Univerzita Pardubice

Dopravní fakulta Jana Pernera

Modernizace řídícího terminálu válcového dynamometru

Luboš Vidner

Diplomová práce

2017

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Luboš Vidner**
Osobní číslo:	**D13739**
Studijní program:	**N3708 Dopravní inženýrství a spoje**
Studijní obor:	**Elektrotechnické a elektronické systémy v dopravě**
Název tématu:	**Modernizace řídícího terminálu válcového dynamometru**
Zadávající katedra:	**Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě**

Zásady pro vypracování:

Nahraďte stávající řídící terminál válcového dynamometru v laboratoři KEEZ novou řídící procesorovou deskou, která rozšíří možnosti pracoviště. Nová řídící deska již existuje, není třeba ji vyvíjet. Propojte jednotlivé části řídícího terminálu mezi sebou (zdroj, pulzní měnič, řídící desku, LCD, klávesnici) a tyto komponenty umístěte do vhodné skříně. Do firmwaru nového řídícího terminálu doplňte komunikaci po sběrnici CAN a terminál komunikačně propojte s vybranými moduly u dynamometru, aby je bylo možné z terminálu ovládat, respektive z nich vyčítat data. Opravte chyby v současné verzi firmwaru terminálu.
Ověřte funkci nového řídícího terminálu ve spojení s dynamometrem v laboratoři KEEZ. Do měřící ústředny, která je součástí dynamometrického pracoviště, vytvořte modul pro měření otáček spalovacího motoru. Ověřte funkci modulu a funkci ústředny s tímto modulem.
Doporučný postup:
- Seznámení s aktuálním stavem dynamometrického pracoviště KEEZ, novou procesorovou řídící deskou a moduly na sběrnici CAN.
- Hardwarová úprava terminálu.
- Oprava a přidání nových funkcí do stávajícího firmwaru terminálu.
- Ověření funkce terminálu ve spojení s dynamometrem a jeho technologií, ověření komunikace s CAN moduly.
- Vytvoření modulu měření otáček spalovacího motoru do měřící ústředny a ověření jeho funkce.
- Provedení závěrečného měření výkonu školního motocyklu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

HEROUT, P. Učebnice jazyka C. Nakladatelství KOPP, 2004, IV. přepracované vydání.
Datasheety výrobců komponent.
BURKHARD, M. C pro mikrokontroléry. BEN - Technická literatura, 2003.
MAŠEK, Zdeněk. Programové řízení dynamometru pro zkoušení pohonu vozidel. Univerzita Pardubice, 2005. Diplomová práce. Univerzita Pardubice.
JIŘÍ, Marek. Řídicí jednotka pro válcový dynamometr. Univerzita Pardubice, 2012. Diplomová práce. Univerzita Pardubice.
LELEK, Tomáš. Měřící ústředna k dynamometru s výstupem na sběrnici CAN. Univerzita Pardubice, 2011. Bakalářská práce. Univerzita Pardubice.
Přednášky z předmětů "Aplikace mikroprocesorů", "Diagnostika silničních vozidel" a "Mikroprocesorová a senzorová technika v konstrukci diagnostice vozidel".

Vedoucí diplomové práce: **Ing. Zdeněk Mašek, Ph.D.**
Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě

Datum zadání diplomové práce: **27. listopadu 2014**

Termín odevzdání diplomové práce: **26. května 2017**

doc. Ing. Libor Švadlenka, Ph.D.
děkan

L.S.

Ing. Dušan Čermák, Ph.D.
vedoucí katedry

V Pardubicích dne 2. března 2017

Prohlášení autora

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 31. 8. 2017 Luboš Vidner

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Zdeňkovi Maškovi, Ph.D., za jeho odborné vedení, cenné rady, připomínky a návrhy v celém průběhu práce. Dále bych chtěl velmi poděkovat za odborné rady pánům Ing. Michalu Zajačikovi a Ing. Zdeňku Tobiášovi. A panu Ing. Petru Hapalovi za seznámení s vývojovým prostředím CodeWarrior 5.1. Závěrem bych chtěl poděkovat celé své rodině, která mě vždy ve studiu podporovala.

ANOTACE

Diplomová práce se zabývá modernizací řídícího terminálu ”Modernizace řídícího terminálu válcového dynamometru”. Předmětem práce je úprava firmware řídícího terminálu, tak aby bylo umožněno dálkové ovládání pomocí nadřazeného PC a přidaní komunikace po sběrnici CAN. Tato práce navazuje na práci Bc. Jiřího Marka. Práce je rozdělena na několik částí. První část obsahuje seznámení se stavem práce Bc. Jiřího Marka. Další části obsahují opravu chyb v původním firmware a přidání nových funkcí. Nakonec následuje ověření funkčnosti řídícího terminálu po úpravách.

KLÍČOVÁ SLOVA

Řídící terminál, dálkové ovládání, sběrnice CAN, platforma Amit

TITLE

Modernization of the roller dynamometer control terminal

ANNOTATION

This thesis deals with modernization of the control terminal „Modernization of the control terminal of a roller dynamometer“. The subject of the thesis is firmware adjustment of the control terminal to be enabled remote control using the parent PC and adding communication over the CAN bus. This thesis follows up on Bc. Jiří Marek´s work. The work is divided into several parts. The first part contains a familiarization with the state of Bc. Jiří Marek´s work. Other parts contain a correction of errors in the original firmware and an addition of new functions. Finally, verification of the functionality of the control terminal follows after adjustment.

KEYWORDS

Control terminal, remote control, CAN bus, Amit platform

OBSAH

ÚVOD

Diplomová práce se zabývá modernizací řídícího terminálu, přesněji odstraněním chyb ve stávajícím firmware (umožnění ovládání řídícího terminálu z nadřazeného PC) a doplněním komunikace po sběrnici CAN.

Tato práce navazuje na práci Bc. Jiřího Marka, který vyrobil HW část a napsal, přesněji přepsal z platformy Amit (16-bit mikrokontrolér Infineon 80C167) pro řídící terminál firmware na novou platformu s 16-bit mikrokontrolérem MC9S12XEP100.

Tento řídící terminál již by měl plně nahradit současný terminál, který je postaven na platformě Amit a umožnit plně využití stanoviště dynamometru, které se nachází v laboratořích katedry KEEZ.

Tento typ mikrokontroléru je vhodnější pro danou aplikaci, protože disponuje požadovaným výpočetním výkonem než mikrokontrolér obsažený v terminálu Amit (viz literatura [2]). A navíc obsahuje integrovanou jednotku pro komunikaci po sběrnici CAN.

Sběrnice CAN díky své odolnosti proti chybám a rušení a také vysoké přenosové rychlosti našla jak uplatnění při řízení průmyslových zařízení, tak především v dopravních prostředcích, kde je zejména kladen důraz na vysokou spolehlivost.

Modul měření otáček spalovacího motoru tato práce neobsahuje, protože byla předmětem ročníkového projektu viz literatura [1].

1. AKTUÁLNÍ STAV

1.1. Hardware

Převzal jsem již vyrobenou a oživenou desku s mikrokontrolérem MC9S12XEP100 a s potřebnými periferiemi, vyrobenou Bc. Jiřím Markem.

1.1.1. Hlavní části

Jádro zařízení:

mikrokontrolér MC9S12XEP100 (16bit)

Podpůrné IO[1]

sériová komunikace

převodník úrovní pro RS232 MAX232

převodník úrovní pro USB FT232

sběrnice CAN:

budiče sběrnice CAN PCA82C250C

napájení mikrokontroléru:

DC/DC měnič AMSR — 7805 — NZ

referenční napětí pro A/D převodník mikrokontroléru:

reference REF50XX

napájení reference:

lineární stabilizátor 7809

IO pro galvanické oddělení vstupů/výstupů:

ADUM1400, ADUM1301

Napájení oddělovacích IO:

DC/DC měniče AM1S—0505SZ

[1] IO – integrovaný obvod, na malé ploše respektive na křemíkovém plátku je vytvořen elektronický obvod z aktivních i pasivních elektronických prvků.

Vstupy/Výstupy

konektor pro připojení LCD displeje 16x2

konektor pro připojení maticové klávesnice 4x3

1x USB — není galvanicky odděleno od mikrokontroléru

1x RS232 — není galvanicky odděleno od mikrokontroléru

2x2 CAN — galvanicky odděleno od mikrokontroléru

1x analogové vstupy/výstupy — nejsou galvanicky oddělené od mikrokontroléru

1x digitální vstupy/výstupy - vstupy jsou galvanicky oddělené od mikrokontroléru, výstupy jištěny proudovými zdroji.

Dále zařízení obsahovalo membránovou klávesnici 4x3 a alfanumerický LCD displej 16x2 od firmy PALM TECHNOLOGY CO., LTD.

Foto převzaté DPS viz příloha D.

1.2. Úprava hardware:

Mým úkolem bylo splnit následující úkoly.

Vyrobit přední panel, tak aby bylo možno umístit na něj LCD displej 16x2 s maticovou klávesnicí. Tento panel je nutno vyrobit s ohledem na umístění v použitém PC šasi.

Vhodně umístit a připevnit DPS v použitém PC šasi.

Vyrobit propojovací kabely na CAN bus, USB a RS232 a panel pro vyvedení USB konektoru na zadní straně PC šasi.

Dále byl upraven kabel na připojení LCD displeje (doplněn upevňovací konektor a modul displeje s lámací lištou).

Výše uvedené bylo splněno.

1.3. Software

Byl zjištěn následující stav přebíraného zařízení.

1.3.1. Místní ovládání:

Terminál správně reagoval na povely zadávané z klávesnice. Nezkoušel jsem funkčnost regulátorů, protože terminál s PC šasi se již nacházel mimo těžké laboratoře a vedoucí práce rozhodnul, že to není potřeba.

1.3.2. Komunikace s nadřazeným PC:

Nešlo zapnout dálkové ovládání. Na PC obrazovce vyskakovala chybová zpráva, že terminál vůbec neodpovídá.

Tato chyba vyskakovala při pokusu o zapnutí přímého řízení, statického a dynamického režimu.

Chyba na straně software běžícího na PC byla vyloučena, protože z analýzy komunikace po sériové lince s řídícím terminálem bylo zjištěno, že řídící terminál nezasílá odpověď´ na přijaté bajty z nadřazeného PC.

1.3.3. Komunikace po sběrnici CAN:

Provedena inicializace. Tato inicializace byla odstraněna a nahrazena vygenerovanou pomocí nástroje Processor Expert (dále PE). Nástroj PE je obsažen v CodeWarrior 5.1, který slouží pro rychlé nastavení jednotlivých periferií mikrokontroléru.

1.3.4. Struktura projektu

Celý projekt je rozdělen do více souborů. Každý soubor obsahuje definice funkcí pro obsluhu určité části řídícího terminálu. Například soubor s názvem Regul.c obsahuje definice funkcí pro obsluhu jednotlivých regulátorů.

Seznam modulů je vyňat z diplomové práce Bc. Jiřího Marka uvedeném na stránkách 47 a 48. Tak to také bylo při otevření projektu.

Brzda.c, Brzda.h – obsahuje hlavní funkci `main`, volá funkce pro nastavení periferií a definuje proměnné použité v celém projektu

Events.c, Events.h – program sem skáče při vyvolání všech přerušení a jsou odtud volány funkce obsluhující přerušení

Dalk_ovl.c, Dalk_ovl.h – obsluha programu při dálkovém řízení jednotky

Nabidka.c, Nabidka.h – zobrazení hlavní nabídky na displeji

Nabidka2.c, Nabidka2.h – zobrazení podrobnější nabídky, přechází z Nabidka.c

Nastav.c, Nastav.h – nastavení jednotlivých periferií

Pomocne.c, Pomocne.h – pomocné funkce, načítání a ukládání proměnných a obsluha A/D převodníku

Regul.c, Regul.h – obsahuje kód pro obsluhu regulace otáček, proudu a momentu

Zapis_E.c, Zapis.h – zapisování a vyčítání hodnot uložených v EEPROM

spolecne.h – definice některých proměnných používaných v celém projektu

Klavesnice.c, Klavesnice.h – obsluha maticové klávesnice

LCD.c, LCD.h – obsahuje funkce pro inicializaci a zobrazovaní na displeji

Serial.c, Serial.h – obsluha sériové linky

termio.c, termio.h – funkce pro formát dat do řetězce umožňující zobrazování na displeji

2. OPRAVA SOFTWARE

2.1. Navázání komunikace s PC

Prvně jsem postupoval tak, že jsem ověřil, zda-li není závada v HW části.

