW časovače a plánovače CanFestivalu (převzato z [25])

Generátor objektového slovníku

Součástí projektu CF je i nástroj na generování kompletního objektového slovníku. Program *objdictEdit* je grafický editor (GUI) vytvořený v jazyku Python. V tomto nástroji je možné vytvářet nové OD a editovat stávající OD. Navíc tento nástroj umí z editovaných OD generovat zdrojové kódy v ANSI C, které tvoří součást knihovny CanFestivalu. *ObjdictEdit* je z velké části tvořen jako vícestránkový tabulkový procesor. Nástroj přehledně zobrazuje položky v objektovém slovníku a pomocí záložek dovoluje editovat obsah jednotlivých položek. Podoba programu *objdictEdit* je vidět na obr. 3.13.

Program dovoluje také nastavit vlastnosti zapisovaných objektů, jako maximální či minimální hodnotu datových typů či počáteční hodnotu pracovních proměnných v použitých profilech. Program navíc podporuje import a export do EDS formátů, čímž umožňuje přenositelnost vygenerovaného OD do jiného nástroje od různých výrobců.



obr. 3.13 – Grafické uživatelské rozhranní pro generování objektového slovníku

4 Dvoukanálový elektrohydraulický akční člen

V této kapitole bude popsán použitý elektrohydraulický akční člen (EHSA). Bude vysvětlena funkce a princip řízení této soustavy. V závěru budou uvedeny konkrétní parametry EHSA.

4.1 Funkce EHSA

Obvod EHSA je dvoukanálový elektrohydraulický akční člen, který slouží k nastavení polohy kormidla letounu. Umístění tohoto členu a jeho propojení do systému FBW je znázorněno na obr. 1.2. Obvod na základě aktuální hodnoty vstupních proudů do cívek servoventilů nastavuje průtok tlakové kapaliny a tím polohu pístnice. Navíc tento člen dovoluje přepnout z elektrického řízení na klasické mechanické řízení s proporcionálním ventilem.

Pro možnost využití tohoto obvodu pro letecké účely je konstrukčně upraven, tak aby splňoval požadavky na potřebnou redundanci akčního členu (záloha). Díky tomu se jeho řízení a tedy i funkce dělí na možnost řídit polohu pístnice současně jedním, nebo dvěma servoventily, popřípadě v době poruchy elektrických komponent úplně přepnout na mechanické řízení.

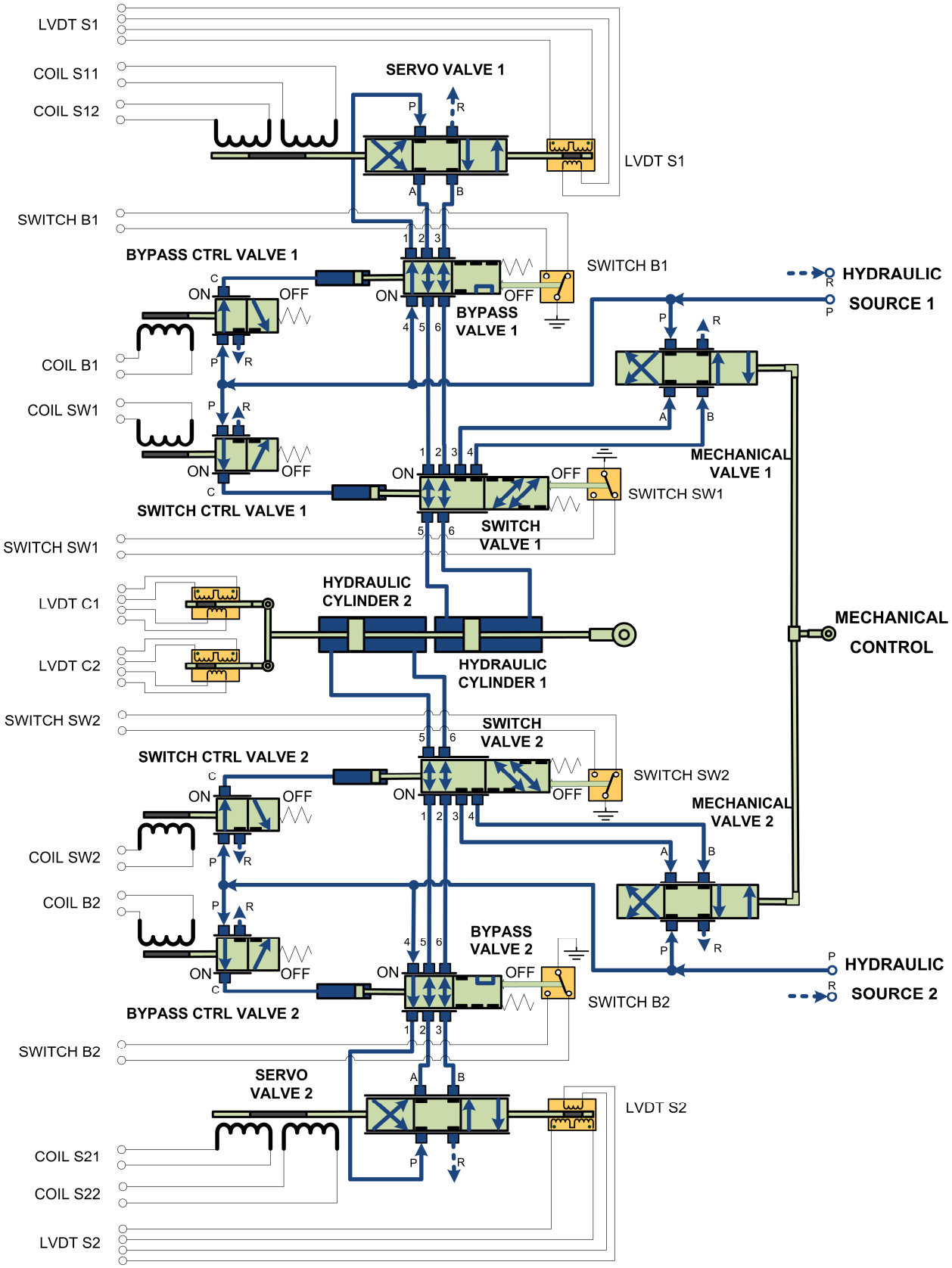
4.2 Popis a struktura EHSA

Hydraulické schéma EHSA je uvedeno na obr. 4.1. Principiálně se EHSA skládá ze dvou redundantních hydraulických okruhů se zdvojeným elektrohydraulickým řízením s centrálně uloženým zdvojeným hydraulickým válcem na společné pístnici.

Centrální částí EHSA je zdvojený hydraulický válec (Hydraulic cylinder 1 a 2), který je umístěný na společné pístnici. Posun resp. poloha těchto válců je řízena velikostí tlaku v hydraulickém potrubí každého ze dvou postraních hydraulických okruhů (dále je používán zástupný symbol $ pro označení čísla okruhu).

Každý okruh EHSA obsahuje, jak elektrohydraulickou řídicí komponentu (servoventil se zdvojeným řízením), tak čistě mechanickou řídicí část ve formě proporcionálního ventilu. Tím je zajištěno hybridní elektrohydraulické řízení. Pro případ poruchy okruhu je navíc zajištěna možnost přemostění tohoto okruhu, čímž dojde k jeho odpojení, tak aby nebránil pohybu druhého válce.

Kromě elektrohydraulických a proporcionálních ventilů a samotných hydraulických válců jsou v každém okruhu umístěny elektrohydraulické přepínače (Bypass control valve, Switch control valve), pomocí kterých je možno ovládat funkci jednotlivých ventilů a přepínat tak EHSA mezi jednotlivými pracovními režimy. Bližší informace ohledně hydraulického zapojení EHSA lze nalézt v [7] a [4].



obr. 4.1 - Hydraulické schéma EHSA (převzato z [6])

4.2.1 Princip ovládání

Řízení výstupní polohy pístnice je možné provádět, jak čistě mechanickým řízením za pomocí proporcionálního ventilu, tak elektrickým způsobem pomocí servoventilů. Dále bude věnována pozornost pouze elektrickému řízení, s ohledem na možnost využití pro elektronické řízení ECU.

Každý elektrohydraulický ventil je řízen dvěma separátními řídicími cívkami (S$1 a S$2, celkově tedy S11, S12 pro servoventil 1 a S21 a S22 pro servoventil 2). Proud těchto cívek ovládá polohu šoupátka servoventilu a tím i výslednou polohu pístnice. Dalšími vstupy pro ovládání EHSA jsou elektrické cívky B$ resp. SW$. Pomocí těchto cívek je možno řídit elektrohydraulické přepínače přemosťovacího (Bypass) ventilu daného okruhu resp. přepínacího (Switch) ventilu, čímž je možno přepínat mezi elektrohydraulickým a mechanickým řízením polohy pístnice.

Daný hydraulický okruh díky použitému zdvojení na vstupech servoventilů umožňuje využití nezávislého zdvojeného řízení, díky čemuž je možno použití až čtyřnásobného nezávislého řízení polohy pístnice. V našem případě bude použito kříženého zdvojeného řízení pro každý okruh. Z tohoto důvodu je rozděleno řízení EHSA na dva nezávislé řídicí kanály 1 a 2 pro každý ze dvou okruhů EHSA (dále je použito zástupného symbolu # pro označení čísla kanálu).

4.2.2 Výstupní měřené signály

Akční člen EHSA obsahuje zabudované LVDT snímače polohy hydraulických válců C1 a C2 a LVDT snímače polohy šoupátek obou servoventilů. Parametry LVDT snímačů jsou uvedeny v kapitole 4.3.

Kromě spojité informace o poloze pístnice a šoupátek servoventilů jsou z EHSA vyvedeny také signály mikrospínačů přemosťovacího (Bypass) a přepínacího (Switch) ventilu. Tím je možno sledovat stav sepnutí daného ventilu a kontrolovat správnou funkci EHSA. Koncové mikrospínače jsou propojeny tak, že krajní polohy přepínacích ventilů vždy zkratují jeden ze dvou kontaktů vzhledem ke společnému vodiči mikrospínače.

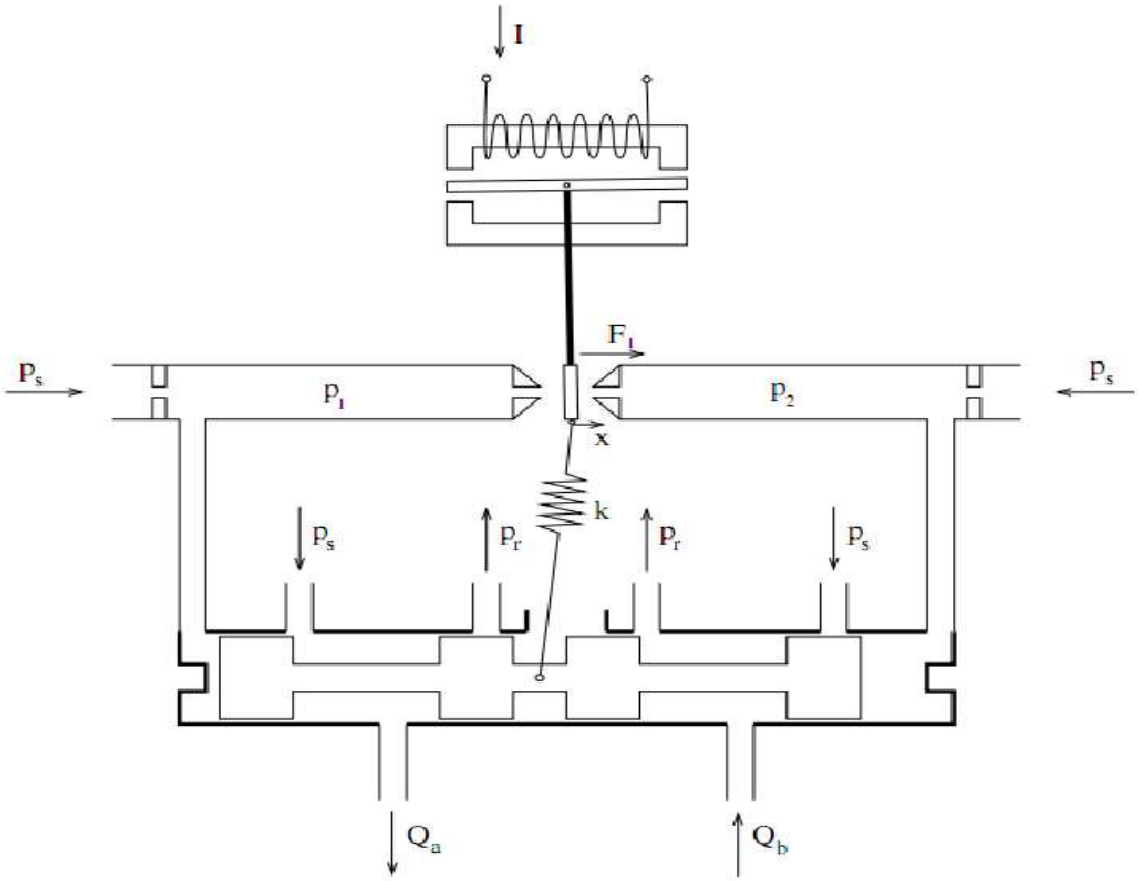
4.2.3 Elektrohydraulický servoventil

Jedním z prvků obvodu EHSA je elektrohydraulický servoventil. Obvod obsahuje dva tyto prvky, každý na nezávislé ovládání jednoho z okruhů EHSA.

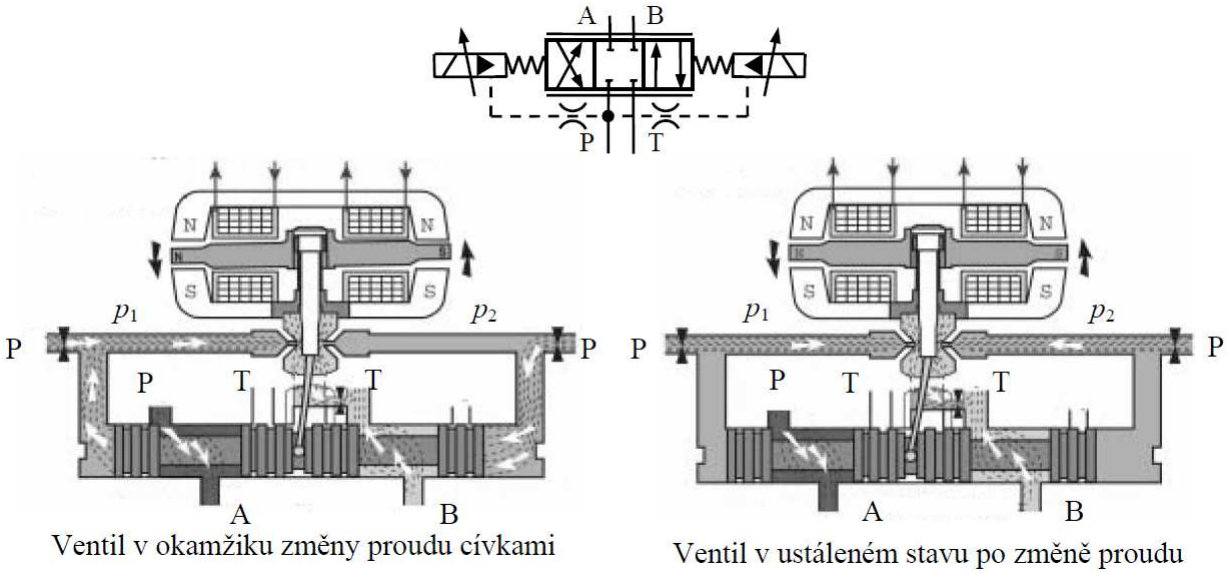
Z pohledu elektronického řízení je tento člen nejdůležitější, protože zajišťuje převod elektrické informace (proud v ovládacích solenoidech) na mechanickou energii (průtok a směr kapaliny). V případě EHSA je použit šoupátkový ventil s dvoustupňovým uspořádáním a zavedenou mechanickou zpětnou vazbou. Tím je docíleno rychlé reakce a přesnosti ovládání.

Struktura servoventilu je zobrazena na obr. 4.2. Základem elektrohydraulického servoventilu je protisměrné uspořádání typu klapka-tryska, které tvoří primární stupeň. Mezi přívodním potrubím k tryskám je umístěno šoupátko, které je propojeno ke klapce primárního stupně. Na opačné straně šoupátka je připevněna pružina, která vytváří mechanickou zpětnou vazbu.

Na straně klapky je umístěn ovládací elektromagnet, pomocí kterého může dojít k ovládání polohy klapky. Při vychýlení klapky na jednu stranu dojde na této straně ke zvýšení tlaku a na straně opačné ke snížení. Tím dojde ke změně tlaků p1 a p2 a v důsledku toho i k posuvu šoupátka. Díky pružině následně dochází ke stabilizaci pozice šoupátka (vyrovnání sil na obou stranách šoupátka). Šoupátko tak vlivem reakce pružiny, ale i suchého tření, zastaví svůj pohyb, který odpovídá požadovanému tlaku a směru toku kapaliny. Elektricky ovládaný servoventil je stejného typu jako servoventil použitý v zařízení popsaném v [3].



obr. 4.2 – Princip činnosti servoventilu v obvodu EHSA (převzato z [4])



obr. 4.3 – Dvojstupňový servoventil (převzato z [44])

4.3 Parametry EHSA

LVDT servoventilu (Servovalve) 1 a 2

- Napětí primární cívky *7 V* (efektivní hodnota, sinusový signál)
- Napětí sekundární cívky *3,6 V* (efektivní hodnota, sinusový signál)
- Pracovní frekvence *3400 Hz*
- Odpor primární cívky *250 Ω*
- Odpor sekundární cívky *545 Ω*
- Sekundární vinutí jsou zapojeny antisériově.

LVDT hydraulických válců (Hydraulic cylinder)

- Typ AF 145/75 parametry popsány v [5]:
- Maximální zdvih *± 37,5 mm*
- Reálný zdvih *± 30 mm*
- Napájení primární cívky *1÷10 V* (efektivní hodnota) pro *0,4÷12,5 kHz* (sinusový signál)
- Napětí sekundární cívky maximálně *3.3 V* (efektivní hodnota)
- Impedance primární cívky *≥ 300* Ω
- Sekundární vinutí jsou zapojeny antisériově.

Řídicí cívky (Servovalve) servoventilů

- Pracovní rozsah proudu *± 10 mA*
- Odpor vinutí *1000* Ω
- Indukčnost vinuti *2,5 H*

Řídicí cívky přepínacích (Switch valve) a přemosťovacích (Bypass valve) ventilů

- Pracovní proud *280 mA* při *28 V*
- Maximální napětí *32 V*
- Napětí připnutí *18 V*
- Napětí odpojení *3 V*
- Odpor vinutí *100* Ω *(± 8 %)*

Přepínače (Switch SW$ a B$) přepínacích a přemosťovacích ventilů

- Zabudovaný 2 polohový přepínač řízený mechanickým táhlem ventilu
- Výstupní poloha ventilu dána připojením kontaktu ke společné zemi

5 Specifikace požadavků

V této kapitole jsou shrnuty hlavní požadavky na hardware a software ECU jednotky. Požadavky je možné rozdělit do dvou hlavních skupin. První skupinu tvoří požadavky na hardware ECU. Ty vyplývají z použitého akční členu a z architektury a propojení komponent v FBW systému. Druhá skupina je tvořena požadavky na funkci a výsledné chování ECU při řízení EHSA. Chování ECU je přímo závislé na implementovaném softwaru (algoritmus řídicího systému), který musí zajistit komunikaci s dalšími komponentami FBW a schopnost plně ovládat EHSA.

5.1 Vstupní požadavky na hardware

Výsledná konstrukce ECU vychází z koncepce navrženého FBW systému, který předepisuje redundantní zapojení FBW komponent a využití zdvojené CAN sběrnice. Důležitým požadavkem na hardware ECU je také schopnost komplexně ovládat obvody akčního členu EHSA. Při návrhu ECU bylo nutné zohlednit tyto vlastnosti a požadované parametry:

- Zkřížená dvoukanálová koncepce vstupních a výstupních obvodů obou ECU.
- Vzájemné galvanické oddělení obou kanálů.
- Využití odolných a bezpečných komponent.
- Redundance komunikačního kanálu pro CAN.
- Centrální jednotku Dostatečný výpočetní výkon.
- Obvody pro řízení EHSA navrženy s ohledem na [4.3].
- Návrh obvodů pro diagnostiku řídicích signálů a *selftest* ECU.
- Dostatečný frekvenční rozsah výstupních řídicích a vstupních měřicích obvodů.

5.2 Vstupní požadavky na chování ECU

Funkce ECU jednotek je přímo závislá na řízené soustavě EHSA a na požadovaném chování v rámci FBW systému, především pak komunikaci s nadřazenými prvky FCC. Řídicí algoritmus ECU navazuje na dosavadní výsledek práce [7]. Výsledkem této práce byl návrh modelu a řídicích algoritmů pro soustavu EHSA. Zároveň zde byl navržen testovací model řídicího systému pro dvoukanálové řízení EHSA včetně stavového automatu a bloku proporcionálně integračního (PI) regulátoru. Práce se také zabývala návrhem vhodného přepínacího algoritmu EHSA. Tento algoritmus podle aktuálního stavu EHSA zajišťoval přepínání konstant PI regulátoru.

Na rozdíl od práce [7], kde byli vstupní požadavky na řídicí algoritmus dány pouze tvarem modelu EHSA, musí software pro reálnou ECU zohledňovat také použité obvody a propojení jednotlivých součástek. Obslužný SW musí zajistit kromě výpočtu samotného regulačního zásahu, také včasnou obsluhu periferních obvodů a komunikaci po CAN sběrnici.

Dalším požadavkem na software ECU je schopnost vyřešit čtení a zápis informací ze dvou komunikačních kanálů CANu. Navíc je nutné, aby obslužný software detekoval případné chyby na komunikační sběrnici a poruchy výstupních obvodů (diagnostika řídicích signálů).

V případě, že dojde k poruše nebo destrukci některé z části ECU jednotky, musí software zajistit bezpečné odstavení a zaslání informace po sběrnici CAN. To je důležité zejména pro včasnou modifikaci řídicího algoritmu sekundární ECU a nadřazené FCC jednotky.

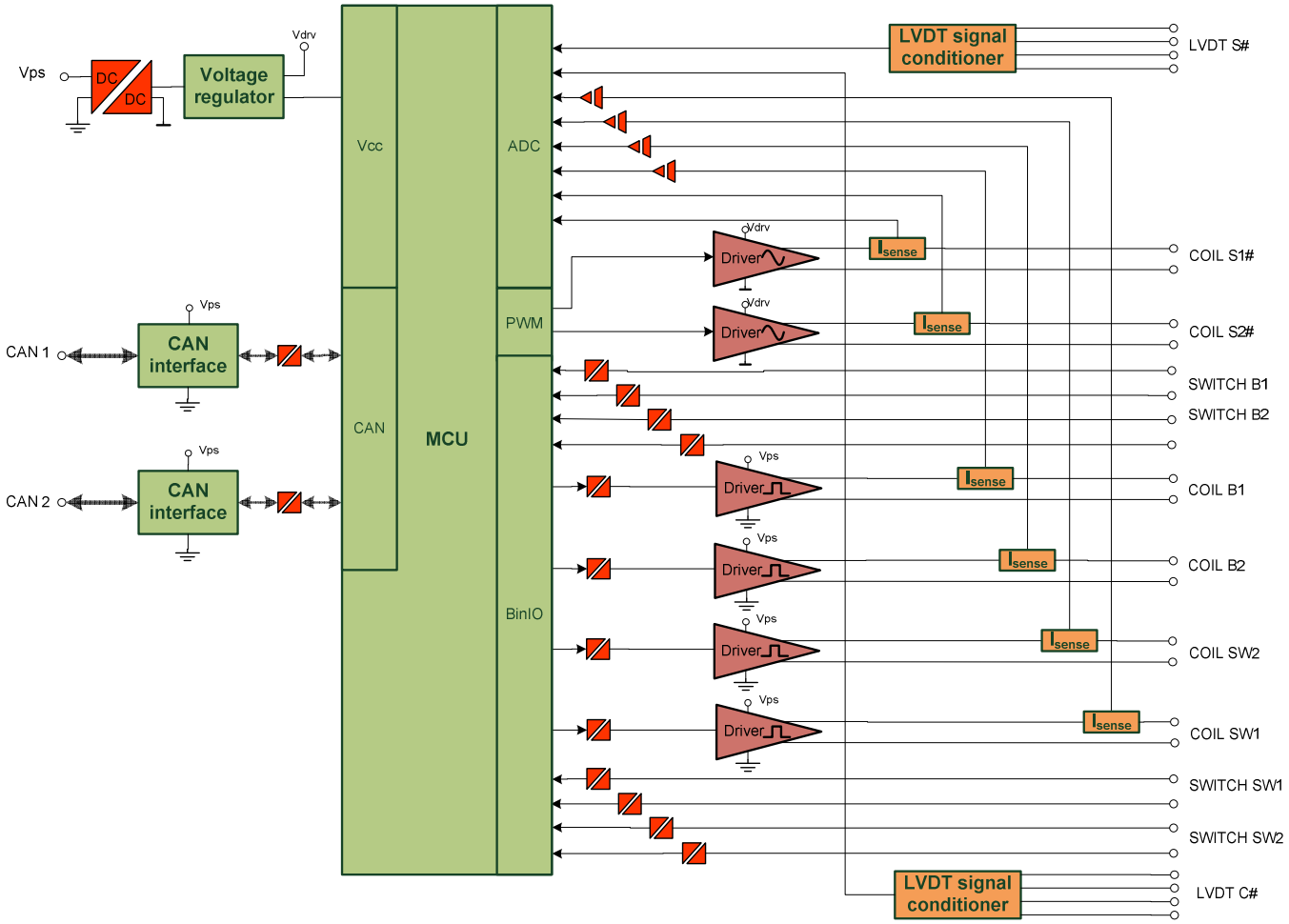
6 Hardware ECU

V této kapitole je popsána architektura a elektronické zapojení ECU. Je vysvětlen princip a funkce jednotlivých podčástí a jejich vzájemné propojení. Zároveň jsou uvedeny informace o použitých součástkách a modulech a definovány jejich přípustné hodnoty.

6.1 Architektura a konceptuální řešení

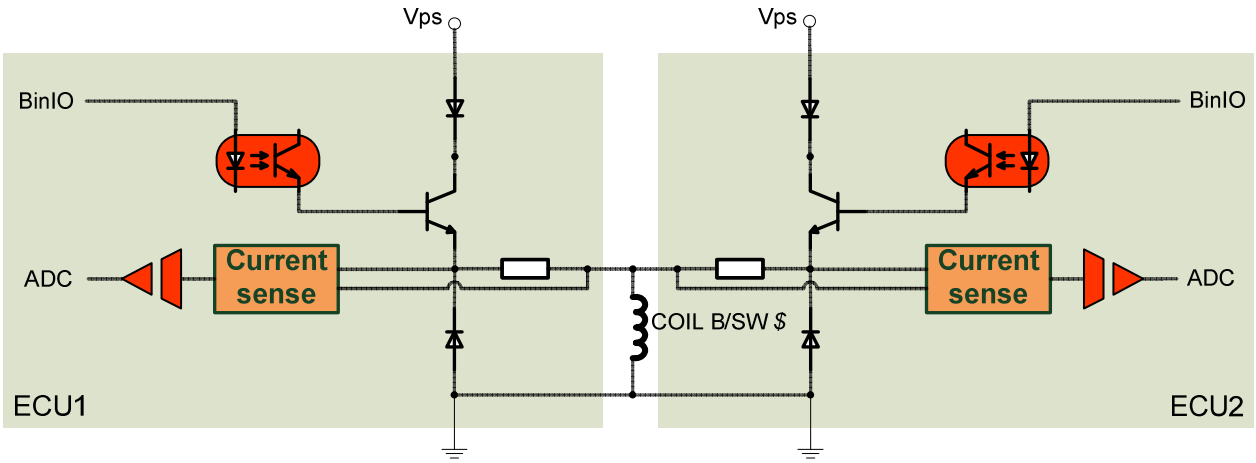
Výsledné HW vlastnosti ECU vychází z požadavků z kapitoly 5.1. Navržená koncepce jedné ze dvou (identických) ECU je znázorněna na obr. 6.1. Z hlediska HW lze ECU rozdělit na komunikační část, výstupní řídicí a vstupní měřící část. Uprostřed ECU je procesorová část, která je sdíleným prvkem pro ostatní periferní obvody. Napájení všech částí ECU zajišťuje izolovaný zdroj. Komunikační část ECU využívá standardizovanou fyzickou vrstvu typu CAN, jejíž princip a popis byl uveden v kapitole 3.

Červené prvky (dělené obdélníkové a trojúhelníkové bloky) na obr. 6.1 znázorňují oddělovací členy, které zabezpečují galvanické oddělení vnitřních obvodů ECU od vnějších periferních částí. Galvanické oddělení bude diskutováno dále v této kapitole.



obr. 6.1 - Blokové schéma dvoukanálové ECU (modifikováno podle [6])

Výstupní řídicí část ECU odpovídá potřebám pro kompletní ovládání EHSA. Díky požadavkům ovládat oba okruhy EHSA současně z obou ECU byla zvolena zdvojená prokřížená koncepce. Z pohledu volby a návrhu řídicích obvodů ECU je nutné zohlednit počet a charakter prvků EHSA. Ty jsou totiž v některých případech zdvojené a lze je přímo použít pro nezávislé redundantní řízení (např. zdvojené řídicí cívky servoventilů S$#), avšak v některých případech EHSA obsahuje části, které mají pouze jedno zastoupení (např. cívky přepínacích a přemosťovacích ventilů, LVDT snímač servoventilu) a nelze je tak použít přímo pro každou ze dvou ECU samostatně. Bylo tedy potřeba vyřešit problém sdíleného ovládání nezdvojených komponent EHSA. Navrhnutý princip sdílení je zobrazen na obr. 6.2 (v tomto případě se jedná o cívku přepínacích ventilů).



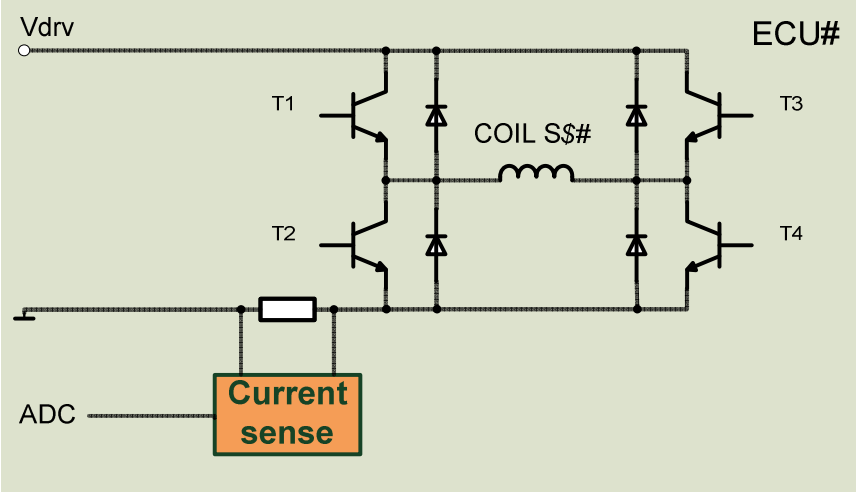
obr. 6.2 - Principielní schéma ovládání solenoidů Coil B$ a SW$ (modifikováno podle [6])

Ovládání jednonásobných částí EHSA bylo voleno tak, aby obě jednotky mohli ovládat tuto část nezávisle na druhé a v případě poruchy jedné z ECU, byla tato část dostupná pro ovládání druhou ECU.

Pro sdílené ovládání bylo voleno zapojení podobné tzv. *Wired-OR* funkci. V tomto případě byl ale volen princip společného emitoru. V případě poruchy jednoho řídicího kanálu jedné ECU musí dojít k přepnutí tohoto řídicího kanálu do stavu vysoké impedance, čímž dojde k jeho odpojení od řízeného výstupu druhé ECU. Tím je dosaženo nezávislého řízení pomocí druhé ECU. Tento princip je používán u všech ovládacích cívek Coil SW$ a Coil B$.

Při tomto typu zapojení je však potřeba počítat s výsledkem logické funkce při současném řízení oběma ECU, protože výsledek řídicího signálu je pak dán logickým součtem obou signálu ECU. Což znamená, že v případě dvoustavového řízení cívek dojde k průchodu řídicího proudu solenoidem, jestliže alespoň jedna z ECU sepne výstupní obvod.

Spojité ovládání zdvojených prvků (řídicí cívky servoventilů S$#) je zajištěno pomocí PWM řízení. Principiální schéma ovládání a snímání řídicí cívky je nakresleno na obr. 6.3 . Zároveň je ukázán princip měření aktuální hodnoty proudu řídicí cívky. Tato informace je využívána pro kontrolu funkce jednotlivých cívek (diagnostika).

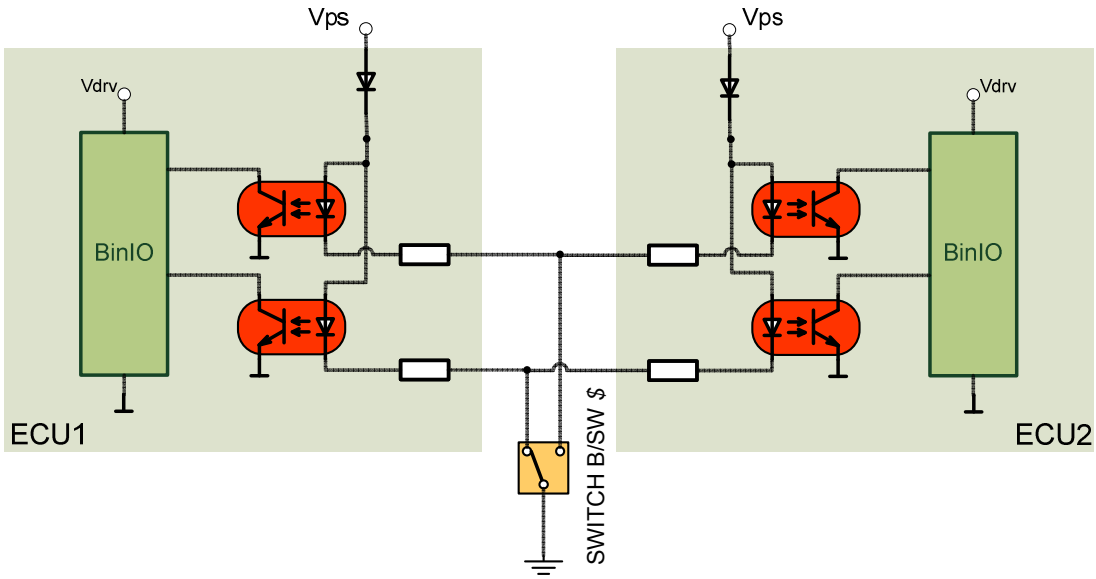


obr. 6.3 - Principielní schéma ovládání jedné cívky servoventilu S$ (modifikováno podle [6])

Koncepce HW pro měření výstupních částí EHSA odpovídá struktuře EHSA. Podobně jako se dělí vstupní prvky jednoho kanálu EHSA na zdvojené a jednonásobné, tak i výstupní obvody akčního členu existují ve zdvojené (např. LVDT snímač hydraulického válce) a jednonásobné podobě (např. LVDT snímač servoventilu, spínače SW$ a B$ přepínacích a přemosťovacích ventilů).

V případě zdvojených prvků je koncepce HW jednoznačná, protože každá ECU je připojena vždy na jeden prvek z dostupné dvojice. Na rozdíl od sdílených jednoprvkových částí EHSA je navíc možné v případě poruchy jednoho ze dvou prvků zajistit softwarové přeposílání aktuální informace ze sesterského ECU.

V případě jednoprvkových částí EHSA je princip snímání složitější. Na obr. 6.4 je znázorněn princip společného snímání stavů spínačů SW$ a B$. Lze pozorovat, že se jedná opět o zapojení *Wired-OR* se společným emitorem. Společným měřeným prvkem je zde mikrospínač zapojený proti zemi. Každý kanál poskytuje při snímání svůj *pull-up* rezistor (definice napěťové úrovně). V případě poruchy jednoho kanálu první ECU musí dojít k přepnutí do stavu vysoké impedance, čímž opět nedojde k ovlivnění druhé sesterské ECU (druhý kanál).



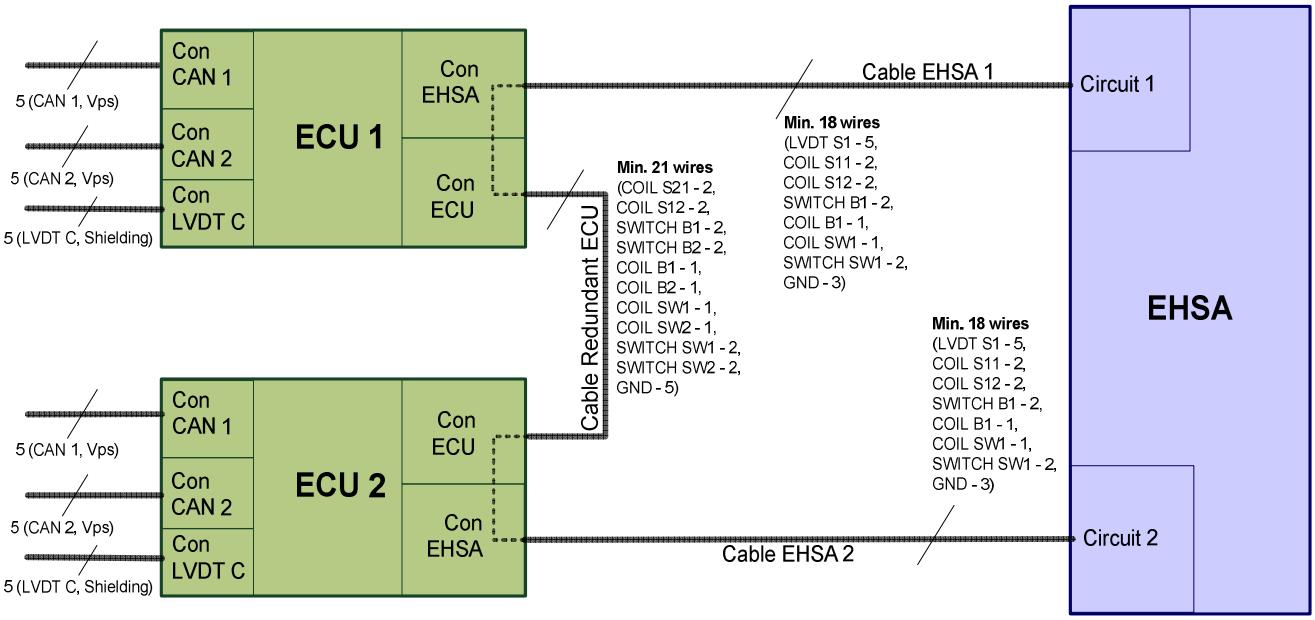
obr. 6.4 - Principielní zapojení čtení stavu spínačů SW$ a B$ (modifikováno podle [6])

Snímání LVDT senzorů hydraulických cylindrů je prováděno průběžně v každé ECU. Snímání LVDT senzorů servoventilu v daném okruhu provádí vždy jen jedna ECU (první ECU snímá LVDT prvního servoventilu a druhá ECU LVDT druhého servoventilu). Aby i zde byla informace z LVDT S1 dostupná pro druhou ECU, která snímá jen LVDT S2, je tato informace přeposílána přes CAN sběrnici a průběžně aktualizována pomocí CANopen komunikace.

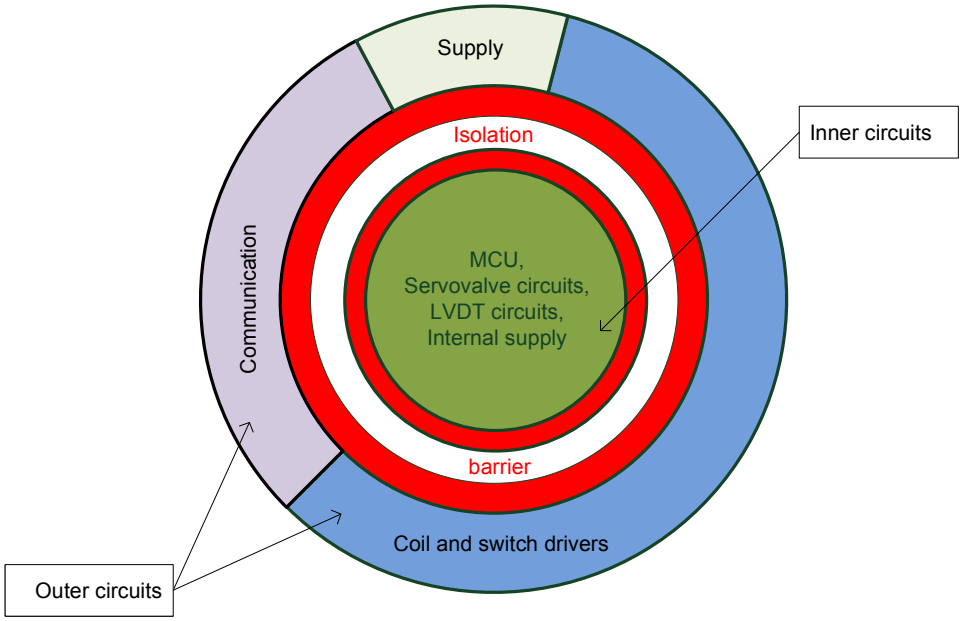
Každá ECU obsahuje dva nezávislé CAN řadiče, které zabezpečují připojení ECU do FBW systému. Tyto komunikační linky musí být galvanicky oddělené od společné CAN sběrnice FBW. V případě poruchy jedné CAN linky na jedné ECU nesmí dojít k ovlivnění linky sekundární, tím by totiž došlo k úplnému odstavení této ECU od nadřazeného řídicího systému a nemožnosti ovládání EHSA přes tento kanál (EHSA by bylo řízeno jen z druhé ECU). Proto je nutné zajistit aby signálové cesty od CAN řadiče ke CAN sběrnici a to včetně oddělovacích členů byly navzájem nezávislé.

Obě ECU jsou z hlediska HW identické, z pohledu propojení na dvouokruhouvou soustavu EHSA jsou jednotlivé ECU připojeny paralelně, vždy na jeden z okruhů EHSA (konektor *Con EHSA*). Každá z ECU je připojena na svůj okruh pomocí přímého kabelu (Cable EHSA *$*). Princip propojení ECU a EHSA i s vyznačenými signály a počtem vodičů je nakreslen na obr. 6.5. Navíc je potřeba, aby každá z ECU poskytovala signály (vodiče) svého okruhu druhé redundantní ECU. Tato křížová výměna signálů mezi oběma ECU dává možnost řídit oba okruhy EHSA (Circuit 1 a Circuit 2) pouze z jedné ECU. Výměna signálů mezi ECU 1 a ECU 2 se uskutečňuje v křížené kabeláži *Redundant ECU* (viz obr. 6.5). Křížení vodičů v kabeláži *Redundant ECU* je proto párové, tak aby se výměna informací z jednoho okruhu (např. Circuit 1) dostala k druhé ECU (např. ECU 2) na pozici druhého řídicího obvodu této ECU. Díky principu párového křížení vodičů v *Redundant ECU* je možné postavit HW obou ECU naprosto identický.

Jedním z primárních požadavků na ECU je odolnost vůči poruchám a negativním účinkům z vnějšího prostředí a samotného FBW systému. Hlavním prostředkem je galvanické oddělení výpočetní části ECU od vnějšího prostředí. V případě vzniku vysokonapěťového signálu na vstupu nebo výstupu nedojde díky izolační barieře k destrukci centrálního výpočetního prvku. Tím je umožněno aby ECU informovala nadřazený systém o vzniku poruchy a změnila způsob řízení EHSA (např. předala řízení redundantní ECU). Nutnou podmínkou pro správnou funkci centrálního prvku je i funkce napájecího subsystému, proto i on musí být vhodně chráněn. Některé budicí obvody pro EHSA nemusí být izolovány od výpočetního prvku, protože jejich galvanické oddělení od vnějšího prostředí je dáno jejich konstrukcí (transformátorová vazba). Principiální oddělení výpočetní části od výkonových periferních a komunikačních obvodů je nakresleno na obr. 6.6.



obr. 6.5 - Detailní schéma propojení ECU1, ECU2 a EHSA



obr. 6.6 - Princip galvanického oddělení subsystémů ECU

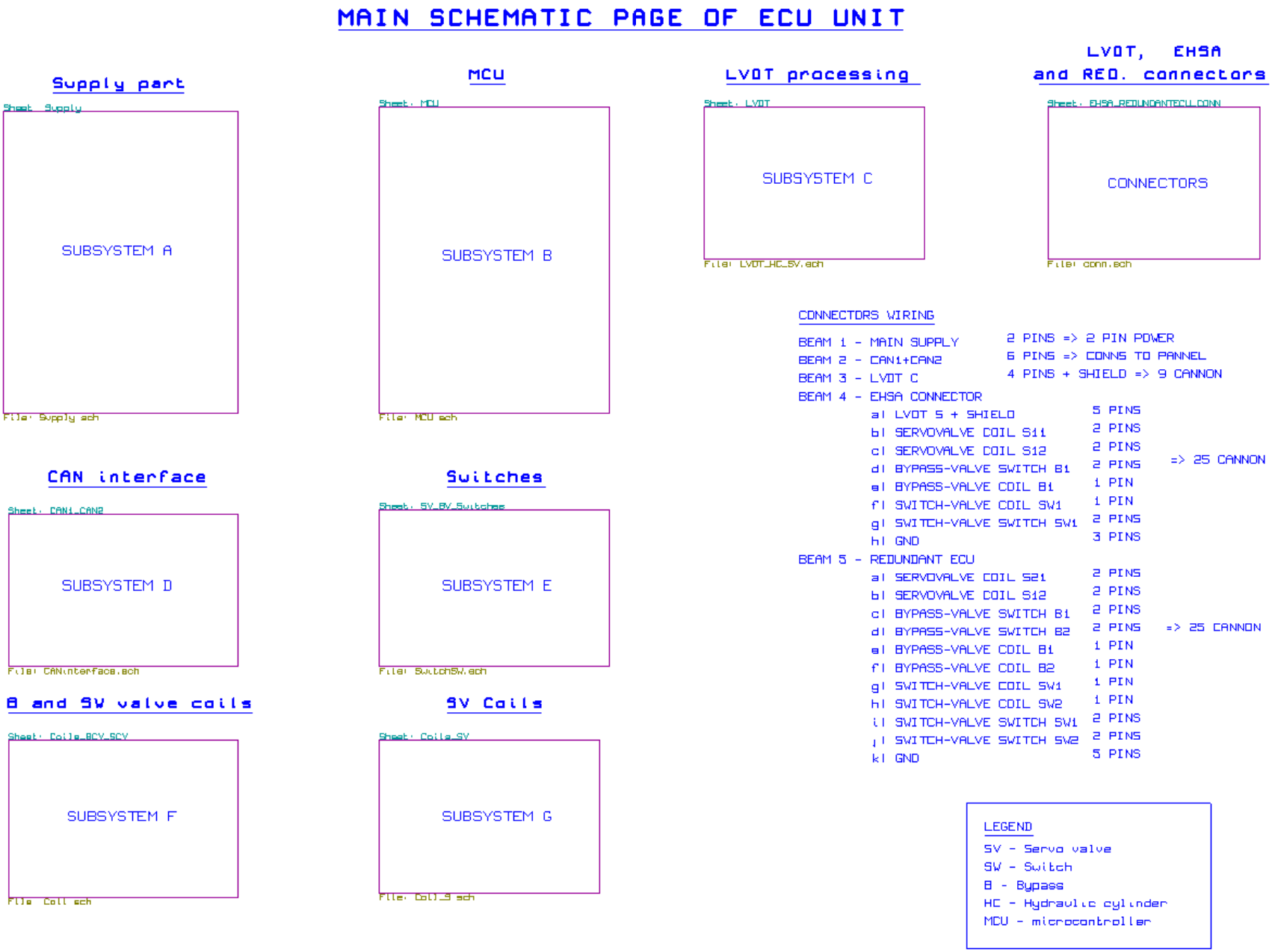
6.2 Rozvržení elektronických subsystémů

V předchozí kapitole byla popsána architektura a koncept HW zapojení. Z hlediska funkce lze elektroniku ECU rozdělit do několika podčástí (subsystémů):

- Napájení (Supply part)
- Centrální procesorová část (MCU)
- Vstupní obvody
  - Obvody zpracování LVDT signálu (LVDT processing)
  - Obvody zpracování stavů přepínačů (Switches)
- Výstupní obvody
  - Řízení solenoidů přemosťovacích a přepínacích ventilů (B and SW valve coils)
  - Řízení cívek servoventilů (SV coils)
- Komunikační obvody CAN sběrnice (CAN interface)
- Křížové propojení konektorů (LVDT, EHSA and RED. connectors)

Hierarchické schéma je zobrazeno na obr. 6.7. Jednotlivé bloky tvoří samostatné celky (stránky) schematického návrhu. Propojení těchto částí je zajištěno přes globální návěští v programu *EESchema*.

Následující podkapitoly popisují princip a funkci výše zmíněných celků. Při návrhu schématu byl brán ohled nejen na požadavky z kapitol 1.3, 1.4, 3, a 4.3 ale také na celkovou spolehlivost a dostupnost použité součástkové základny.



obr. 6.7 – Hierarchické schéma ECU

6.3 Schematický návrh

Základem pro finální podobu elektroniky ECU je vytvoření elektronického schématu. Návrh schématu a desky plošných spojů (PCB) byl vytvořen v programu KiCAD, který je šířen pod GPL licencí. KiCAD je komplexní integrované návrhového prostředí s otevřeným kódem, které poskytuje možnost vytváření elektronických schémat, schematických značek, obsahuje editor desky plošných spojů, prohlížeč Gerber souborů. Bližší informace lze nalézt v [9].

Následující podkapitoly popisují postupně jeden z výše uvedených subsystémů z kapitoly 6.2. Nejprve je vždy popsána funkce subsystému, uveden seznam požadavků, možnosti řešení, následně výběr vhodného řešení včetně zvolené součástkové základny a popis elektrického zapojení. Všechna elektronická zapojení lze nalézt na přiloženém CD (viz kapitola 11).

6.3.1 Napájecí obvody

Napájecí část poskytuje hlavní zdroj energie pro ostatní subsystémy ECU. Zároveň slouží k úpravě a stabilizaci napěťových úrovní pro digitální i analogové obvody.

Hlavním zdrojem ECU jednotky je stejnosměrné napětí *28 V* (maximální proudový odběr není stanoven). Toto napětí je generováno v palubním zdroji letounu. V laboratorních podmínkách bylo simulováno podobným zdrojem s hodnotou *24 V*. Oba tyto zdroje mají dostatečný výkon pro pokrytí proudové spotřeby ECU. Napětí z těchto zdrojů není stabilizované a je vztažené ke společné zemi letounu. Při návrhu je potřeba počítat s kolísáním napěťové úrovně o ± *1 V* od hodnoty *28 V*, je tedy potřeba počítat s hodnotou napětí až *29 V*.

Návrh napájecí části je přímo závislý na průměrné a špičkové spotřebě dalších subsystémů. Je také nutné vytvořit požadované stabilizované napájecí úrovně. Při návrhu zdroje tak bylo potřeba splnit několik základních požadavků:

- Pokrytí proudové spotřeby všech částí ECU
- Pokrytí všech napěťových úrovní
- Dostatečný rozsah vstupního napětí a účinnost
- Filtrace a vhodné oddělení zemí
- Minimální zahřívání součástek a velikost
- Odolnost proti zkratu a přehřátí
- Galvanické oddělení vstupů
- Spolehlivost součástek

Při návrhu schématu ale i výsledné PCB byl brán ohled na doporučené zapojení od výrobce. Jednotlivé datové listy vybraných komponent jsou uvedeny v seznamu referencí v kapitole 10. Další odstavce diskutují výše uvedené požadavky a shrnují podstatné údaje nutné k návrhu.

Pokrytí proudové spotřeby

Napájecí systém ECU musí pokrýt maximální možnou spotřebu dalších subsystémů ale i klidovou spotřebu vlastních stabilizačních prvků. V tab. 6.1 je uvedena (maximální) proudová spotřeba subsystémů a ovládaných obvodů EHSA. Průměrná spotřeba je vždy menší než maximální hodnoty. Při počátečním návrhu zdroje ji nelze jednoznačně určit a proto zde není uvažována.

Navrhovaný zdroj ECU napájí jen některé části z tab. 6.1. Subsystémy, které nevyžadují stabilizaci (solenoidy) se do výkonu navrhovaného zdroje nezapočítávají. Tyto části jsou pak napájeny přímo z palubního nestabilizovaného zdroje.

tab. 6.1 – Spotřeba jednotlivých částí ECU a EHSA

Název části ECU a EHSA	Maximální proudový odběr [mA]	Vyžaduje stabilizaci napájení
Obvody napájení - sekundární část	*20*	*NE*
Centrální procesorová část	*150*	*ANO*
Vstupní obvody	*40*	*ANO*
Výstupní obvody	*20*	*ANO*
Komunikační obvody	*140*	*ANO*
Solenoid B1	*280*	*NE*
Solenoid B2	*280*	*NE*
Solenoid SW1	*280*	*NE*
Solenoid SW2	*280*	*NE*
Cívka servoventilu S1	*10*	*ANO*
Cívka servoventilu S2	*10*	*ANO*
napájení LVDT servoventilu - symetrické napájení	*75*	*ANO*
napájení LVDT hydr. cylindru - symetrické napájení	*75*	*ANO*

Celkový maximální proudový odběr přes ECU do EHSA [mA]	*1660*

Nutný proudový výdej napájecího systému ECU [mA]	*540*

Pozn.: Části ECU a EHSA, které vyžadují pro svou funkci stabilizaci napájecího napětí jsou zobrazeny bílým pozadím v tab. 6.1. Ty části, které nevyžadují stabilizaci nemusí být napájeny napájecím systémem ECU a mohou být napájeny přímo (popř. přes vstupní dělič) z hlavního palubního zdroje *28 V* (resp. *24 V*) jsou zobrazeny šedým pozadím v tab. 6.1.

Klidová hodnota odebíraného proudu z palubního zdroje, při kterém ECU nebudila obvody EHSA byla na finální jednotce změřena *260 mA* ± *5 mA*.

tab. 6.2 – Přehled napájecích úrovní

Název subsystému	Napájení	Symetrické napájení
Centrální procesorová část	*3,3 V* digitální částí. 3,3 V analogová část	NE
Komunikační obvody	*3,3 V* digitální část. *5 V* digitální část	NE
Řídicí obvody solenoidů	*3,3 V* digitální část. *5 V* digitální část	NE
Řídicí obvody servoventilu	*12 V* digitální část. *12 V* analogová část	NE
Obvody zpracování LVDT	*12 V* analogová část	ANO

Pozn.: Symetrické napájení značí nutnost vytvoření kladného a záporného potenciálu vůči společné zemi.

Pokrytí všech napěťových úrovní

Napájecí systém musí zajistit všechny potřebné napěťové úrovně a jejich dostatečnou stabilizaci s povoleným rozkmitem výstupního napětí. Pro obvody ECU a ovládání EHSA je nutné vytvořit několik stupňů napěťových úrovní. Přehled a účel těchto úrovní je uveden v tab. 6.2.

Dostatečný rozsah vstupního napětí a účinnost

Hlavní napájení do ECU zajišťuje nestabilizovaný palubní zdroj *28 V* (resp. *24 V* pro laboratorní testování) s rozkmitem ± *1 V*. Vstupní napájení může tedy nabývat až *29 V*, teoreticky při vysoké nestabilitě vstupní úrovně může dojít až ke špičkovému zvýšení nad *30 V.*

Zároveň je potřeba zohlednit účinnost zdroje. Rozdíl vstupního napětí a výstupních dodávaných napěťových úrovní je v některých případech více než *20 V* a odběr obvodů ECU činí ve špičce až *540 mA*. Při tomto proudovém odběru by v případě lineárního stabilizátoru bylo potřeba vyzářit ve formě tepla výkon až *11 W* (nepřípustné zahřátí komponent) .

Řešením je volba kaskádového zapojení. Primární (spínaný) zdroj s vysokou účinností vytvoří dostatečně malé napětí pro sekundární stupeň stabilizačních prvků (lineární stabilizátory). Díky tomu nevznikne na stabilizátorech velký rozdíl potenciálu a tím i zbytečně vysoký tepelný výkon. Výsledné zahřívání zdroje ale i rozdíl potenciálu je tak z větší části přenesen na primární a z menší části na sekundární stupeň.

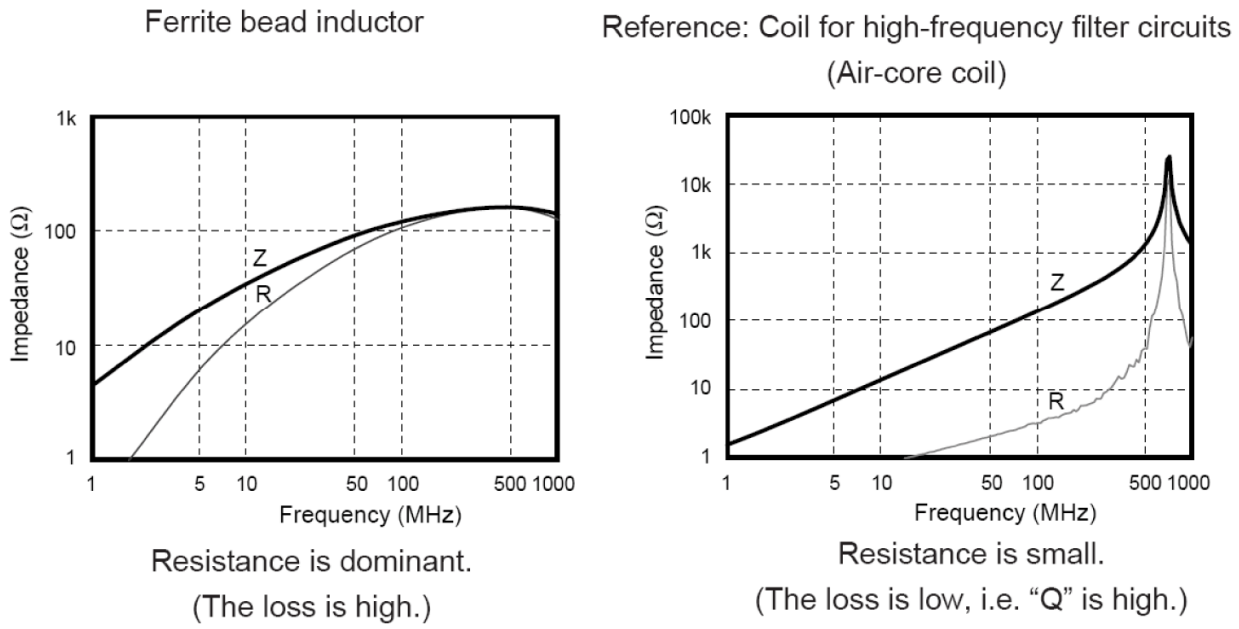
Díky lineárním stabilizátorům navíc dochází k prvotní filtraci (průchodem přes samotný lineární stabilizátor) primárního stupně, důsledkem toho je na výstupu sekundárního stupně menší zvlnění.

Filtrace a vhodné oddělení zemí

Při návrhu zdroje je nutné vytvořit napájení, jak pro spínanou vysokofrekvenční část, tak pro analogovou nízkofrekvenční část. Proto bylo potřeba vhodně oddělit země a napájení, tak aby nedocházelo k přenosu rušivých složek signálů ze spínaných částí do nízkofrekvenčních částí ECU. K tomuto účelu byly použity pasivní filtry složené z kondenzátorů a ferittový kotoučků. Zároveň byl vytvořen samostatný zdroj stabilizovaného napětí pro analogové nespínané části.

Použití ferritových kotoučků (FB) namísto klasických induktorů má několik výhod. Hlavní rozdíl oproti induktorům je ve frekvenční charakteristice obou prvků. Na obr. 6.8 jsou ukázány typické průběhy frekvenčních charakteristik ferritového kotoučku a induktoru. Frekvenční charakteristiky obou prvků jsou si podobné jen v oblastech nízkých a středně-vysokých kmitočtů (řádově do desítek *MHz*). V těchto oblastech impedance roste na hodnoty okolo *100 Ω*. V oblastech vysokých frekvencí (stovky *MHz* a výše) se začíná projevovat odlišná konstrukce obou prvků. Zatímco v případě induktoru je impedance nadále rostoucí a obsahuje dominantní rezonanční špičku, tak v případě FB se impedance již nezvětšuje a začíná mírně klesat. Důležitá je v tomto případě rezistivní složka impedance, protože určuje celkové množství vyzářené energie ve formě tepla a tedy výsledné ztráty při filtraci. V případě induktoru je ve většině případů rezistivita malá a energie v oblastech velkých frekvencí se akumuluje do magnetického pole induktoru. Pouze na úzké rezonanční oblasti dochází k velkému nárůstu rezistivní složky. Z hlediska návrhu filtračního členu je proto nutné naladit filtr s induktorem do těchto rezonančních oblastí. Rezonanční oblast induktoru je ale velice úzká (rozsah okolo 100 Mhz) a na vysokých frekvencích tak opět klesají požadované filtrační účinky. Vhodné použití LC filtrace je tedy pro úzkopásmové filtry, méně vhodné pak pro dolní propusti.

V případě FB se na vysokých frekvencích uplatňuje odporový charakter a FB se ve vysokých frekvencích chová jako rezistor s hodnotou okolo jednotek až stovek *kΩ*. Navíc je tato hodnota stabilní i na vysokých frekvencích. Pro návrh dolních propustí je tak kombinace FB - kapacitor velice výhodná.



obr. 6.8 – Frekvenční charakteristiky ferritových jader a induktorů (převzato z [46])

Zahřívání součástek a velikost

Návrh zdroje byl také optimalizován z hlediska minimálního zahřívání součástek, z tohoto důvodu byl vybrán spínaný zdroj s vysokou účinností až *82 %*. Díky tomu nebylo nutné na PCB použít přídavných aktivních chladičů ale stačilo využít pasivního chlazení na povrchu PCB.

Velkou výhodou použití spínaného zdroje pro primární stupeň je jeho velikost. Oproti klasickému nespínanému zdroji je rozměr i hmotnost výrazně menší. Rozměry lze navíc snížit použitím integrované verze. V případě použití monobloku v kovovém pouzdře dojde ke zlepší nejen chlazení spínaného zdroje, ale také snížení vyzařovacích vlastností samotného spínaného zdroje (elektromagnetická kompatibilita).

Odolnost proti zkratu a přehřátí

Veškeré prvky použité ve zdroji jsou odolné proti zkratu, přepólování a přehřátí. Díky tomu se zvýšila odolnost celé jednotky a není potřeba při krátkodobém vzniku nežádoucích pracovních podmínek vyměňovat součástky na PCB. Odolnost proti poruchám konkrétního prvku lze nalézt v datovém listu dané součástky v seznamu referencí.

Galvanické oddělení vstupů

Napájecí napětí na výstupu zdroje musí být od vstupního palubního napájení galvanicky odděleno. Řešením bylo využití izolovaného spínaného zdroje. Velikost průrazného napětí tohoto zdroje činí až *1500 V*.

Volba řešení a výběr součástek

Napájecí zdroj byl navržen jako dvoustupňový systém. Primární stupeň je tvořen izolovaným symetrickým spínaným monoblokem TEN-12-4822. Ten tvoří vnitřní napětí izolované od vnějšího palubního napájení. Na tento stupeň je napojen sekundární stupeň tvořený lineárními *3,3 V* stabilizátory MC33269. Mezi analogovým a digitálním napájením jsou umístěny pasivní filtrační obvody z ferritových jader a kondenzátorů.

Paralelně k primárnímu stupni je vytvořen *5 V* stabilizovaný zdroj L78M05CDT. Ten slouží k napájení všech výstupních a budicích obvodů za izolační barierou.

Popis zapojení

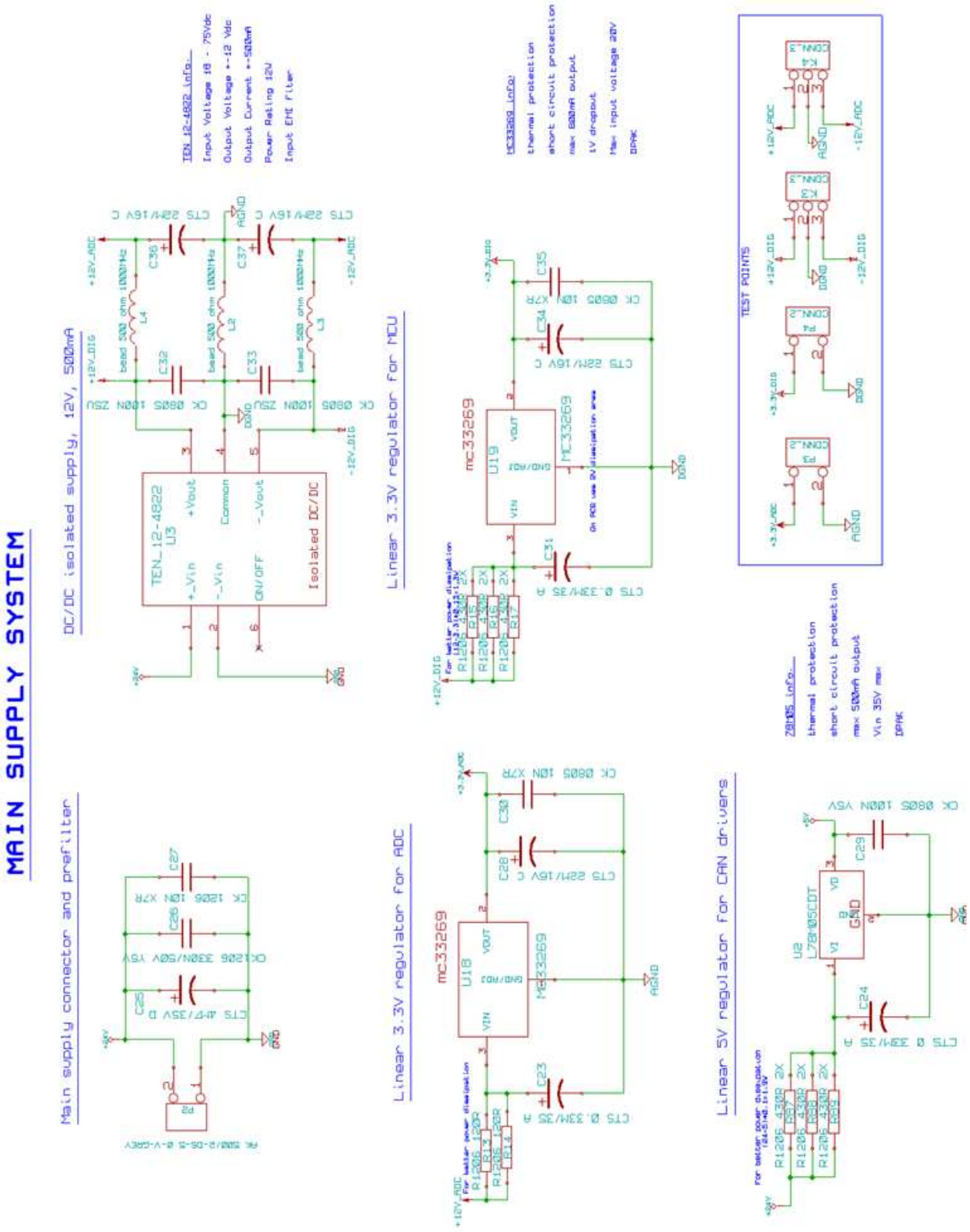
Elektrické schéma navrženého zdroje ECU je zobrazeno na obr. 6.9. Palubní (vstupní) napájení je přivedeno přes konektory CAN sběrnic. Obě nezávislá napájení z těchto sběrnic jsou spojena a vedena do vstupního konektoru P2. Zde dochází k první filtraci vstupního napájení.

Palubní napájení je přivedeno na vstupní svorky integrovaného DC/DC modulu TEN12-4822 (U3). TEN12-4822 je symetrický *±12 V* izolovaný spínaný zdroj s maximálním výkonem *12 W*, proto plně pokrývá špičkový odběr ECU. Zároveň splňuje veškeré požadavky na rozsah vstupního napětí a odolnost.

Napětí *±12 V* a země jsou filtrována přes pasivní filtr 2. řádu (C32, C33, C36, C37 L2, L3, L4). Nefiltrovaných *+12 V* je přivedeno přes rezistory R15, R16 a R17 na lineární stabilizátor napětí MC33269 (U19), který vytváří *+3,3 V*. Podobným způsobem je vytvořeno napětí *+5 V* (U2). Tyto nefiltrované napětí jsou použita pro napájení vysokofrekvenčních digitálních částí ECU.

Filtrované *+12 V* napětí z DC/DC konvertoru je upraveno na *3,3 V* pomocí lineárního stabilizátoru MC33269 (U18) a poskytuje napájení pro analogové nízkofrekvenční části ECU.

Na napájení LVDT snímačů v EHSA je použito filtrovaných *±12 V*. Vytvořená napětí jsou pro testovací účel vyvedena na konektory P3, P4, K3, K4.



obr. 6.9 – Schéma napájecího zdroje ECU

6.3.2 Centrální procesorová část

Funkcí centrální procesorové jednotky je řídit celkovou činnost ECU. Tím je myšleno ovládání vstupně-výstupních obvodů pro ovládání EHSA, komunikační linky CAN, měření signálů od senzorů LVDT a mikrospínačů přemosťovacích a přepínacích cívek. Činnost procesoru je řízena na základě aktuálního programu uloženého v paměti.

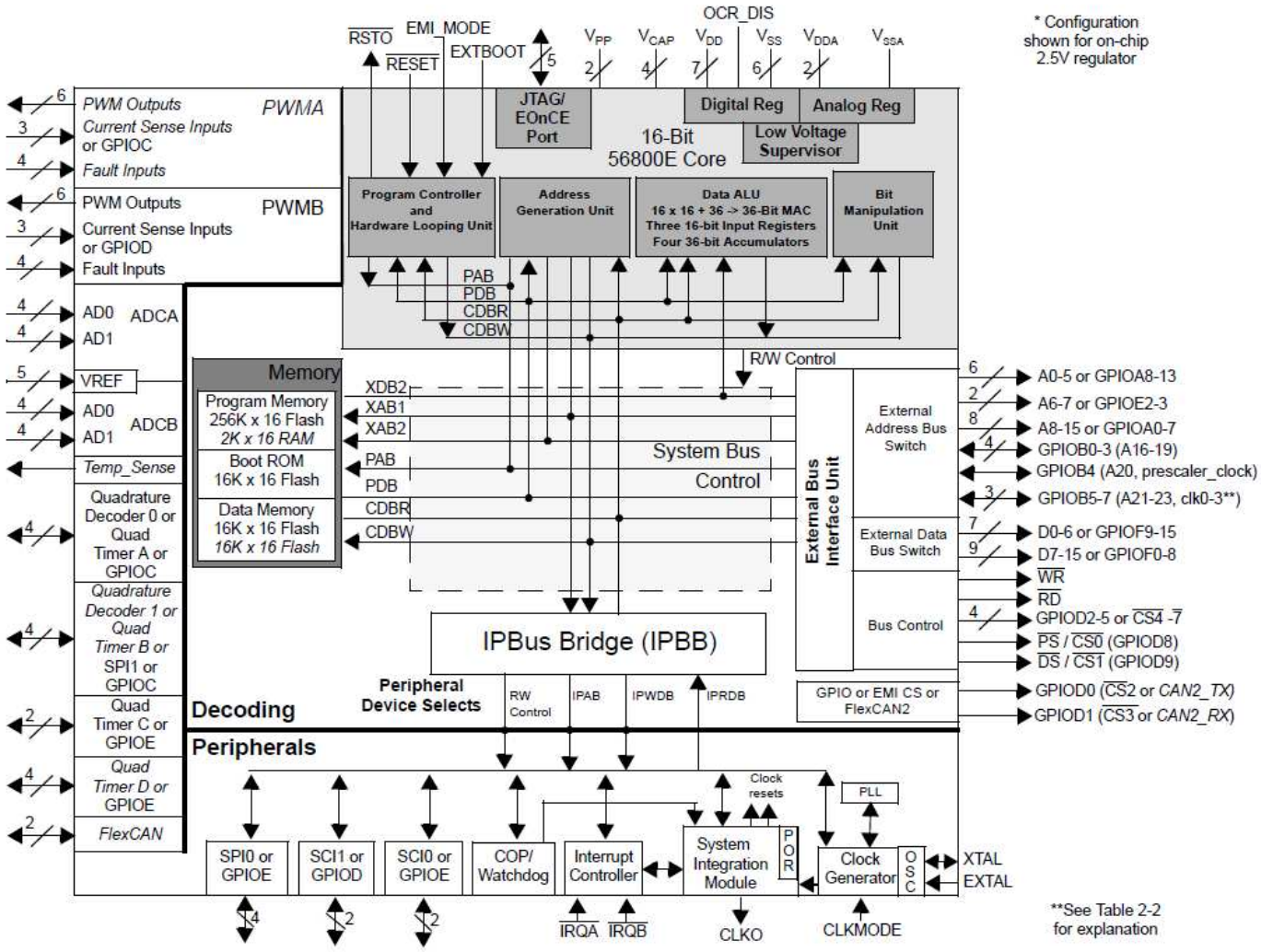
Hlavními požadavky na centrální část je především dostatečný výpočetní výkon a pokrytí všech požadovaných vstupně-výstupních signálových cest. Nutnou podmínkou pro zvolené řešení je výběr platformy s dvěma nezávislými CAN řadiči.

Jedna z variant jak řešit tento problém bylo využití dvou procesorových platforem, každá pro jeden řídicí a komunikační obvod. To by obnášelo výrazně složitější návrh PCB ale především složitý návrh softwaru a synchronizaci paralelního běhu programu ve dvou výpočetních jednotkách. Druhou možností byla volba platformy s dvěma CAN řadiči na jednom čipu se sdílenou aritmeticko-logickou jednotkou (ALU). Z tohoto pohledu bylo potřeba vybrat takovou procesorovou platformu, která by měla dostatečný výpočetní výkon, tak aby při obsluze více periferií včetně komunikaci po CANu došlo ke včasné reakci a obsluze obvodu.