To jsem provedl tak, že jsem si založil vlastní projekt a naprogramoval jsem si program, který měl za úkol přijmout bajt po sériové lince a v nezměněné formě ho poslat nazpátek do PC.

Řídící terminál po spuštění projektu dělal přesně to, co jsem od něj vyžadoval. Tím jsem vyloučil HW chybu.

Dále jsem tento vlastní program nahrál do projektu Bc. Jiřího Marka a zkoušel jsem, jestli řídící terminál bude plnit zadaný úkol.

Na žádná přijímaná data nereagoval.

Pak jsem následně zkontroloval nastavení sériové linky a porovnal jsem to s nastavením vlastního projektu.

Moje nastavení a nastavení sériové linky od Bc. Jiřího Marka se lišilo v tom, že velikost paměti pro příjem/vysílání měl nastaven na 1 bajt. A při této velikosti paměti bohužel MC9S12XEP100 nevysílá žádné bajty.

Dále je nutné po odvysílání zprávy vymazat tuto zprávu z přidělené paměti pro vysílání, protože při zaplnění mikrokontrolér také přestává vysílat.

Správný postup zprovoznění sériové linky pomocí Processor Expert (dále PE)

- přidat pomocí PE modul sériové linky
- nastavit vstupní/výstupní bránu
- nastavit prioritu přerušení
- velikost paměti pro příjem/vysílání — na více než na jeden bajt
- délka zprávy, parita, přenosová rychlost a aj.
- povolit přijímač, vysílač
- povolit inicializaci modulu, přerušení při startu programu a aj.

Příjem a Vysílání je řešeno pomocí přerušení.

- přijmout data, zpracovat data
- vysílání – zápisem do vysílacího registru je spuštěno vysílání, využívá se přerušení
- Po odvysílání je nutno smazat přidělenou paměť pro vysílání, jinak další data nebudou odvysílána respektive ve velikosti této přidělené paměti.

Na řešení nízkoúrovňového přístupu k sériové lince slouží funkce, které jsou vygenerované PE.

Po zjištění správného nastavení sériové linky jsem testoval, jestli mikrokontrolér správně přijímá data nebo ne. Testování jsem provedl tak, že jsem posílal data po RS232 a co se přijalo, tak jsem posílal zpět po USB do PC. Dále jsem využil výstupů na připojení panelu s LED diodami. Led diody mně ukazovaly, jestli se daný kód vykoná nebo ne.

Nejprve jsem zkusil, jestli se vykoná přerušení po příjmu — ano.

Pak následovalo otestování, jestli mikrokontrolér přijme správná data po sériové lince.

Obsluhu přerušení po příjmu po sériové lince obstarává funkce *void Prerus_od_RS232_Rx(void).*

Tato funkce přijala stejné bajty, které jsem vyslal po sériové lince. Pak následovalo zjištění, jestli jsou prováděny dále kroky, které následují po úspěšném příjmu paketu.

Kroky byly správně dále provedeny, protože na displeji jsem dostal správné zobrazení. Terminál nadále nespolehlivě komunikoval s PC, jenom šlo vypnout dálkové ovládání.

Nakonec řešením bylo úprava funkce *NastavNuluMomentu(),* která je volána při výběru regulátoru. Funkce nastavuje nulu momentu.

Tato funkce zbytečně zdržovala provoz na sériové lince. Níže, původní řešení.

```
void NastavNuluMomentu(void)
{
int32_t sum = 0;
int16_t i;
Cpu_DisableInt();
for (i=0; i<1000; i++)
        {
        sum += read_adc_m(1);
        Cpu_Delay100US(5);
        }
Msn_nula = (int16_t)(sum/1000)+1;
Cpu_EnableInt();
}
```

Popis

Inicializace proměnných sum a i. Dále následuje vypnutí globálního přerušení (funkce *Cpu_DisableInt();*). Po vypnutí globálního přerušení následuje cyklus s opakováním 1000. V tomto cyklu se načítá vstup z A/D převodníku, na který je připojen tenzometr. Po převodu hodnoty následuje čekací smyčka o délce 500 μs. Po skončení cyklu následuje výpočet momentu a povolení globálního přerušení.

Funkce je volána třikrát za sebou. A průchod trvá (počítám jenom čekací smyčku, která trvá v jednom průchodu cyklem 500 μs) 500 μs. 1000 (průchodů cyklem) = 0,5s . A tato funkce je volána třikrát za sebou. To je 0,5s . 3 = 1,5 s!!!

Po této krátké analýze nastávají otázky – z jakého důvodu zakazovat/povolovat globální přerušení, a také proč čekat 500μs, když převod z A/D převodníku trvá podle PE kolem 20μs.

Moje úprava funkce spočívala v tom, že jsem zrušil zakazování/povolování globálního přerušení a čekací smyčku jsem zkrátil na 100 μs.

Po úpravě těchto částí bylo dálkové ovládání zprovozněno. Nefungovalo pouze periodické vyčítání dat z terminálu.

2.1.1. Souhrn

Název problému: Zprovoznění dálkového ovládání

Chyba:

- malá velikost přidělené paměti, pro příjem/vysílání

- funkce NastavNuluMomentu() moc dlouhá čekací smyčka
- zákaz/povolení globálního přerušení

Řešení:

- přidělena větší velikost přidělené paměti, pro příjem/vysílání
- úprava funkce void NastavNuluMomentu(void),
- zkrácení čekací smyčky,
- odstranění vypnutí/zapnutí globálního přerušení

2.2. Periodické vyčítání dat z terminálu

Pokud máme „periodickou“ činnost, tak je vykonávána za určitý časový úsek. Má-li to dělat mikrokontrolér, tak vykonávání této činnosti vhodné dát do přerušení od časovače. Tím bude zajištěno, že činnost bude vykonávána stále za stejný čas.

Po zhlédnutí zdrojového kódu bylo zjištěno, že periodické vysílání je v přerušení od časovače T2. Zkusil jsem dát do tohoto přerušení blikání LED. LED neblikala, tak jsem zkontroloval, jestli je komponenta zinicializována a atd.

Nikde jsem nenašel inicializaci, tak jsem do hlavní funkce main () na začátek přidal inicializaci a problém byl vyřešen. Na inicializaci tohoto přerušení slouží funkce vygenerovaná PE *T2_Enable().*

2.2.1. Souhrn

- Název problému: periodické vyčítání dat
- Chyba:

  není inicializován časovač T2
- Řešení:

  přidání inicializace na začátku funkce main () pomocí funkce T2_Enable()

Časovač T2 vyvolá přerušení každých 25 ms. Nastavení časovače se nalézá v souboru T2.c

2.3. Periodické vykonávání regulátorů

Regulátory jsou vykonávány v přerušení od časovače T0. Postupoval jsem jako při zprovoznění periodického vyčítání dat.

2.3.1. Souhrn

- Název problému: Periodické vykonávání regulátorů
- Chyba:

  není inicializován časovač T0
- Řešení:

  inicializace na začátku funkce main () pomocí funkce T0_Enable()

Časovač T0 vyvolá přerušení každou 1 ms. Nastavení časovače se nalézá v souboru T0.c

2.4. Vypnutí dálkového ovládání z PC

Možnost dálkového ovládání se rozumí možnost ovládání řídícího terminálu povely z nadřazeného PC. Při dálkovém ovládání terminálu je z důvodu konfliktu mezi dálkovým ovládáním a místním ovládáním terminálu, znemožněno místní ovládání.

Ukončení módu dálkového ovládání je možno pouze povelem z PC.

Dále nešlo vypnout dálkové ovládání z PC. Po zhlédnutí kódu jsem zjistil, že když se vypne dálkové ukládání, tak následuje uložení nastavených parametrů do EEPROM. Pokud jsem toto ukládání zrušil, tak dálkové ovládání šlo z PC vypnout normálně.

Tak tedy příčina je jasná. Upravit funkci pro uložení dat do EEPROM (*zapis_EEPROM()).*

Zjednodušil jsem metodu pro ukládání dat do EEPROM z Safe write na Destructive write a ve funkci *zapis_EEPROM()* jsem zakázal globální přerušení před voláním nízkoúrovňové funkce pro ukládání dat vygenerované PE, po skončení této funkce následuje samozřejmě opět povolení globálního přerušení, tak aby uložení dat nebylo ničím rušeno.

Dále jsem zrušil, aby byla tato funkce volána neustále v cyklu, pokud dojde k jakékoliv chybě v ukládání dat. Hrozí zamrznutí programu. Lepší variantu bych například viděl

v rozsvícení led diody error na čelním panelu terminálu nebo v tom, že by po pěti nezdarech v ukládání byl tento cyklus přerušen a následovalo by rozsvícení ledky error.

Zdrojový kód funkce *void zapis_EEPROM(void)* nebudu uvádět (je příliš dlouhý). Uvedu pouze ten úsek, kde jsem provedl úpravu.

```
Cpu_DisableInt();                    //  zákaz globálního přerušení, mé přidání
for (i=0; i<POCET_BAJTU_EEPROM; i++)
{
        //while (write_eeprom((word)(i), poleBajtu[i]) == 1);
//      do
//      {
          status = IEE1_SetByte(((IEE1_TAddress)(i+offset)), poleBajtu[i]);

//      } while (status == ERR_BUSY);  //  zakomentován cyklus do – while, který
                                       //testuje úspěšnost zapsání do EEPROM
                                       // má úprava
}
Cpu_EnableInt();      //  povolení globálního přerušení, mé přidání
```

2.4.1. Souhrn

- Název problému: nelze vypnout dálkové ovládání
- Chyba:

  Příliš složitá metoda pro ukládání dat do EEPROM, rušení ostatními přerušeními.

  Zamrznutí programu, při variantě, že nízkoúrovňová funkce pro ukládání dat, vrátí chybu.

- Řešení:

  Zjednodušení metody pro ukládání dat do EEPROM. Přidání zákaz/povolení globálního přerušení. Zrušen cyklus pro testování návratové hodnoty.

2.5. Vypisování naměřených hodnot v souladu s terminálem Amit

Po úspěšném rozběhnutí dálkového ovládání jsem se zaměřil na to, aby terminál vypisoval stejné hodnoty naměřených veličin, jaké vypisuje na obrazovce terminál na platformě Amit.

Po odzkoušení jsem zjistil, že je chybně vypisován AD_proud, AD_moment a otacky_skutecne.

AD_proud byl dvojnásobný, AD_moment čtyřnásobný a otáčky poloviční oproti hodnotám, které vypisoval terminál na platformě Amit.

Úprava spočívala pouze v doplnění jednoduchých početních operací tak, aby se dosáhlo měřených hodnot v souladu s terminálem na platformě Amit.

2.6. Ukládání dat

OSVD (program na PC, který komunikuje pomocí sériové linky s řídícím terminálem) vyžaduje rychlou odpověď z terminálu o úspěšném ukončení módu dálkového ovládání. A uložení dat do EEPROM by zbytečně zdržovalo komunikaci po sériové lince. Z tohoto důvodu je přesunuto ukládání dat do EEPROM do nekonečné smyčky ve funkci *main()* .

Vyjmutí je provedeno následujícím způsobem. V těle funkce *uint8_t prepni_ovl(byte* pBajt, byte pocet_dat_bajtu*) je zrušeno volání funkce *void zapis do EEPROM(void)* a přidáno nastavení příznaku (proměnná *priznak_uloz_do_eeprom = 1*). Číselná hodnota příznaku v nekonečné smyčce rozhodne o tom, jestli se bude ukládat do EEPROM nebo ne.

Zdrojový kód funkce *uint8_t prepni_ovl(byte* pBajt, byte pocet_dat_bajtu)* nebudu uvádět (je příliš dlouhý). Uvedu pouze ten úsek, kde jsem provedl úpravu.

```
if (RUN_STOP == 0)
{
        dalkove_ovl = 0;
        priznak_uloz_do_eeprom = 1;       // moje přidání
        // zapis do EEPROM zakomentováno volání funkce
                                // funkce volána až v nekonečné smyčce
        clear_display();
        /*printf("Dalkove ovladani vypnuto!")*/
        /*dl3();*/
```

```
            zobrazovat = 1;
        }
```

Úsek kódu přidán do nekonečné smyčky v main().

```
if(priznak_uloz_do_eeprom)
    {
    zapis_EEPROM();
    priznak_uloz_do_eeprom = 0;
    }
```

Pokud proměnná s názvem priznak_uloz_do_eeprom nabývá hodnoty větší než nula, tak je provedeno zavolání funkce *zapis_EEPROM()* a přiřazení nuly do proměnné *priznak_uloz_do_eeprom*. Ukládat do EEPROM je nutno pouze jednou.