V centrální části byl proto umístěn digitální signálový procesor (DSP) MC56F8367 od společnosti Freescale, jehož základní vlastnosti jsou:

- Výpočtový výkon *60 MIPS* při frekvenci procesorového jádra *60 MHz*
- 16-ti bitové jádro založené na duální Harvardské architektuře
- Simultánní přístup do třech míst paměti: 1x programová, 2x datová
- 16-ti bitová paralelní násobička
- Interní programová paměť velikosti *512 kB* Flash, *4 kB* RAM, *32 kB boot* ROM
- Interní datová paměť velikosti *32 kB* Flash, *32 kB* RAM
- 2 x 6 kanálů PWM
- 4 x 4 kanály 12 bitových A/D převodníků
- 4 x Quad časovače (Quad Timer)
- 2 x FlexCan moduly (vyhovující standardu CAN 2.0 B)
- 2 x SPI (Serial Peripheral Interface), 2 x SCI (Serial Communication Interface)
- 2 x vyhrazený vstup externího přerušení (IRQA, IRQB)
- 76 x univerzálních vstupně-výstupních vývodů procesoru (General Purpose I/O)
- JTAG a emulátor na čipu (OnCE)

MC56F8367 je vyroben technologií CMOS a jeho vstupní obvody jsou kompatibilní s TTL logikou. Přímo na čipu obsahuje dva stabilizátory *3,3 V* a *2,6 V* , jeden pro digitální části druhý pro analogové části. Obsahuje obvody pro sníženou spotřebu (Wait mode) a úsporný režim (Stop mode). Navíc lze každou nepoužitou periferii samostatně přepnout do Stop módu. Blokové schéma MC56F8367 je zobrazeno na obr. 6.10.



obr. 6.10 – Blokové schéma MC56F8367

Architektura signálového procesoru dovoluje paralelizovat některé z výpočetních činností. Jádro MC56F8367 má Harvardskou architekturu a obsahuje tři paralelní výpočetní jednotky. Díky tomu je provedeno za jeden instrukční cyklus až šest různých operací. Instrukční sada MC56F8367 byla optimalizována pro efektivní překlad z jazyka C a C++. Program může být spouštěn jak z vnitřní, tak vnější paměti a díky zdvojení datové sběrnice je načítání operandů vykonáno v jednom instrukčním cyklu

MC56F8367 je primárně určen pro aplikace s PWM řízením a *motor control*. Pro tyto účely je vybaven dvěma pulzně šířkovými modulátory. Ty mohou generovat navzájem nezávislé PWM signály v celkovém rozsahu 12 nezávislých PWM výstupů. PWM modulátory dovolují komplementární operace a nastavení spínacích dob (Dead time) pro můstkové aplikace. Generovaný signál je nastavitelný v rozsahu *0 %* až *100 %* se symetrickým (Center-alignment) nebo krajním (Edge- alignment) zarovnáním. PWM výstupy jsou výkonově posíleny pro přímé řízení optoizolačních prvků. Díky vnitřnímu propojení PWM modulátorů s ADC obvody je možné synchronizovat spouštění měření s každou novou periodou PWM.

MC56F8367 je vybaven dvěma analogově-digitálními převodníky. Každý ADC obsahuje dva kanály s nastavitelným zesílením. Každý kanál obsahuje vlastní *sample/hold* obvod a multiplexer 4:1. Díky tomu je možné měřit celkově až 16 signálů. Maximální rozlišení měření je 12 bitů s volitelným zarovnáním. Frekvence vzorkování je *1,66 MHz* s možností dávkového měření a HW průměrování. Oba kanály umí měřit jak v diferenciálním, tak standardním režimu. Obvody ADC obsahují filtrační členy pro potlačení rušení při nezapojeném vstupu.

Použité DSP integruje na svém čipu hned dva nezávislé FlexCAN moduly. FlexCAN (FC) modul je komunikační řadič pro asynchronní komunikaci po CAN. FC implementuje protokol CAN verze 2.0 až do rychlosti *1 Mbit/s*. Každý komunikační kanál obsahuje 16 *bufferů* konfigurovatelných pro příjem nebo

vysílání. Zároveň dovoluje nastavit přerušovací systém pro reakci na každý *buffer* zvlášť. Délka zpráv může být 0 až 8 bytů s 11-ti bitovým nebo 29-ti bitovým identifikátorem. Každý z *bufferů* má přiřazenu vlastní filtrační masku pro rychlou prioritní arbitraci zpráv.

Výhodou MC56F8367 je také dostatečně velká paměť programu a dat, díky tomu je tento DSP vhodný i pro náročné aplikace postavené na metodice MBD s automatickou generací kódu.

Popis zapojení

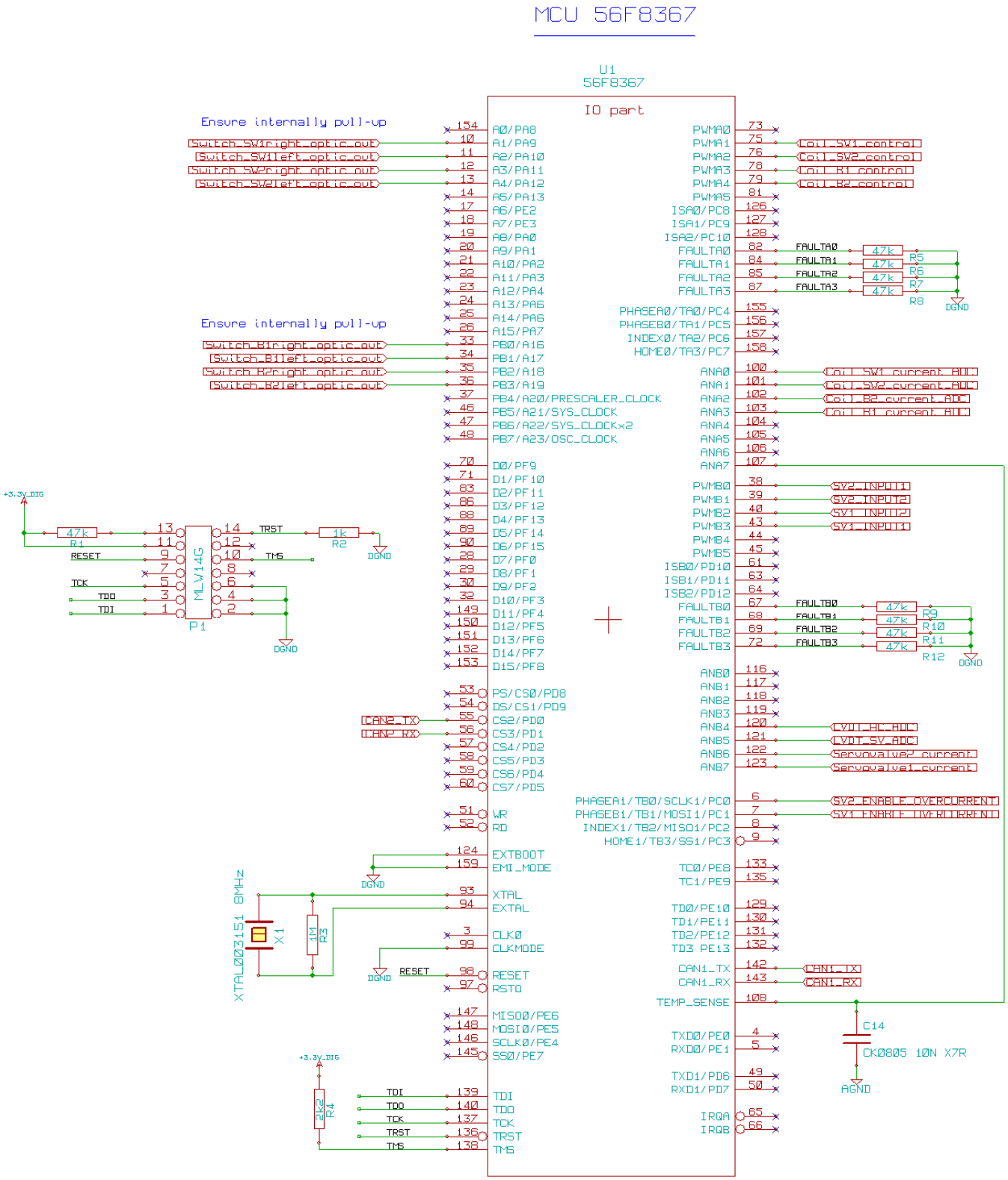
Zapojení MC56F8367 bylo navrhováno dle doporučení a specifikací výrobce. Z hlediska funkčnosti lze schéma rozdělit na dvě části. První částí je zapojení vstupně-výstupních pinů a obvodů zpracování signálů (obr. 6.11) a druhou část tvoří zapojení napájecího subsystému (obr. 6.12).

Pro potřeby ECU bylo použito osm kanálů AD převodníku a osm vstupně-výstupních pinů (GPIO), těchto osm GPIO je určeno na snímání stavu spínačů přepínacích (SW) a přemosťovacích (B) ventilů, čtyři kanály AD převodníku jsou určeny k měření proudu procházejícího přes solenoidy SW a B ventilů, dva kanály AD slouží k měření LVDT signálů a dva kanály AD měří aktuální proud řídicích cívek servoventilů. Dále jsou k U_1 připojeny řídicí signály solenoidů a řídicích cívek servoventilů. Tyto signály jsou připojeny na výstupní piny PWM modulátorů.

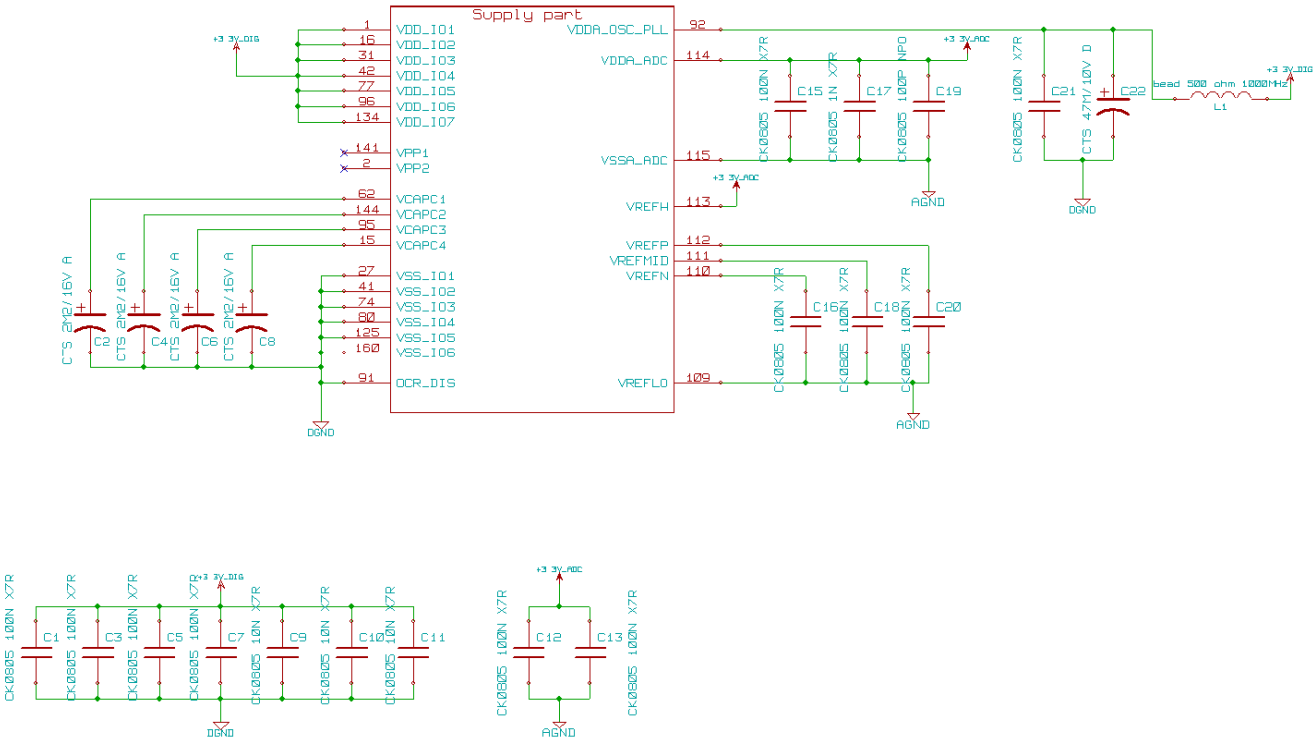
MC56F8367 obsahuje dva nezávislé CAN řadiče. Ty jsou zapojeny na izolační členy a následně budiče CAN komunikace. Spolu tvoří kompletní fyzickou vrstvu CAN komunikace ECU. Pro správnou funkci DSP je nutné dodat externí zdroj hodin. V tomto případě je použito krystalového rezonátoru (X_1) o hodnotě *8 MHz*. Díky fázovému závěsu DSP dochází k násobení tohoto kmitočtu až na hodnotu *60 MHz*. Pro ladící účely je také zapojen na vstup AD převodníku interní teplotní senzor. V zapojení také nechybí konektor (P_1) pro připojení ladícího nástroje komunikující přes rozhranní JTAG.

Napájecí část MC56F8367 vyžaduje ke správné funkci stabilizované napájení *3,3 V*. Ty jsou přivedeny ze sekundárního stupně napájení ECU. Dále je zapojen dostatečný počet blokovacích ($C_1, C_3, C_5, C_7, C_9, C_{11}, C_{13}, C_{16}, C_{18}, C_{20}$) a skupinových (C_2, C_4, C_6, C_8) kondenzátorů.

Ze stabilizovaného zdroje *3,3 V* je přivedeno filtrované napětí pro napájení AD převodníku a interní napěťové reference. MC56F8367 potřebuje také napájet obvody fázového závěsu, ty mají z hlediska možného vzniku rušení filtrováno napájení přes pasivní filtr 2. řádu (C_{21}, C_{22}, L_1).



obr. 6.11 – Schéma zapojení digitální části MC56F8367



obr. 6.12 – Schéma zapojení napájecí části MC56F8367

6.3.3 Vstupní obvody a předzpracování signálu

Vstupní část ECU tvoří obvody pro zpracování a úpravu signálu z LVDT snímačů EHSA a obvody pro čtení stavu spínačů přepínacích (SW) a přemosťovacích (B) ventilů.

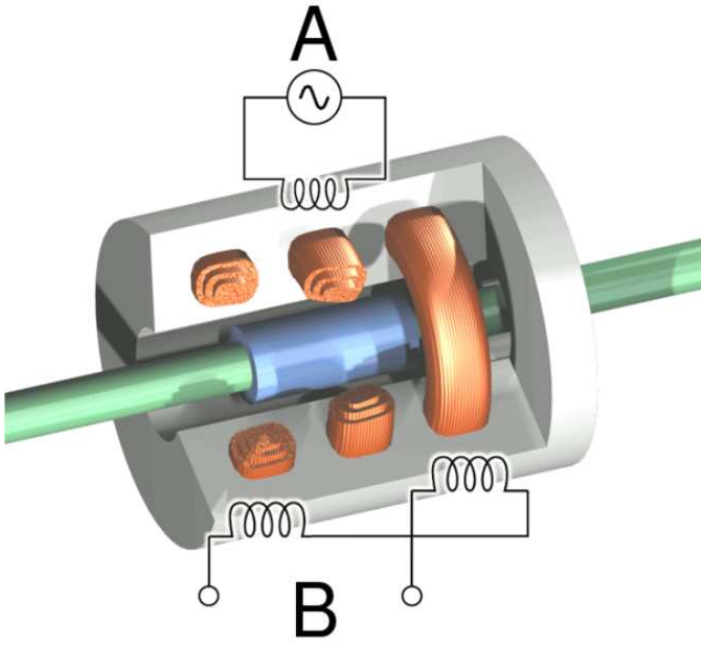
Obvody pro zpracování signálu LVDT senzorů

Aktuální poloha pístnice EHSA a servoventilů je měřena LVDT (Linear Variable differential transformer) snímačem, jehož principiální schéma je nakresleno na obr. 6.13.

LVDT snímač se skládá ze tří solenoidů navinutých okolo společné trubice, ve které se pohybuje feromagnetické jádro. Jeden solenoid (A) tvoří primár a slouží k excitaci snímače, druhý a třetí solenoid jsou spojeny začátky vinutí do série a tvoří sekundár LVDT. Detaily ohledně principu a funkce LVDT snímače lze nalézt v [38]. Parametry použitého LVDT v EHSA lze nalézt v kapitole 4.3.

Pro vyhodnocení a předzpracování signálů z LVDT snímačů je nejdříve potřeba zajistit správnou excitaci primárního vinutí. Pro LVDT snímače existuje několik způsobů, jak navrhnout vyhodnocovací část. Ideálním řešením je využít kompletního integrovaného obvodu pro současnou excitaci a vyhodnocení sekundárního napětí LVDT snímače. Pro tyto účely lze použít obvod AD698. Základními parametry AD698 jsou:

- Linearita 0.05%
- Maximální výstupní napětí ±11 V
- Drift zesílení 20 ppm/℃
- Drift offsetu 5 ppm/℃



obr. 6.13 – Čtyřvodičový LVDT snímač (převzato z [45])

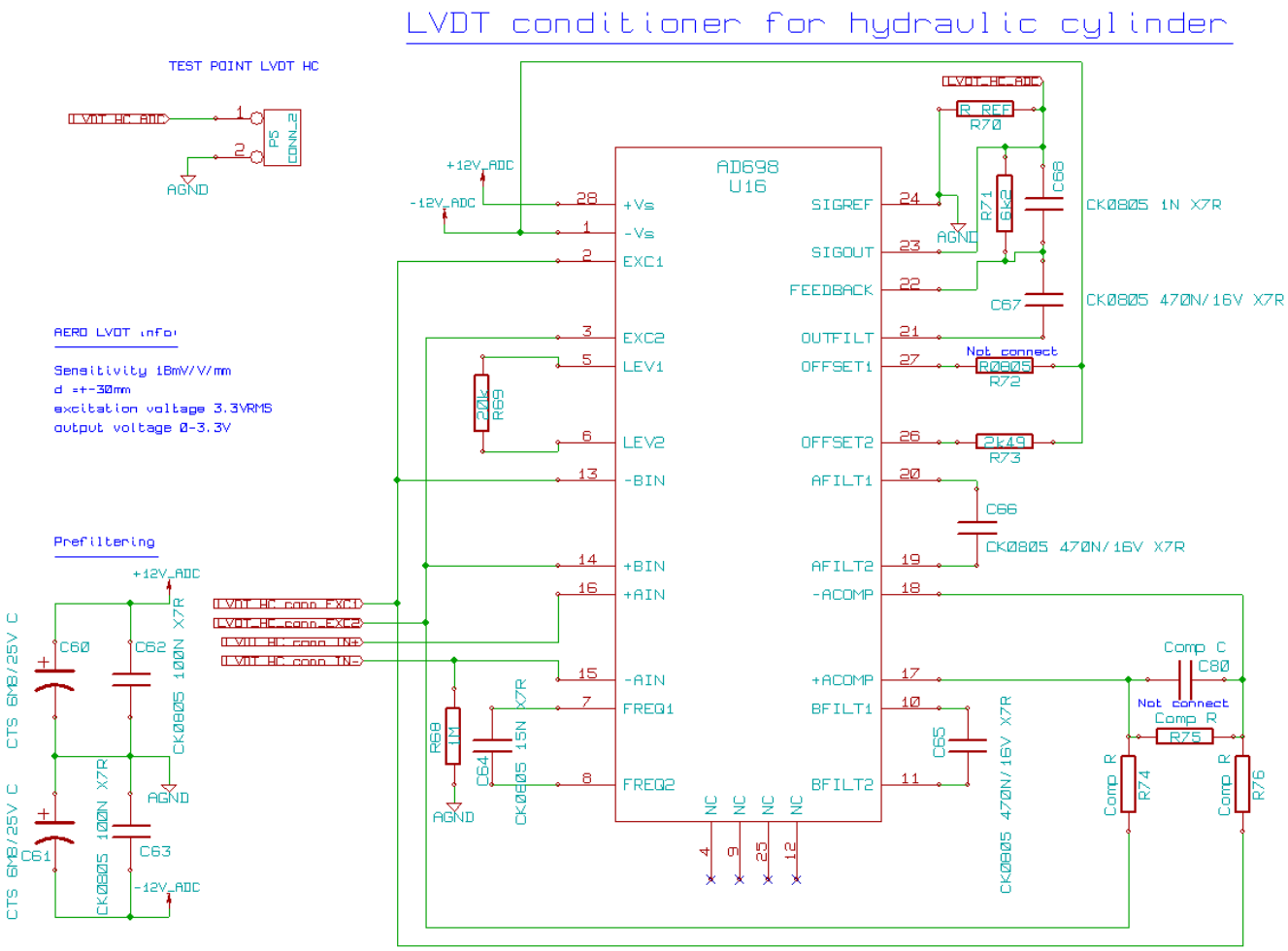
Obvod AD698 je monolitický vyhodnocovací systém určený pro čtyřvodičové nebo pětivodičové LVDT snímače. Uvnitř AD698 jsou integrovány tyto části:

- Interní sinusový oscilátor *20 Hz* až *20 kHz* pro napájení LVDT
- Napěťová reference
- Rozhranní pro připojení čtyř a pětivodičového LVDT snímače
- Linearizační obvody LVDT signálu
- Obvody pro zesílení a úpravu LVDT signálu
- Obvody pro unipolární nebo bipolární výstup
- Obvody fázové kompenzace

Elektronické schéma pro zapojení s AD698 bylo vytvořeno dle doporučení z [14]. Konkrétní hodnoty součástek byli vypočítány pro LVDT s parametry z kapitoly 4.3. Zapojení obou obvodů AD698 pro předzpracování signálů z LVDT snímače servoventilu a hydraulického válce je uvedeno na obr. 6.14 a obr. 6.15.

Bylo voleno zapojení pro čtyřvodičový LVDT snímač. Výsledné vlastnosti pro zpracování signálu byli nastaveny pomocí pasivní součástek. Nejdříve bylo potřeba nastavit excitační frekvenci f_{EXC} a amplitudu excitačního napětí V_{EXC} pro LVDT snímač. Poté bylo vhodně nastaveno zesílení a offset výstupní LVDT signálu, tak aby odpovídal vstupním rozsahům AD převodníku MC56F8367. Výstupní signál z AD698 byl nastaven tak, aby nulová poloha LVDT odpovídala polovičnímu napětí napěťové reference AD převodníku 56F8367. Zároveň byl výstupní signál normován tak, aby maximální rozsah odpovídal krajním napěťovým mezím vstupu AD převodníku.

Z katalogových listů LVDT pro hydraulické válce byla zjištěna citlivost $S = 18mV/V/mm$ při pracovním efektivním napětí primární cívky $V_{EXC} = 3{,}3V_{RMS}$ a frekvenci $f_{EXC} = 2500Hz$. Maximální pracovní zdvih LVDT je $d = 30mm$. Tyto údaje byli použity i pro LVDT servoventilů, protože jejich pracovní zdvih ani citlivost nebyla udána.



obr. 6.14 – Schéma zapojení obvodu pro zpracování LVDT hydraulického cylindru

Rezistory R_{69} a R_{78} určují velikost excitačního napětí V_{EXC}. To je požadováno pro oba LVDT $V_{EXC} = 3{,}3V$. Z tabulky 9 na straně 7 v [14] byla určena hodnota $R_{69} = R_{78} = 20k\Omega$. Kapacitory C_{64} a C_{74} určují frekvenci excitačního napětí. Jejich hodnota byla určena na základě vzorce:

$$C_{74} = C_{64} = \frac{35.10^{-6}}{f_{exc}} = \frac{35.10^{-6}}{2500} = 14nF \approx 15nF \qquad (1)$$

Kapacitory C_{65}, C_{66}, C_{67} a C_{75}, C_{76}, C_{77} nastavují šířku pásma výstupního filtru a filtrů obou synchronních demodulátorů v AD698. Jako nominální šířka pásma pro oba filtry byla volena $1/10$ z excitační frekvence f_{EXC}. Hodnota kapacitorů pak byla vypočítána na základě:

$$C_{65} = C_{66} = C_{67} = C_{75} = C_{76} = C_{77} = \frac{10^{-4}}{0{,}1.f_{exc}} = \frac{10^{-4}}{250} = 400nF \approx 470nF \qquad (2)$$

AD698 dekóduje aktuální pozici LVDT pomocí synchronního demodulátoru (SD). První vstup do SD je zavedeno excitační napětí V_{EXC}. Výstupní sekundární napětí z LVDT vstupuje do druhého vstupu synchronního demodulátor. Poměr těchto dvou napětí je následně filtrován a normován na požadovaný maximální rozsah ADC. Finální normování je určeno dvěma parametry. První parametr je hodnota výstupního napětí $Vout$. $Vout$ je hodnota napětí odpovídající maximálnímu jednosměrnému zdvihu

přičemž nulová pozice jádra odpovídá $Vout = 0V$. $Vout$ je nastaveno prvky R_{71} a R_{80} . Jejich hodnota je určena na základě:

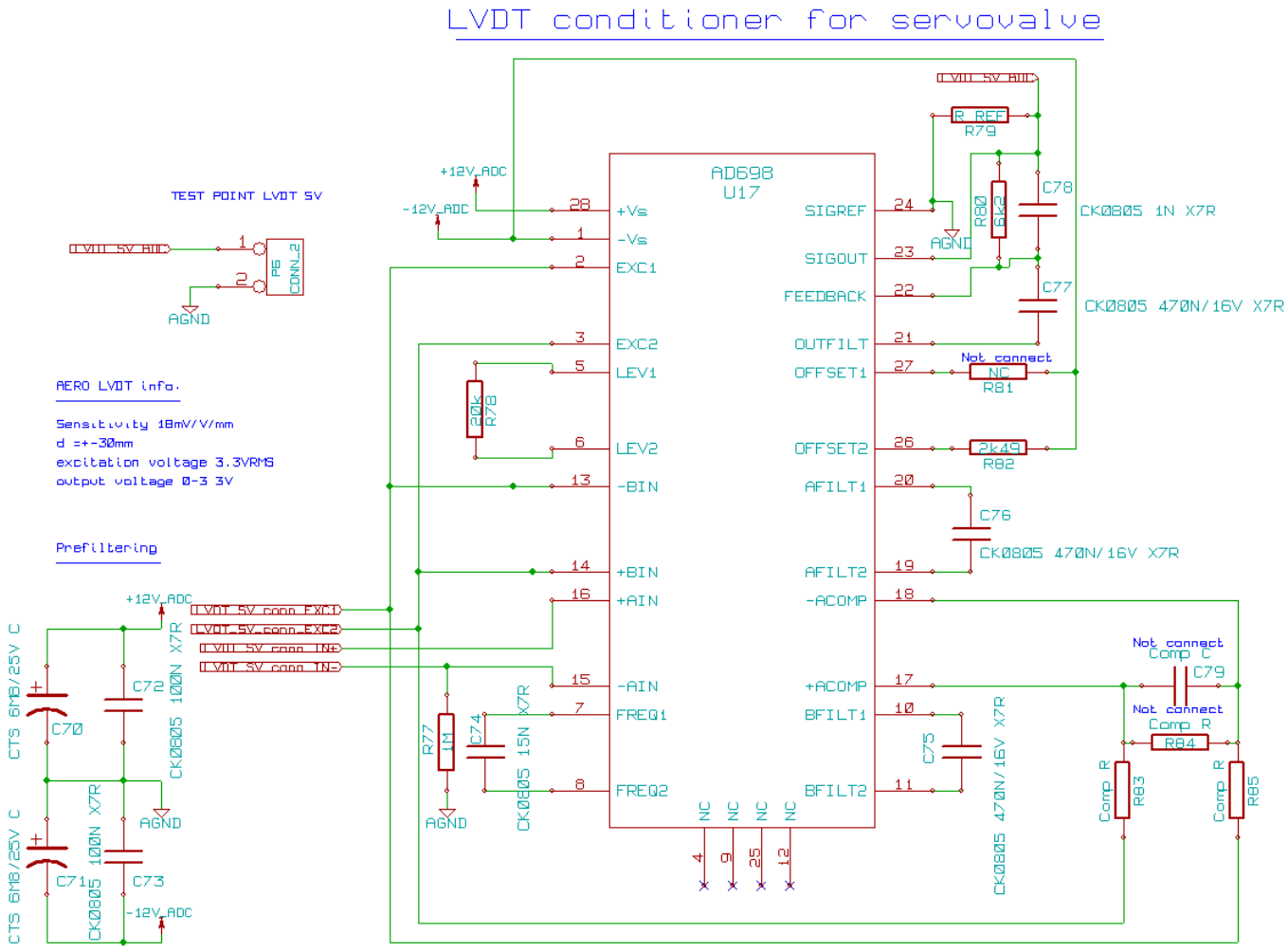
$$R_{71} = R_{80} = \frac{Vout}{S.2.d.500.10^{-6}} = \frac{3,3}{0,018.2.30.500.10^{-6}} = 6,111k\Omega \approx 6,2k\Omega \qquad (3)$$

Jelikož je vypočtené napětí $Vout$ symetrické okolo nuly. Musí být ještě posunuto do kladných napětí v rozmezí ADC. Druhým parametrem normování je V_{Offset} . Ten se vypočítá jako:

$$V_{Offset} = 1,2.R_{71}.\left(\frac{1}{R_{73} + 2000}\right) \rightarrow$$

$$R_{73} = R_{82} = \frac{1,2.R_{71}}{V_{Offset}} - 2000 = \frac{1,2.6200}{1,65} - 2000 = 2509\Omega \approx 2,49k\Omega \qquad (4)$$

Fázová kompenzace nebyla při výpočtu uvažována. Při finálních testech vstupních obvodů pro LVDT byli vypočtené hodnoty korigovány na přesnější výstupní napětí pro ADC a byla nastavena fázová kompenzace.



obr. 6.15- Schéma zapojení obvodu pro zpracování LVDT servoventilu

Obvody čtení stavů přepínačů přemosťovacích a přepínacích cívek

Čtení stavů přepínačů (mikrospínačů) přemosťovacích (Bypass) a přepínacích (Switch) ventilů zajišťuje zapojení z obr. 6.16. Přepínač je funkčně zapojen tak, že uzemní vždy jeden ze dvou vývodů podle typu krajní polohy. Snímáním obou těchto vývodů detekujeme přerušení jednoho nebo obou vodičů, vzájemný zkrat obou vodičů a zkrat obou vodičů proti zemi. Chybně by bylo vyhodnoceno pouze současné přerušení jednoho vodiče (toho který je přepínačem uzemněn) a zkrat druhého na zem.

Poloha každého přepínače je snímána oběma ECU. Principiálně je sdílení zapojené dle [6.1]. Aby na společných svorkách přepínače nebyly oba kanály ECU jednotek galvanicky spojeny, jsou vstupy ECU opticky odděleny. Optické oddělení zajišťuje optočlen ILD206, jehož vstup je připojen na přepínač daného ventilu. Výstup optočlenu je přiveden na vstup MC56F8367, kde dochází k jeho logickému vyhodnocení. Vysoká logická úroveň je dána *pull-up* rezistorem v MC56F8367. Ten musí být softwarově nastaven.

6.3.4 Výstupní obvody a budiče

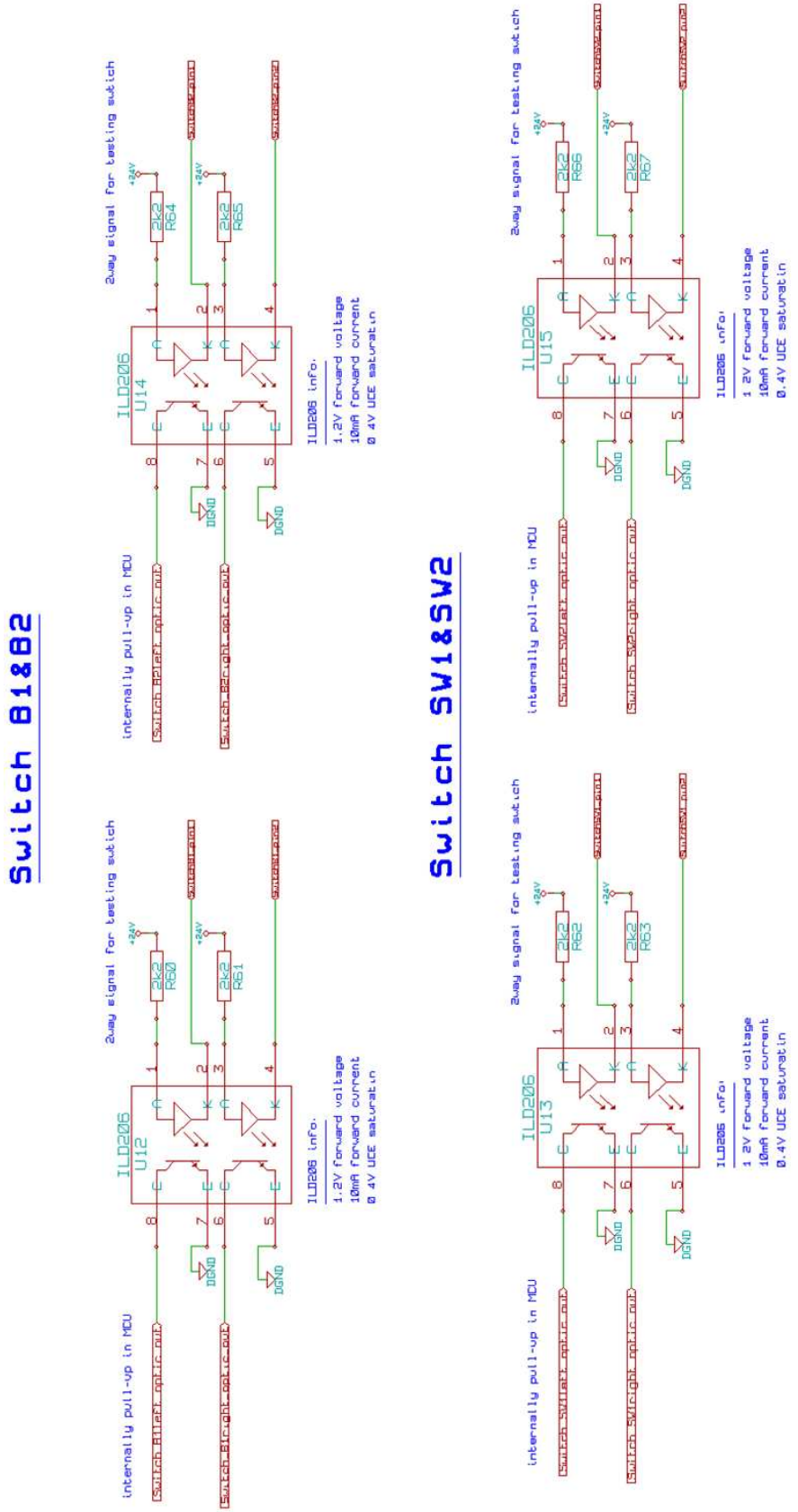
Mezi výstupní obvody ECU patří obvody ovládání přemosťovacíh (Bypass) a přepínacích (Switch) ventilů EHSA a obvody pro řízení polohy servoventilů EHSA.

Obvody ovládání a snímání přemosťovacích a přepínacích cívek

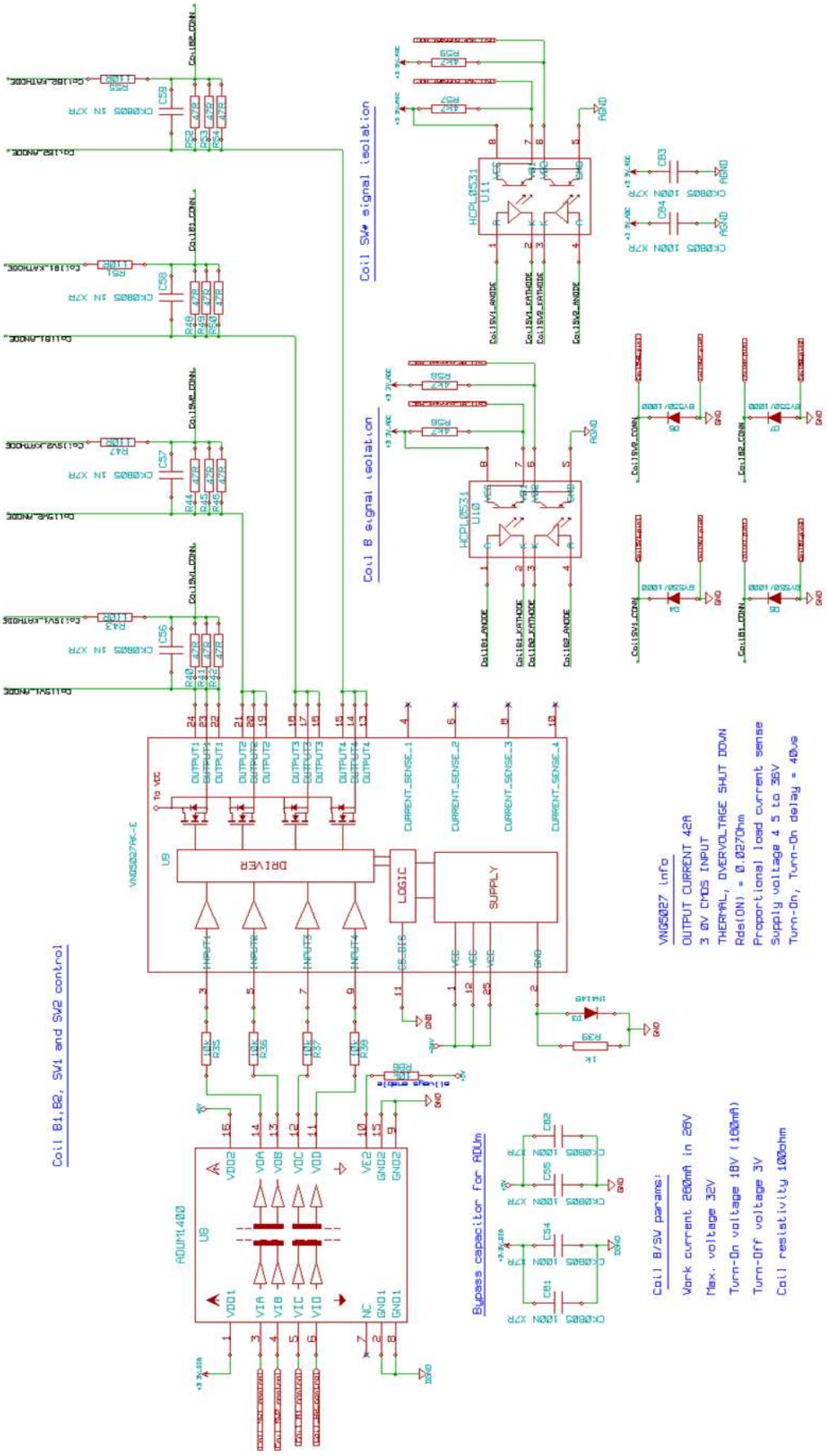
Obvody řízení solenoidů musí být podle požadavků z [5.1] galvanicky odděleny. Proto obsahují galvanický oddělovač ADUM1400, který kromě oddělení zabezpečuje také translátor napěťových úrovní. Obvod ADUM1400 obsahuje 4 jednosměrné digitální kanály. Každý kanál obsahuje izolační obvod s transformátorovou vazbou. Výhoda tohoto zapojení oproti klasickému optickému oddělení je především v nižší spotřebě, vyšší šířce pásma a jednoduchosti zapojení.