2.7. Vyčítání dat z EEPROM při spuštění terminálu

Dle požadavku pana vedoucího práce by měla funkce nového terminálu být co nejvíce podobná funkci terminálu, který je naprogramován na platformě Amit. Proto bylo nutné přidat při spuštění terminálu vyčítání dat z eeprom.

To lze provést tak, že se na začátek funkce *main()* přidá následující úsek zdrojového kódu.

```
clear_display();
if(!cti_EEPROM()) { print_string("Obsah EEPROM nacten!");}
else {print_string("Data neplatna,konec cteni EEPROM");}
dl3();
```

Nejprve je smazán display (*funkce clear_display()*). Bez této úpravy je špatně zobrazován následující text na displeji. Potom je zavolána funkce *cti_EEPROM(),* která vyčte data z EEPROM. Její návratová hodnota dává informaci o tom, jestli data byla vyčtena úspěšně nebo ne.

Vrací: 0 — úspěšně provedeno načtení hodnot, 1 — data jsou chybná

Tato návratová hodnota je potom použita pro větvení programu. Pokud jsou data úspěšně zapsána, tak na displeji je vypsáno "Obsah EEPROM nacten!", jinak "Data neplatna,konec cteni EEPROM".

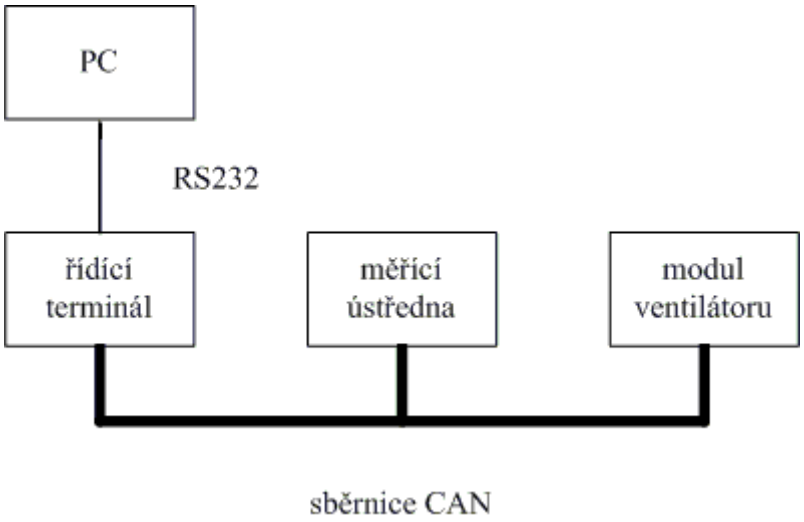
3. SBĚRNICE CAN

Do řídícího terminálu byla přidána komunikace po sběrnici CAN, z důvodu komunikace s ostatními moduly, které jsou součástí pracoviště válcového dynamometru v těžkých laboratořích katedry KEEZ.

Jedná se o modul ventilátoru a měřící ústřednu.

Modul ventilátoru ovládá přes frekvenční měnič motor ventilátoru, který chladí daný měřený objekt. Modul ventilátoru vyžaduje informaci o rychlosti měřeného objektu, v budoucnu i aktuální teplotu daného měřeného objektu. Podle aktuální přijaté rychlosti nastaví požadované otáčky motoru ventilátoru.

Modul měřící ústředna slouží k připojení různých čidel. Obsahuje osm analogových vstupů, osm vstupů na termočlánek, vstupy na dva platinové teploměry, modul měření otáček a možnost připojení sondy HB_85.



Obrázek 3-1 Síťový diagram

Do projektu jsou přidány následující soubory.

Všechny id zpráv přítomných na sběrnici CAN jsou umístěny do souboru CAN_id.h.

Výpis souboru CAN_id.h

```
#define ven_rych_can_id                 0x10  // ventilator Hladik


#define analog_vstupy1_CAN_id           0x11  // merici ustredna Lelek
#define analog_vstupy2_CAN_id           0x09      // viz příloha A 0x10
```

```
#define termoclanky_1_CAN_id                0x13
#define termoclanky_2_CAN_id                0x14        // 0x13
#define stav_vstupu_termoclanky_CAN_id      0x15
#define platinovy_teplomer_CAN_id           0x16
#define stav_vstupu_platina_RTD1_CAN_id     0x17
#define stav_vstupu_platina_RTD2_CAN_id     0x19
#define otacky_motoru_CAN_id                0x12        // 0x12
#define sonda_HXB_85_CAN_id                 0x1B
#define ledky_can_id                        0xA1  // zkouska ovladani terminalu pomoci CAN Analyzatoru
```

3.1. Vysvětlení pojmů

MSB – více významný bajt (v případě 16-bit čísla se jedná o jeho horní bajt)

LSB – méně významný bajt (v případě 16-bit čísla se jedná o jeho dolní bajt)

3.2. Nastavení sběrnice CAN

Inicializace a nastavení CAN linky (oba dva kanály) byla provedena pomocí nástroje Processor Expert.

Nastavení

- komunikační rámec CAN2.0A
- přenosová rychlost 1Mbit/s

Měřící ústředna může být připojena k oběma CAN linkám. Modul ventilátoru se připojuje k lince 2.

3.3. Princip řízení zpráv po sběrnici CAN

Princip řízení zpráv po CANu je založen na použití kruhové fronty.

Na tomto principu je založen princip komunikace po sběrnici CAN.

Jednotlivé zprávy po CANu jsou přijímány, a pokud mikrokontrolér nestíhá jednotlivé zprávy zpracovat, tak se může stát, že všechny zprávy nebudou přijaty a některé se ztratí. To může být vyřešeno tím, že mokrokontrolér nejprve uloží zprávy do paměti (fronta) a když na ně bude mít čas, tak je může obsloužit (vytáhne si je z paměti). Ukládání zpráv mikrokontroléru zabere mnohem méně času, než kdyby ještě musel na ně ihned reagovat. Musí je pouze umět hned přijmout a uložit.

Z této krátké analýzy lze hned určit činnosti, které mikrokontrolér bude konat.

Pro příjem zprávy po CANu je použito přerušení, a v tomto přerušení se ukládají přijaté zprávy do paměti.

V hlavní funkci main (v nekonečné smyčce) je umístěna funkce, která vyčte z paměti jednotlivé CAN zprávy a na ně bude reagovat.

3.3.1. Přerušení po příjmu CAN zprávy

Jednotlivé zprávy jsou načítány do struktury s názvem frameRx. Pak obsah struktury je uložen do pole zpráv přijatých po CANu. Zpráva, která nebyla přijata úspěšně, tak není uložena do pole zpráv. Pole zpráv slouží jako místo vyhrazené v paměti RAM, kde se fyzicky ukládají jednotlivé bajty přijaté zprávy.

3.3.2. Nekonečná smyčka

V nekonečné smyčce ve funkci main (), pokud pole zpráv obsahuje nezpracovanou zprávu, tak jednotlivé nezpracované zprávy jsou vyčítány do struktury s názvem frameW. Podle CAN_id je tato zpráva podle kódu patřičně zpracována. Toto trvá tak dlouho, dokud nejsou všechny zprávy zpracovány.

Vzhledem, že pole je indexované, tak podle indexů se pozná, jestli byly všechny zprávy zpracovány.

Jsou vytvořeny dvě proměnné, jejichž hodnoty říkají, jestli byly všechny zprávy zpracovány. V přerušení od příjmu se navyšuje hodnota proměnné s přijatou zprávou, která říká, kolik bylo přijato zpráv. S každou novou zprávou se hodnota proměnné navyšuje o jedna. Proměnná se použije pro index, do jaké pozice ve frontě zpráv tato zpráva bude uložena. Hodnota se navyšuje až po uložení přijaté zprávy.

V nekonečné smyčce je zase navyšována hodnota při načtení a zpracování každé CAN zprávy. Pokud se hodnoty obou proměnných rovnají, tak jsou zpracovány všechny přijaté zprávy.

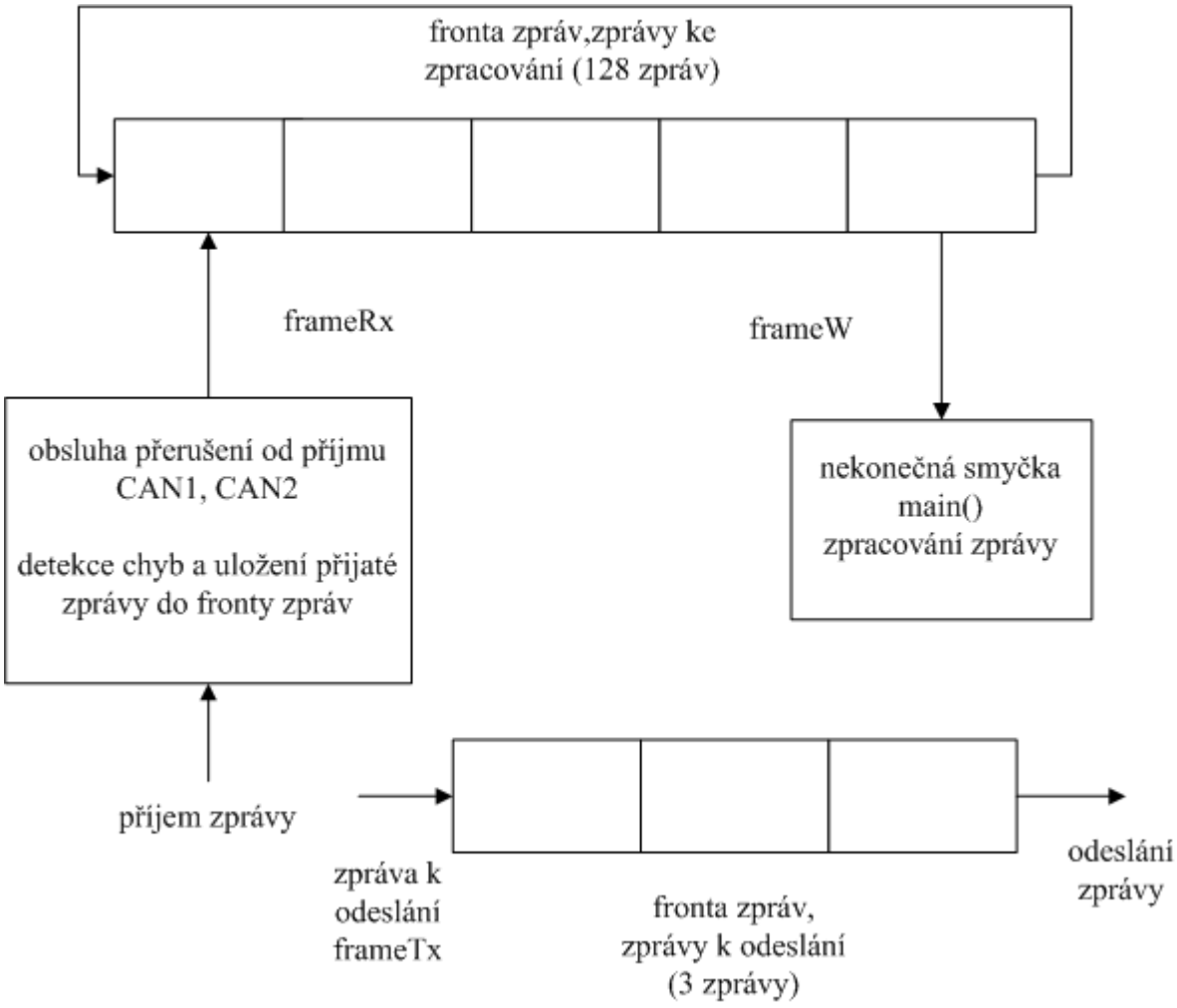
Tyto dvě proměnné se musí limitovat na velikost fronty zpráv. Jinak hrozí, že zprávy by byly ukládány do jiné části paměti a nemohly by být později načteny. Limitace má za následek, že pokud mikrokontrolér by nezpracovával CAN zprávy s dostatečnou rychlostí, tak nastává přepsání zpráv, které přišly dříve.

Při dosažení konce pole zpráv jsou hodnoty proměnných vráceny na začátek fronty. Tedy na prvek ve frontě zpráv s indexem 0.

3.3.3. Vysílání zpráv po CANu

Vysílání zpráv po CANu je vytvořeno pomocí funkce *unsigned char can_sendFrame(unsigned char can, CanFrame *frameTx*). V těle této funkce se nachází funkce (*byte CAN1_SendFrameExt(dword MessageID,byte FrameType,byte Length,const byte *Data),* (pro odeslání do druhého kanálu změna hlavičky funkce z CAN1 na CAN2), která využívá k odeslání kruhovou frontu. Do kruhové fronty jsou nejvýše zapsány tři zprávy. Prvně se vyšle zpráva s nejvyšší prioritou (nejnižší id). Funkce je vygenerována PE.

3.3.4. Řízení toku zpráv po CANu



Obrázek 3-2 Řízení toku zpráv po CANu

3.3.5. Nové datové typy

Byl nadefinován nový datový typ (název CanFrame) pro uložení/zpracování a vyslání zprávy po CANu. Datový typ je použit pro deklarace struktur s názvem frameRx, frameW a frameTx.

Definice datového typu se nalézá v souboru can.h.

```
typedef struct
{
   unsigned long id;          // id zprávy
   unsigned char type;        // typ zprávy (žádost o data, datová zpráva)
   unsigned char format;      // CAN2.0A nebo CAN2.0B, typ přenášeného rámce
   unsigned char length;      // počet datových zpráv
```

```
    unsigned char data[8];        // přenášená data (datové pole)
} CanFrame;
```

3.3.6. Seznam funkcí pro řízení zpráv po CANu

Funkce jsou umístěny v souborech can.c , can.h.

void prerus_od_CAN_Rx(unsigned char can)

void can_zpracovani_zpravy(void)

can_sendFrame(unsigned char can, CanFrame *frameTx)

unsigned char can_receiveFrame(unsigned char can, CanFrame *frameRx)

3.3.7. Význam a popis jednotlivých funkcí

- void prerus_od_CAN_Rx(unsigned char can)

Název funkce:	prerus_od_CAN_Rx
Vstupní parametry:	unsigned char can – výběr CAN kanálu, v jakém přerušení je volána
Výstupní parametry:	nemá
Vrací:	nemá návratovou hodnotu
Význam:	Zjistí, jestli přijatá zpráva obsahuje chybu a pokud ne, tak ji uloží do fronty zpráv.
Poznámka:	Mikrokontrolér obsahuje dvě CAN linky. Na tuto funkci jsou kladeny maximální požadavky na rychlost, tak není věnován čas na inicializaci proměnných a ani ukazatelů. Může být součástí pouze tohoto projektu. Využívá funkce generované PE.

- void can_zpracovani_zpravy(void);

Název funkce:	can_zpracovani_zpravy
Vstupní parametry:	žádné
Výstupní parametry:	nemá
Vrací:	nemá návratovou hodnotu

Význam: Zpracuje dané přijaté CAN zprávy.

Poznámka: Pro zpřehlednění zdrojového kódu v nekonečné smyčce v hlavní funkci a pro umožnění rychlejší úpravy kódu.

- can_sendFrame(unsigned char can, CanFrame *frameTx)

Název funkce: can_sendFrame

Vstupní parametry: can – výběr CAN kanálu, ve kterém budou data poslána

CanFrame *frameTx – ukazatel na strukturu, která je připravena k odeslání

Výstupní parametry: nemá

Vrací: nemá návratovou hodnotu

Význam: Vyšle danou CAN zprávu.

Poznámka: nízkoúrovňová funkce

- unsigned char can_receiveFrame(unsigned char can, CanFrame *frameRx)

Název funkce: can_receiveFrame

Vstupní parametry: can – výběr CAN kanálu, v jakém je přijímaná zpráva

CanFrame *frameRx – ukazatel na strukturu, ve které jsou přijatá data

Výstupní parametry: nemá

Vrací: pokud došlo k chybě, tak vrací 0, jinak 1 – zpráva OK

Význam: Přijme danou CAN zprávu.

Poznámka: nízkoúrovňová funkce

3.4. Funkce pro obsluhu modulu ventilátor.

Funkce jsou umístěny v souborech ventilator.c , ventilator.h

Do modulu ventilátoru se zasílá údaj o rychlosti měřeného objektu s periodou vysílání 1 sekunda. Rychlost je zasílána ve formě celočíselného čísla, bez desetinné části.

Tabulka 3-1 Formát zasílané zprávy po CANu do modulu ventilátoru

CAN ID	typ	formát	délka	data
0x10	datová zpráva	CAN2.0A	1	uint8_t (údaj v km/h)

3.4.1. Seznam funkcí pro obsluhu modulu ventilátoru

void ven_rych_CAN_TX (unsigned char can_kanal, unsigned char can_id,float *p_rychlost)

3.4.2. Význam a popis jednotlivých funkcí

- void ven_rych_CAN_TX (unsigned char can_kanal, unsigned char can_id,float *p_rychlost)

Název funkce: ven_rych_CAN_TX

Vstupní parametry: unsigned char can_kanal – výběr CAN kanálu, v jakém kanále bude zpráva vyslána

unsigned char can_id – předáno CAN_id

float *p_rychlost – ukazatel na proměnou, ve které je uložena rychlost

Výstupní parametry: nemá

Vrací: nemá

Význam: složí zprávu s informací o rychlosti a zašle jí do modulu ventilátoru po CANu.

3.5. Funkce pro obsluhu modulu měřící ústředna

Funkce jsou umístěny v souborech merici_ustredna.c, merici_ustredna.h.

Modul měřící ústředna zasílá zprávy o stavu analogových vstupů, termočlánkových vstupů, údaje z platinových teploměrů, z modulu měření otáček a ze sondy HB_85.

Jednotlivé zprávy obsahují čísla, které jsou roztržena po osmi bitech. Je to proto, že datové pole ve zprávě zasílané po CAN obsahuje maximálně osm datových polí po osmi bitech.

Funkce pouze složí výsledná čísla a tyto čísla jsou poté uložena do struktury s názvem merici_ustredna datového typu MERICI_USTREDNA (popis datového typu níže).

Definice datového typu MERICI_USTREDNA , se nalézá v souboru can.h

```
typedef struct
{
   unsigned int analogove_vstupy[8];          // pole,ve kterém jsou uloženy analog. vstupy
   unsigned int termoclankove_vstupy[8];   // pole,ve kterém jsou uloženy termo. vstupy
   unsigned char stav_vstupu_termoclanky; // proměnná, info o stavu vstupů termocl.
   unsigned long int RTD1;                    // proměnná, teplota RTD1
   unsigned long int RTD2;                    // proměnná, teplota RTD2
   unsigned char stav_vstupu_RTD1;         // proměnná, stav vstupu RTD1
   unsigned char stav_vstupu_RTD2;         // proměnná, stav vstupu RTD2
   unsigned long int otacky_motor_CAN;    // proměnná, info o změřených otáčkách
   signed int sonda_HXB_85_vstupy[4];      // pole,ve kterém jsou uloženy údaje ze sondy
                                                      // HXB-85
} MERICI_USTREDNA;
```

Formát zasílaných zpráv a frekvence zasílání je uveden v Příloze A.

3.5.1. Seznam funkcí pro obsluhu modulu měřící ústředna

void nacteni_vstupu_AI_TC(uint8_t *p_data,uint16_t z[],uint8_t zacatek)

uint8_t stav_vstupu_TC(uint8_t *p_data)

void natceni_teploty_RTD(uint8_t *p_data,uint32_t *RTD1,uint32_t *RTD2)

uint8_stav_vstupu_RTD(CanFrame *p_pracovni)

void otackomer_CAN(uint8_t *p_data,uint32_t *p_otacky_motor_CAN)

void sonda_HXB_85(uint8_t *p_data,int16_t z[])

uint32_t nacti_celociselne_cislo (uint8_t *p_bajt,uint8_t *p_MSB,uint8_t pocet_bajtu)

3.5.2. Význam a popis jednotlivých funkcí

- void nacteni_vstupu_AI_TC(uint8_t *p_data,uint16_t z[], uint8_t zacatek)

Název funkce: nacteni_vstupu_AI_TC

Vstupní parametry: *p_data – ukazatel na první prvek v poli předané funkci

uint8_t zacatek – pozice od které se čísla budou ukládat do pole z[]

Výstupní parametry: z[] – ukazatel na pole, do kterého se ukládají složená čísla

Vrací: nemá návratovou hodnotu

Význam: Vytvoří z pole (složeno z 8 bajtů) 4 proměnné datového typu uint16_t, a tyto proměnné uloží do vybraného pole čísel. Proměnné předpokládá first LSB. Je nutno určit od jaké pozice v poli čísel se čísla začnou ukládat.

Poznámka: Údaj o analogových vstupech či teplotě termočlánků je zasílán v datové zprávě o obsahu osmi datových zpráv. Každý údaj je složen ze dvou bajtů.

Ukázka zdrojového kódu a příkladu použití je v části 3.5.3 .

- uint8_t stav_vstupu_TC(uint8_t *p_data)

Název funkce: stav_vstupu_TC

Vstupní parametry: *p_data – ukazatel na první prvek v poli předané funkci

Výstupní parametry: nemá

Vrací: návratovou hodnotu, datový typ uint8_t

Význam: Vrací první prvek v poli dat

Poznámka: Stav vstupů termočlánku je zasílán v datové zprávě o obsahu jednoho datového bajtu.

- void natceni_teploty_RTD(uint8_t *p_data,uint32_t *RTD1,uint32_t *RTD2)

Název funkce: natceni_teploty_RTD

Vstupní parametry: *p_data – ukazatel na první prvek v poli předané funkci

Výstupní parametry: uint32_t *RTD1 – ukazatel na proměnou RTD1, obsahuje složenou hodnotu teploty přečtenou z teploměru RTD, na vstupu RTD1

uint32_t *RTD2 – ukazatel na proměnou RTD2, obsahuje složenou hodnotu teploty přečtenou z teploměru RTD, na vstupu RTD2

Vrací: nemá návratovou hodnotu

Význam: Vytvoří z pole dat (složeno z 8 bajtů) 2 proměnné datového typu uint32_t . Proměnné předpokládá first LSB.

Poznámka: Údaj o teplotách je zasílán v datové zprávě o obsahu osmi datových bajtů.

- uint8_stav_vstupu_RTD(uint8_t z[]);

Název funkce: stav_vstupu_RTD

Vstupní parametry: z[] – pole ukazatelů na předané pole funkci

Výstupní parametry: nemá

Vrací: vrací 4. prvek z pole, předaného do funkce

Význam: Vyčte hodnotu ze 4. prvku pole, které je funkci předáno. Údaj o stavu vstupu RTD se přenáší ve čtvrtém datovém bajtu v přijaté datové CAN zprávě.

Poznámka:

- void otackomer_CAN(uint8_t *p_data,uint32_t *p_otacky_motor_CAN)

Název funkce: otackomer_CAN

Vstupní parametry: uint8_t *p_data – ukazatel na první prvek v poli předané funkci

Výstupní parametry: uint32_t *p_otacky_motor_CAN – ukazatel na proměnou, ve které jsou uloženy výsledné otáčky motoru.

Vrací: nemá návratovou hodnotu

Význam: Vytvoří z pole předaného funkci (složeno ze 4 bajtů) proměnnou datového typu uint32_t . Proměnnou předpokládá first LSB.

Poznámka: Údaj o otáčkách je zasílán v datové zprávě o obsahu čtyř datových bajtů.

- void sonda_HXB_85(uint8_t *p_data,int16_t z[])

Název funkce: sonda_HXB_85

Vstupní parametry: *p_data – ukazatel na první prvek v poli předané funkci

Výstupní parametry: z[] — ukazatel na pole předaného funkci, do kterého se ukládají složená čísla

Vrací: nemá návratovou hodnotu

Význam: Vytvoří z pole dat (složeno z 8 bajtů) 4 proměnné datového typu int16_t . Proměnnou předpokládá LSB – MSB.

Poznámka: Údaj ze sondy HXB_85 je zasílán v datové zprávě o obsahu osmi datových bajtů.

- uint32_t nacti_celociselne_cislo (uint8_t *p_bajt,uint8_t *p_MSB, uint8_t pocet_bajtu)

Název funkce: nacti_celociselne_cislo

Vstupní parametry: *p_bajt – ukazatel na první prvek v poli předané funkci

uint8_t *p_MSB – ukazatel na prvek v poli, který obsahuje MSB

uint8_t pocet_bajtu – počet bajtů, z kterého se skládá číslo

Výstupní parametry: nemá

Vrací: složené číslo, datového typu uint32_t

Význam: poskládá číslo o velikosti maximálně 4 bajty, vrací číslo datového typu uint_32 . Předpokládá číslo v přenášeném formátu LSB – MSB.