Galvanicky oddělený řídicí signál od MCU je následně zesílen přes výkonový budič VNQ5027AK, který obsahuje 4 výkonové spínací tranzistory a příslušnou řídicí logiku. Spínání solenoidů probíhá vždy vůči vstupnímu palubnímu napětí. Solenoidy jsou tak jednou stranou zapojeny na zem palubního napětí a druhým koncem na výstup výkonového tranzistoru VNQ5027AK. Konkrétní zapojení budícího obvodu je znázorněno na obr. 6.17.

Pro kontrolu správné funkce solenoidů a pro ověření sepnutí daného solenoidu je v zapojení navrženo také měření protékajícího proudu. Proud solenoidem je měřen nepřímo z úbytku napětí na čtyřech trojicích snímacích rezistorů R_{40}, R_{41}, R_{42}, dále R_{44}, R_{45}, R_{46}, dále R_{48}, R_{49}, R_{50} a R_{52}, R_{53}, R_{54}. Jelikož je nutné měřený obvod solenoidů oddělit od vstupů AD převodníků (MC56F8367) jsou použity optočleny HCPL0531 (U_{10}, U_{11}). Přenosová funkce tohoto optočlenu je nelineární a dochází tak ke zkreslení měřeného proudu. Z hlediska ověření spínací meze však toto zkreslení není důležité a stačí ověřovat mezní nebo hysterezní charakteristiky spínacích proudů. Tento krok je implementován v SW. Nelinearitu HCPL0531 lze také pomocí SW plně odstranit.



obr. 6.16 - Schéma obvodu pro čtení stavů přepínačů přemosťovacích a přepínacích ventilů



obr. 6.17 - Schéma zapojení obvodu pro ovládání a snímání solenoidů SW a B ventilů

Obvody pro řízení a snímání proudů cívek servoventilů

Rychlost pohybu pístnice EHSA je možno spojitě ovládat pomocí řízení proudu do cívek servoventilů EHSA. Servoventil hydraulického okruhu 1 a 2 obsahuje dvě řídicí cívky, tedy celkově EHSA obsahuje 4 cívky pro řízení pístnice EHSA.

Každá ECU obsahuje vždy jeden řídicí obvod pro cívku servoventilu v okruhu 1 a jeden řídicí obvod pro cívku servoventilu v okruhu 2. Tím je zajištěno, že lze řídit oba servoventily v obou okruzích v případě poruchy sekundární ECU.

Konkrétní zapojení pro ovládání proudu řídicích cívek servoventilů je zobrazeno na obr. 6.19. Jelikož jsou řídicí cívky sami o sobě galvanicky oddělené od společné země EHSA není potřeba zajistit galvanické oddělení, jako tomu bylo v případě řízení solenoidů přemosťovacích a přepínacích ventilů.

Elektronické schéma vychází z koncepce uvedené v [6.1]. Základem zapojení je obvod L6225. Tento obvod obsahuje dvojnásobný plný tranzistorový H-můstek s kompletní řídicí a budící logikou pro všechny výkonové tranzistory. Zároveň na každém výkonovém spínači jsou umístěny ochranné diody proti možnému naindukovanému napětí při spínání zátěže. Hlavní parametry L6225 jsou:

- Napájecí napětí od *8 V* do *52 V*
- Trvalé proudové zatížení až *1,4 A*
- Odpor tranzistorů při sepnutém stavu 0,73 Ω
- Frekvence spínání tranzistorů až *100 kHz* (možnost využití pro PWM řízení)
- Plně paralelní řízení obou tranzistorových můstků
- Ochrana proti přetížení a přehřátí
- Integrované ochranné diody pro každý výkonový tranzistor
- Vyvedeny piny pro připojení snímacího odporu

Další informace ohledně L6225 lze nalézt v [21]. Výsledné řízení cívek servoventilů je zajištěno pomocí PWM modulátoru v MC56F8367, který vstupuje do L6225 a ten následně budí H-můstek do něhož je zapojena jedna z cívek servoventilu. Pomocí PWM modulace lze pak spojitě měnit střední hodnotu proudu protékaného cívkou servoventilu a tím i polohu pístnice EHSA.

Šířkově modulovaný signál lze rozložit na nekonečnou Furrierovu řadu:

$$sig_{PWM}(t) = A_i\left(\frac{\tau_{\min}}{T_i} - \frac{\tau_{\max}}{T_i}\right)\sin(\Omega t) + \sum_{k=1}^{\infty}\sum_{n=-\infty}^{\infty}\frac{1}{k\pi}J_m(k\omega_i\tau_{\max})\sin[(k\omega_i - n\Omega)t - k\omega_i\tau_{\max}] - \sum_{k=1}^{\infty}\frac{1}{k\pi}\sin(k\omega_i t - \tau_{\min}) \quad (5)$$

Kde:

τ_{min}	-	minimální šířka impulzu
τ_{max}	-	maximální šířka impulzu
T_i	-	perioda PWM signálu
ω_i	-	úhlová frekvence sledů impulzů ($\omega_i = 2\pi / T_i$)
Ω	-	úhlová frekvence řídícího signálu
J_m	-	Besselova funkce *m*-tého řádu

Z předchozí rovnice (5) jasně plyne, že spektrum PWM signálu obsahuje stejnosměrnou složku a složky s frekvencemi $f\omega_i$ s f_Ω , složky s bočními frekvencemi ($k\, f\omega_i \pm f_\Omega$) a složky s bočními frekvencemi ($k\, f\omega_i \pm nf_\Omega$). (pro k a $n \in$ celá čísla) .

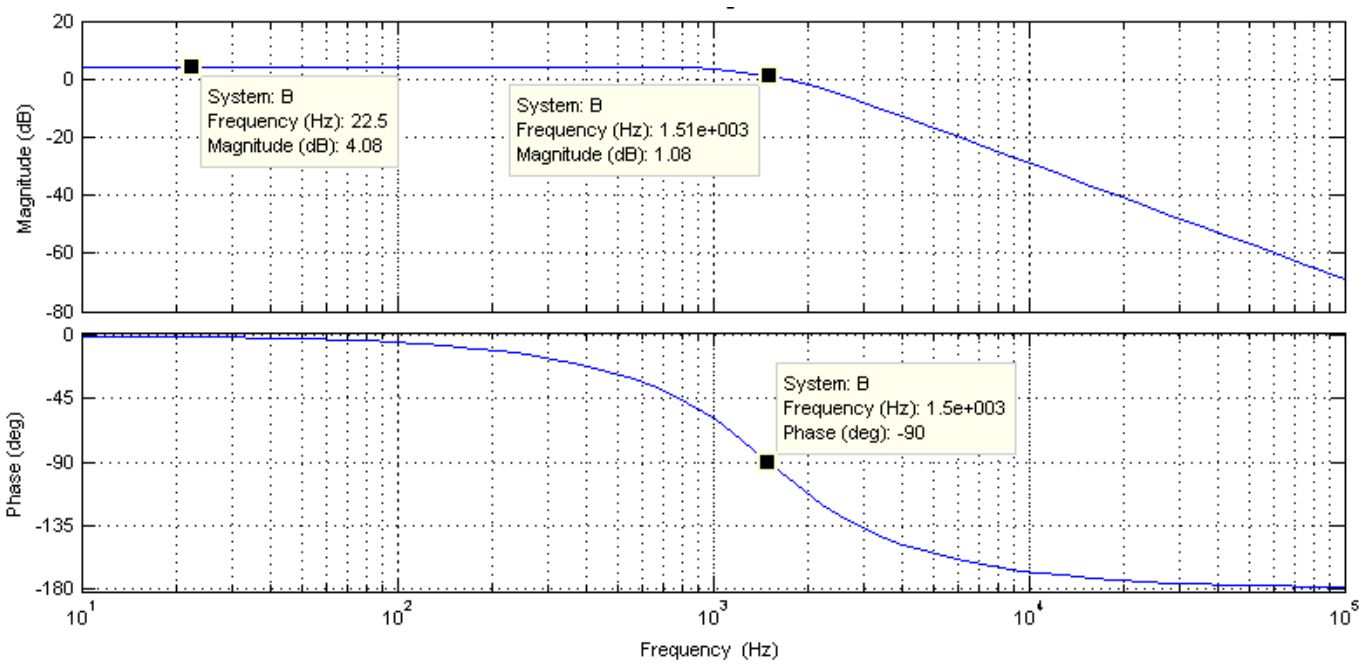
Přičemž amplitudy postranních složek velice prudce klesají, a tak většina výkonu je soustředěna do menšího pásma a lze ho orientačně vypočítat podle vzorce (6).

$$\Delta Fs \approx \frac{2}{\tau} \qquad \text{pro (95\% výkonu),} \qquad (6)$$

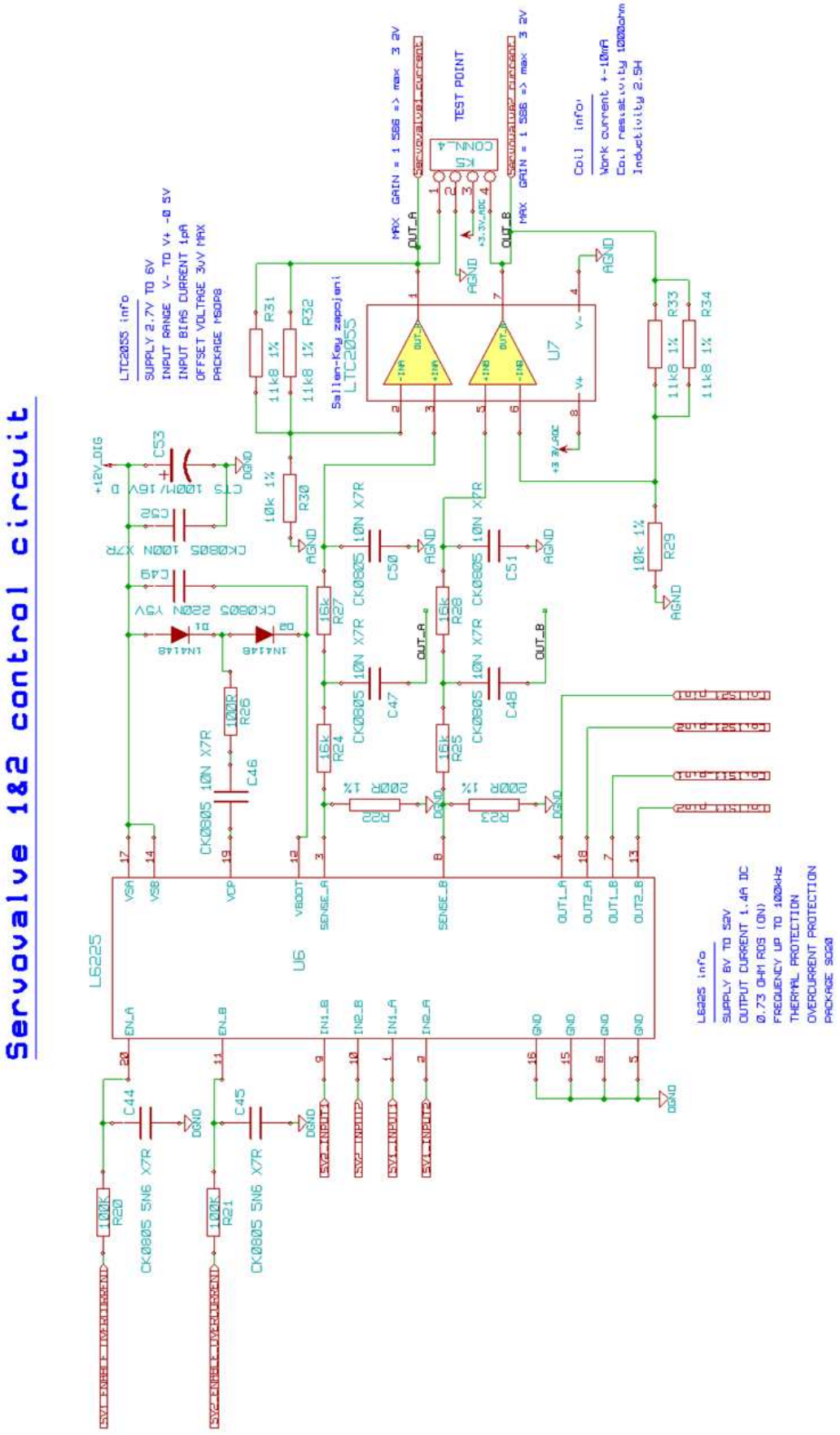
kde *ΔFs* je šířka pásma, které zaujímají modulované impulzy a τ je šířka impulzu.

Z výše uvedených rovnic přímo plyne velikost řídicího proudu do cívek servoventilů ale také princip měření střední hodnoty proudu. Zapojení s H-můstkem, v jehož středu je zapojen řízený solenoid nepotřebuje explicitní filtraci proudu, protože samotný solenoid spolu s můstkem vytváří dolnofrekvenční propust (DP) vyššího řádu. Zanedbáním kapacit tranzistorů se jedná o DP 1. řádu. Na výstupu H-můstku (piny 3 a 8 obvodu U_6) tak dostaneme nízkofrekvenční signál s výrazně potlačenými vysokofrekvenčními složkami. Další informace ohledně PWM modulace a PWM signálech lze nalézt v [39].

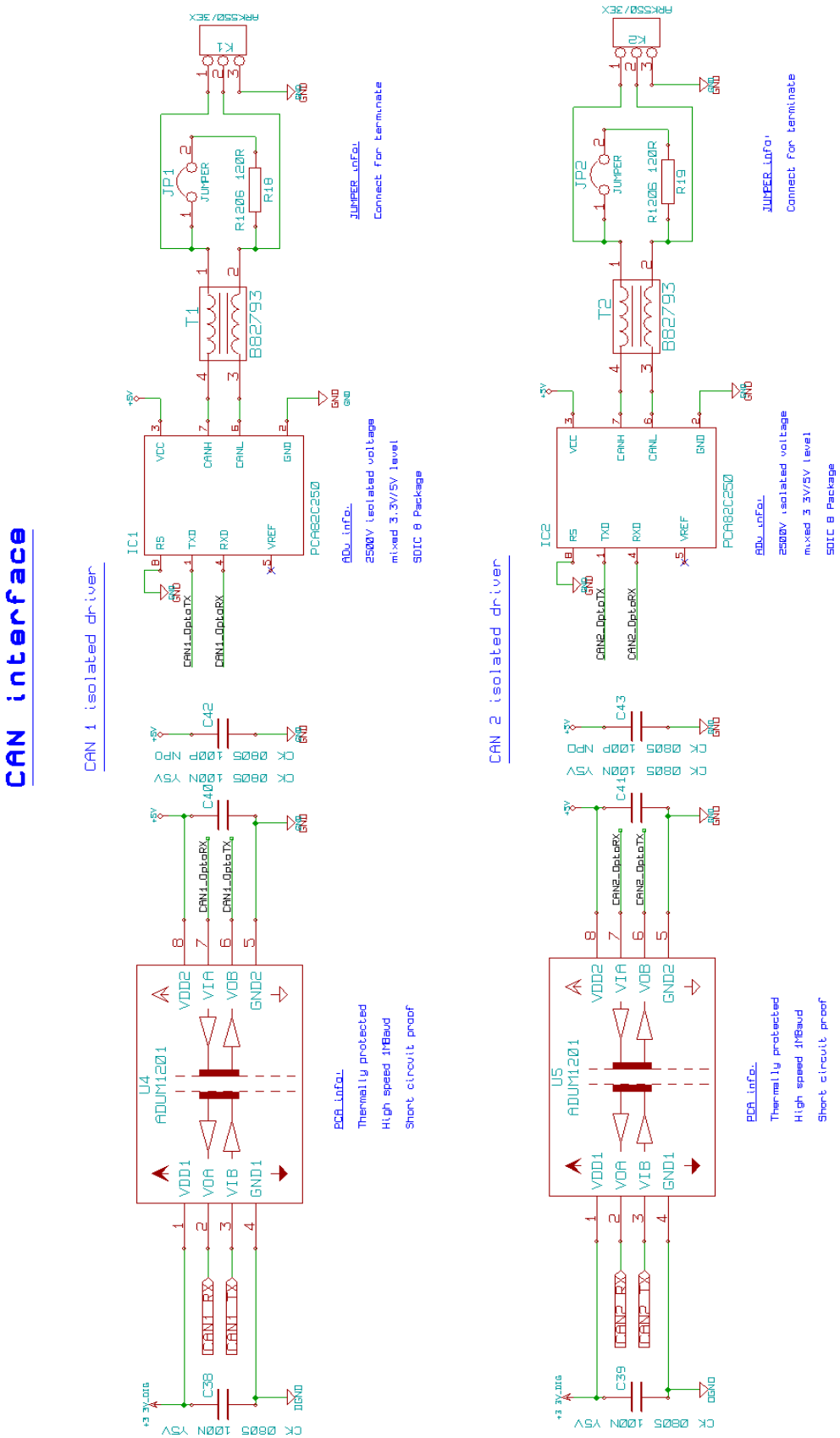
Z důvodu kontroly protékaného proudu cívkami je do výstupního obvodu L6225 připojen snímací odpor (R_{22} a R_{23}) protékaného proudu. Úbytek na těchto snímacích odporech je přímo závislý na velikosti protékaného proudu cívkou. Cívka servoventilu má odpor vinutí *1 kΩ*. Maximální protékaný proud má být ± *10 mA*. Z těchto údajů vyplývá, že při napájení *12 V* je napěťový úbytek na snímacím rezistoru $R_{22} = R_{23} = 200\Omega$ maximálně *2V*. Aby bylo využito celého rozsahu AD převodníku MC56F8367 je toto napětí zesíleno *1,6-krát* a filtrováno aktivní propustí 2. řádu typu Butterworth v modifikaci Sallen-Key (mezní frekvence f_C je nastavena na *1,5 kHz*). Zapojení aktivního filtru je navrženo tak, aby zesílený signál za filtrem byl v rozmezí *0 V* až *3,2 V*. Výpočet Butterworthova filtru byl proveden podle [40] ze strany 70. Na obr. 6.18 je umístěna výsledná frekvenční charakteristika DP filtru pro měření proudu v cívkách servoventilů.



obr. 6.18 – Bodeho frekvenční amplitudová a fázová charakteristika navrženého DP filtru



obr. 6.19 - Schéma zapojení obvodu pro řízení a snímání proudů cívek servoventilu



obr. 6.20 - Schéma zapojení obvodů pro komunikaci po CAN sběrnici

6.3.5 Komunikační obvody

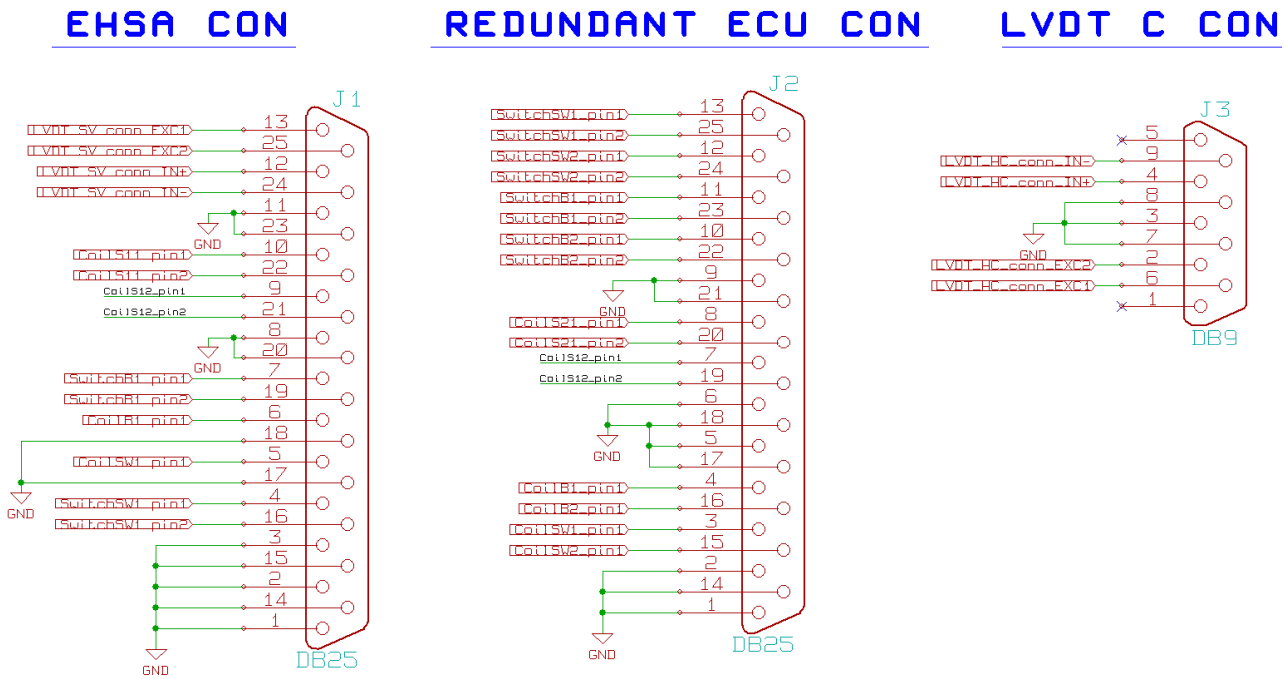
Každá ECU obsahuje dvě nezávislé komunikační linky pro CAN. MC56F8367 obsahuje dva nezávislé řadiče typu FlexCAN. Tyto řadiče zprostředkovávají obsluhu, vysílání a příjem zpráv po CANu.

MC56F8367 obsahuje z celkové fyzické vrstvy CANu jen příslušný řadič. Budiče CAN sběrnice musí být implementovány zvlášť. Navíc bylo požadováno galvanické oddělení logických signálů z MC56F8367 a signálů přicházejících z CAN sběrnice.

Zapojení kompletního komunikačního kanálu je na obr. 6.20. Pro galvanické oddělení signálové cesty bylo užito obvodu ADUM1201 (U_4, U_5). Tento obvod poskytuje galvanické oddělení až do *2500 V*. Podrobnosti o obvodu ADUM1201 lze nalézt v [19].

Po galvanickém oddělení vstupuje signál do budiče PCA82C250 (IC_1, IC_2). Základní parametry PCA82C250 jsou:

- Plně kompatibilní s normou *ISO 11898*
- Rychlost až do *1 Mbit/s*
- Ochrana sběrnice před rušivým prostředím
- Redukce RFI a EMI rušení
- Ochrana proti přehřátí, přepólování a zkratu



obr. 6.21 - Schéma zapojení konektorů ECU a EHSA

6.3.6 Propojení konektorů

Každá ECU je zapojena k jednomu okruhu EHSA. Pro tento účel je použit 25-ti vodičovým přímý kabel zapojený z konektoru J_1. Obě ECU jsou spojeny 25-ti vodičovým párově zkříženým kabelem zapojený na konektor J_2. Křížení je potřeba na párových signálech EHSA, tak aby jedna ECU přijímala vždy signál z druhého okruhu EHSA, než na který je sama zapojena. Dále je ke každé ECU zapojen jeden z dvojice LVDT hydraulických cylindrů. Ty jsou zapojeny na konektor J_3. Propojení konektorů je zobrazeno na obr. 6.21.

6.4 Fyzický návrh

V této kapitole je popsán návrh desky plošného spoje (PCB) a výroba ECU jednotky.

6.4.1 Deska plošného spoje

Deska plošného spoje byla navrhována v prostředí *KiCAD* v programu *PCBnew*. Na výslednou PCB byli kladeny následující požadavky

- Dvouvrstvá PCB
- Rozměr PCB *160 mm x 160 mm*
- Typ materiálu *FR4*
- Minimální křížení analogových a digitálních signálů
- Minimální ohřev PCB při špičkovém odběru ze všech periferních obvodů
- Rozlévané plochy mědi pro zemnění a odvod tepla
- Správné rozmístění komponent z hlediska EMC a rušení

Při umísťování součástek a kreslení propojovacích vodičů byli dodržovány všechna pravidla podle [22], [23] a [24]. Zároveň byl brán zřetel na správné a doporučené zapojení a umístění komponent podle datových listů jednotlivých součástek.

Výsledná deska plošného spoje a jednotlivé vrstvy jsou umístěny v příloze a na doprovodném CD. PCB je pro oba řídicí kanály (ECU1 a ECU 2) identická.

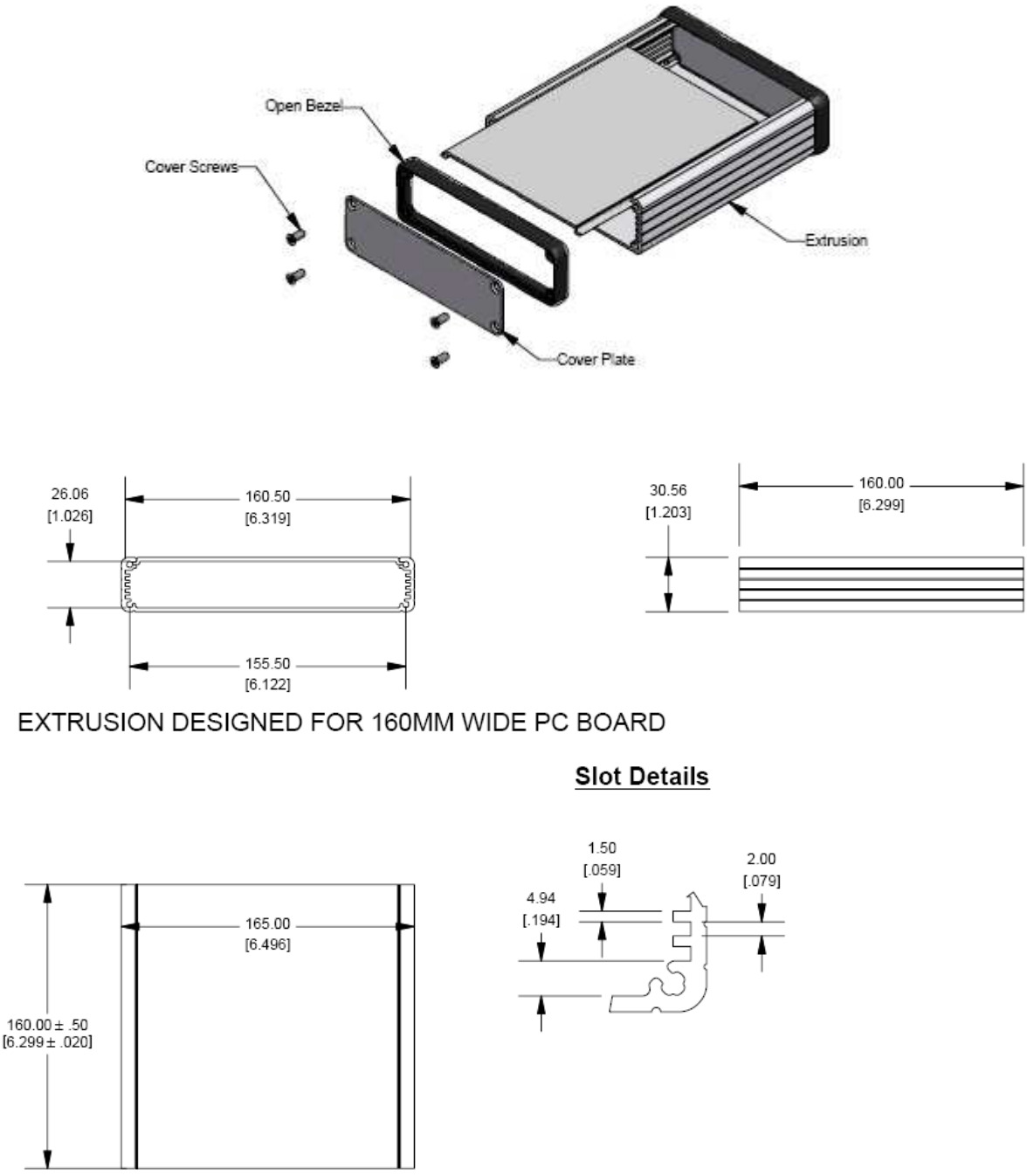
Pro možnost zasunutí PCB do slotů ochranného boxu je nutné dodržet nejen šířku *160 mm*, ale i okraj bez součástek *4 mm*. PCB je zasunuta do nejnižšího slotu ochranného boxu, proto bylo současně nutné zachovat okraj *6 mm* kvůli profilu krabičky pro montážní šrouby v předním a zadním panelu. Zároveň bylo potřeba při návrhu počítat, že na spodní straně PCB mohou být jen součástky nižší než *2mm*.

Routování a rozmísťování komponent na PCB bylo prováděno s ohledem na standard IPC600.

6.5 Mechanické provedení a realizace

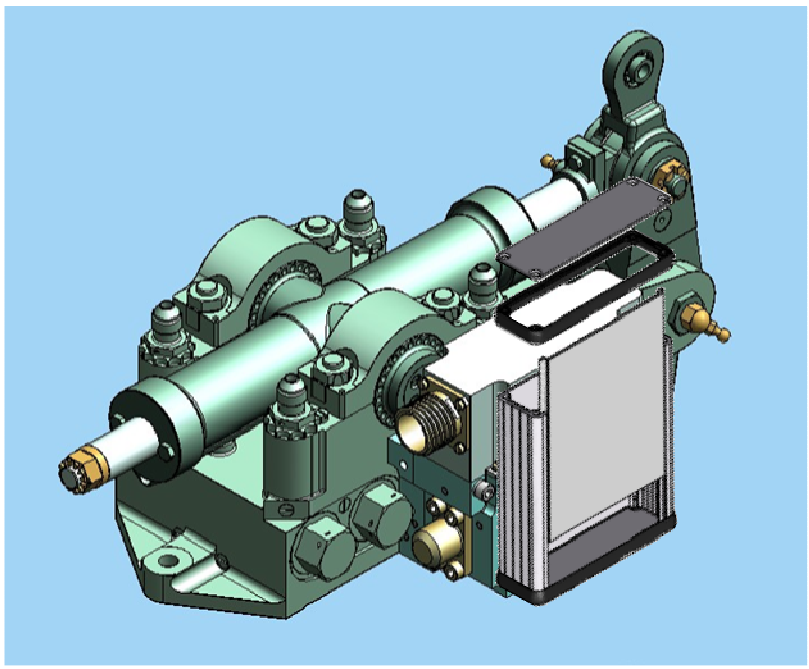
6.5.1 Ochranný kryt

Vestavba každého kanálu ECU byla plánována do hliníkové krabičky Hammond 1455R1601 (viz obr. 6.22) s kovovými bočními kryty jejíž délka je zkrácena na *130 mm* (délka PCB + místo na konektory CAN sběrnice).



obr. 6.22 – Nákres ochranného boxu pro uložení PCB (převzato z [6])

Krabička byla namontována svisle na boku příslušného kanálu EHSA. Umístění ochranného boxu je zobrazeno na obr. 6.23.

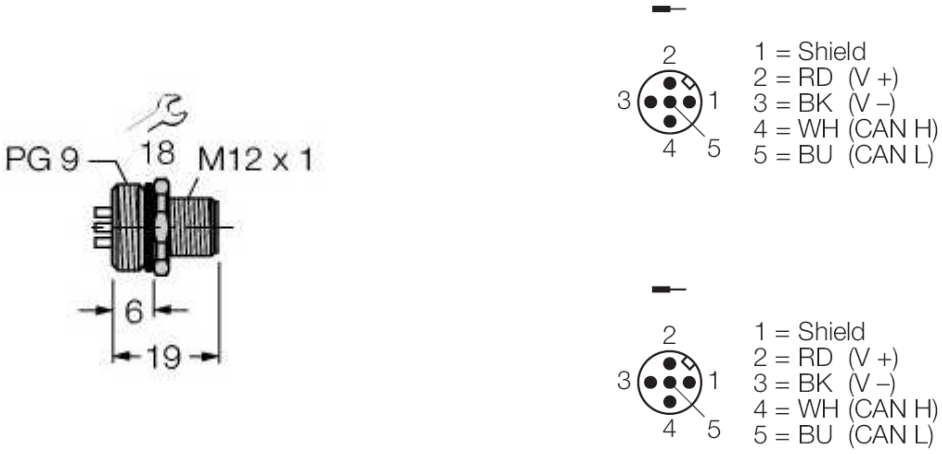


obr. 6.23 – Schéma umístění jednoho řídicího kanálu na EHSA (převzato z [6])

6.5.2 Konektory

Konektory pro CAN sběrnici jsou umístěny na jednom z čelních panelů ochranného boxu a na opačné straně boxu jsou umístěny konektory pro připojení EHSA (*EHSA con*) a druhé ECU (*ECU con*). Zároveň je na tentýž čelní panel vyveden konektor pro připojení LVDT snímače hydraulické válce.

Pro komunikaci po CAN byli použity standardní kulaté konektory M12 (viz obr. 6.24).



obr. 6.24 – Konektory pro CAN sběrnici (převzato z [6])

Pro propojení s druhou sesterskou ECU byl použit konektor typu CANNON DB25. Díky tomu je možné použít pro propojení ECU standardní kabeláž pro paralelní komunikaci mezi PC. Je však nutné párově prokřížit odpovídající signálové vodiče (viz obr. 6.21).

7 Software ECU

V této kapitole bude popsána architektura a kompletní návrhový proces softwaru (SW). Zároveň budou popsány softwarové vrstvy a použité techniky při tvorbě řídicího algoritmu. Návrh SW byl úzce spjat s koncepcí HW z kapitoly 6.1 a se zvolenou procesorovou platformou. Při vývoji SW byla použita metodika Model-Based Design a automatické generování kódu řídicího algoritmu z prostředí Simulink.

7.1 Architektura softwaru

Architektura SW se přímo odvíjí od použitého procesoru (56F8367). Zároveň bylo nutné zohlednit propojení jednotlivých komponent na PCB, hlavně pak z pohledu generování a časování řídicích signálů pro vstupně-výstupní obvody.

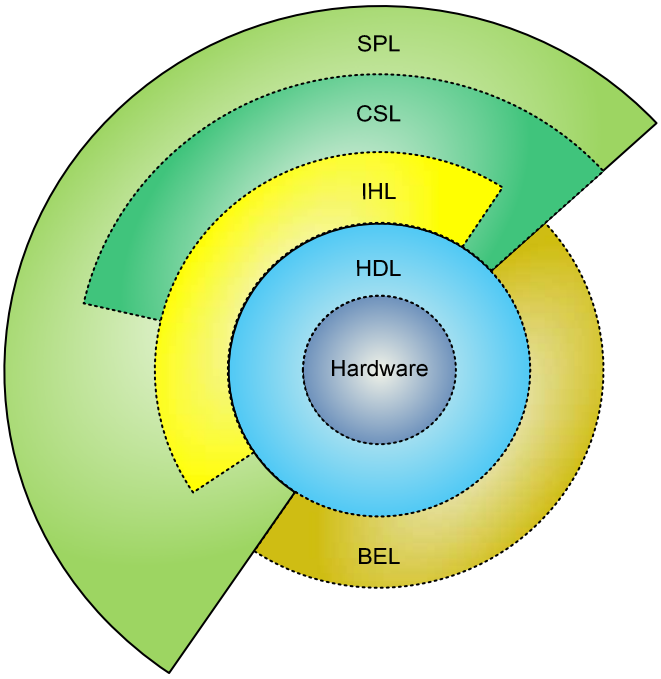
V této kapitole jsou popsány SW komponenty (vrstvy) a jejich vzájemné propojení. K tomuto účelu je vhodné použít vrstvový model, který ukazuje vazby mezi SW komponentami. Jednotlivé komponenty (vrstvy) vykonávají požadované úkoly a poskytují služby pro vyšší vrstvy. Na obr. 7.1 je zobrazen vrstvový model softwaru ECU. Každá z pěti softwarových vrstev vykonává skupinu jasně definovaných funkcí potřebných pro správnou činnost ECU. Komunikace mezi vrstvami je obousměrná, vždy od středu vrstvového modelu.

Komunikace mezi vrstvami SW probíhá v následujícím pořadí. Na počátku odezvy řídicího algoritmu musí vzniknout požadavek některé z periferie procesoru (GPIO, A/D převodník, HW časovač, apod.). Následně příslušný řadič vyvolá přerušení popř. čeká dokud nedojde k otestování stavového bitu (metoda bit polling). Obsluhu takto vzniklých přerušení či testování stavů periferií zabezpečuje HDL vrstva (Hardware dependent layer). Tato vrstva zaobaluje celý HW a obsahuje veškeré ovladače. Zároveň poskytuje množinu funkcí pro nadřazené vrstvy, čímž tvoří první rozhranní pro následující zpracování informací.

Nad HDL vrstvou budou umístěny dvě vyšší vrstvy IHL (Interrupt handling layer) a BEL (Background execution layer). Vrstva IHL zajišťuje veškerou obsluhu přerušovacího systému MC56F8367 a poskytuje funkce a služby pro vyšší vrstvy CSL (Communication services layer) a SPL (Signal processing layer). Mimo pravidelné vykonávání podprogramů přerušení je „na pozadí“ spuštěn normální synchronní běh programu. Veškeré podprogramy běžící mimo přerušovací systém zaštiťuje vrstva BEL. V ní se odehrává postupné nepreemptivní vykonávání instrukcí nejméně prioritních programů, jejichž pořadí není z hlediska závažnosti a časové kritičnosti ECU důležité.

Nad IHL vrstvou jsou postaveny dvě preemptivně vykonávané vrstvy. Vrstva CSL zajišťuje kompletní CANopen komunikaci pro obě CAN linky. Protokol CANopen je tedy implementován právě v této vrstvě. Informaci o přijetí nové zprávy z CANu zpracovává vrstva IHL. Ta spolu s drivery CAN řadičů, implementovaných ve vrstvě HDL, poskytuje služby pro příjem a vysílání na CAN sběrnice.

Nejvyšší vrstva SPL završuje celý řídicí software ECU. Vrstva SPL zajišťuje výpočet regulačního zásahu do EHSA a pomocí služeb nižších vrstev, aplikuje vypočtený akční zásah na výstup ECU. Referenční hodnotu pro regulaci EHSA dostává od CSL vrstvy a v opačném směru SPL zasílá do CSL veškeré synchronizační a chybové zprávy na CAN. Podstatnou část vrstvy SPL je navíc možné implementovat pomocí metodiky MBD z programu Simulink. To bylo současně jedním z hlavních cílů této diplomové práce. Detailní rozbor vrstev z hlediska implementace je uveden v kapitole 7.4.



obr. 7.1 – Vrstvový model software pro ECU

7.2 Návrhový proces

Implementovaný software lze z hlediska použitých vývojových prostředků a prostředí rozdělit do několika částí. Na obr. 7.2 jsou přehledně zobrazeny veškeré SW moduly, soubory a vývojová prostředí nutná pro vývoj řídicího softwaru ECU. Zároveň je vidět provázanost mezi použitými prostředky a schematicky naznačena skladba celého projektu. Význam a provázání kódu z obr. 7.2 ve velké míře odpovídá také adresářové struktuře z [28].

Software ECU se skládá ze tří samostatných podprojektů tvořených v programu MATLAB a Processor Expert (PE). Společné propojení zdrojových kódů z těchto nástrojů je uskutečněno v prostředí Codewarrior. V něm je utvořena základní struktura projektu (uživatelské moduly v jazyce C, generované moduly z použitých beanů v PE a Simulinku a soubory z projektu CanFestival). Implementace části programu, linkování všech programových modulů, knihoven a kompilace zdrojových kódů byla tedy prováděna v prostředí Codewarrior (CW) s nástavbou ProcessorExpert (PE).

Software pro ECU lze rozdělit na tři hlavní části:

- Funkční části psané v Codewarrioru, generovaný kód z PE, knihovní funkce MSL C 568000E
- Modely regulátoru a stavového automatu z MATLABu a generovaný kód z RTW
- CANopen komunikace a projekt CanFestival

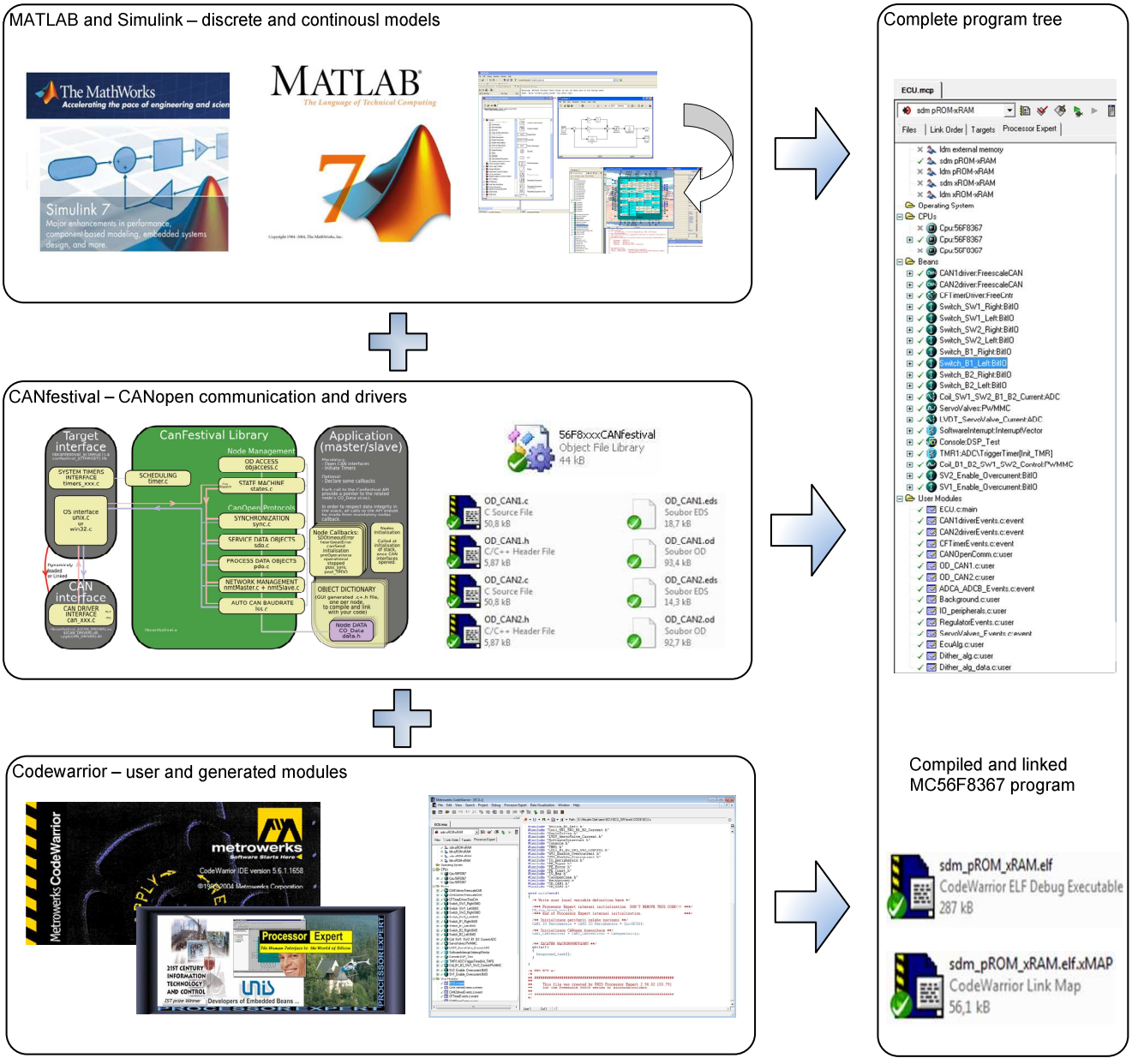
CANopen aplikační vrstva byla postavena na *opensource* projektu CanFestival. Modely regulátoru a stavového automatu byly navrženy v MATLABu a simulačním prostředí Simulink. Navržené modely v Simulinku byly přegenerovány pomocí nástroje Real-Time Workshop do jazyka C. Určité části softwaru ECU jsou tak vázány na platformu MC56F8367 (generované drivery a moduly z PE a knihovna CanFestivalu), nicméně většina implementovaného algoritmu je díky využití metodiky MBD přenositelná i na jiný typ procesoru.

Drivery jednotlivých periferií byli generovány za pomocí PE, přičemž bylo využito předem předpřipravených beanů. Architektura celého SW byla optimalizována na výpočetní náročnost a vlastnosti

platformy MC56F8367 (např. fixed point aritmetika). Byl kladen důraz na včasné vykonávání jednotlivých (kritických) úkolů, jako výpočet regulačního zásahu či měření vstupních signálů. Z tohoto důvodu bylo nutné správně volit hodnoty priorit přerušení jednotlivých periferií procesoru. Pro případy pozdních odezev byly implementovány diagnostické a signalizační prvky (zápisy do *Error* registrů CANopenu).

Pro potřebu CANopenové komunikace byl implementován kompletní komunikační stack. Zde bylo využito projektu CanFestival, který implementuje veškeré funkční algoritmy pro bezproblémovou komunikaci mezi uzly na CAN sběrnici. Bylo však nutné vytvořit drivery pro CAN periferie procesoru 56F8367 a navázat je na stávající interface CanFestivalu. Projekt CF byl popsán v kapitole 3.4.

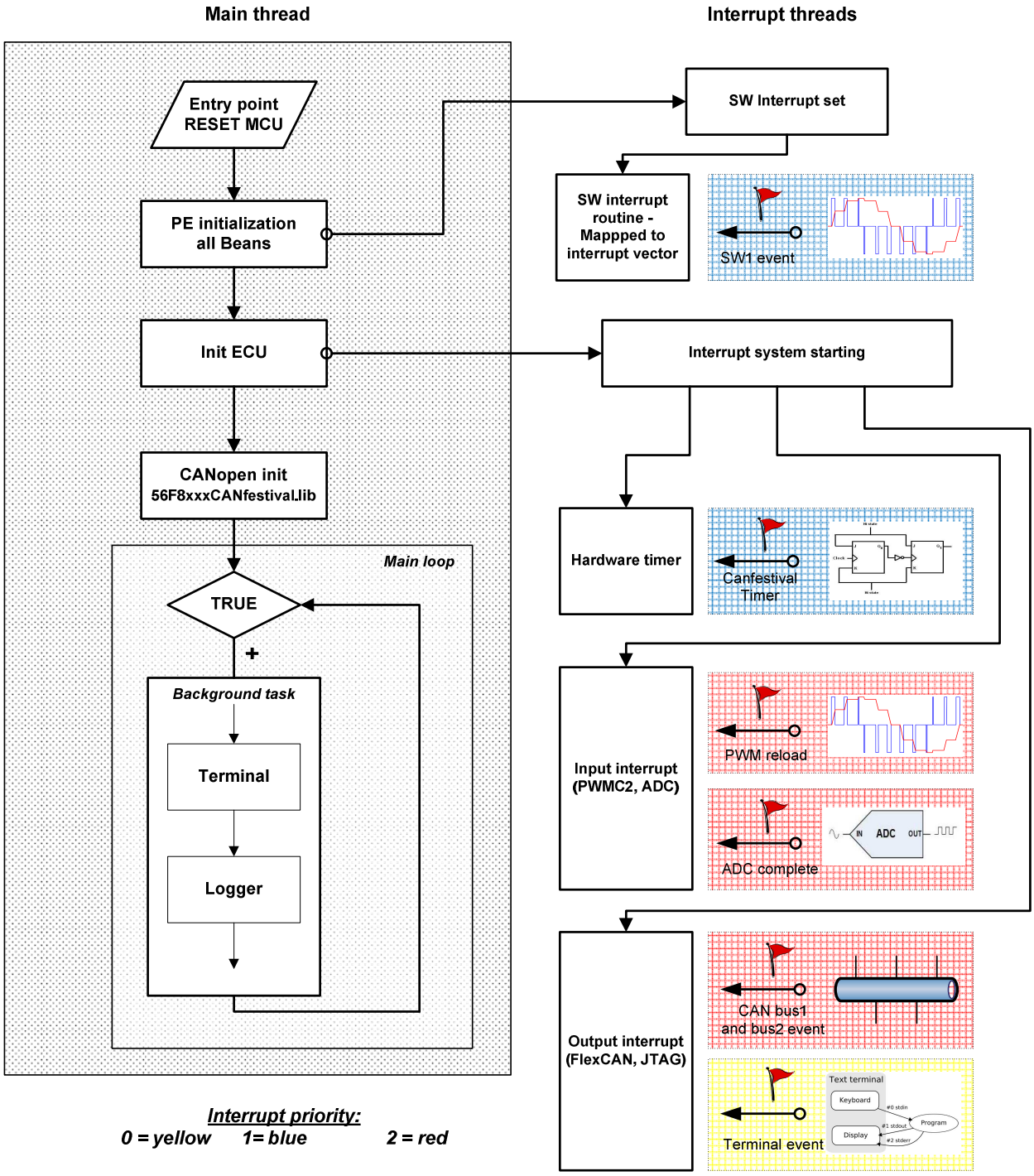
Další podrobnosti o použitých prostředích jsou napsány v kapitole 7.5. Kompletní zdrojové kódy programu lze nalézt na SVN na adrese [28] a [29] nebo na přiloženém CD.



obr. 7.2 – Softwarové části ECU a struktura projektu

7.3 Pracovní cyklus programu

Chování ECU jednotky je dáno požadavky na ovládání akčního členu EHSA. Pro splnění těchto požadavků je nutné, aby SW postupně nastavil periferie MC56F8367 a správně inicializoval komunikační a přerušovací systém. Teprve poté lze spustit hlavní regulační smyčku ECU a komunikaci po CANopenu.



obr. 7.3 – Vývojový diagram programu

Časová souslednost vykonávání jednotlivých úkolů programu je zobrazena ve formě vývojového diagramu na obr. 7.3. Program lze rozdělit na hlavní vlákno (Main thread), které zpracovává časově nekritické úlohy a na vlákna obsluhy přerušení. V hlavním vlákně začíná běh programu ECU. Nejdříve se inicializují periferie MC56F8367 a nastaví se parametry přerušovacího systému. Následně dojde k počátečnímu nastavení výstupních obvodů, spuštění generátoru PWM signálu a analogově-digitálních převodníků. Teprve poté jsou povoleny odezvy na příznaky vzniku přerušení.

Poté je inicializována komunikace po CANu a spuštěn stavový automat CANopenu. Jednotka ECU vystupuje v systému FBW, z hlediska CANopen komunikace, jako dvojnásobný *slave*. Spuštění komunikace proto nastává až po přepnutí ECU do operačního módu (*Operational)* nadřazenou komponentou FCC.

Po dokončení inicializačních rutin přechází ECU jednotka do hlavní smyčky *backgroud tasku*. V této chvíli je ECU jednotka připravena na aktivní ovládání EHSA. V *background tasku* se vykonávají časově nekritické úlohy (logování změřených dat do souboru, výpisy ladících informací do terminálu vývojového prostředí).

Po nastavení a spuštění přerušovacího systémů začíná běžet paralelně druhé vlákno (Interrupt thread). Přerušovací systém zajišťuje preempci hlavního vlákna a přepíná vykonávání instrukcí mezi podprogramy přerušení a *background taskem*. Pořadí, v jakém se vykonávají podprogramy přerušení závisí na hodnotě priority. Na obr. 7.3 jsou barevně odlišeny priority a tedy i pořadí, v jakém se vykonávají odezvy na výskyt přerušení od jednotlivých periferií.

7.4 Struktura programu

Vykonávaný program v ECU zajišťuje požadované vnější chování ECU jednotky v rámci celého FBW systému. Z vnitřního pohledu lze strukturu implementovaného programu rozdělit do vzájemně propojených a mezi sebou komunikujících vrstev. Každá vrstva zabezpečuje specifickou úlohu a vymezuje rozsah svého použití pro další navazující vrstvy. Vrstvy jsou složeny z několika programových modulů. V následujících podkapitolách jsou detailně rozebrány programové části a implementované moduly.

7.4.1 Vrstva HDL

Tuto vrstvu tvoří funkce pro inicializaci periferií, nastavení vektorů přerušení a drivery jednotlivých periferií (*beanů*). V hierarchickém pořadí je nejnižší vrstvou. HDL zaobaluje hardware a poskytuje rozhranní pro softwarové ovládání periferních obvodů MC56F8367.

Tato vrstva je tvořena programovými moduly generovanými z PE (*ADCA_ADCB_Events*, *CAN1driver*, *CAN1driverEvents*, *CAN2driver*, *CFTimerDriver*, *Cpu*, *ECU*, *CFTimerEvents*, *Coil_B1_B2_SW1_SW2_Control*, *Coil_SW1_SW2_B1_B2_Current*, *LVDT_ServoValve_Current*, *ServoValves*, *SoftwareInterrupt*, *Switch_B1_Left*, *Switch_B1_Right*, *Switch_B2_Left*, *Switch_B2_Right*, *Switch_SW1_Left*, *Switch_SW1_Right*, *Switch_SW2_Left*, *Switch_SW2_Right, Vectors*) a moduly *CANOpenComm*, a *IO_peripherals*. Ve vrstvě HDL se spouští běh vrstvy IHL. Inicializační vlákno vrstvy HDL zde končí a program přechází do vykonávání funkce *background_task()* BEL vrstvy. Programové moduly HDL vrstvy jsou využívány při každém volání z IHL a BEL vrstvy.

7.4.2 Vrstva BEL

Normální běh programu mimo dobu obsluhy přerušení setrvává ve funkci *background_task()*. Ta tvoří hlavní část BEL vrstvy. V této funkci se volají veškeré nekritické funkce. Funkce *background_task()* je volána v hlavní smyčce vrstvy HDL. Pro ladící účely byli implementovány do aktuálního kódu výpisové funkce na konzoli prostředí CodeWarrior. Z hlediska chování a ovládání EHSA nemá BEL vrstva žádný efekt a neovlivňuje vrstvy IHL, CSL a SPL.

7.4.3 Vrstva IHL

Tato vrstva zabezpečuje obsluhu části událostí generovaných od přerušovacího systému MC56F8367. Jsou zde implementovány hlavní obslužné funkce (ISR), ve kterých jsou pak volány příslušné podprogramy z vrstvy komunikačního systému (CSL) a vrstvy signálového zpracování (SPL).

IHL tak zabezpečuje reakci na přerušení hlavního vlákna od AD převodníků, HW časovačů, periferních obvodů, komunikačních obvodů a softwarového přerušení. Přerušení od AD převodníků (převodník A a B) je zpracováno v rutinách ISR (Interrupt service routine) v modulu ADCA_ADCB_Events. Rutina LVDT_ServoValve_Current_OnEnd() obsluhuje převodník ADCB pro LVDT senzor a snímání proudu v cívkách servoventilů. Přerušení od tohoto AD převodníku je voláno po každém dokončeném odběru všech měřených kanálů. Frekvence měření je $f_{ADCA} = 40kHz$ a je synchronizováno s generátorem PWM signálu pro řízení proudu v cívkách servoventilů. Měření jednoho kanálu trvá $1{,}7\mu s$. Změřená data jsou v této ISR kopírována do objektových slovníků CAN 1 a CAN 2.

Rutina *Coil_SW1_SW2_B1_B2_Current_OnEnd()* obsluhuje přerušení od převodníku ADCA. Na tento převodník je připojeno měření stavu sepnutí v solenoidech SW a B. Dvoustavové snímání proudu tak bylo nahrazeno *12*-ti bitovým měřením. Tento signál je následně zpracován ve vyšší vrstvě SPL. Nastavení ADCA je totožné s výše uvedeným měřením v ADCB. V této rutině jsou současně čteny aktuální hodnoty mikrospínačů solenoidů SW a B. Díky tomu je měření proudů v solenoidech SW a B a měření polohy mikrospínačů těchto solenoidů vzájemně synchronizováno. Obsluha ostatních zdrojů přerušení je prováděna ve vyšších vrstvách SPL a CSL.

Pro včasné odezvy na vzniklé požadavky od hardwaru ECU bylo potřeba zvolit hodnoty priorit k daným přerušením. Díky tomu bylo možné určit pořadí v jakém budou obsluhy přerušení vykonávány. Přerušovací systém MC56F8367 byl nastaven tak, aby výpočet regulačního zásahu do EHSA byl ukončen v časovém rozmezí $350\mu s$. Pak bylo možné volit periodu řízení EHSA $f_{control} = 2kHz$. Z tohoto pohledu bylo nutné implementovat „vysokoprioritní“ ISR jako krátké a rychlé podprogramy. Pro splnění výše uvedených časových charakteristik řídicího algoritmu byl navrhnut tzv. systém časových značek. Více o tomto principu výpočtu regulačního zásahu a samotné koncepci systému časových značek bude uvedeno na závěr této podkapitoly.

Nastavení přerušovacího systému a rozvržení priorit

Preempce hlavního vlákna, zpracovávající nekritické funkce vrstvy BEL, zajišťuje reakci na asynchronně generované události. Rozlišení významu a tedy i pořadí v jakém jsou obsluhy vykonávané, určuje nastavená priorita přerušení. Platforma MC56F8367 rozlišuje 3 stupně priorit. Nejvyšší priorita, značená číslem *2*, je přiřazena funkčně i časově kritickým událostem. V případě, že by nebyly tyto typy událostí včasně obslouženy, může dojít k chybné reakci a chování ECU. Druhý stupeň priority je stupeň *1*. Do těchto priorit jsou umístěny všechny časově nekritické ale funkčně kritické události. Mezi tyto typy události patří všechny služby, u kterých malé zpoždění reakce na vznik přerušení nezapříčiní chybu celé ECU. Poslední stupeň, značený jako *0*, je určen pro časově a funkčně nekritické služby.

Priority přerušení byli nastaveny následujícím způsobem:

- Nejvyšší (interně priorita 2)
    - AD převod (po dokončení měření)
    - *Reload* výstupního PWM signálu
    - Komunikace po CAN 1 a CAN 2

- Střední (interně priorita 1)
    - Výpočet regulačního zásahu (SW1 interrupt)
    - Hardwarový časovač pro CANopen komunikaci

- Nejnižší (interně priorita 0)
    - Pro ladící účely

Harmonogram měření a výpočet regulačního zásahu

Vedle přiřazení jednotlivých priorit daným zdrojům přerušení bylo nutné vyřešit synchronizaci naměřených dat a určit okamžiky začátku a konce výpočtu regulačního zásahu. Přitom bylo nutné ošetřit případ pozdního dokončení výpočtu regulačního zásahu.

Tento kritický bod regulace byl vyřešen pomocí systému časových značek. Tato metoda vychází z konceptu generování pevné časové základny (synchronní s generátorem PWM), od které se odvíjí okamžiky začátků (značka začátku), konce výpočtů regulace (značka konce) a okamžiky zpracování dat ze vstupních periferních obvodů.

Princip systému časových značek, jeho časové rozvržení a pořadí vykonávaných akcí je zobrazeno na obr. 7.4. Z něho lze pozorovat, že časová základna generuje periodické hodiny. Zdroj časové základny je generátor PWM signálu (frekvence generátoru byla zvolena $f_{PWM} = 40kHz$, což je současně hodnota frekvence generátoru časových značek).

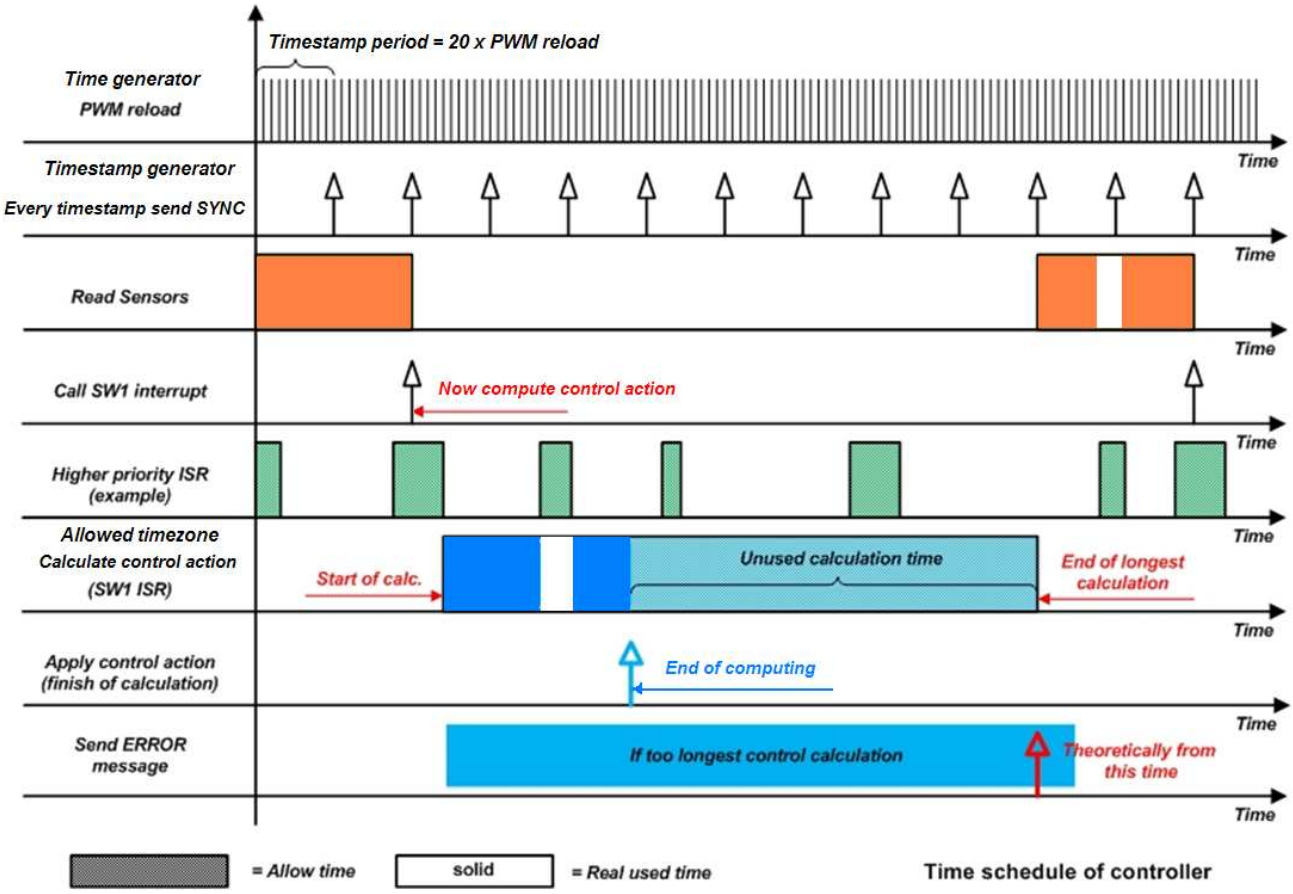
Při každém přetečení (reload) PWM čítače dochází k vygenerování příznaku přerušení. Tento příznak je odchycen přerušovacím systémem a reakcí na něho je inkrementace čítače časových značek. Čítač je implementován tak, že při hodnotě *19-ti* načítaných pulzů (časových značek) vynuluje svou hodnotu. Během *20-ti* načítaných pulzů musí dojít ke kompletnímu výpočtu regulačního zásahu pro akční člen EHSA. Z toho vyplývá hodnota frekvence řídicího zásahu (regulačního výpočtu) do EHSA, která vychází $f_{control} = 2kHz$. Časová vzdálenost dvou značek tak činí $T_{TSG} = 25\mu s$, nový krok regulačního zásahu je pak $T_{TSG} = 500\mu s$. Během $T_{TSG} = 500\mu s$ musí dojít k přečtení vstupních naměřených dat z periferních obvodů, následně výpočtu regulačního zásahu a aplikaci vypočteného zásahu na výstupní řídicí obvody.

Čtení vstupů pro regulátor se odehrává v intervalu $0\mu s$ až $100\mu s$. Následně jsou tyto přečtené hodnoty kopírovány na vstup regulátoru a poté v intervalu $100\mu s$ až $450\mu s$ probíhá výpočet regulačního zásahu. Po dokončení jsou vypočtené hodnoty aplikovány na výstupní obvody ECU.

V případě, že výpočet regulačního zásahu přesáhne dobu $350\mu s$, tj. se dokončí po čase $450\mu s$, zajistí systém odeslání chybové zprávy na CANopen (služba Error control protokol - NMT). Nadřazený systém FCC je tak včas informován o vzniku poruchy a může porouchaný kanál odpojit od řízení EHSA.

Výhodou použitého algoritmu časových značek je možnost rychlé úpravy a nastavení potřebné délky výpočtu regulačního zásahu. Při ladících a testovacích procesech lze tak efektivně nalézt hranici minimálního času potřebného pro výpočet regulačního zásahu. Tento fakt je důležitý zejména v případě automaticky generovaného kódu regulátoru a stavového automatu ze Simulinku (obtížně odhadnutelný výpočetní nárok navrhnutého algoritmu).

V případě, že by byla doba výpočtu regulátoru nedostatečná, je ještě možné snížit frekvenci časové základny PWM generátoru nebo zvětšit počet časových značek v čítači časových značek. V obou případech dojde k prodloužení periody řízení, a tím i k prodloužení vzdálenosti mezi jednotlivými značkami.



obr. 7.4 - Časový harmonogram měření a výpočtu regulačního zásahu ECU

7.4.4 Podvrstva CSL – CANopen

Nad vrstvou IHL je umístěna komunikační vrstva CSL. Tato vrstva implementuje protokol CANopen a poskytuje služby pro vyšší podvrstvu řídicího algoritmu (SPL). Základem pro implementaci CANopen protokolu je projekt Canfestival (CF), který byl popsán v kapitole 3.4. Pro využití CF bylo potřeba *naportovat* zdrojové kódy tohoto projekt na procesor 56F8367. To obnášelo vytvoření ovladačů pro hardware CANu a hardwarového časovače. Navíc bylo nutné navázat stávající rozhranní CF na zdrojový kód vrstvy SPL a IHL.

Pro *portaci* kódu CF bylo využito vygenerovaných funkcí Processor Expertu a přednastavených beanů řadičů *FlexCAN* a časovače *FreeCntr*. Byli tak vytvořeny služby pro časovač CANopen (funkce *timerInit(),setTimer(),getElapsedTime()*), služby pro inicializaci, vysílání a příjem z CAN řadičů (funkce *onCan2Receive(), onCan1RTRmessage(), onCan2RTRmessage(), canSend(), Can1Init(), Can2Init(), onCan1Receive()*). Pro komunikaci po CAN bylo využito všech *16-ti* přijímacích a vysílacích bufferů. Výsledný kód byl otestován a následně v prostředí Codewarrior slinkován do jediné knihovny *56F8xxxCANfestoval.lib*. Tato knihovna byla následně zaintegrována do stávajícího kódu ECU. Veškeré zdrojové kódy CF s implementovanou podporou pro rodinu 55F8xxx lze nalézt v [29]. Kromě knihovních funkcí CF bylo potřeba implementovat objektový slovník (OD) resp. slovníky obou CAN řadičů. Z aplikačního hlediska OD zastřešuje pohled na aktuální stav komunikačního uzlu. K implementaci OD byl použit nástroj *objdictEdit*. V něm byli vytvořeny dva objektové slovníky, každý pro jeden z CAN řadičů a následně generovány zdrojové soubory slovníků v jazyce C (*OD_CAN1 a OD_CAN2*). Vygenerované soubory byli linkovány ke stávajícímu projektu ECU.

Aktualizace, synchronizace a konzistence dat v OD

Jedním z problematických míst při implementaci softwaru bylo řešení konzistence dat v obou objektových slovnících dané ECU a synchronnost obou ECU při čtení dat. Synchronnost čtení dat je důležitá zejména z hlediska rozdílných začátků výpočtu regulačního zásahu na obou ECU. Konzistence dat je pak důležitá z hlediska výběru zdroje dat pro výpočet regulačního zásahu. V opačném případě hrozí chybný nebo pozdní výpočet regulačního zásahu do EHSA. Vrstva CSL byla implementována tak, aby zajistila automatickou aktualizaci dat a synchronnost čtení v obou OD dané ECU. Výběr zdroje dat je řešen ve vrstvě SPL.

Aktualizace a konzistentnost dat je zajištěna v jednotlivých ISR při čtení nových dat, synchronizace čtení v obou ECU je zajištěna použitím synchronního režimu při komunikaci po CANopen (globální synchronizace pomocí SYNC).

Dalším problémem, při implementaci softwaru byla aktualizace dat sekundární ECU při čtení z nenásobných prvků EHSA. Typickým příkladem bylo čtení LVDT senzoru servoventilu v daném okruhu EHSA, kde není k dispozici signál pro druhou ECU na druhém řídicím kanále. Proto bylo potřeba implementovat aktualizaci dat z jednoho řídicího kanálu (ECU 1) do druhého řídicího kanálu (ECU 2). Pro tento účel byla využita možnost synchronní výměny dat přes CANopen komunikaci. Aktualizovaná informace je v tomto případě ale časově opožděna o jeden synchronizační úsek (vzdálenost značek SYNC), který činí přibližně *100 ms*.

7.4.5 Podvrstva SPL – regulátor polohy

Nejvyšší softwarovou vrstvou je podvrstva SPL. Tato vrstva implementuje řídicí algoritmus a tedy i výsledné chování ECU jednotky. Řídicí algoritmus realizuje dvě hlavní funkce. První funkcí je stavový automat ECU pro ovládání módů EHSA. Druhou funkcí je regulace polohy pístnice EHSA.

Návrh regulátoru pro ovládání polohy pístnice byl proveden v prostředí MATLAB a Simulink. V Simulinku byl nejdříve vytvořen model regulátoru pro řízení modelu soustavy EHSA. Následně byl k regulátoru navržen stavový automat pro ovládání pracovních módů EHSA. Vytvořený model regulátoru byl testován na modelu EHSA, který byl vytvořen v bakalářské práci [7]. Testování řídicího systému ECU na modelu a skutečné soustavě EHSA je popsáno v kapitole 8. Aktuální model navrženého řídicího algoritmu a vygenerovaný kód v jazyce C lze nalézt v [28].

Vrstva SPL je tvořena programovými moduly *RegulatorEvents*, *ServoValves_Events, EcuAlg, Dither_alg* a *Dither_alg_data*. Modul *RegulatorEvents* implementuje ISR softwarového přerušení SW1. V této funkci je umístěno načtení dat , spuštění a aplikace regulačního zásahu pro EHSA, zároveň je vypočítán jeden krok stavového automatu ECU. Data jsou čtena z obou OD. Výpočet řídicího algoritmu je prováděn ve funkci *EcuAlg_step()*, která je umístěna v modulu *EcuAlg*. Aplikace vypočteného zásahu je realizována službou *applyCoilsToOut()*, implementovanou ve vrstvě HDL v modulu *IO_peripherals*.

V modulu *ServoValves_Events* je implementován systém časových značek, aplikace *ditheru* a inicializace softwarového přerušení SW1 pro výpočet regulačního zásahu. Všechny výše zmíněné algoritmy jsou vykonávány v rutině *ServoValves_OnReload()*. V téže rutině je také implementován kontrolní algoritmus konce výpočtu regulačního zásahu. Výsledek této kontroly je navázán na službu vrstvy CSL.

Struktura řídicího algoritmu

Modul *EcuAlg* je generován z nástroje RTW ze *Simulinku*. V tomto modulu jsou obsaženy funkce a struktury pro výpočet regulačního zásahu. Hlavní službou je funkce *EcuAlg_step()*, která vypočte při každém volání jeden krok regulátoru.

Základem pro vygenerování funkce *EcuAlg_step()* je model v Simulinku. Model řídicího algoritmu ECU obsahuje čtyři podčásti:

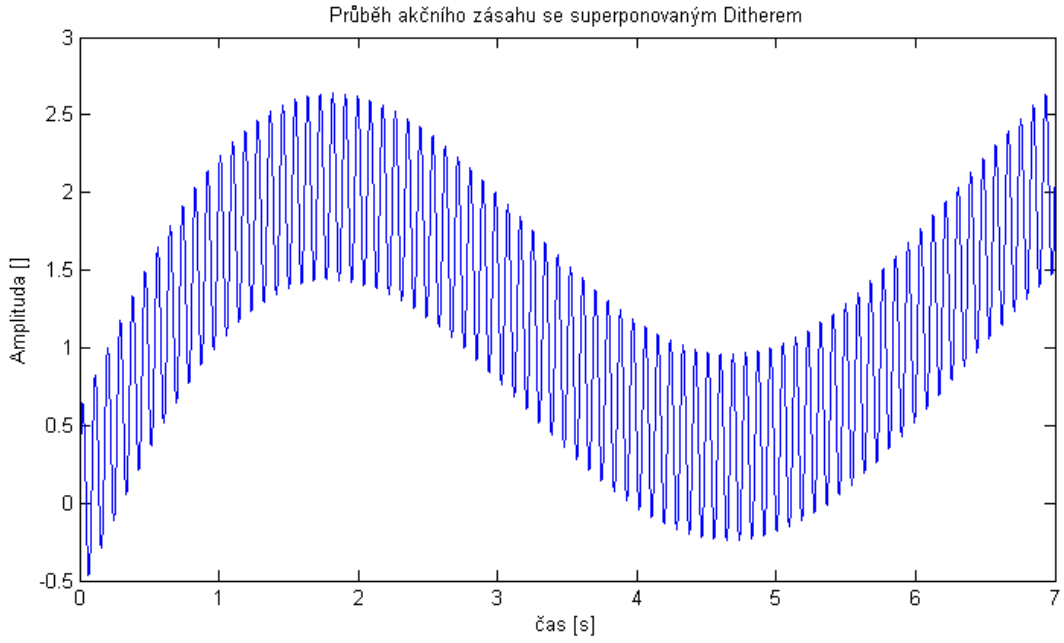
- Model regulátoru polohy pístnice EHSA (Servovalve position control)
- Model detektoru poruch (Error detection)
- Model detektoru stavu CAN sběrnice
- Model stavového automatu ECU

Regulátor polohy pístnice EHSA byl realizován jako proporcionálně integrační (PI) regulátor s antiwindup zapojením. Konstanty PI regulátoru jsou navázány na objektové slovníky CAN a jsou mapovány do PDO zpráv CANopenu. Díky tomu bylo možné nastavit PI regulátor přímo z nadřazeného systému. V budoucnu je možné implementovat do řízení EHSA také metodu adaptivního řízení zesílení (Gain sheduling). Nadřazený systém může také díky tomuto mechanismu flexibilně reagovat na poruchy ve FBW řízení přenastavením konstant regulátoru dané ECU. Součástí regulátoru je také člen výpočtu regulační odchylky. Detektor poruch obsahuje bloky pro indikaci a detekci chyb a poruch v periferních obvodech ECU. Detektor stavu CAN sběrnice sleduje aktuální stav na CAN sběrnici a vybírá zdroj dat pro regulátor polohy pístnice. Poslední součástí je stavový automat ECU, který je určen pro volbu pracovního módu EHSA.