Poznámka: Podpůrná funkce pro funkce, které se zabývají zpracováním zpráv přijatých po CANu. Realizuje základní složení proměnných.

3.5.3. Ukázka

Zdrojový kód

```
void nacteni_vstupu_AI_TC(uint8_t *p_data,uint16_t z[],uint8_t zacatek)
{
  uint8_t i;
  uint8_t MSB = 2; // čísla jsou zasílána ve formátu first LSB
  for(i = zacatek;i<(4+zacatek);i++) // pouze čtyři čísla
  {
  z[i] = (uint16_t)nacti_celociselne_cislo (p_data,&MSB,2);
  MSB +=2;                          // MSB = 2,4,6,8
  }
}
```

Příklad použití (vykopírováno z těla funkce *void can_zpracovani_zpravy(void)*

```
 case analog_vstupy1_CAN_id:  // 0x17
        nacteni_vstupu_AI_TC(&frameW.data,&merici_ustredna.analogove_vstupy,0);
        SendArray_USB(&merici_ustredna.analogove_vstupy,16);
  break;
 case analog_vstupy2_CAN_id: // 0x18
        nacteni_vstupu_AI_TC(&frameW.data,&merici_ustredna.analogove_vstupy,4);
        SendArray_USB(&merici_ustredna.analogove_vstupy,16);
  break;
  case termoclanky_1_CAN_id:
   nacteni_vstupu_AI_TC(&frameW.data,&merici_ustredna.termoclankove_vstupy,0);
        SendArray_USB(&merici_ustredna.termoclankove_vstupy,16);
  break;
```

3.6. Vývojové diagramy

3.6.1. Vývojový diagram funkce nacteni_vstupu_AI_TC

*Obrázek 3-3 Vývojový diagram funkce void nacteni_vstupu_AI_TC(uint8_t *p_data,uint16_t z[], uint8_t zacatek).*

Na začátku funkce jsou předány vstupní parametry (*p_data – ukazatel na první prvek v poli dat, zacatek – od které pozice se má ukládat do pole, do něhož jsou ukládána výsledná

čísla a výstupní parametr z[] – pole ukazatelů na pole, do kterého se budou ukládat výsledná čísla.

Dále následuje inicializace lokálních proměnných. Pak je nutno nastavit, kde se nachází bajt s nejvyšší váhou (MSB). V této funkci je to dvojka, protože MSB se nachází ve druhém bajtu v přijaté CAN datové zprávě. Pak následuje cyklus for, ve kterém je volána funkce nacti_celociselne_cislo, která poskládá z přijatých bajtů číslo a její návratová hodnota je uložena do pole, kam se ukládají čísla. Cyklus probíhá 4x , protože jedna datová zpráva od CANu přenáší čtyři proměnné.

Ukládání čísel od dané pozice je uděláno tak, že řídící proměnná cyklu i je zároveň její hodnota použita pro index v poli, do kterého se ukládají výsledná čísla.

Pokud chci uložit daná čísla od začátku, tak řídící proměnná je inicializována hodnotou 0. A index nula je také zároveň první pozice v poli. Cyklus probíhá od i = 0 až do i = 3, tedy 4x . Výchozí inicializace je provedena přes vstupní parametr zacatek.

Pokud je třeba začít od pozice 4, tak řídící proměnná cyklu se inicializuje hodnotou 4.

A následně je tato hodnota využita pro index v poli. Cyklus probíhá od i = 4 až do i = 7.

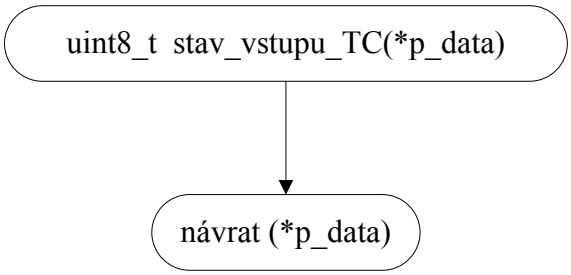
Tabulka 3-2 Formát přenášených zpráv po CANu analogové vstupy

CAN-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x11	AI4		AI3		AI2		AI1	

Tabulka 3-3 Formát přenášených zpráv po CANu termočlánkové vstupy

CAN-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x13	TC1		TC2		TC3		TC4	

3.6.2. Vývojový diagram funkce stav_vstupu_TC

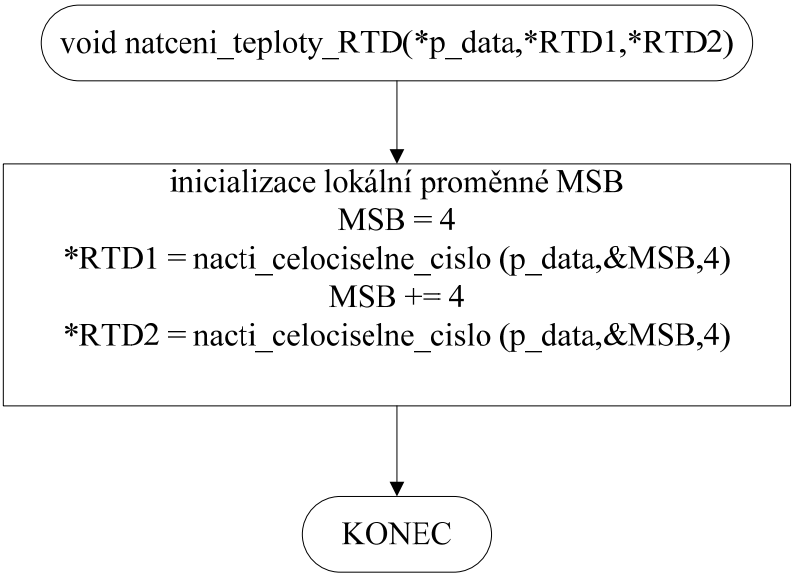


*Obrázek 3-4 Vývojový diagram funkce uint8_t stav_vstupu_TC(uint8_t *p_data)*

Tabulka 3-4 Formát přenášených zpráv po CANu, stav vstupu TC

CAN-ID 0x15	byte 0
	stav vstupů TC1 až TC8

3.6.3. Vývojový diagram funkce natceni_teploty_RTD



*Obrázek 3-5 Vývojový diagram funkce void natceni_teploty_RTD(uint8_t *p_data,uint32_t *RTD1,uint32_t *RTD2)*

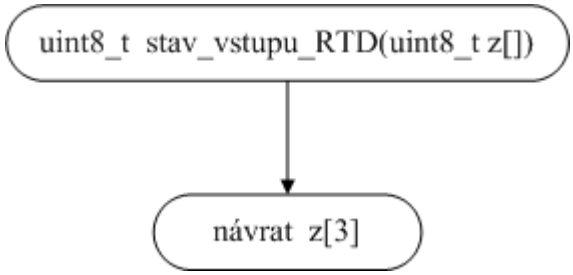
Nejprve jsou funkci předány vstupní parametry.

Pak je inicializována lokální proměnná MSB. Do této proměnné je přiřazeno číslo čtyři. Je to z toho důvodu, že MSB v přijaté zprávě od CANu je ve čtvrtém bajtu (první číslo). Druhé číslo má MSB v obsažené zprávě až v osmém bajtu. Po tomto přiřazení následuje složení čísla a následně zápis přes ukazatel do příslušné proměnné. Čísla jsou o velikosti čtyři bajty. Prvně následuje převod prvního čísla a až pak převod druhého čísla.

Tabulka 3-5 Formát přenášených zpráv po CANu, RTD1, RTD2

CAN-ID 0x16	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	RTD1				RTD2			

3.6.4. Vývojový diagram funkce uint8_t stav_vstupu_RTD(uint8_t z[])



Obrázek 3-6 Vývojový diagram funkce uint8_t stav_vstupu_RTD(uint8_t z[])

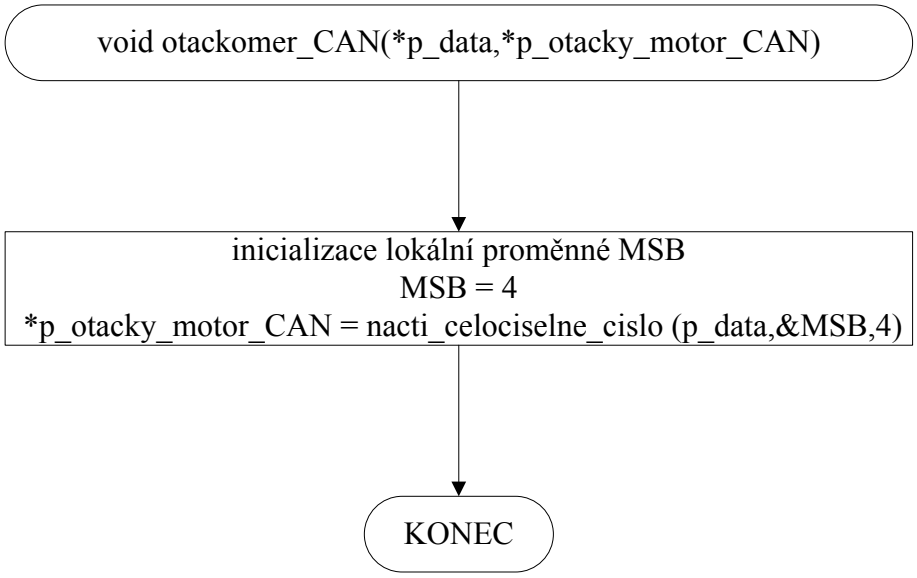
Tabulka 3-6 Formát přenášených zpráv po CANu, stav vstupu RTD1

CAN-ID 0x17	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
				stav vstupu RTD1				

Tabulka 3-7 Formát přenášených zpráv po CANu, stav vstupu RTD2

CAN-ID 0x19	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
				stav vstupu RTD2				

3.6.5. Vývojový diagram funkce otackomer_CAN



*Obrázek 3-7 Vývojový diagram funkce void otackomer_CAN(uint8_t *p_data,uint32_t *p_otacky_motor_CAN)*

Nejprve je inicializovaná proměnná MSB a přiřazena hodnota 4. V zasílané zprávě jsou otáčky motoru přenášeny ve čtyřech bajtech, v pořadí LSB – MSB. Dále je tato hodnota předána do funkce nacti_celociselne_cislo. Její návratová hodnota je použita pro přepsání hodnoty v proměnné, na kterou ukazuje ukazatel *p_otacky_motor_CAN. Návratová hodnota obsahuje již posílané otáčky.

Tabulka 3-8 Formát přenášených zpráv po CANu, otáčky

CAN-ID	Byte 0	Byte 1	Byte 2	Byte 3
0x12	Doba mezi dvěma posledními zápaly			

3.6.6. Vývojový diagram funkce sonda_HXB_85

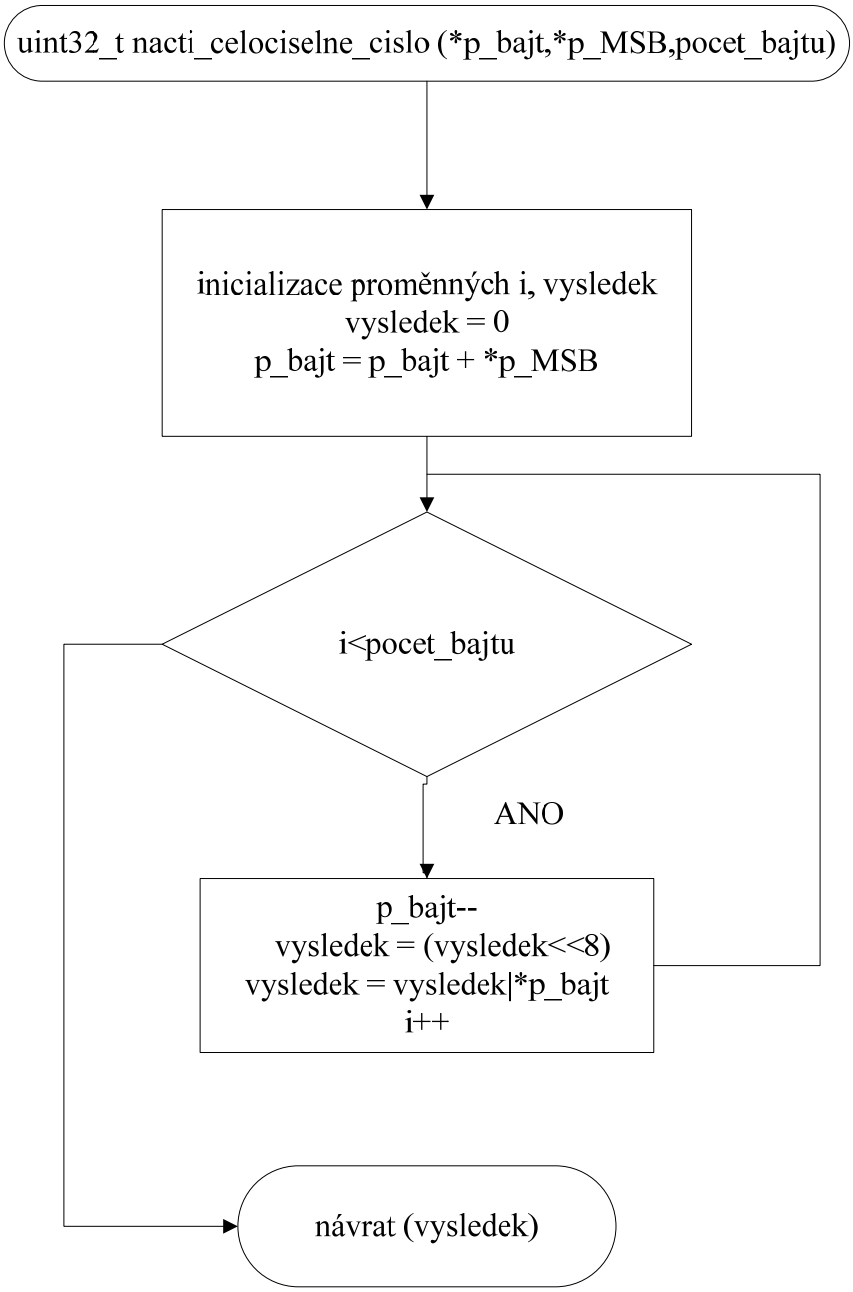
*Obrázek 3-8 Vývojový diagram funkce void sonda_HXB_85(uint8_t *p_data,int16_t z[])*

Nejprve je inicializována lokální proměnná MSB a je do ní přiřazena hodnota 2. Pak následuje cyklus for, ve kterém jsou složena a uložena jednotlivá čísla. Tento cyklus proběhne 4x, protože jsou zasílaná 4 čísla ze sondy HXB_85. Převedená čísla jsou uložena do pole, na něhož ukazuje pole ukazatelů z. Posun indexu MSB je o dva, protože MSB jednotlivých čísel se nacházejí ve druhém, čtvrtém, šestém a osmém bajtu v datové zprávě zaslané po CANu.

Tabulka 3-9 Formát přenášených zpráv po CANu, sonda HXB-85

CAN-ID 0x1B	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	Rosný bod		Teplota		Tlak		Relativní vlhkost	

3.6.7. Vývojový diagram funkce nacti_celociselne_cislo



*Obrázek 3-9 Vývojový diagram funkce uint32_t nacti_celociselne_cislo (uint8_t *p_bajt,uint8_t *p_MSB,uint8_t pocet_bajtu)*

Vzhledem, že tato funkce je páteřní pro všechny funkce, tak bude vysvětlena podrobně její činnost.

Vysvětlení funkce:

Nechť je přijata zpráva po CANu, která má osm datových bajtů a jednotlivé datové bajty obsahují následující čísla viz tabulka 3-10.

Dále je uložena v mikrokontroléru do paměti. Každý uložený bajt se ukládá na předem danou vyhrazenou adresu v paměti. Adresa je dána při inicializaci proměnné, do které jsou tyto bajty ukládány. Vzhledem k tomu, že je přijímána datová zpráva o osmi bajtech, tak toto je vhodné uložit do pole čísel. Díky tomu jsou jednotlivé bajty uloženy v paměti za sebou a tato skutečnost se pak využije efektivně pro převod jednotlivých proměnných.

Vzhledem k tomu, že konkrétní adresy, do kterých se ukládají data, jsou neznámé a nejsou důležité pro napsání programu, ale pro tento příklad již nabývají významu, tak jsou vymyšlená.

Dále pro příklad budeme uvažovat pouze čísla, která se skládají ze dvou bajtů. Budeme převádět pouze jedno číslo, protože jednotlivé kroky se jenom opakují.

Tabulka 3-10 Přijatá zpráva po CANu

pořadí přij. bajtu	0.	1.	2.	3.	4.	5.	6.	7.
adresa bajtu hexadecimálně	A	B	C	D	E	F	10	11
číselné hodnoty jednotlivých bajtů								
hexadecimálně	78	AC	0	0	0	0	0	0
binárně	01111000	10101100	0	0	0	0	0	0
každý bajt má 8 bitů, pro úsporu místa je napsána pouze jedna 0 u binárního vyjádření čísla 0								

Činnost funkce

Nejdříve jsou načteny vstupní parametry.

*p_bajt - ukazatel na první prvek v poli dat.

Hodnota v ukazateli je rovna *p_bajt $= (A)_{16}$. Adresa prvního prvku v poli dat.

Dále *p_MSB je přiřazena adresa proměnné, ve které je uložena pozice MSB. Například proměnná je na adrese $(CD)_{16}$. Na této adrese musí být uloženo číslo 2, protože číslo je přijato ve formátu first LSB a čísla jsou dvoubajtová.

Je nutno funkci předat počet bajtů, z kterého se skládá číslo. V našem případě uložíme 2. Číslo je složeno ze dvou bajtů.

Dále následuje inicializace lokálních proměnných vysledek a i. V proměnné vysledek je ukládáno výsledné složené číslo. Proměnná i má význam řídící proměnné následujícího cyklu.

i = 0

vysledek =(0)$_{10}$ = (0)$_{16}$ = (00000000 00000000 00000000 00000000)$_2$

Dále je přesunut ukazatel na adresu, kde se nachází v poli dat MSB.

p_bajt = A + 2 = C (hexadecimálně)

Tedy ukazatel se sice přesune až za pozici, kde je uložen MSB, ale v cyklu hned odečítáme jedničku, tedy na již správné MSB. Musíme to takhle udělat, abychom se mohli posouvat v cyklu vždy na předchozí adresu. A funkce byla co nejjednodušší.

Pak následuje cyklus. Cyklus má tolik opakování, podle toho jaké číslo obsahuje proměnná pocet_bajtu. V našem příkladě pocet_bajtu = 2, takže cyklus se bude opakovat dvakrát.

V cyklu se provádějí níže uvedené operace:

- Ukazatel je přesunut na předchozí adresu.

  p_bajt = C – 1 = B (hodnota na adrese B, hodnota = (AC)$_{16}$

- V proměnné vysledek se provede posun 8 bitů doleva. Tím je uvolněno místo pro přijatý bajt.

  vysledek = (0)$_{10}$ = (0)$_{16}$ = (00000000 00000000 00000000 00000000)$_2$

- Do proměnné výsledek je načtena hodnota, na kterou ukazuje ukazatel p_bajt.

  vysledek = 0 + AC = AC (hexadecimálně)

  vysledek = (172)$_{10}$ = (00000000 00000000 00000000 101011000)$_2$

Tyto výše uvedené kroky se opakují ještě jednou, protože je zadán počet opakování 2x.

- Ukazatel je přesunut na předchozí adresu.

  p_bajt = B – 1 = B (hodnota na adrese A, hodnota = (78)$_{16}$

- V proměnné vysledek se provede 8 bitů doleva. Tím je uvolněno místo pro přijatý bajt.

vysledek = (44032)$_{10}$ = (AC00)$_{16}$ = (00000000 00000000 10101100 00000000)$_2$

- Do proměnné výsledek je načtena hodnota, na kterou ukazuje ukazatel p_bajt.

  vysledek = AC00 + 78 = AC78 (hexadecimálně)

  vysledek = (44152)$_{10}$ = (00000000 00000000 101011000 0111 1000)$_2$

Operace je provedena dvakrát, tak je ukončen cyklus. Ve zprávě je obsaženo číslo ve dvou bajtech v pořadífirst LSB. Jsou zpracovány dva bajty, to je v pořádku.

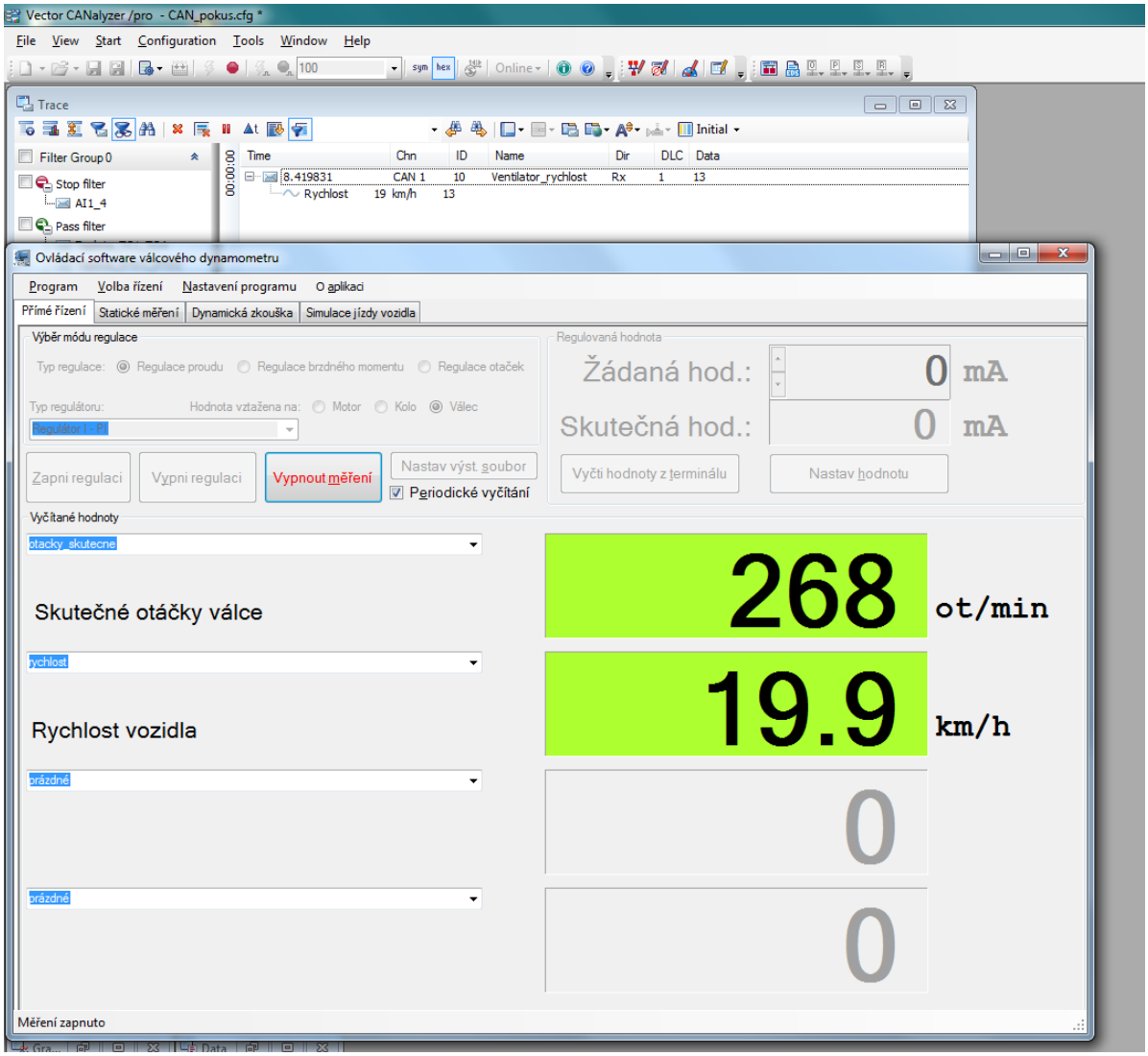
Dále je ukončena funkce a přes její návratovou hodnotu je poskytnuta hodnota proměnné vysledek ostatním funkcím.

4. OVĚŘENÍ FUNKCE

Funkce terminálu, který běží na mikrokontroléru MC9S12XEP100, byla porovnána s funkcí terminálu, který běží na platformě Amit (mikrokontrolér 80C166). Pokud se naměřená data shodují, tak tím je prokázána funkčnost terminálu.