Dither

Dalším generovaným algoritmem je modul *Dither_alg* a *Dither_alg_data*. Tyto moduly obsahují funkci generování přídavného signálu k regulačnímu zásahu. Tento přídavný signál (*dither*) je superponován na akční zásah regulátoru polohy. Úkolem *ditheru* je minimalizovat, v lepším případě úplně odstranit, suché tření při přesouvání šoupátka servoventilu, díky tomu se zvětší rozlišení posuvu a zmenší hysterezní vlastnosti šoupátka. Frekvence *ditheru* je závislá na použitém akčním členu. Průběh akčního zásahu se superponovaným *ditherem* je ukázán na obr. 7.5.

Pro servoventil použitý v EHSA je doporučeno od výrobce použít frekvenci v rozmezí *100 Hz* až *400 Hz* o amplitudě *±20%* z rozsahu akčního zásahu. Tvar *ditheru* má mít sinusový průběh. Výpočet jednoho kroku algoritmu *ditheru* probíhá v modulu *RegulatorEvents* ve funkci *ServoValves_OnReload()*. Perioda výpočtu *ditheru* je $25\mu s$. Aktuální model generátoru *ditheru* je umístěn v [28].



obr. 7.5 – Akční zásah se superponovaným ditherem

7.5 Vývojové prostředky a použité nástroje

Pro vývoj softwaru bylo použito hned několik programovacích prostředků a prostředí. Návrh SW probíhal podle [7.2]. V následujících odstavcích jsou popsány a shrnuty nejdůležitější vlastnosti použitých vývojových prostředí.

7.5.1 Prostředí Codewarrior

Základním prostředím pro vývoj SW bylo integrované vývojové prostředí CodeWarrior (CW) od společnosti Metrowerks. Prostředí integruje editor, kompilátor a linker zdrojových kódů. Zároveň poskytuje množinu knihovních funkcí pro vyvíjené platformy.

Součástí prostředí CW je programová nadstavba Processor Expert (PE). PE je komponentově orientovaný vývojový nástroj pro rychlý návrh a nastavení ovladačů cílové platformy. PE odstiňuje programátora od cílového hardwaru díky využití grafického rozhranní s generátorem kódu. Základními prostředky pro vývoj v PE je použití tzv. Embedded beans (EB). Každý z EB představuje jednu z použitých periferních částí cílové platformy. PE obsahuje pro každou platformu množinu těchto přednastavených beanů. Díky unifikovanému vzhledu vývojového nástroje je návrh ovladačů pro různé platformy jednotný. Programátor se tak nemusí seznamovat s každou novou platformou a studovat odlišnosti jednotlivých procesorů.

7.5.2 Prostředí MATLAB, Simulink a StateFlow

MATLAB je integrované vývojové prostředí pro numerické výpočty, návrh algoritmů, zpracování signálů a návrh řídicích a komunikačních systémů. Součástí MATLABu je grafická nadstavba Simulink. Ta poskytuje vývojové prostředí pro návrh, simulaci, modelování dynamických systémů. V Simulinku lze vytvářet jak lineární, tak nelineární spojité i diskrétní systémy. Simulink také dovoluje modelovat schémata spouštěná jen za určité podmínky či výsledku logické funkce. Výhodou je otevřená architektura, která dovoluje uživateli vytvářet si vlastní funkční bloky a rozšiřovat již tak bohatou knihovnu. Modely lze stavět do přehledné hierarchické struktury, díky čemuž lze modelovat i velmi složité systémy. Funkce modelů nemusí být definována pouze skripty MATLABu nebo diferenciálními rovnicemi modelu ale lze implementovat uživatelem definované programové moduly v jazyce C.

Součástí MATLABu je i program StateFlow, ten slouží k modelování událostmi řízených systémů (stavových automatů). Pro návrh cílového automatu je k dispozici hned několik možných reprezentací - stavový popis, vývojové diagramy atd. Navržené systémy ze StateFlow lze navíc propojit s modely vytvořenými v Simulinku. Tím je možné efektivně skloubit vývoj řídicích systémů pro spojité a událostmi řízené ovládání.

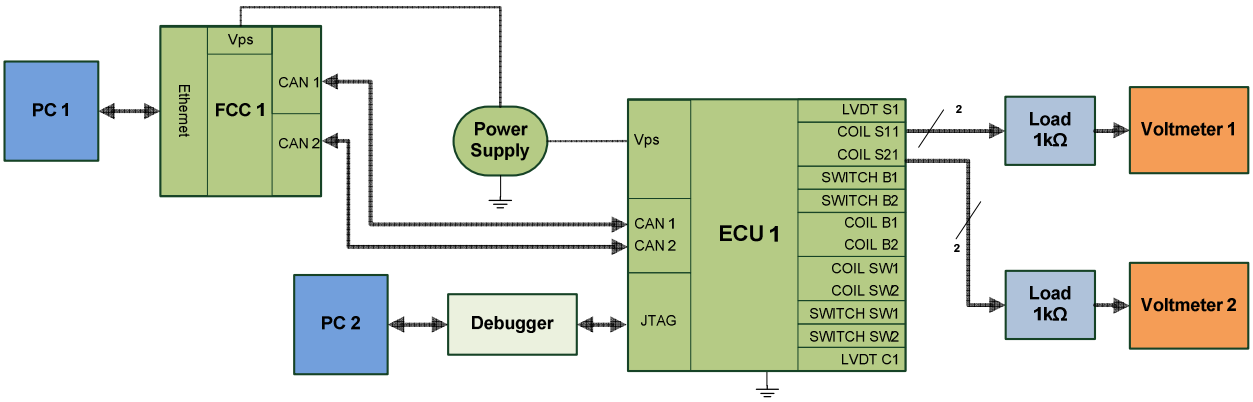
7.5.3 Nástroj Real-Time Workshop

Základem metodiky MBD je užití simulačních prostředků pro vytvoření modelu řídicího systému. Plynulý přechod od simulačního modelu k reálné aplikaci na cílové platformě je nedílnou součástí principu návrhu pomocí MBD. Pro tento účel musí mít vývojář k dispozici nástroj pro konverzi modelu do jazyka cílové platformy. Tímto nástrojem je právě Real-Time Workshop (RTW). RTW dokáže rychle a efektivně přeložit model ze Simulinku nebo Embedded MATLABu do zdrojového kódů v jazyce C. Díky tomu je generovaný kód z větší části nezávislý na cílové platformě. Jako cílová platforma v RTW byla volena rodina procesorů 56F8000. Překlad modelu byl optimalizován pro fixed-point target (*ert.tlc*).

8 Testování elektronické řídicí jednotky

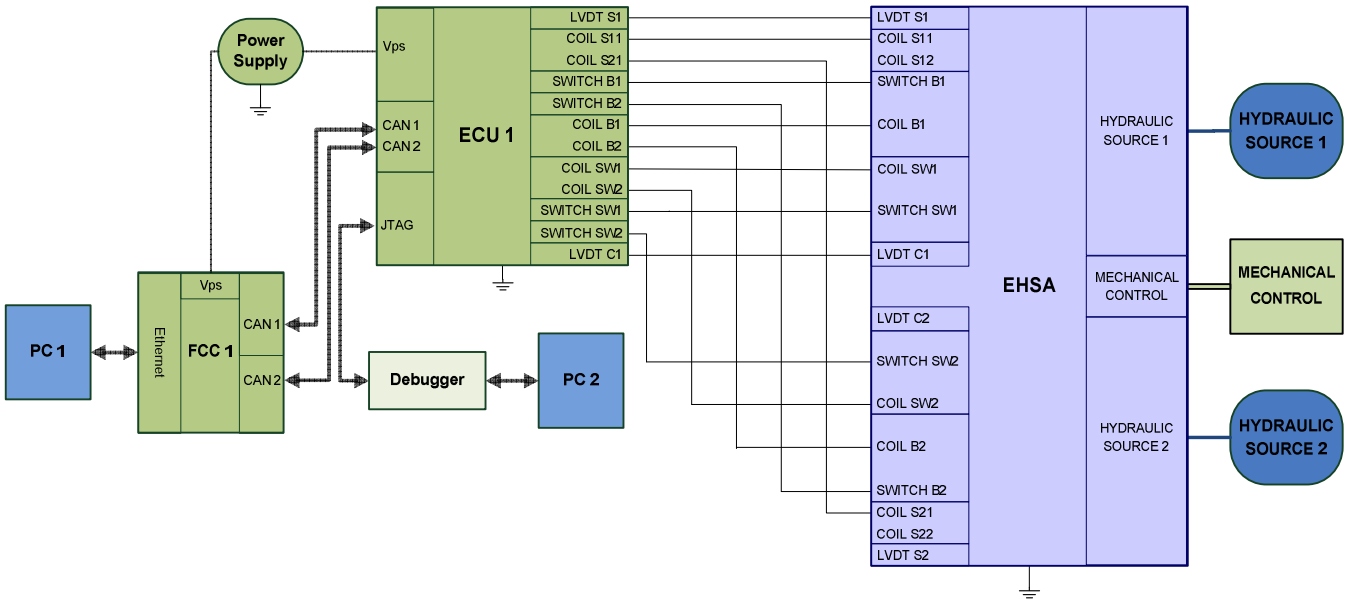
V této kapitole bude popsán proces testování ECU jednotky. Navržený model regulátoru ECU včetně stavového automatu byl nejprve testován na modelu řízené soustavy EHSA. Při testování byl použit identifikovaný model z práce [7]. V této simulaci byl naladěn PI regulátor polohy a otestována správná funkce stavového automatu. Po naladění PI regulátoru byl navržený model řídícího systému přenesen na cílovou platformu MC56F8367 (generování zdrojového kódu pomocí RTW).

Další testování ECU bylo prováděno metodou HIL. Nejdříve bylo provedeno testování ECU v otevřené smyčce, kdy byla změřena statická převodní charakteristika ECU (vstup-výstup). Princip tohoto měření je nakreslen na obr. 8.1.



obr. 8.1 – Měření statické převodní charakteristiky ECU

Při následném testování ECU byli změřeny data se zapojenou zpětnovazební smyčkou, kdy byla zavedena záporná zpětná vazba od výstupu (LVDT hydraulického cylindru) EHSA. V tomto případě bylo testováno chování ECU jak pro jeden (Coil S11), tak pro oba řídicí obvody EHSA (Coil S11 a S21). Princip měření a testování ECU zapojené do zpětnovazební smyčky je zobrazen na obr. 8.2 .



obr. 8.2 - Testování zpětnovazebního zapojení ECU a EHSA

8.1 Popis měřícího experimentu a testování

Měření probíhalo podle zapojení z obr. 8.1 a obr. 8.2. V prvním případě byl referenční signál (vstup do ECU) generován z programovacího PC připojeného přes rozhranní JTAG. Výstupy řídicích kanálů S11 a S21 byly přes *1 kΩ* zátěž měřeny voltmetrem 1 a 2.

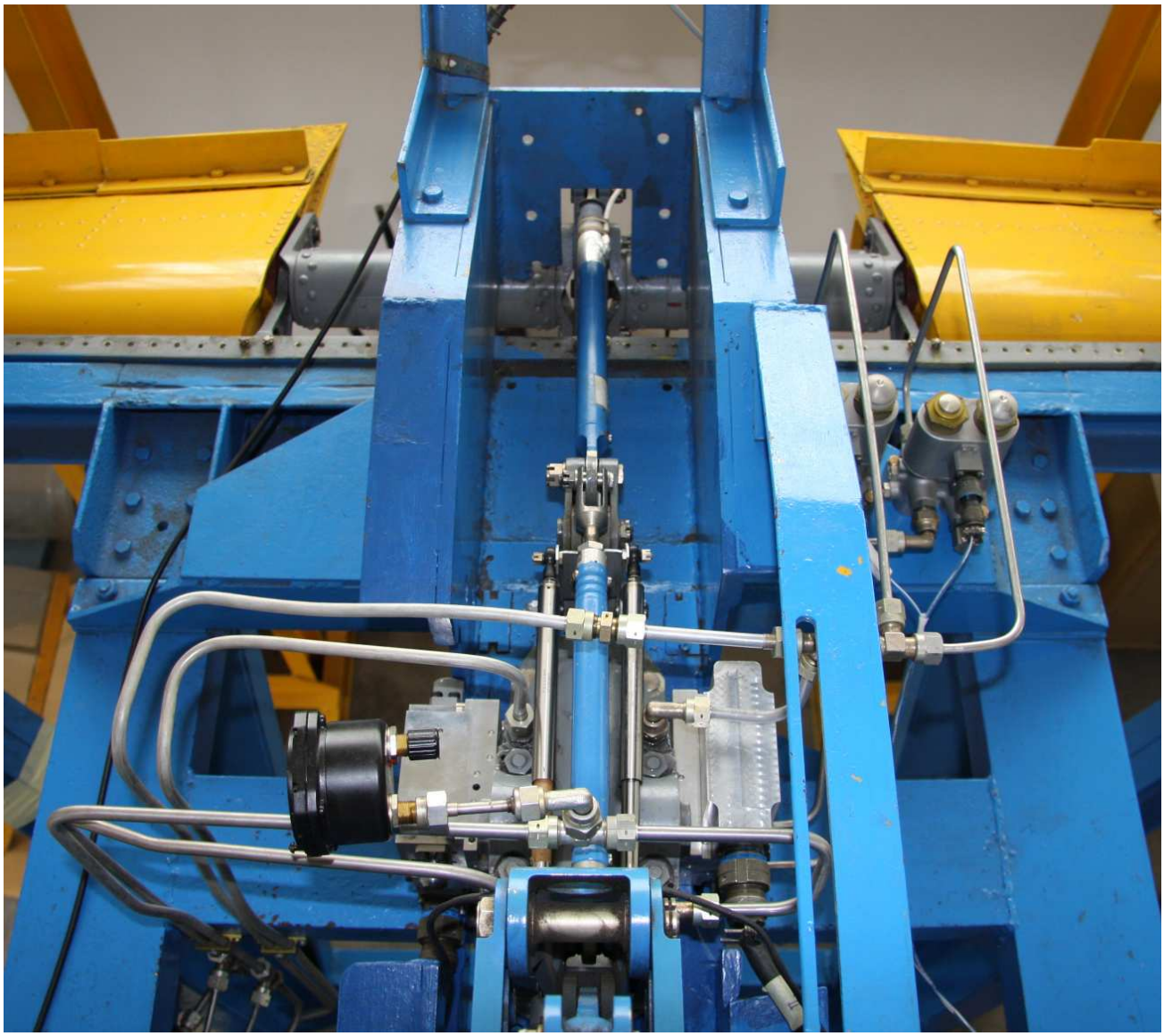
V druhém případě byl experiment zapojen do zpětnovazební smyčky s řízenou soustavou EHSA. Jako generátor referenčního signálu byl použit FCC, který byl zapojen přes Ethernet k programovacímu PC 1. Na tomto PC bylo spuštěno prostředí MATLAB a Simulink v external módu. Referenční signál byl definován v prostředí Simulink a přeposílán do FCC. Následně byl tento referenční signál uložen do objektových slovníků FCC. Komunikace pro CANopen byla nastavena tak, že FCC byl *master* a ECU *slave*. Pro komunikaci byl zvolen synchronní režim s periodou $\tau_V = 100ms$. Do vysílaného PDO na FCC byly namapovány konstanty PI regulátoru a referenční signál. Přijímací PDO bylo nastavené pro data ze senzoru natočení SENDIX a měřená data z LVDT z ECU. Samotné měření zprostředkovávala ECU. Perioda měření byla $\tau_M = 25\mu s$. Změřené data byly s periodou *100 ms* odesílány na CAN sběrnici do FCC. V FCC byly data přeposílána do programovacího PC 1 a následně vizualizovaná a ukládána do paměti. Díky poměrně dlouhé periodě vizualizace dat lze ve změřených vysokofrekvenčních datech (řídicí signál do EHSA) nalézt *aliasingové* efekty. Důvodem takto dlouhé periody byla relativně dlouhá časová odezva operačního systému v FCC.

Při měření nebyly k písnici EHSA připojeny žádné zatěžovací prvky ani jiná přídavná redukovaná hmota, nicméně zatěžovací válec byl mechanicky připojen k EHSA a spolu s *bypass* propojením druhého okruhu EHSA vykazoval malou tlumící sílu.

Zpětnovazební experiment s ECU a EHSA byl aplikován na dva typy zkušebních zařízení. První měření probíhalo na menším lokálním zatěžovacím standu (obr. 8.3). Ten byl zkonstruován pro samostatné testování a zatěžování EHSA pomocí hydraulického zatěžovacího mechanismu [41].

obr. 8.3– Hydraulické zkušební zařízení dvoukanálového servomechanismu EHSA

Po otestování ECU na lokálním zatěžovacím standu, byla řízená soustava EHSA instalována na větší letounové zkušební zařízení (obr. 8.4), který imituje skutečný drak proudového letounu. V tomto případě bylo EHSA spolu s ECU zapojeno na cílové místo v rámci FBW systému. Díky přesunu EHSA na letadlový stand a připojení kormidlových ploch (zátěž EHSA) došlo ke změně dynamiky řízené soustavy. Proto bylo nutné znovu naladit PI regulátor polohy.



obr. 8.4 – Letounové zkušební zařízení dvoukanálového servomechanismu EHSA

8.2 Měření v otevřené smyčce

Po osazení desky plošného spoje ECU byla PCB opticky a elektricky kontrolována na možné vady (studené spoje, mechanické vady, porušení vodičů). Postupně byli oživeny všechny subsystémy ECU. Z hlediska řízení ECU jednotky bylo důležité ověřit správnou funkčnost výstupních obvodů, hlavně linearitu výstupního řízení.

Tyto vlastnosti jsou popsány měřenou statickou převodní charakteristikou. Ta popisuje ustálenou výstupní odezvu systému na ustálený vstupní signál. V případě ECU byla, jako referenční budící signál, brána aktuální hodnota v 16-ti bitovém registru příslušného PWM modulátoru. Změřené a vypočtené data

výstupních řídicích obvodů jsou uvedena v tab. 8.1. Statická převodní charakteristika ECU jednotky pro řídicí obvod 1 je zobrazen na obr. 8.5, pro obvod 2 pak na obr. 8.6.

Referenční signál (hodnota v PWM registrech) byl generován z programovacího PC přes rozhranní JTAG. Jako simulace zátěže cívek servoventilů byla použita odporová zátěž *1 kΩ* zapojená do výstupního H-můstku. Díky měření ustálené hodnoty výstupního napětí (střední hodnota) nebylo potřeba uvažovat indukční parametry odporové zátěže (statické vlastnosti rezistoru a cívek servoventilu jsou shodné).

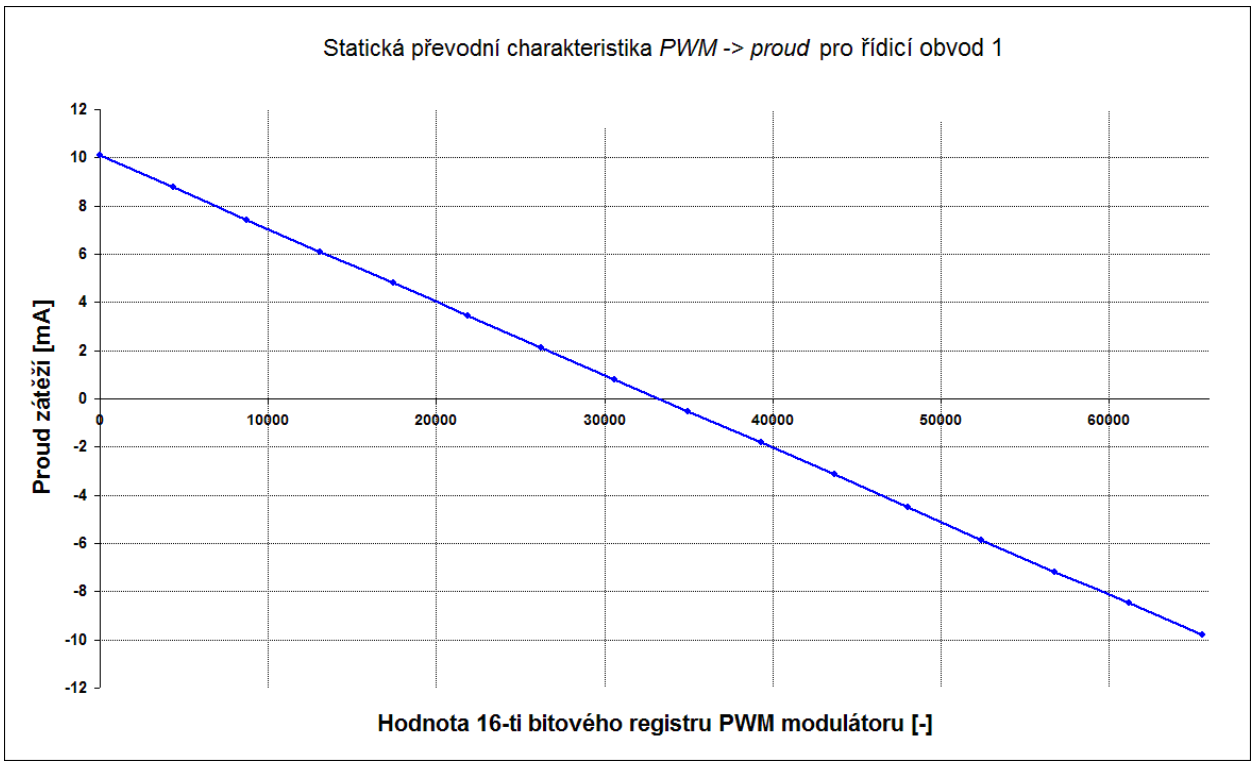
Výstupní napětí na zátěži bylo dáno střídou PWM signálu. Jako statická hodnota na výstupu byla brána střední hodnota PWM signálu na zátěži. Měření střední hodnoty bylo uskutečněno multimetrem CEM DT9602 .

Z naměřeného výstupního napětí byli vypočteny odpovídající hodnoty proudu v zátěži. Ze zobrazených statických převodních charakteristik vyplývá téměř lineární převod mezi hodnotou PWM registru a výstupním proudem v zátěži. Díky tomu lze konstatovat, že výstupní obvod nevnáší do řídicího signálu nelineární zkreslení. V simulacích tak bylo možné výstupní obvod modelovat jako lineární člen s daným zesílením. V oblastech okolo nuly jsou převodní charakteristiky bez výrazné necitlivosti.

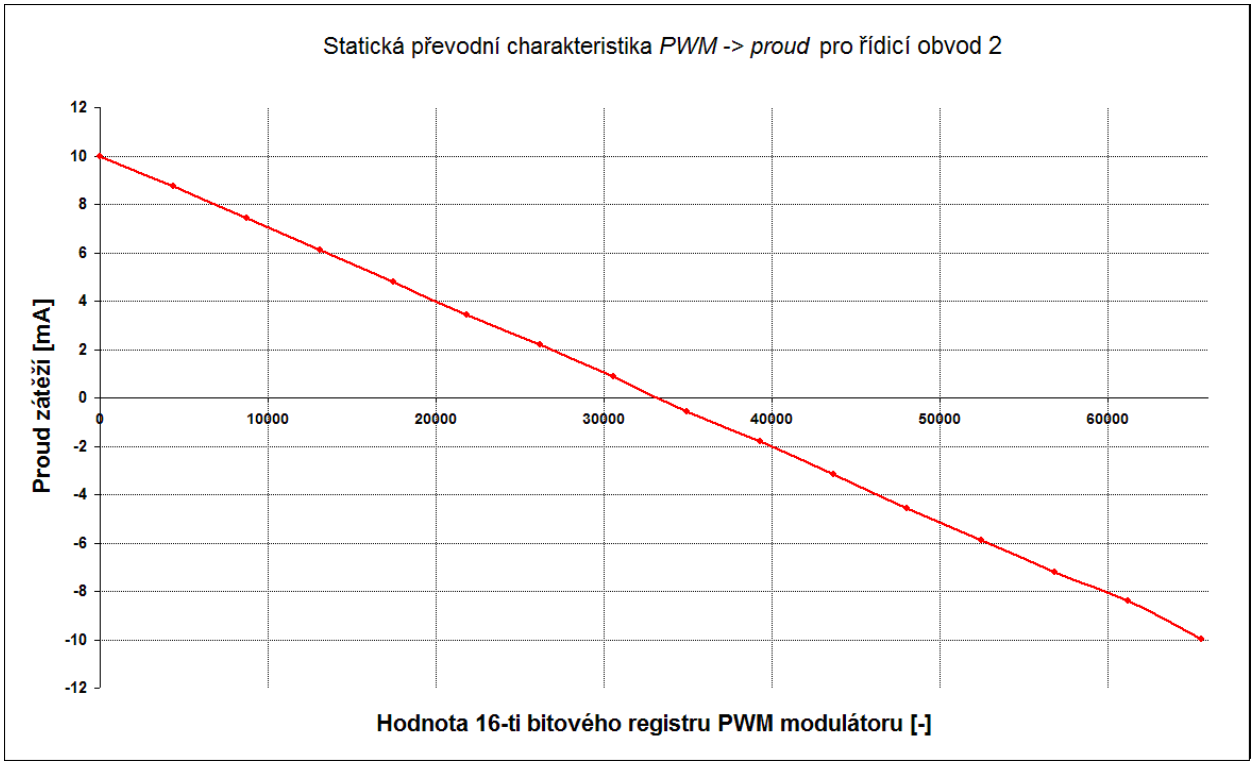
tab. 8.1 – Změřené hodnoty výstupních řídicích obvodů ECU

Číslo měření [-]	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
16 bit registr PWM [-]	0	4369	8738	13107	17476	21845	26214	30583
Napětí voltmeteru 1 [V]	10,1	8,78	7,43	6,11	4,79	3,45	2,12	0,8
Proud zátěží 1 [mA]	10,1	8,78	7,43	6,11	4,79	3,45	2,12	0,8
Napětí voltmeteru 2 [V]	10	8,75	7,42	6,1	4,81	3,42	2,2	0,88
Proud zátěží 2 [mA]	10	8,75	7,42	6,1	4,81	3,42	2,2	0,88

Číslo měření [-]	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
16 bit registr PWM [-]	34952	39321	43690	48059	52428	56797	61166	65535
Napětí voltmetru 1 [V]	-0,53	-1,8	-3,14	-4,48	-5,85	-7,17	-8,48	-9,8
Proud zátěží 1 [mA]	-0,53	-1,8	-3,14	-4,48	-5,85	-7,17	-8,48	-9,8
Napětí voltmetru 2 [V]	-0,58	-1,81	-3,17	-4,55	-5,91	-7,23	-8,39	-9,98
Proud zátěží 2 [mA]	-0,58	-1,81	-3,17	-4,55	-5,91	-7,23	-8,39	-9,98



obr. 8.5 – Statická převodní charakteristika řídicího obvodu 1



obr. 8.6– Statická převodní charakteristika řídicího obvodu 2

8.3 Měření v uzavřené zpětnovazební smyčce

Po otestování všech subsystémů ECU, oživení centrálního obvodu 56F8367 a změření výstupních charakteristik byla testována ECU ve zpětnovazebním zapojení podle obr. 8.2. Pro požadovanou regulaci polohy pístnice EHSA bylo nutné správně nastavit PI regulátor, nejdříve v simulačním prostředí na modelu EHSA, poté na skutečné soustavě.

8.3.1 Testování navrženého modelu řídicího systému

Po návrhu struktury PI regulátoru a stavového automatu ECU byl regulátor polohy naladěn na požadovanou dynamiku zpětnovazebního systému. Pro nalezení integrační a proporcionální časové konstanty byli použity hodnoty z předešlé práce [7]. Pro otestování takto naladěného regulátoru bylo sestaveno zpětnovazební schéma s modelem EHSA. Do simulačního schématu byli přidány všechny systémy, které řídicí řetězec při procesu regulace ovlivňují (Senzory, AD převod, výstupní silové obvody). Celé simulační schéma zpětnovazebního obvodu je ukázáno na obr. 8.7. Ve schématu jsou také umístěny ovládací prvky pro přepínání módů EHSA. Odezvy naladěného PI regulátoru byli shodné jako v práci [7], proto zde nejsou uvedeny. Veškeré simulace probíhaly ve 32-bitové floating-point aritmetice. Cílová platforma 56F8367 neobsahuje jednotku pro výpočty v plovoucí řádové čárce, proto bylo nutné testovaný obvod modifikovat pro výpočet v pevné řádové čárce.

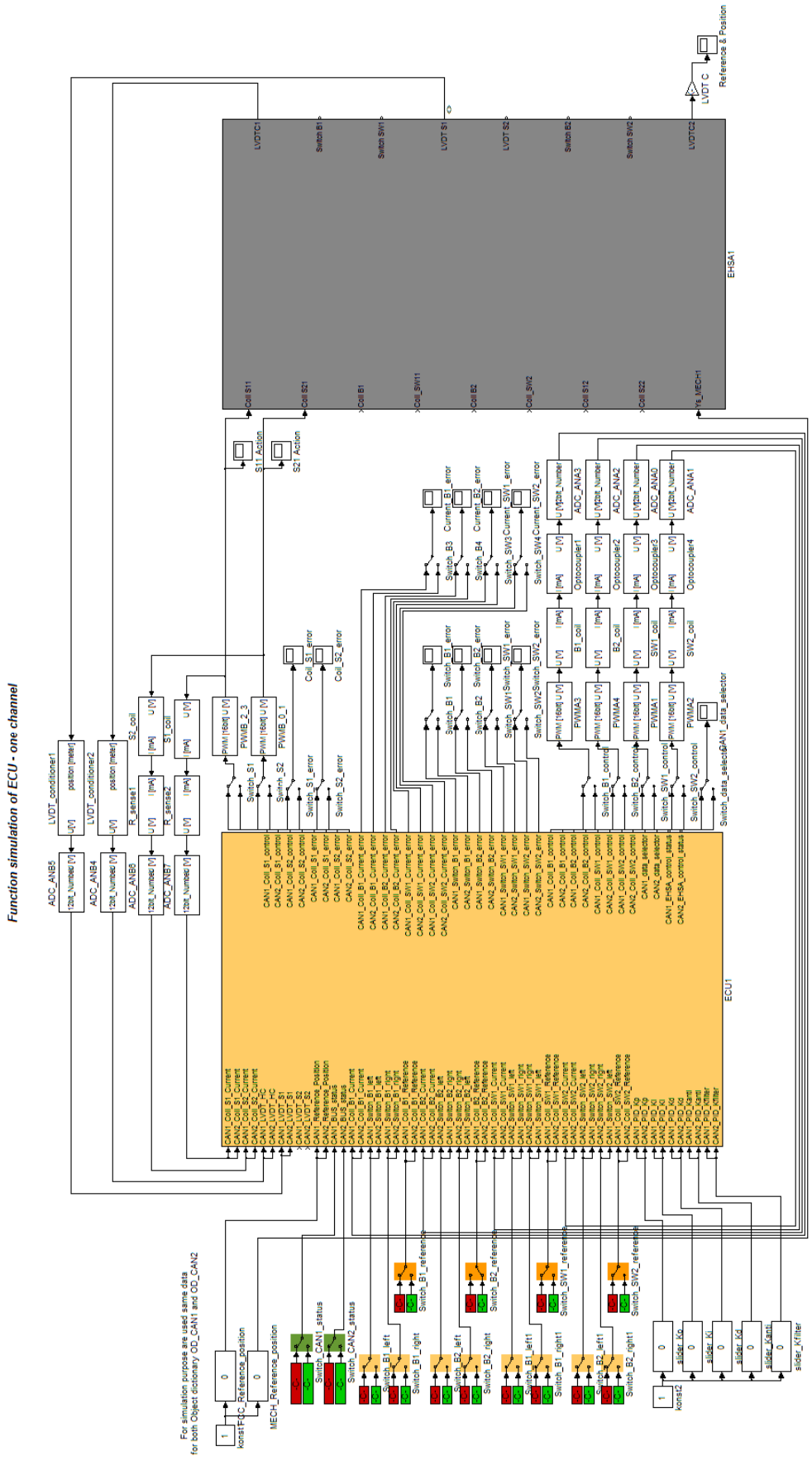
Přenesení do fixed-point aritmetiky obnáší omezení použitelných rozsahů na vstupu a výstupu systému. Při správném nastavení rozsahů není dynamika řízení ovlivněna. Nastavení fixed-point čísel bylo prováděno s ohledem na předpokládané rozsahy výsledků matematických operací ve struktuře regulačního obvodu (násobení, sčítání 16-ti bitových čísel). Nativně je rozsah čísel na platformě 56F8367 omezen na 16 bitů. S pomocí matematických knihoven lze používat i větší *32*-bitové rozsahy. Výpočetní náročnost se tímto však mírně zvýší.

Regulátor polohy byl nastaven tak, aby výstupní signál EHSA (LVDT poloha pístnice) nepřekmitl o více než *5 %* požadované polohy. Z této hodnoty vyplývá také hodnota doby náběhu, která je svázána s dosaženým překmitem. Čím větší je překmit nad požadovanou úroveň tím strmější je náběžná hrana odezvy. Naopak při nižším překmit je dynamika odezvy pomalejší.

8.3.2 Testování ECU na lokálním zatěžovacím stojanu

Po otestování a splnění všech požadavků na tvar a rychlost odezev na výstupu EHSA bylo možné přejít na testy se skutečnou soustavou EHSA. Nejdříve na menším lokálním zatěžovacím stroji (obr. 8.3), poté na letounovém zkušebním zařízení (obr. 8.4). Pro efektivní přechod do cílového procesoru byl generován zdrojový kód modelu regulátoru. Konstanty PI regulátoru byly mapovány do objektových slovníků a bylo je možné měnit z hlavního FCC. Vygenerovaný kód byl následně zaintegrován do projektu Codewarrioru a generované funkce regulátoru byli implementovány do vrstvy IHL do funkcí obsluhy přerušení PWM *reload*.

Při HIL ale i PIL simulacích dochází k nežádoucímu zpožďovacímu efektu. Ten vnáší do zpětnovazebního systému jednokrokové zpoždění. V případě dostatečně malého kroku lze tento efekt zanedbat. V případě ECU bylo signálové zpoždění referenčních dat do ECU $\tau_M = 25\mu s$. Vzhledem k relativně malé dynamice EHSA (desítky milisekund) ho tak lze zanedbat.