Funkce sběrnice CAN byla prokázána tím, že byly posílány zprávy po CANu do terminálu a terminál přijímaná data odesílal po sériové lince do PC.

4.1. Komunikace s modulem ventilátoru



Obrázek 4-1 Test komunikace po sběrnici CAN s ventilátorem

Na obrázek 4-1 lze vidět oskenované okno programu Vector CANalyzer, který zobrazuje zprávy, které jsou posílány po sběrnici CAN. Pod tímto oknem se nachází okno programu OSVD.

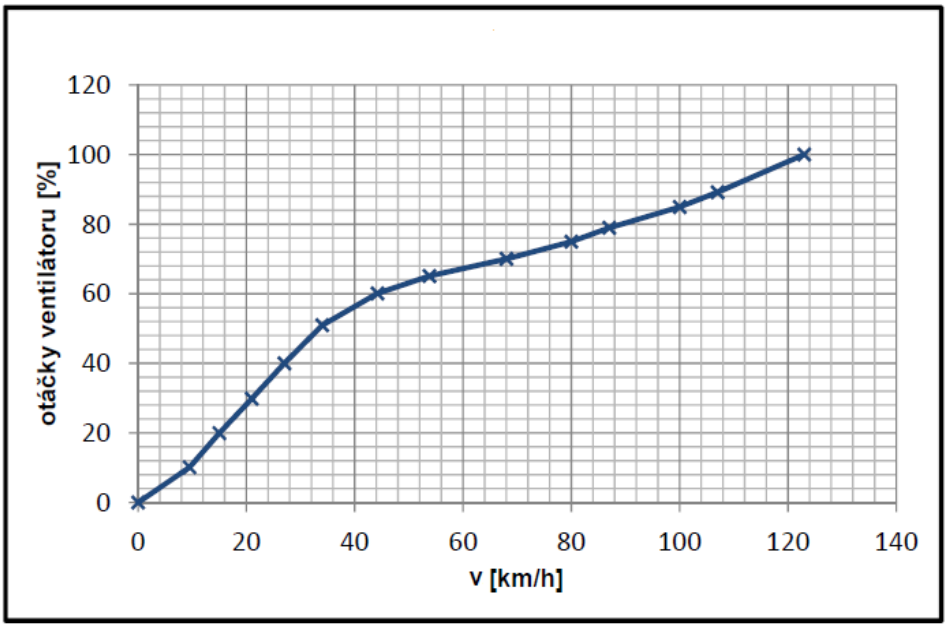
Porovnáním obsahu zprávy zaslané po CANu a výpisu z programu OSVD můžeme potvrdit, že řídící terminál zasílá informaci o rychlosti správně. Rychlosti se shodují. Vzhledem k tomu, že modul ventilátoru zpracovává pouze osmibitová celočíselná čísla, tak desetinná část je z posílané rychlosti vynechána. Zasílá se pouze celé číslo.

Z výše uvedeného lze určit, že nejvyšší posílaná rychlost může nabývat hodnoty 255. Platí $(255)_{10} = (FF)_{16} = (\ 1111\ 1111\ )_2$.

Modul ventilátoru ovládá frekvenční měnič, který ovládá motor ventilátoru.

Tabulka 4-1 Otáčky ventilátoru v závislosti na rychlosti

rychlost [km/h]	frekvence [Hz]	otevření měniče [%]
19,9	12,95	25,9
59,9	33,53	67,06
99,9	42,04	84,08
Při frekvenci 50 Hz je měnič otevřen na 100%.		



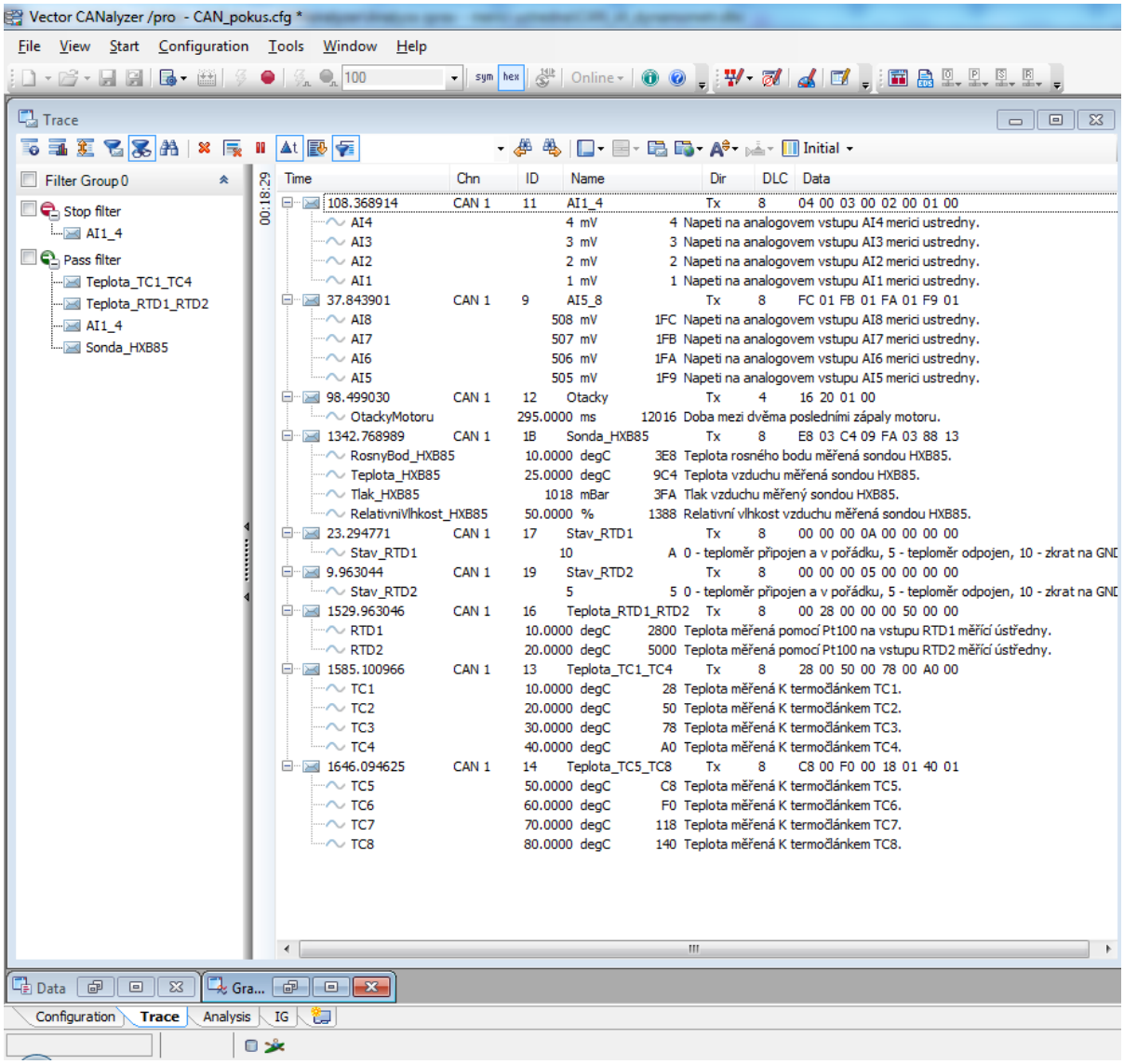
Obrázek 4-2 Závislost otáček ventilátoru na rychlosti motorky či motoru (převzato z literatury [5])

Vzhledem k tomu, že naměřené hodnoty na měniči, který ovládá samotný motor ventilátoru viz tabulka 4-1, se shodují s teoretickými předpoklady viz obrázek 4-2, tak tím je prokázáno, že modul ventilátoru správně vyčítá zprávy, které jsou posílány po sběrnici CAN.

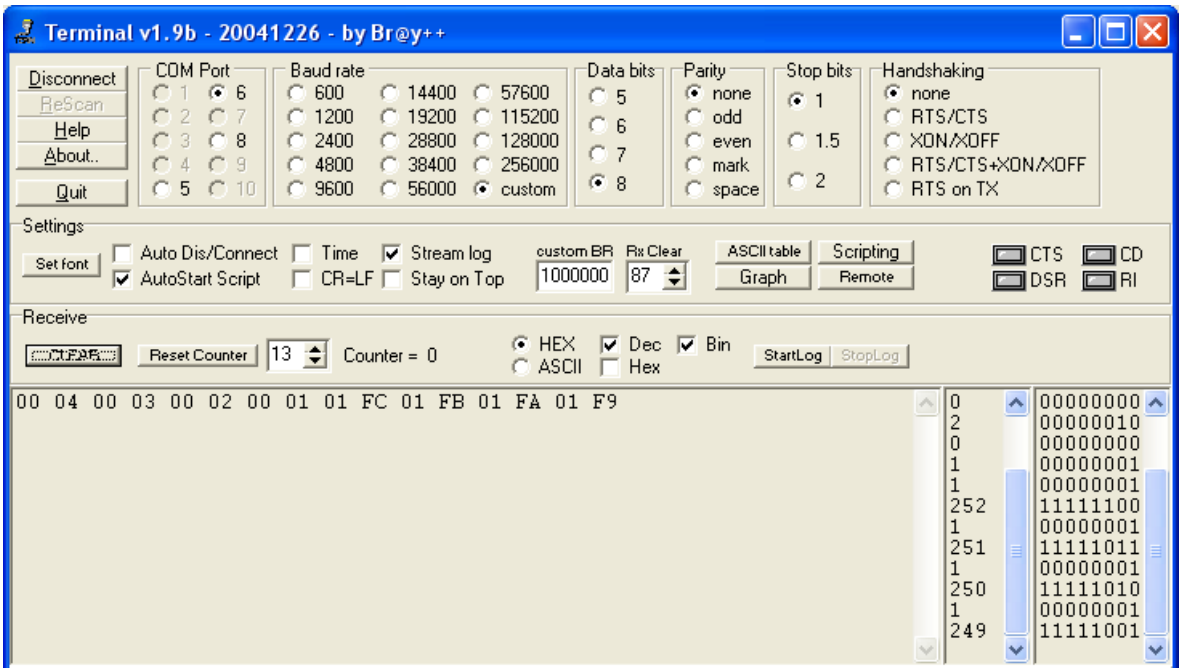
4.2. Komunikace s modulem měřící ústředna

Pro zkoušení funkce s modulem měřící ústředna byl použit simulační software běžící na PC. Náhrada umožňuje například možnost zasílání vybraných zpráv a tím snadnější zjištění a opravení následných chyb v programu řídícího terminálu.

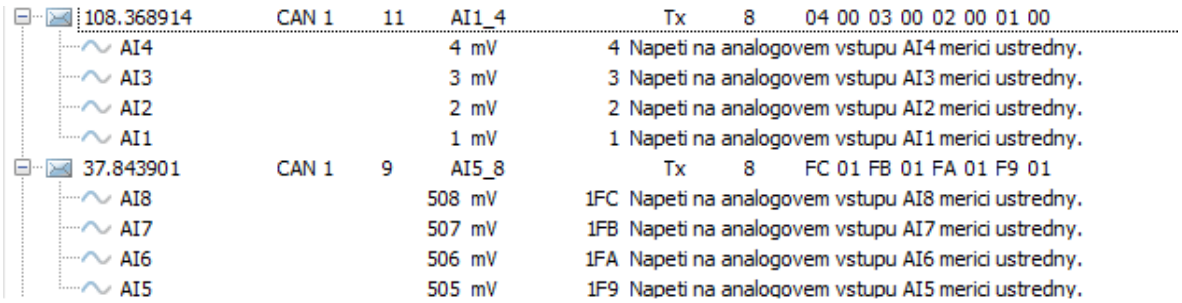
Dále simulační program nahradil fyzickou nepřítomnost modulu měřící ústředna, protože v době zkoušek se nacházela mimo laboratoře KEEZ.