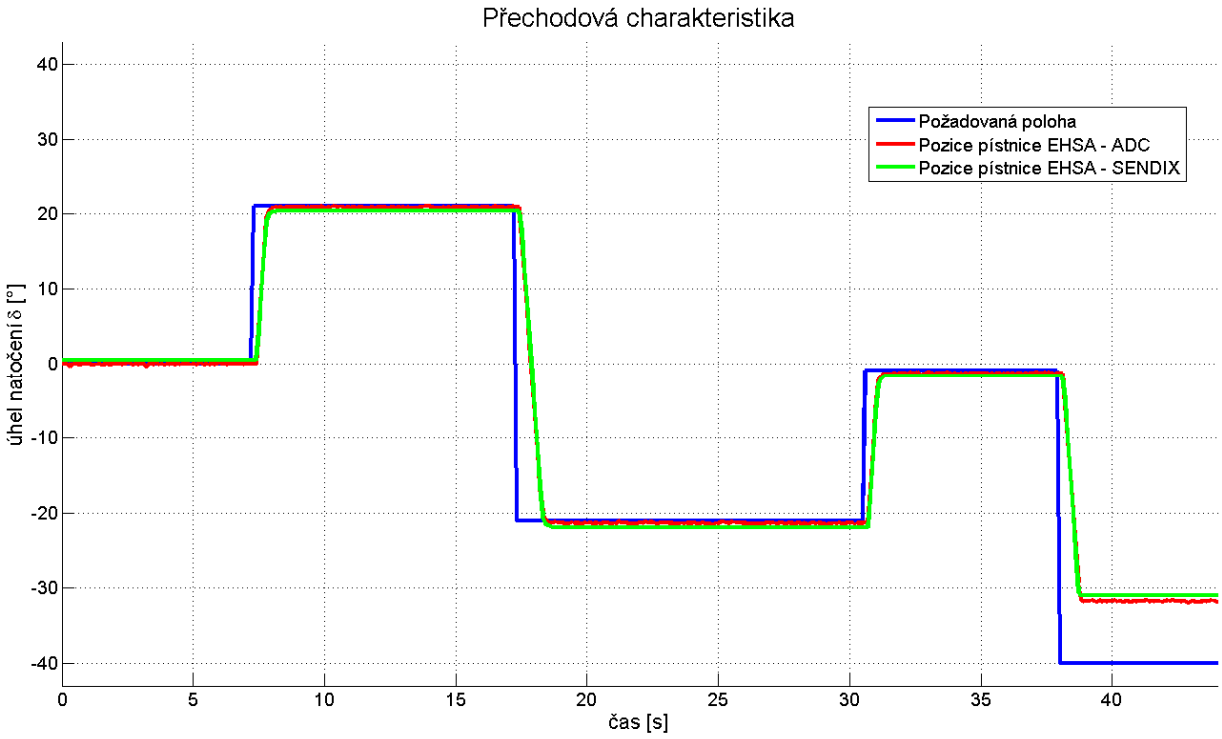
obr. 8.7 – Zpětnovazební zapojení ECU a EHSA v simulačním prostředí

Jednokanálové řízení EHSA s jedním řídicím obvodem

První testování ECU na zatěžovacím zařízení probíhalo podle obr. 8.2 s rozpojeným řídicím obvodem pro S21 (druhý řídicí obvod). Hydraulicky byl druhý okruh EHSA přepojen do *bypass* režimu. PI regulátor byl průběžně nastavován až splnil požadovaný překmit a tvar výstupního signálu. Pro jednokanálové řízení byly nalezeny tyto hodnoty PI regulátoru:

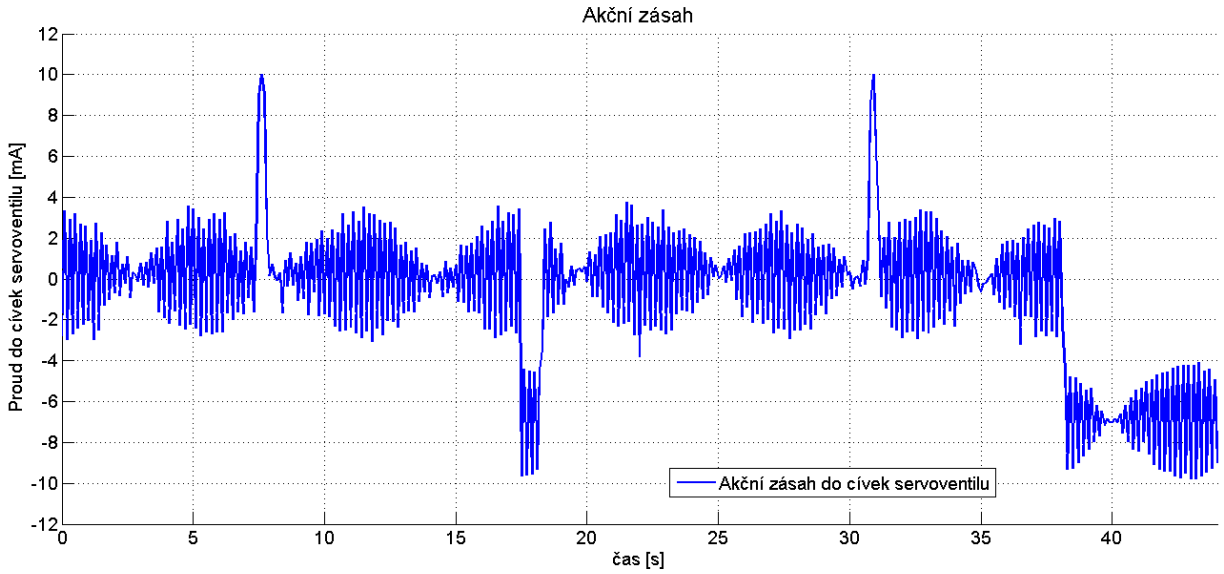
- $K_P = 64000$, při 16-ti bitovém operandu v rozlišení *5* bitů pro pevnou část, *11* bitů pro desetinnou část
- $K_I = 4000$, při 16-ti bitovém operandu v rozlišení *2* bity pro pevnou část, *14* bitů pro desetinnou část
- $K_{Antiwindup} = 8000$, při 16-ti bitovém operandu v rozlišení *2* bity pro pevnou část, *14* bitů pro desetinnou část

Na obr. 8.8 je zobrazena změřená přechodová charakteristika pro naladěný regulátor ECU. Referenčním signálem (modrý signál) byl obdélníkový signál generovaný z FCC. Reálný rozsah polohy pístnice (úhlu natočení kormidla) byl změřen ± *33*°. Ze změřené přechodové charakteristiky je vidět dynamika zpětnovazebního obvodu. Překmit výstupního signálu je menší než *5 %*, doba náběhu je $\tau_N = 950ms$ při přejezdu z +*22*° do -*23*°. Mechanické omezení úhlu natočení kormidla lze pozorovat od času *38 s*, kdy FCC požaduje regulaci na polohu -*40*°, EHSA je však schopna dosáhnout maximální výchylky jen -*33*°. Přechodová charakteristika byla změřena jak ze signálu LVDT hydraulického cylindru (červený signál), tak pomocí externího snímače polohy SENDIX [42] (zelený signál).



obr. 8.8 – Přechodová charakteristika ECU, jednokanálové řízení EHSA na zatěžovacím zařízení

Na obr. 8.9 je zobrazen akční zásah z ECU do řízené soustavy (proud do cívek servoventilu *S11*). Lze pozorovat, že regulátor generuje akční zásah v povolených mezích bez viditelných omezení v amplitudě. Navíc díky malému překmitu v přechodové charakteristice nejsou v akčním zásahu generovány vysoké špičky. Na akčním zásahu lze také pozorovat superponovaný *dither*.



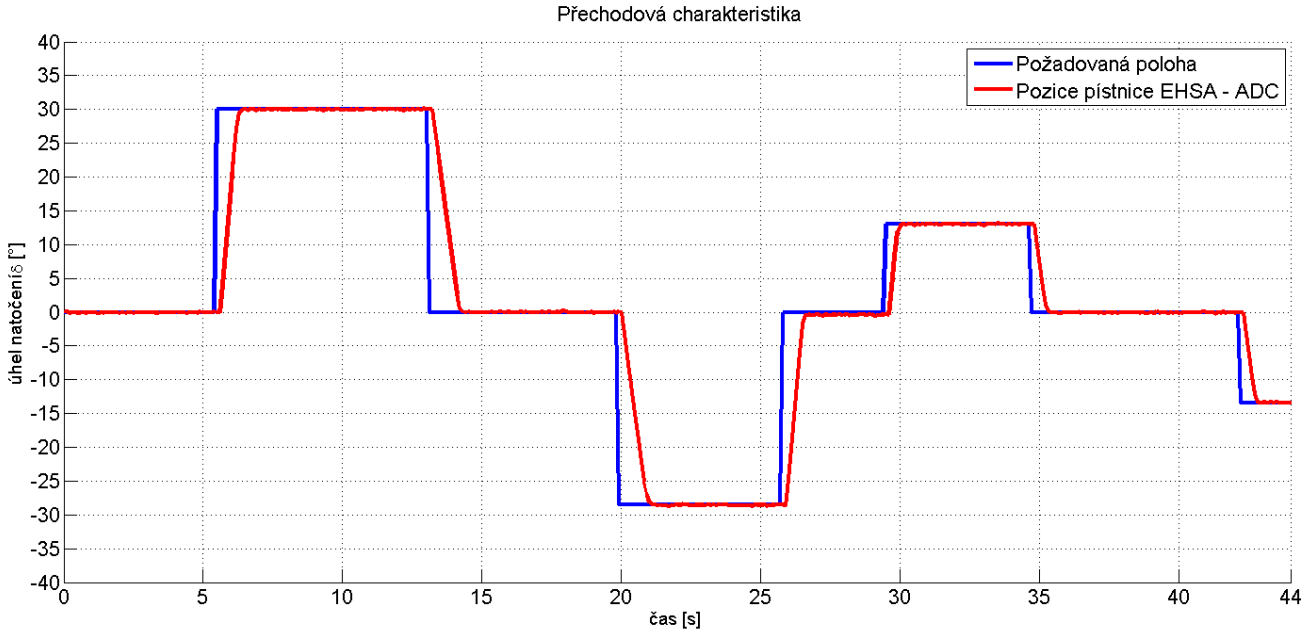
obr. 8.9 – Akční zásah ECU, jednokanálové řízení EHSA na zatěžovacím zařízení

Jednokanálové řízení EHSA s dvěma řídicími obvody

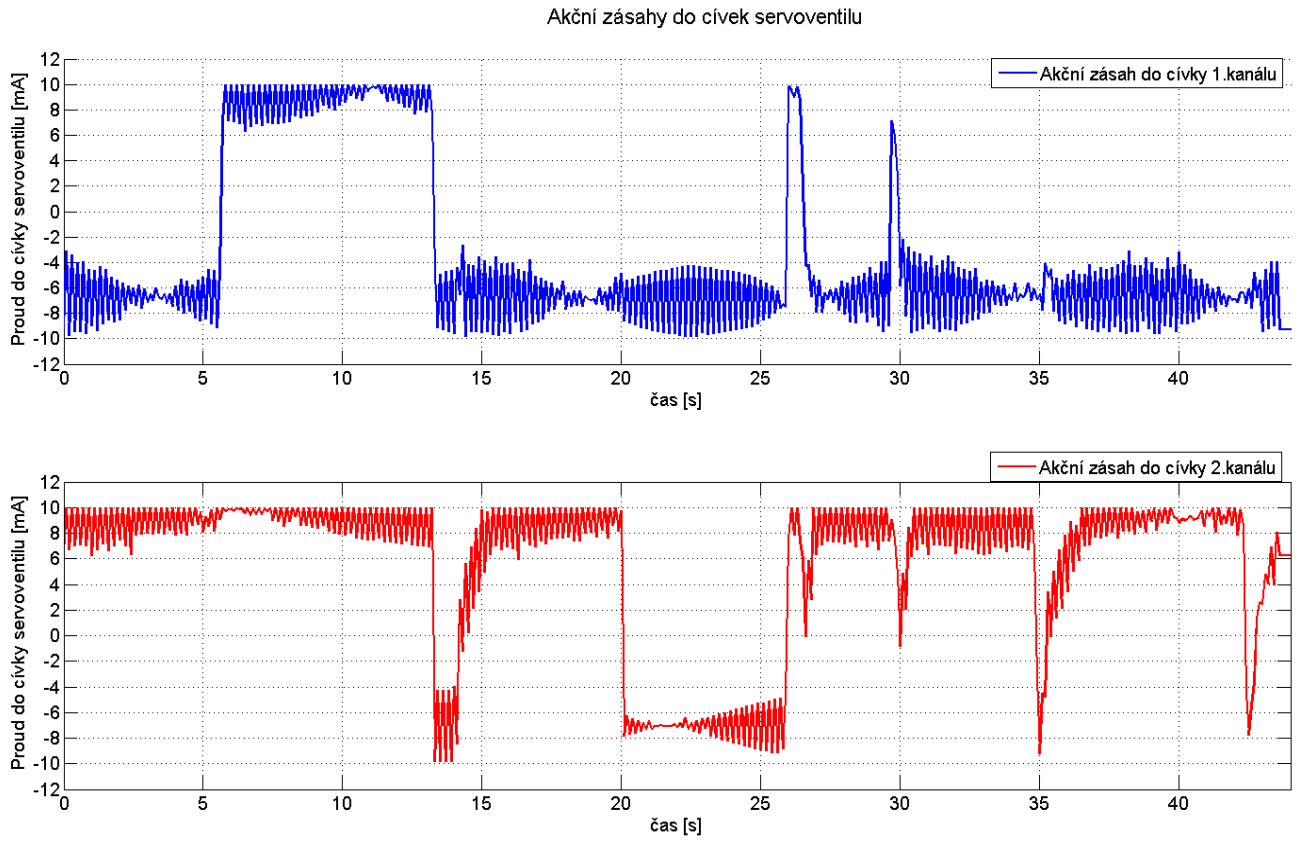
Po změření jednokanálového řízení s jedním řídicím obvodem byl zapojen také druhý řídicí obvod (řízení *S21*) podle obr. 8.2 . Každý řídicí obvod ECU řídil pouze jedinou cívku v odpovídajícím okruhu EHSA. PI regulátor byl ponechán v nastavení pro jednokanálové řízení.

Naměřená přechodová charakteristika je zobrazena na obr. 8.10. Referenční signál (modrá barva) je opět obdélníkový signál se skoky v rozmezí ± 30°, m ěřený výstup EHSA resp. poloha cylindru je zobrazena červenou barvou. Výstup je měřen AD převodníkem 56F8367 ze vstupních obvodů LVDT snímače hydraulického cylindru. Opět lze pozorovat, že pístnice EHSA sleduje požadovanou polohu. Výstup je bez překmitu s téměř stejnou dobou náběhu jako v případě jednokanálového řízení. Z tohoto měření lze pozorovat, že není potřeba použít systém *gain shedulingu a* přepínat regulátory pro jednokanálové jednoobvodové a jednokanálové dvouobvodové řízení.

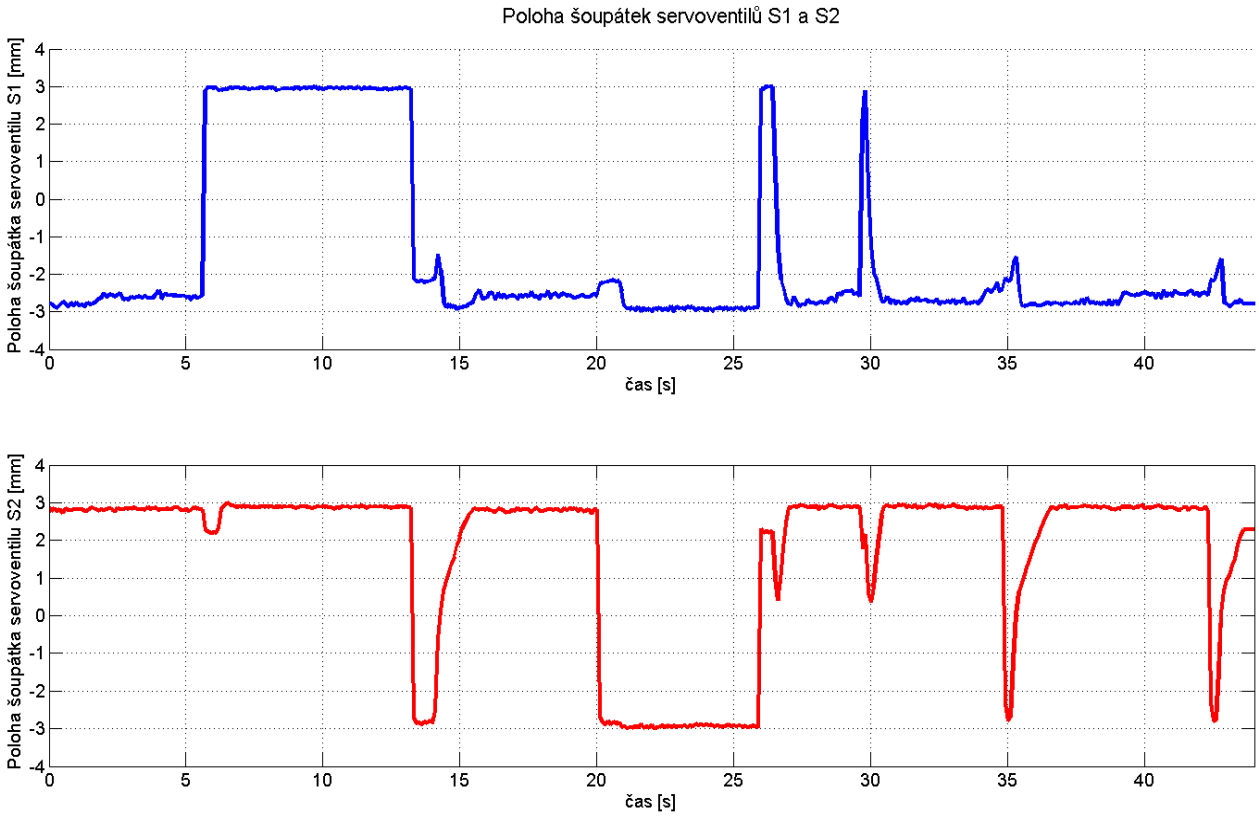
Odpovídající akční zásah obou řídicích obvodů je zobrazen na obr. 8.11. Tomuto akčnímu zásahu odpovídá také aktuální poloha šoupátek servoventilů S1 a S2. Jak je patrné z jejich výchylek. V ustáleném stavu (pístnice je v poloze požadované referencí), kdy jsou oba servoventily vychýleny do krajích poloh je vidět, že oba válce tlačí proti sobě a vzniká tak rovnováha sil. Díky tomu pístnice stojí v ustálené poloze. Dojde-li k poruše nebo změně referenčního signálu, servoventily opustí krajní polohy a dojde k požadovanému pohybu pístnice. Výsledkem této součinnosti je kvalitní sledování reference.



obr. 8.10 – Přechodová charakteristika, jednokanálové řízení EHSA s 2 řídicími obvody



obr. 8.11 – Akční zásah ECU, jednokanálové řízení EHSA s 2 řídicími obvody



obr. 8.12 – Poloha šoupátek servoventilů, jednokanálové řízení s 2 řídicími obvody

8.3.3 Testování ECU na letounovém zkušebním zařízení

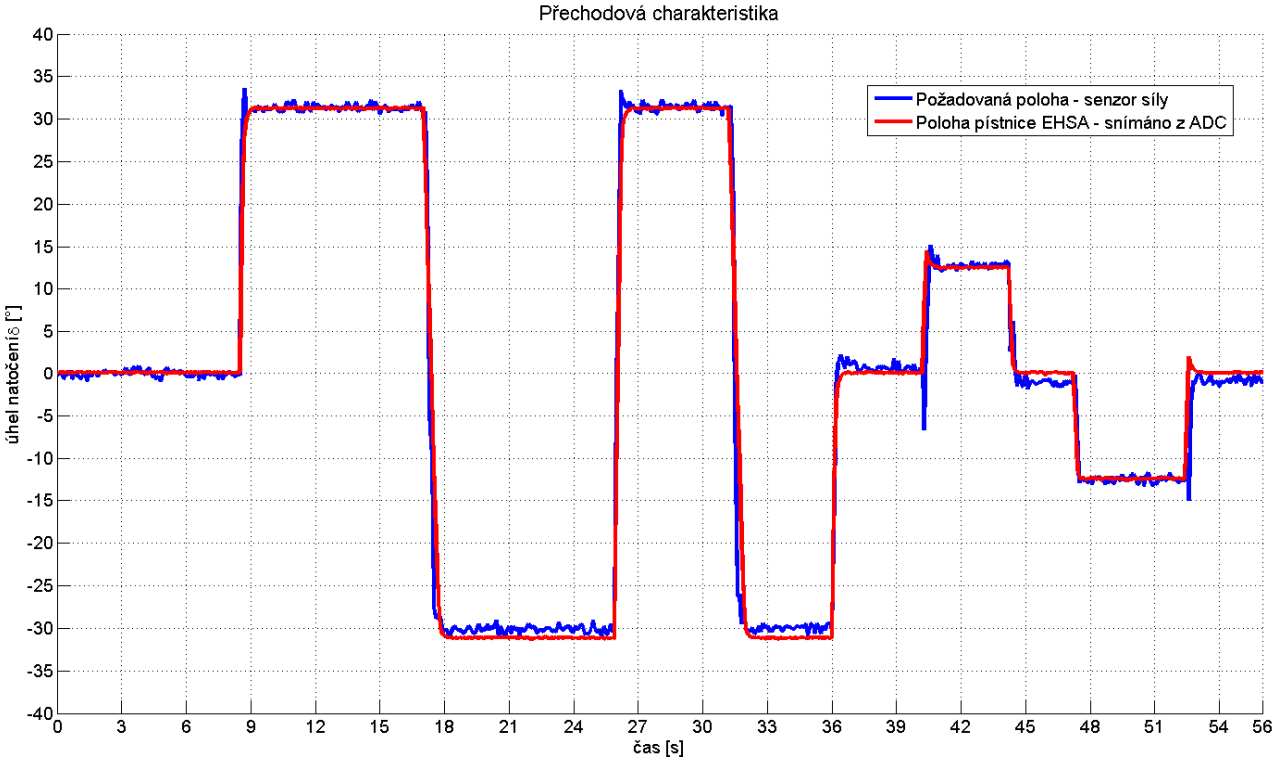
Třetí testovací experiment byl proveden na letounovém zkušebním stojanu (viz obr. 8.4). Tato konfigurace odpovídá reálnému umístění EHSA v proudovém letounu. Reference zde byla generována z řídicí páky z pilotní kabiny. Snímání aktuální polohy páky zajišťoval senzor síly. Principiálně je řízení zapojeno tak, že řídicí páka je spojena s řídicí plochou táhlem řízení. V táhle řízení vzniká síla, která je úměrná aktuální výchylce. Tato síla byla měřena senzorem síly. Jeho aktuální hodnota byla snímána přes CANopen. Podle této hodnoty byla vypočítávána odchylka od požadovaného výstupu. V průběhu reference jsou patrné derivační špičky způsobené setrvačnou hmotou páky.

Na letounovém zkušebním zařízení byl EHSA nainstalován pouze v jednokanálové verzi. Druhý kanál EHSA byl zapojen do *bypass* režimu. Pro řízení byla použita jedna ECU (jednokanálové řízení) s 1 řídicím obvodem (EHSA řízena jedinou cívkou). Mechanicky byl EHSA spojen se skutečnou řídicí plochou a táhli mechanického řízení. Díky této nové konfiguraci se změnila dynamika mechanické části soustavy. Vlivem toho bylo nutné PI regulátor polohy přeladit na nové hodnoty ($K_P = 12800$, $K_I = 640$, $K_{Antiwindup} = 8000$, při stejných FP parametrech).

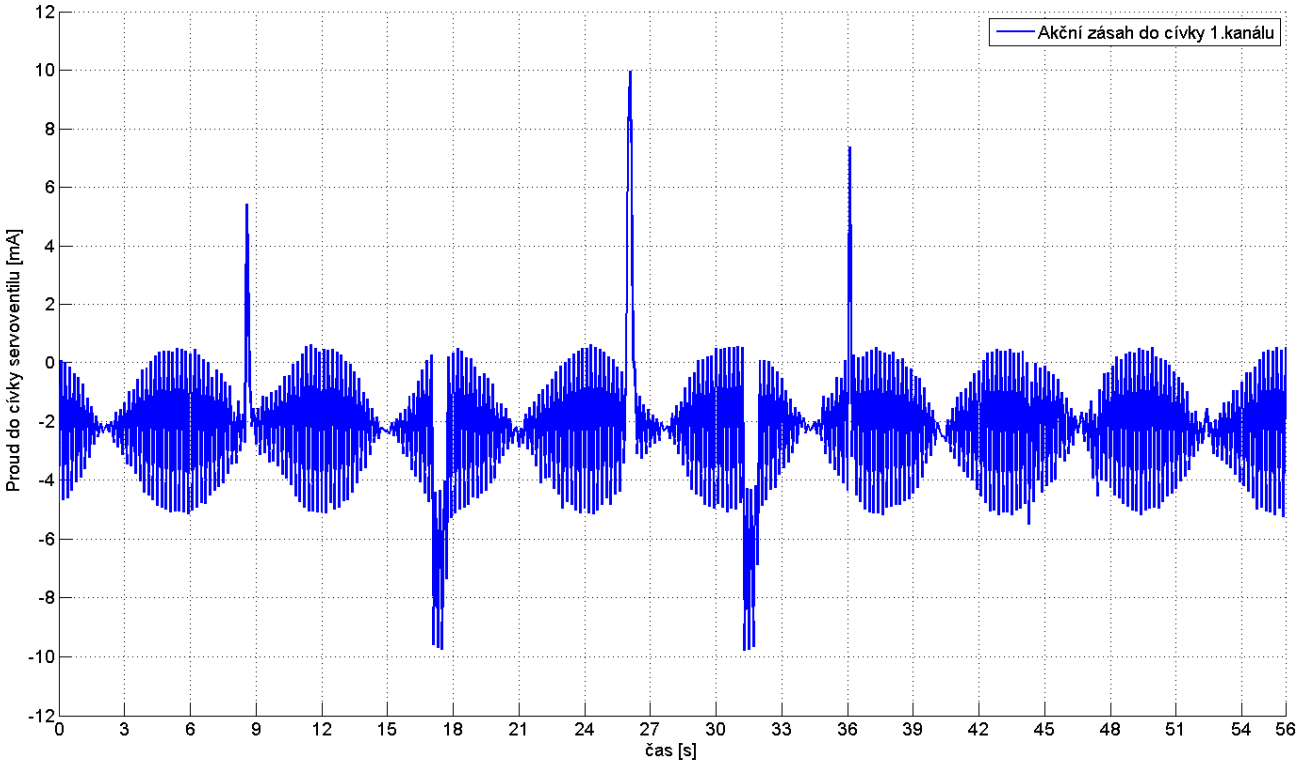
Změřená přechodová charakteristika je zobrazena na obr. 8.13. Oproti předchozímu případu, kdy byla reference generována z počítače FCC, lze pozorovat, že referenční signál je zašuměný s malými derivačními špičkami. Část vysokofrekvenčních složek na referenčním signálu je filtrována pasivní filtraci na vstupech AD převodníků. Z obr. 8.13 je vidět, že naladěný regulátor sleduje kvalitně referenci. Lze také pozorovat, že regulátor filtruje vysokofrekvenční složky z referenčního signálu, sledování reference je tak plynulejší. Vysokofrekvenční nežádoucí složky se nepřenášejí na výstup EHSA.

Na obr. 8.14 je zobrazen akční zásah ECU při řízení pouze s jedním řídicím obvodem. Lze pozorovat, že se oproti testování ECU na zatěžovacím zařízení posunula střední hodnota řízení. To je

způsobeno odlišnou konfigurací EHSA a připojené soustavy. Díky zapojení mechanické zátěže do horizontální polohy působí na písnici EHSA také gravitační síla samotného kormidla, proto musí ECU pro zajištění středové polohy kormidla působit na cívky servoventilu nenulovým akčním zásahem.



obr. 8.13 – Přechodová charakteristika, jednokanálové řízení EHSA na letounovém standu



obr. 8.14 – Akční zásah ECU, jednokanálové řízení EHSA na letounovém standu

9 Závěr

Hlavním cílem předložené práce byl návrh dvoukanálové elektronické řídicí jednotky pro elektrohydraulický akční člen EHSA s pomocí metodiky Model-Based design. Navržená řídicí jednotka byla integrována do demonstrátoru Fly-By-Wire systému řízení pro lehký proudový letoun společnosti AERO Vodochody a.s.

V úvodu práce byl popsán stávající systém řízení a systém řízení pomocí elektronického řídicího systému Fly-By-Wire. Byly shrnuty zásadní požadavky na řízení pomocí FBW systému a navržen vhodný koncept elektronické řídicí jednotky pro řízenou soustavu EHSA. Architektura řídicí jednotky byla navržena s ohledem na veškeré požadavky řídicího systému letounu a integraci do celkového systému řízení FBW. Výsledkem konceptu byla plně redundantní dvoukanálová křížově propojená architektura. Křížové propojení řídicích jednotek ECU zajišťuje, že potencionální porucha dvou různých částí ECU, byť v odlišných kanálech, nezpůsobí výpadek celého FBW systému.

Jedním z požadavků na návrh softwaru bylo využití metodiky Model-Based Design. V úvodních kapitolách byl proto detailně probrán princip a popis metodiky MBD. Bylo popsáno srovnání klasické návrhové metody bez použití modelování a nové perspektivní metodiky využívající modelování. Pro praktickou ukázku vývoje řídicího systému pomocí metodiky MBD byl uveden návrhový postup v prostředí MATLAB a Simulink, včetně nástroje pro automatické generování kódu. Pro úspěšný přechod mezi modelem řídicího systému a výsledným zdrojovým kódem cílové platformy 56F8367 byl probrán problém výpočtů v pevné a plovoucí řádové čárce. Byly shrnuty výhody a nevýhody obou výpočetních technik a možná rizika při přechodu mezi oběma formáty.

Pro komunikaci s dalšími komponentami FBW systému bylo nutné implementovat do vyvíjené řídicí jednotky komunikační protokol CANopen. Z požadavků na řídicí systém FBW vyplynula dvoukanálová verze CAN sběrnice. Z hlediska finální implementace komunikačního kanálu se bylo nutné seznámit s fyzickou a spojovou vrstvou CAN. Ty jsou základem pro vyšší vrstvy komunikačního systému a protokol CANopen. Proto byla probrána specifikace protokolu CAN a CANopen. Širší popis těchto dvou protokolů byl nezbytný pro správné rozhodnutí o způsobu implementace CANopen. Hlavní část tohoto popisu je věnována objektovému slovníku protokolu CANopen. Výsledná implementace protokolu CANopen vychází z projektu CanFestival, který je podrobně popsán v závěru kapitoly věnované komunikaci CAN.

Jedním s cílů diplomové práce bylo seznámení se s funkcí a parametry dvoukanálového hydraulického členu EHSA, který je řízenou soustavou pro navrhovanou řídicí jednotku. Dvoukanálový akční člen EHSA byl proto popsán v samostatné kapitole, kde byly vysvětleny jeho funkce a principy ovládání. Od jeho struktury byla odvozena také architektura řídicích jednotek. Návrh jednotlivých podsystémů ECU vycházel z konkrétních parametrů EHSA. Při návrhu vstupně-výstupních obvodů řídicí jednotky byl vyřešen problém sdíleného snímání a ovládání EHSA.

Po analýze systému FBW, komunikace pomocí sběrnice CAN, a vysvětlení principů řízení EHSA bylo navrženo blokové schéma ECU. Hardware byl rozdělen na vstupní měřící část, výstupní řídicí část, komunikační obvody pro CAN, napájecí obvody a centrální procesorovou část. Všechny obvodové části jednotky jsou z důvodu bezpečnosti a ochrany galvanicky odděleny. Díky tomu je procesorová část chráněna před negativními vlivy z vnějšího okolí. Pro řízení soustavy EHSA bylo navrženo redundantní dvoukanálové propojení řídicích jednotek. Řídicí jednotky jsou tak vzájemně identické a z hlediska propojení na dvouokruhovou soustavu EHSA jsou jednotlivé ECU připojeny paralelně, vždy na jeden z okruhů EHSA. Toto křížové propojení ECU dává možnost v případě výskytu poruchy jedné z řídicích jednotek, řídit oba okruhy EHSA pouze z jedné ECU.

Rozložení jednotlivých obvodů spolu s požadavky na řízení EHSA umožnilo definovat architekturu řídicího softwaru. Byl sestaven vrstvový model, na základě kterého bylo celkové chování ECU rozděleno na samostatné vzájemně komunikující programové moduly.

Z konceptuálního modelu hardwaru byly vybrány konkrétní elektronické prvky a obvody. Následně byl vytvořen schematický návrh všech subsystémů. Při návrhu schématu bylo bráno v potaz zahřívání součástek, odolnost součástek a doporučené zapojení od výrobce komponenty. Ohled byl brán také na

filtraci rušivých signálů a oddělení napájecích úrovní. Hlavní procesorovou platformou byl vybrán signálový procesor 56F8367. Návrh schématu byl vytvořen v integrovaném vývojovém prostředí KiCAD.

Po vytvoření elektronického schématu následoval návrh desky plošného spoje. Rozmístění komponent a propojení bylo optimalizováno vzhledem k navržené architektuře a vhodnému rozložení konektorů pro připojení EHSA. Kreslení vodivých drah a rozmísťování komponent na PCB bylo prováděno s ohledem na mezinárodní standard IPC600. Deska plošného spoje byla navrhnuta ve dvou vodivých vrstvách, součástková základna byla z větší části typu SMD. Po výrobě desky byla PCB osazena a umístěna do ochranného krytu. V ochranném boxu byly vytvořeny montážní otvory pro upevnění na EHSA a pro propojovací konektory. Před připojením ECU k programovacímu PC byla PCB elektricky a opticky kontrolována, jednotlivé subsystémy ECU byli oživovány postupně.

Řídicí software byl implementován podle navrhnutého vrstvového modelu. K vývoji softwaru bylo využito prostředí Codewarrior s aplikační nadstavbou Processor Expert. Software byl rozdělen do pěti vzájemně propojených programových komponent. Každá z komponent implementuje určitou oblast chování (komunikace po CAN, aplikační vrstva CANopen, výpočet regulátoru, apod.). Pro prioritní rozlišení zpracovávaných úkolů bylo využito přerušovacího systému 56F8367. Návrh protokolu CANopen vychází z projektu CanFestival. Jedním v výstupů této práce je otestovaný ovladač pro CAN řadič platformy 56F8000 využitelný pro projekt CanFestival. Tento ovladač využívá všech 16-ti bufferů platformy 56F8367 pro paralelní komunikaci na sběrnici CAN, díky tomu lze komunikovat rychlostí až 1Mbit/s s aplikační odezvou pod $500\mu s$. Pro využití programových částí CanFestivalu bylo nutné portovat zdrojové kódy na cílovou platformu. Zdrojový kód CF byl proto upraven, přeložen a v prostředí Codewarrior slinkován do jediné knihovny *56F8xxxCANfestoval.lib*. Tato knihovna byla následně zaintegrována do stávajícího kódu ECU. Funkčnost této knihovny byla otestována na demonstračním programu. Veškeré zdrojové kódy programových komponent byly psány v jazyce C.

Regulátor polohy pro soustavu EHSA byl navržen v prostředí Simulink, kde byl testován na modelu soustavy EHSA. Po dosažení požadovaných parametrů regulace byl tento model pomocí nástroje Real-Time Workshop generován do jazyka C. Zdrojové kódy algoritmu regulátoru byly integrovány do zbývajícího softwarového projektu ECU. Perioda regulačního zásahu byla zvolena $f_{control} = 2kHz$. Výpočet regulačního zásahu byl spouštěn synchronně s PWM modulátorem pro řízení proudu v cívkách servoventilu. Pro zajištění včasného dokončení výpočtu regulátoru byl zvolen systém časových značek, který hlídá konce a začátky výpočetních kroků řídicího algoritmu.

Ladění PI regulátoru probíhalo nejdříve na modelu soustavy EHSA, kde byli nalezeny integrační a proporcionální konstanty pro spojité řízení. Následně byl model regulátoru zdiskretizován a generován na cílovou platformu. Po integraci řídicího algoritmu do softwarového projektu ECU byl regulátor naladěn na reálném akčním členu. Testování ECU proběhlo na dvou odlišných testovacích zařízení. První experiment se zapojenou zpětnovazební smyčkou probíhal na menším lokálním zatěžovacím zařízení, kde byla ECU testována v jednokanálové verzi s jedním nebo dvěma řídicími obvody. úspěšně byl naladěn PI regulátor polohy a byly změřeny přechodové charakteristiky včetně akčních zásahů. Následně byl EHSA instalován na větší letounový demonstrátor, který simuloval finální umístění ECU a EHSA v rámci FBW systému. Na tomto demonstrátoru byla ověřena správná funkčnost ECU v jednokanálové verzi s jedním nebo dvěma řídicími obvody. Opět zde byl PI regulátor naladěn na požadované dynamické vlastnosti zpětnovazebního obvodu. Referenční signál byl generován z řídicí páky z kabiny letounu. Výsledkem testování ECU na letounovém demonstrátoru byla přechodová charakteristika a změřený akční zásah do cívek servoventilů EHSA.

V budoucnu lze elektronickou řídicí jednotku rozšířit o perspektivnější regulační algoritmus (např. stavové řízení, LQ regulaci, apod.), lze také rozšířit stávající software o diagnostické funkce pro prvky EHSA. Jako další zdokonalení FBW systému by bylo možné stávající komunikační systém s protokolem CANopen modernizovat na jiný komunikační standard (Flexray, ARINC, apod.). Hardwarovou část ECU lze také s použitím rozměrově menších komponent miniaturizovat. Závěrem lze konstatovat, že vyvinutá řídicí jednotka ECU splňuje všechny vstupní požadavky na řízení EHSA a požadavky na integraci do celkového systému FBW řízení.

10 Reference

[1] Jirásek V., Klepetko M., Trnobranský M., Waszniowski L., *Koncepční návrh systému FBW* , Zpráva 22/RKM/2007, Aero VODOCHODY a.s., 2007.

[2] LEE Company, Lee Direct Acting Solenoid Valve, *Electromagnet SDBB2131113B*, 2001.

[3] Jihostroj Velšín a.s. Jednotka ovládání kola LUN 6874-8, *LUN servoblok.*

[4] MOOG, Servovalve, *MoogServoventil.*

[5] Penny & Giles, LVDT DISPLACEMENT TRANSDUCERS, *LVDT P+G 199.*

[6] Waszniowski L., *Specifikace HW elektronické řídicí jednotky hydraulického servomechanismu řídicí plochy letounu*, Zpráva 22/8/2008, Katedra řídicí techniky, ČVUT, FEL, 2008.

[7] Pich J., *Návrh elektronické řídicí jednotky založený na modelování*, bakalářská práce, Katedra řídicí techniky, 2009.

[8] CiA, Control Area network,*CAN 2.0 specification*, http://www.can-cia.org/, 2010.

[9] KiCAD, *KiCAD documentation,* http://kicad.sourceforge.net/, 2010.

[10] Traco Power, *Datový list TEN-12-4822*, http://www.farnell.com/datasheets/26085.pdf, 2010.

[11] On Semiconductor, *Datový list MC33269*, http://www.onsemi.com/pub_link/Collateral/MC33269-D.PDF, 2010.

[12] ST, *Datový list L78M05CDT*, http://www.st.com/stonline/products/literature/ds/2146.pdf, 2010.

[13] Freescale, *Datový list MC56F8367*, http://www.freescale.com/files/dsp/doc/data_sheet/MC56F8367.pdf, 2010.

[14] Analog Devices, *Datový list AD698*, http://www.analog.com/static/importedfiles/Data_Sheets/AD698.pdf, 2010.

[15] Vishay, *Datový list ILD206*, http://www.farnell.com/datasheets/100933.pdf, 2010.

[16] Analog Devices, *Datový list ADUM1400*, http://www.analog.com/static/importedfiles/data_sheets/ADuM1400_1401_1402.pdf, 2010.

[17] ST, *Datový list VNQ5027AK-E*, http://www.st.com/stonline/products/literature/ds/12730/vnq5027ak-e.pdf, 2010.

[18] Agilent Technologies, *Datový list HCPL0531*, http://www.fairchildsemi.com/ds/HC/HCPL0531.pdf, 2010.

[19] Analog Devices, *Datový list ADUM1201*, http://www.analog.com/static/importedfiles/data_sheets/ADuM1200_1201.pdf, 2010.

[20] Philips, *Datový list PCA82C250*, http://www.nxp.com/documents/data_sheet/PCA82C250.pdf, 2010.

[21] ST, *Datový list L6225*, http://www.st.com/stonline/products/literature/ds/9451.pdf, 2009.

[22] Záhlava V. *, Návrh a konstrukce desek plošných spojů, Skripta, Vydavatelství ČVUT*, Praha 2005.

[23] Vobecký J., Záhlava V., *Elektronika, součástky a obvody, principy a příklady*, Grada, Praha 2005.

[24] Záhlava V. , *OrCAD 10*, Grada, Praha 2004.

[25] CanFestival, *Canfestival documentation*, http://www.canfestival.org/, 2009.

[26] Mathworks, *dokumentace MATLAB a Simulink*, http://www.mathworks.com/, 2010.

[27] Metrowerks, *Codewarrior documentation*, www.freescale.com/codewarrior, 2009.

[28] Zdrojové kódy software ECU, svn://rtime.felk.cvut.cz/wasz/ECU, 2010.

[29] Zdrojové kódy softwaru pro CANopen komunikaci ECU, svn://rtime.felk.cvut.cz/wasz/canfestival, 2010.

[30] CiA, *CAN specificaion*, http://www.semiconductors.bosch.de/pdf/can2spec.pdf, 2009.

[31] CiA, *CANopen, CAL-based Communication Profile for Industrial Systems*, Version 4.0, June 1999.

[32] CiA, *CANopen Device Profile for I/O Modules*, Version 1.4, Dec 1996.

[33] CiA, *CANopen Device Profile for Measuring Devices and Closed-Loop Controllers*, Version 1.11, Nov 1997.

[34] CiA, *CANopen Application layer and communication profile,* Feb. 2002.

[35] The MathWorks*, Real-Time Workshop – User's Guide*, 2009.

[36] The MathWorks, *Real-Time Workshop – Target Language Compiler*, 2009.

[37] Repositář projektu CanFestivalu, http://lolitech.fr/dev/, 2009.

[38] Ripka. P., *Senzory a převodníky*, , Vydavatelství ČVUT, 2005.

[39] Kocourek P., Novák J., *Přenos informace*, Vydavatelství ČVUT, 2004.

[40] Vysoký O., *Elektronické systémy 2*, Vydavatelství ČVUT, 2002.

[41] Kovář J., *Regulátor hydraulického zatěžovacího zařízení*, bakalářská práce, 2008.

[42] Fritz Kubler GmbH, *documentation Sendix series 5858*, absolute encoders, 2009.

[43] Hanzálek Z., Hospodář P., Hromčík M., Waszniowski L, Doubrava J., *Design and Validation of the Flight Recovery Control System*, 2010.

[44] Hyniová K., Stříbrský A., Skočdopole J., *Technické prostředky řízení*, Vydavatelství ČVUT, 1994.

[45] Wikipedia, *Linear variable differential transformer*, http://en.wikipedia.org/wiki/Linear_variable_differential_transformer, 2009.

[46] Murrata, *The elegance of ferrite beads*, http://www.murata.com, 2009.

11 Obsah přiloženého CD

Na doprovodném CD jsou umístěny tyto adresáře:

- Hardware_ECU – obsahuje datové listy k použitým součástkám, projekt s elektronickým schématem a deskou plošného spoje, knihovnu schematických značek a modulů, soupisku součástek,

- Software_ECU – obsahuje kompletní zdrojový kód programu ECU, model regulátoru, dokumentaci softwaru v programu Doxygen,

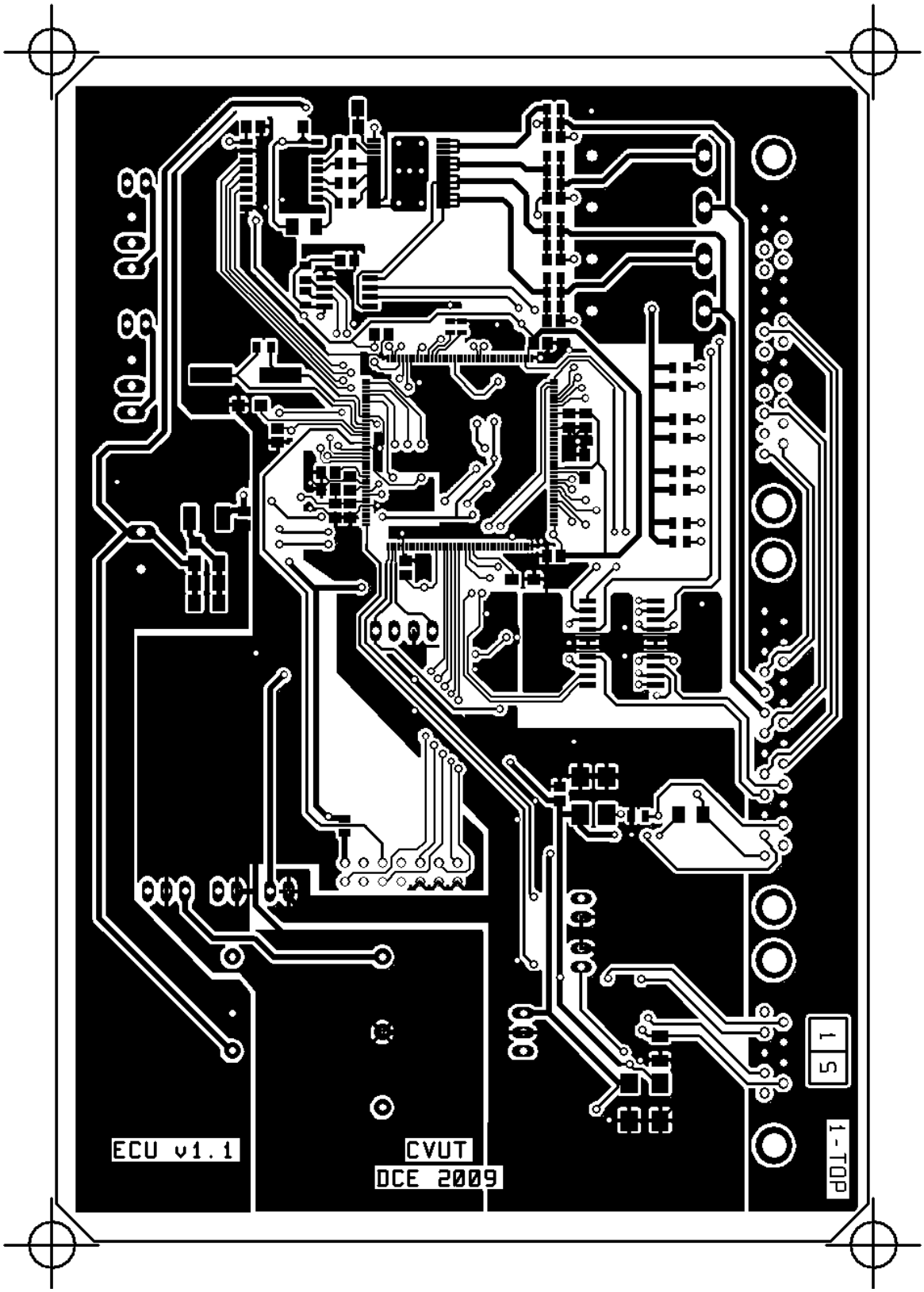
- CanFestival – obsahuje kompletní projekt CanFestival, zdrojové kódy k ovladačů a protokolu CANopen, knihovnu *56F8xxxCANfestival.lib* pro implementaci komunikace CANopen na platformě 56F8000,

- Fotodokumentace – obsahuje fotografie elektronické řídicí jednotky, soustavy EHSA, zatěžovacího zařízení a zkušebního letounového zařízení,

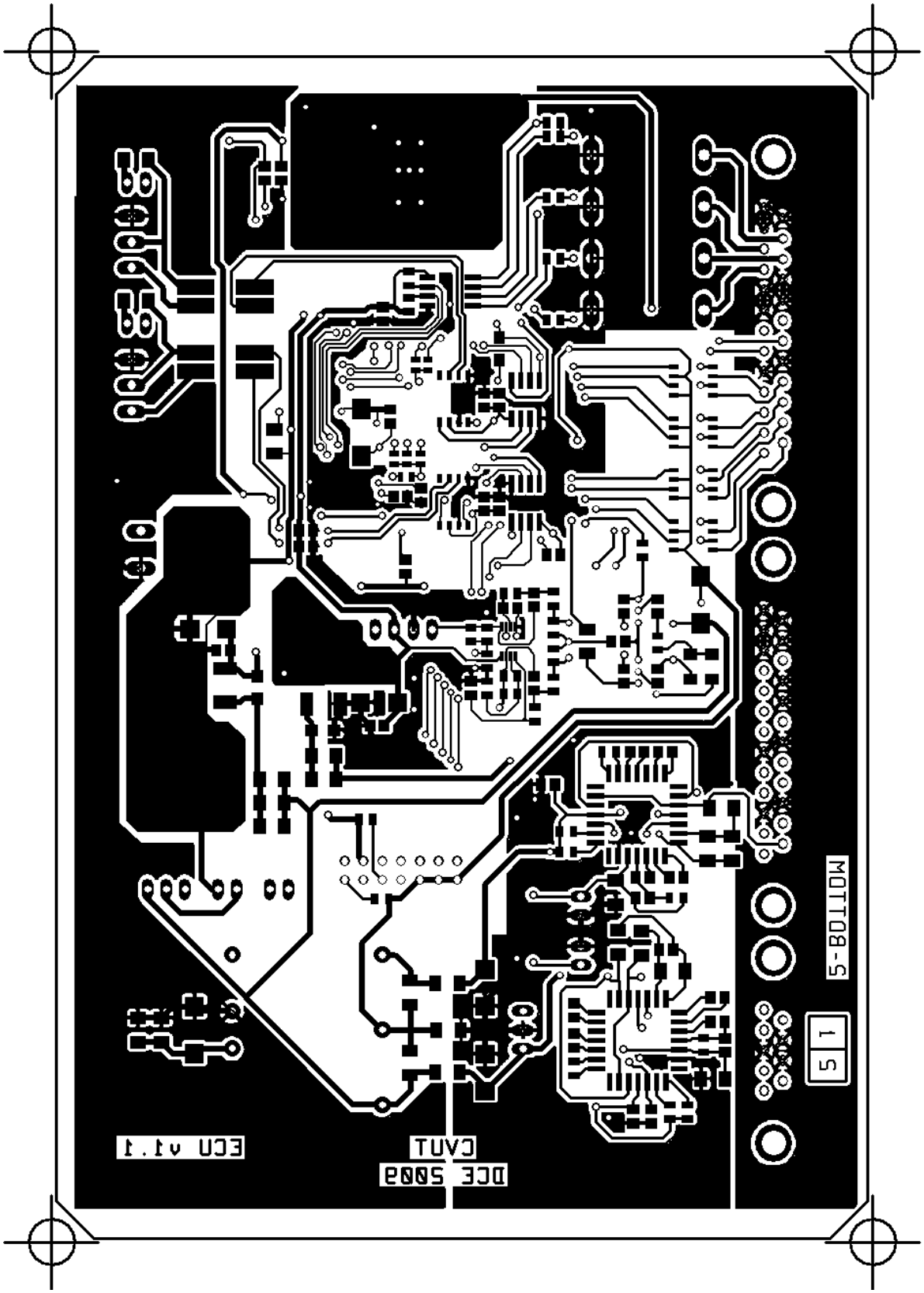
- Diplomova_prace – obsahuje zdrojové soubory této diplomové práce ve formátu DOC a PDF.

12 Příloha

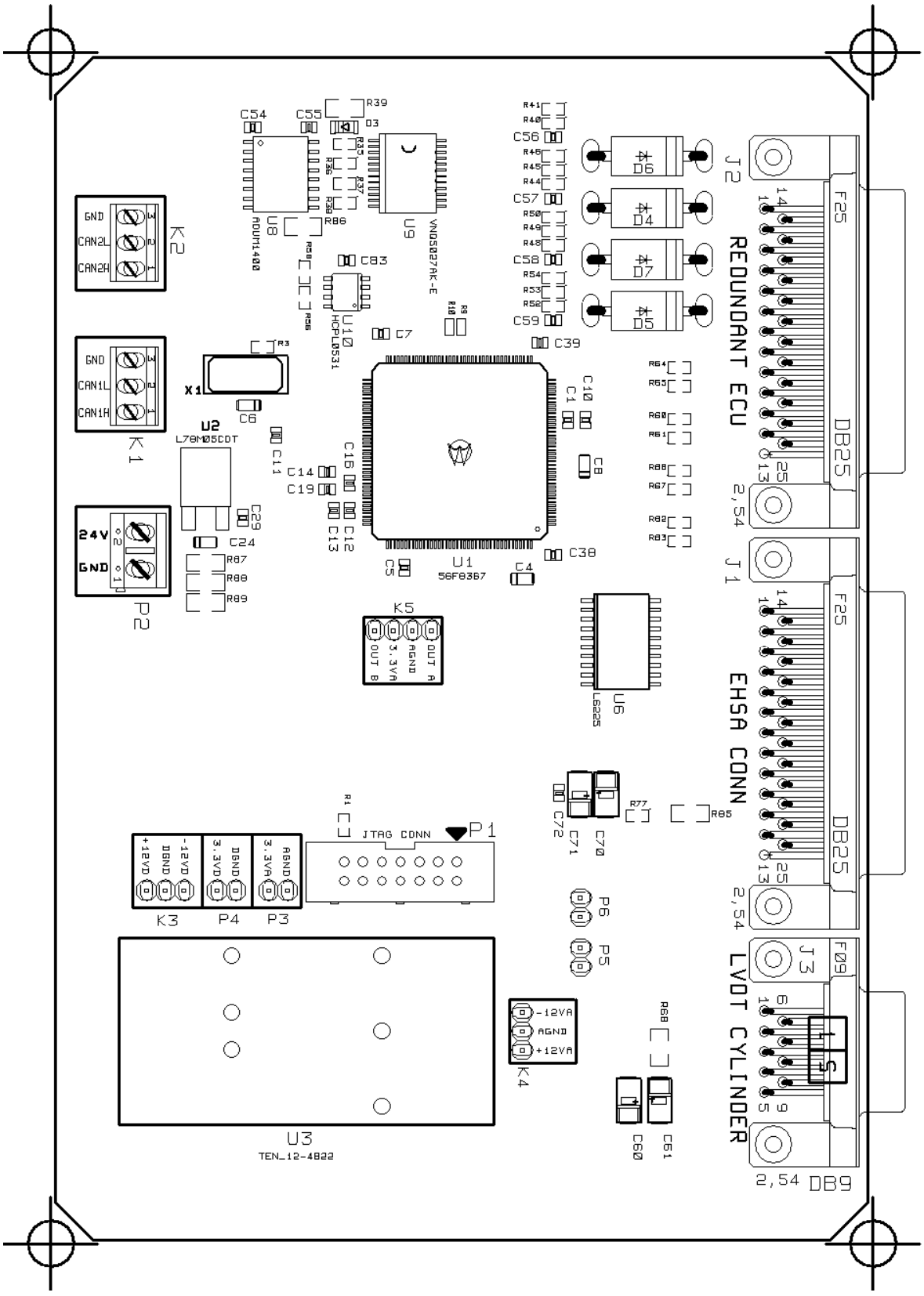
12.1 Deska plošného spoje



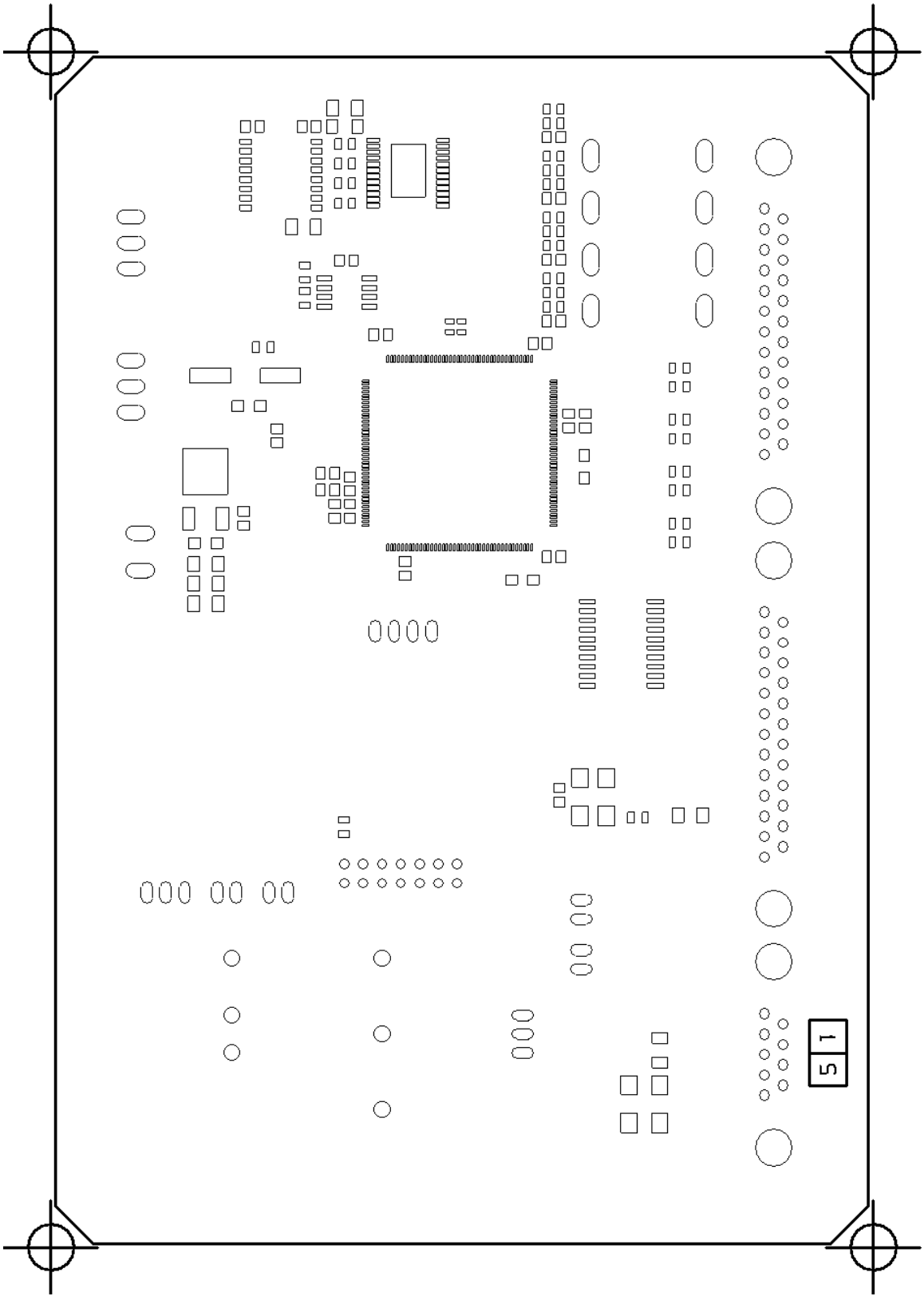
obr. 12.1 – Deska plošného spoje, vrstva TOP



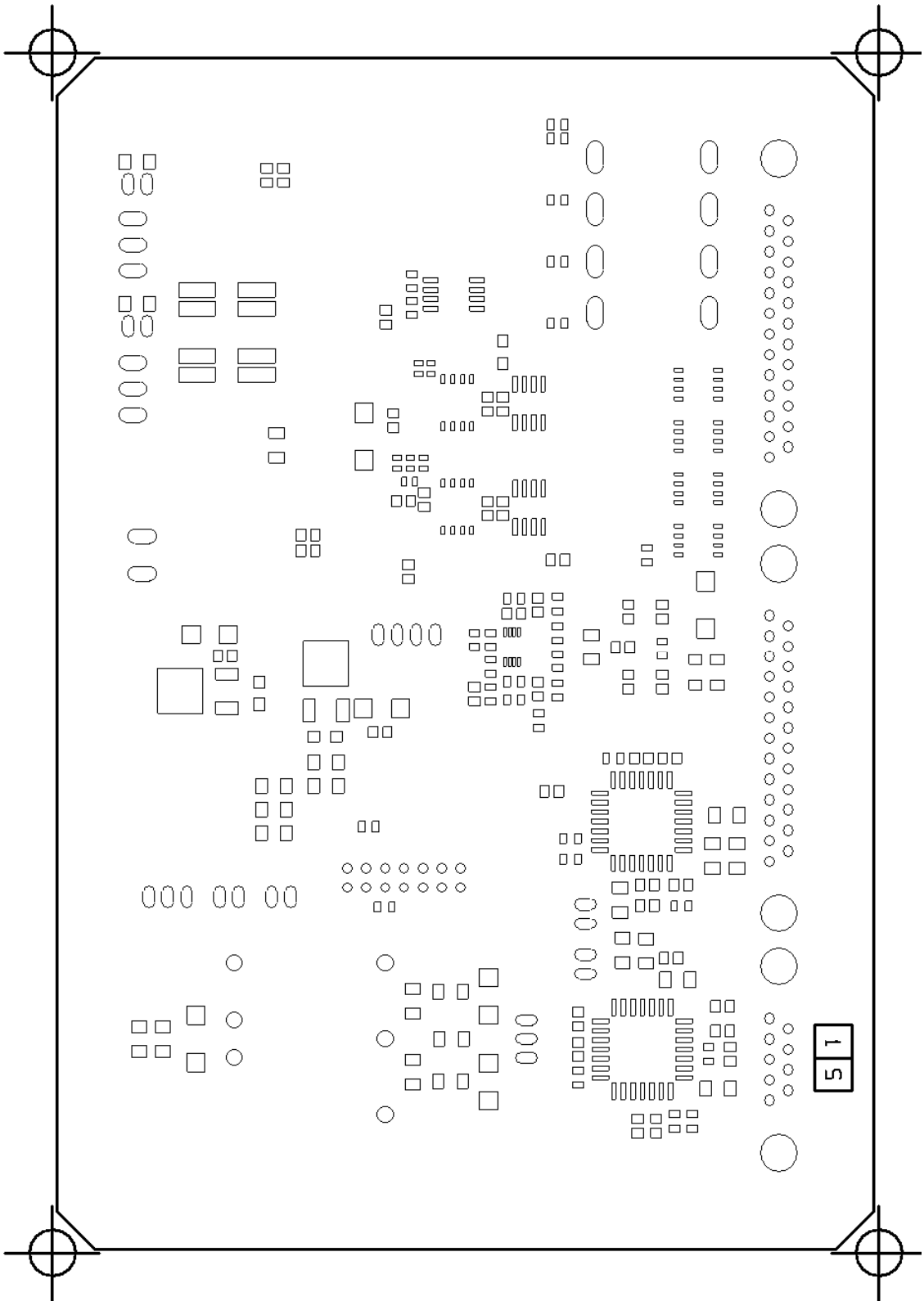
obr. 12.2 – Deska plošného spoje, vrstva BOTTOM



obr. 12.3 – Deska plošného spoje, potisk TOP

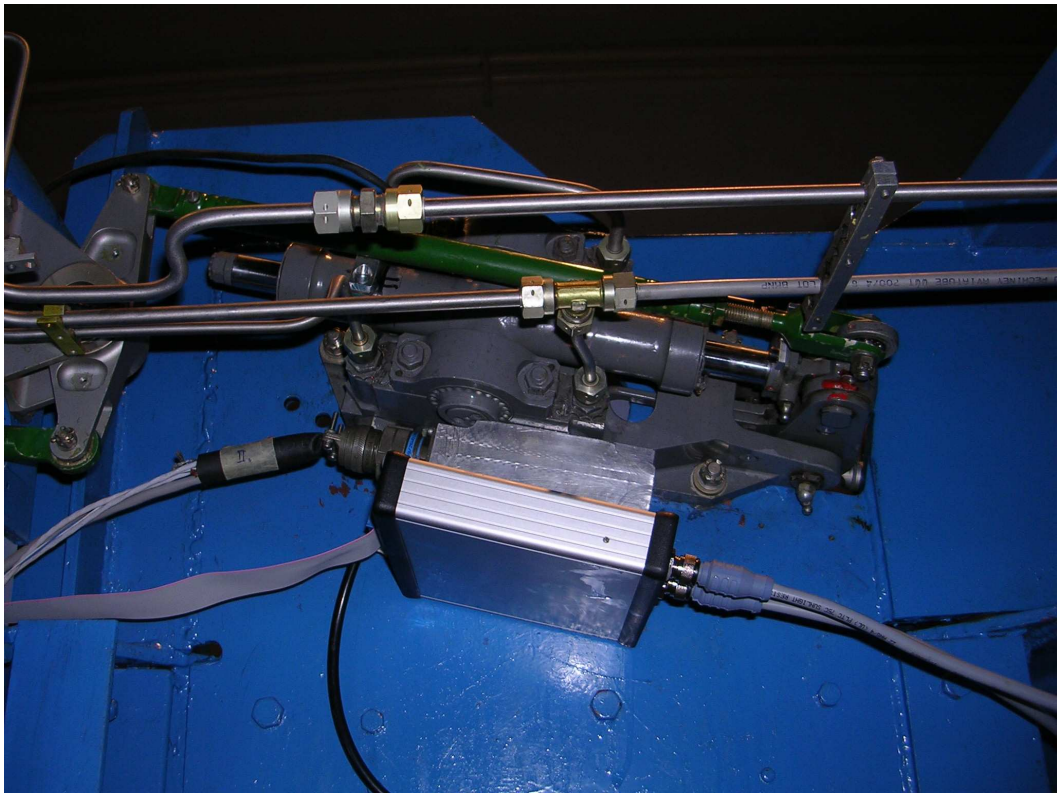


obr. 12.4 – Deska plošného spoje, maska TOP

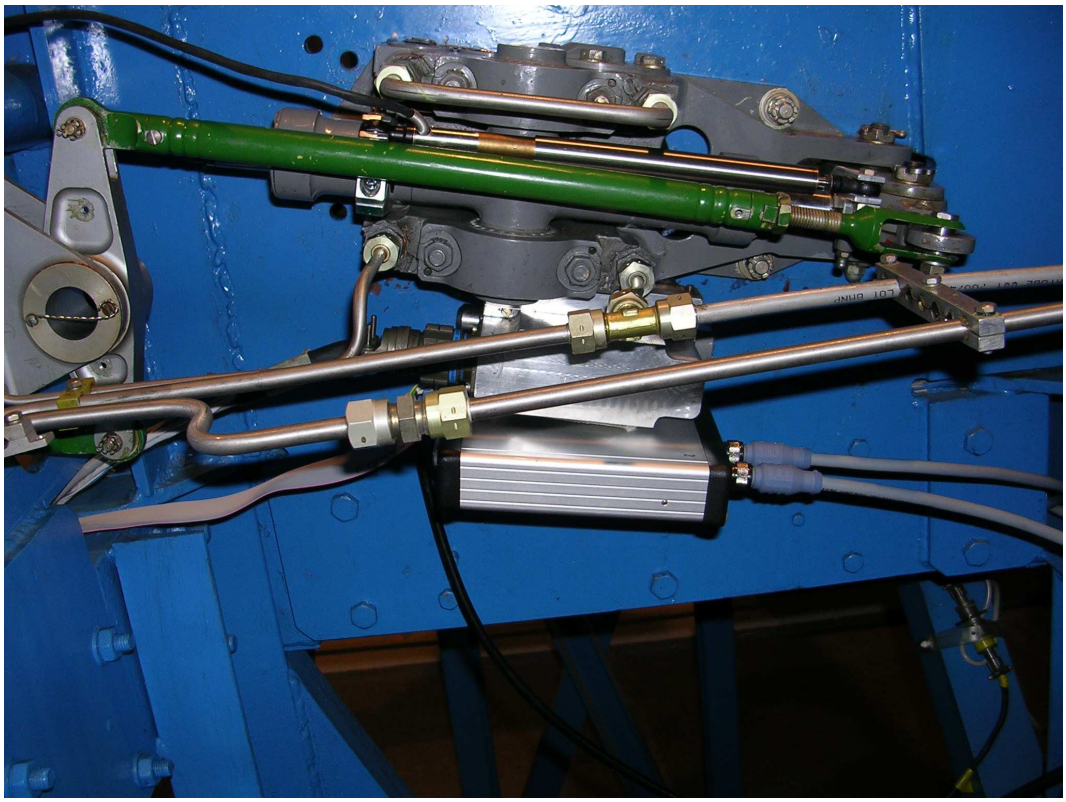


obr. 12.5 – Deska plošného spoje, maska BOTTOM

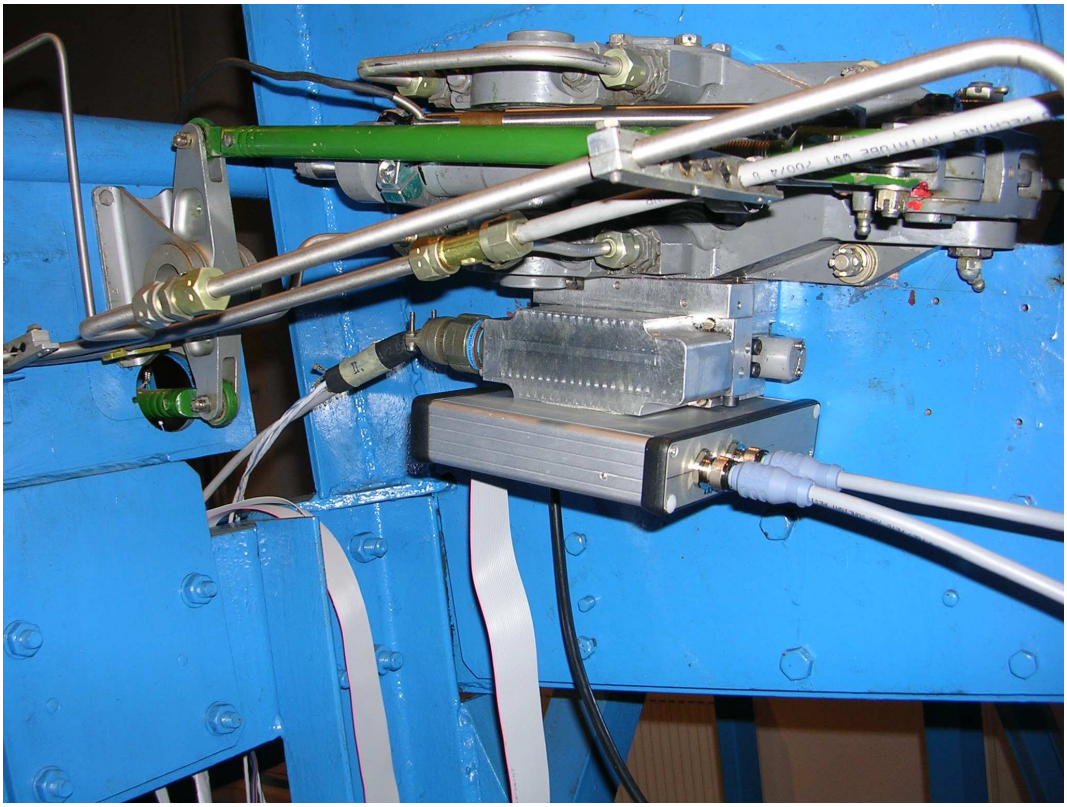
12.2 Fotodokumentace



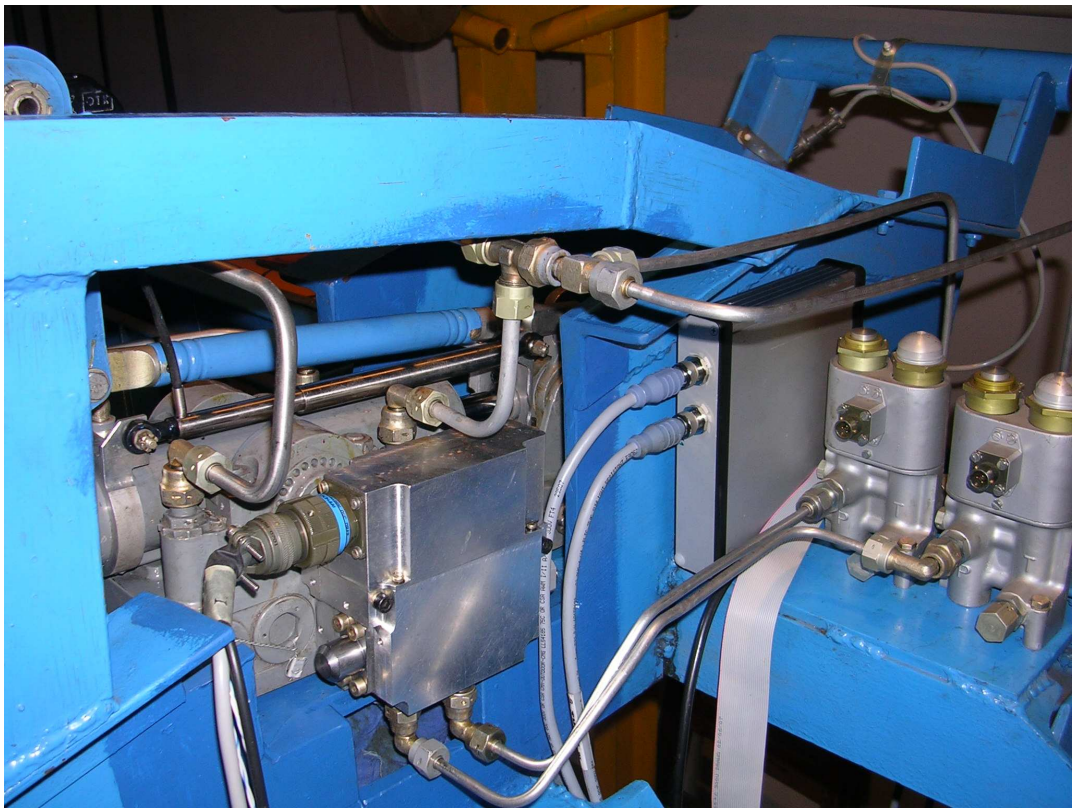
obr. 12.6 – Vrchní pohled na propojenou ECU a EHSA



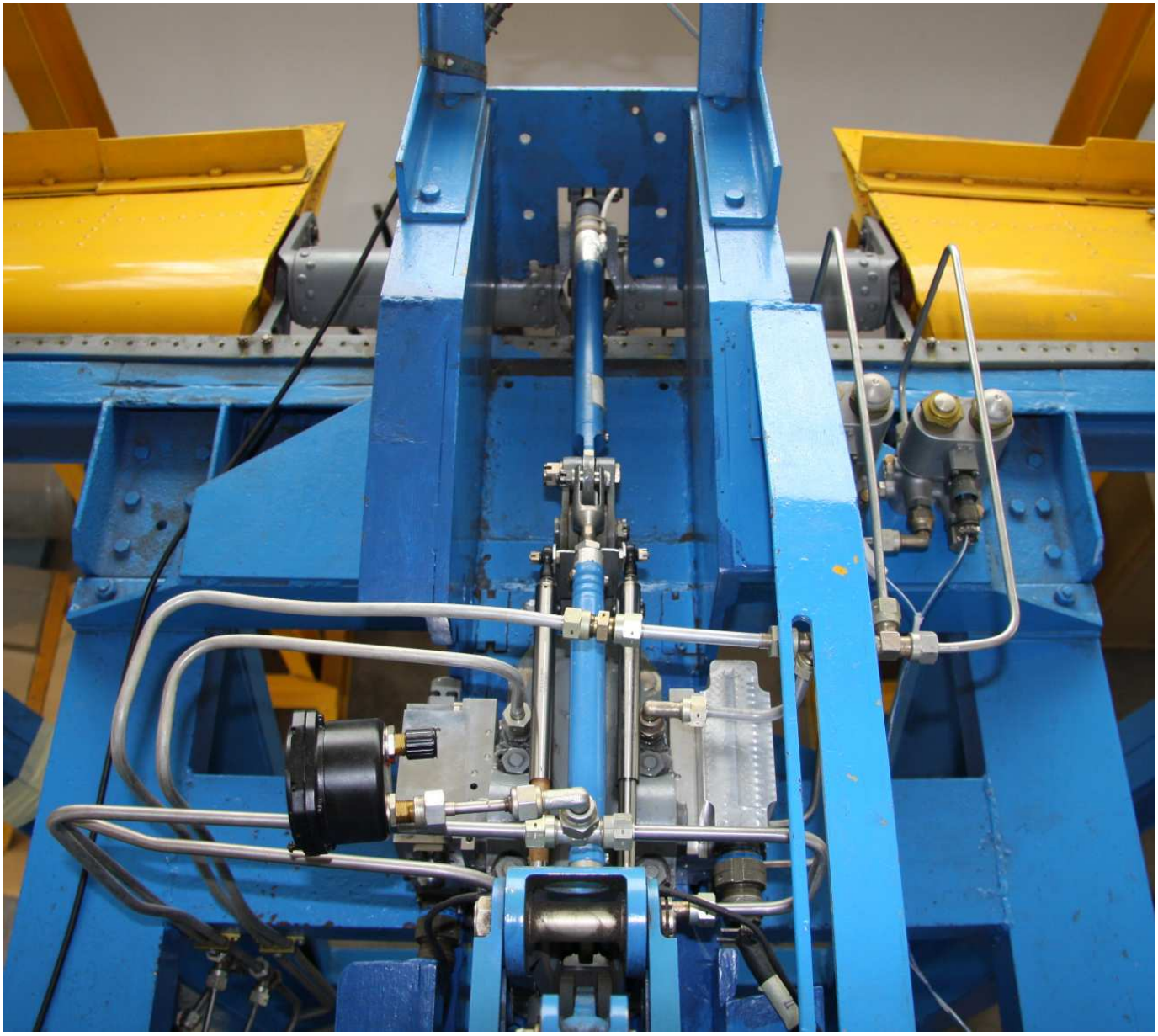
obr. 12.7 – Boční pohled na propojenou ECU a EHSA



obr. 12.8 – Umístění ECU na řízenou soustavu EHSA



obr. 12.9 – Alternativní umístění a propojení ECU 1 a EHSA



obr. 12.10 – Umístění EHSA na letounovém zkušebním zařízení, pohled shora

obr. 12.11 – Soustava EHSA

obr. 12.12 – Lokální zatěžovací zařízení

obr. 12.13 – Vrchní pohled na zatěžovací zařízení, umístění EHSA a zatěžovacího systému