Obrázek 4-3 Výpis zasílaných zpráv po CANu



Obrázek 4-4 Okno programu Terminal - Analogové vstupy ai1 až ai8



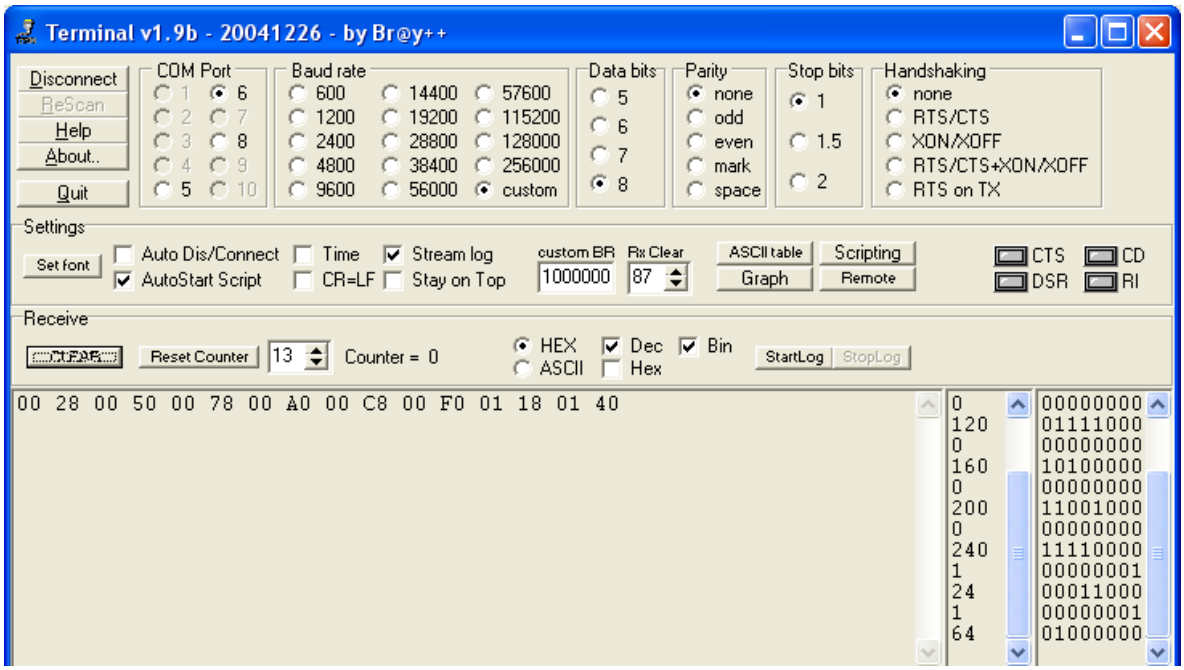
Obrázek 4-5 Výřez z programu Vector CANalyzer, analogové vstupy

Pokud se podíváme na obrázek 4-5 a obrázek 4-4, tak zjistíme, že odesílaná data v odeslané zprávě a v přijatých bajtech po sériové lince (program Terminal pro sledování komunikace na sériové lince v PC) se shodují. Tím je prokázáno, že řídící terminál přijatou zprávu po CANu zpracoval.

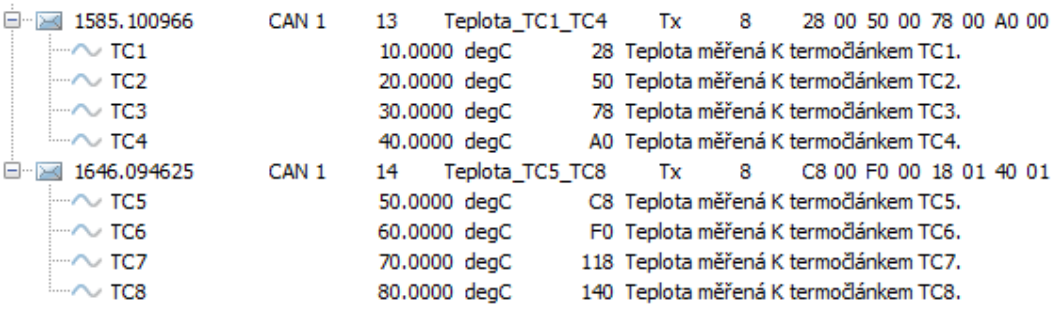
Čísla (datový typ unsigned int) jsou v řídícím terminálu převedena do formátu MSB – LSB. A poté následně uložena do pole, ve kterém jsou ukládány hodnoty analogových vstupů viz tabulka 4-2.Ve zprávě posílané po CANu je to ve formátu first LSB. Podrobnější informace o zasílaných zprávách jsou uvedené v Příloze A.

Tabulka 4-2Formát pole (merici_ustredna.analogove_vstupy[8])

ai4 (16bit)	ai3	ai2	ai1	ai8	ai7	ai6	ai5



Obrázek 4-6 Okno programu Terminal - termočlánkové vstupy tc1 až tc8



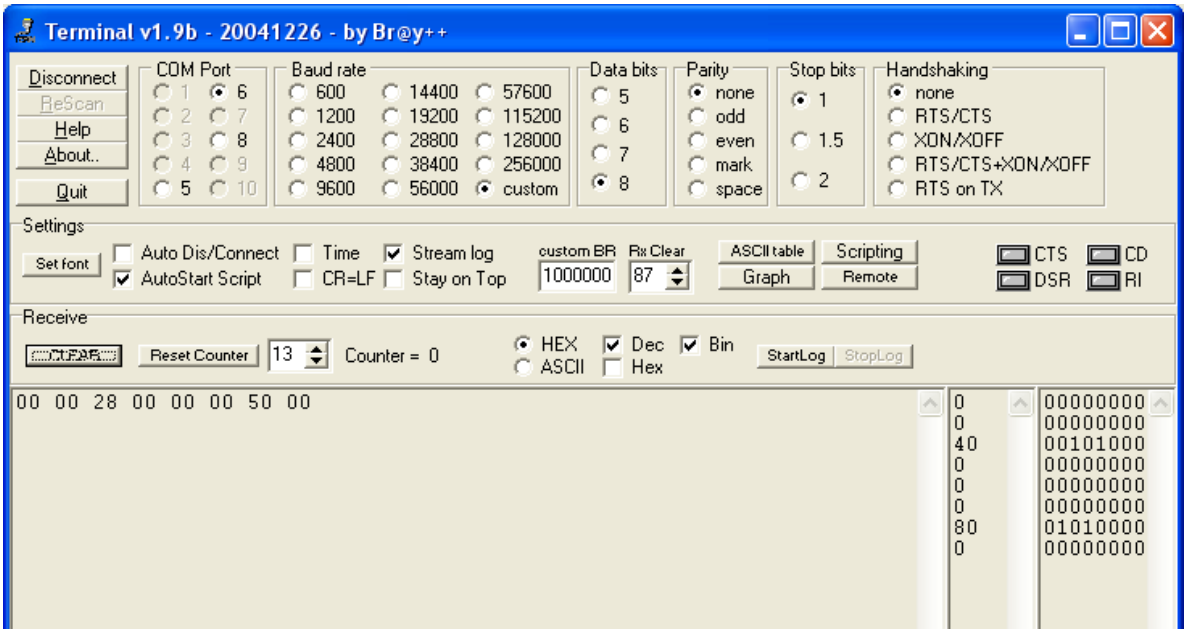
Obrázek 4-7 Výřez z programu Vector CANalyzer, termočlánkové vstupy

Pokud se podíváme na obrázek 4-6 a obrázek 4-7, tak zjistíme, že odesílaná data v odeslané zprávě a v přijatých bajtech po sériové lince (program Terminal pro sledování komunikace na sériové lince v PC) se shodují. Tím je prokázáno, že řídící terminál přijatou zprávu po CANu zpracoval.

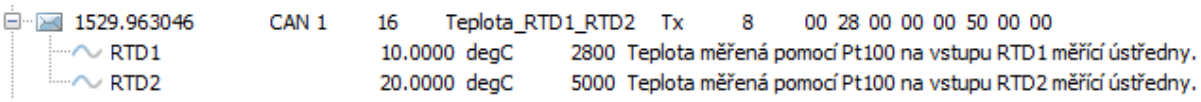
Čísla (datový typ unsigned int) jsou v řídícím terminálu převedena do formátu MSB – LSB. A poté následně uložena do pole, ve kterém jsou ukládány hodnoty termočlánkových vstupů viz tabulka 4-3. Ve zprávě posílané po CANu je to ve formátu first LSB. Podrobnější informace o zasílaných zprávách jsou uvedené v Příloze A.

Tabulka 4-3 Formát pole (merici_ustredna.termoclankove_vstupy[8])

tc1(16bit)	tc2	tc3	tc4	tc5	tc6	tc7	tc8



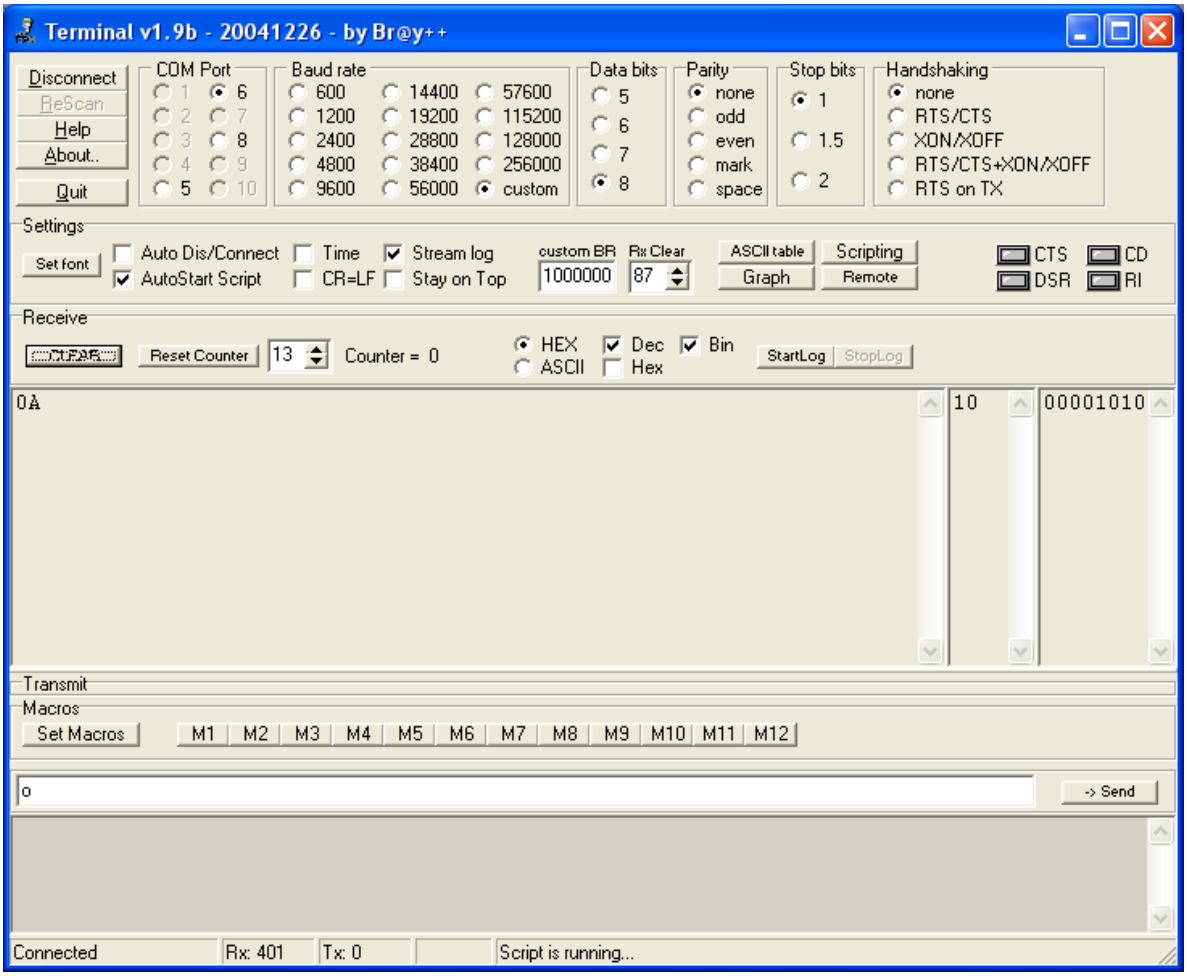
Obrázek 4-8 teplota RTD1, RTD2



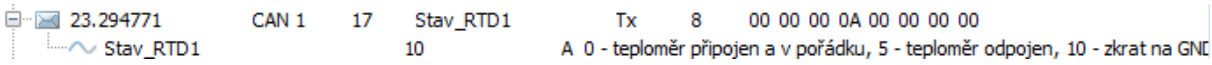
Obrázek 4-9 Výřez z programu Vector CANalyzer, platinové teploměry

Pokud se podíváme na obrázek 4-8 a obrázek 4-9, tak zjistíme, že odesílaná data v odeslané zprávě a v přijatých bajtech po sériové lince (program Terminal pro sledování komunikace na sériové lince v PC) se shodují. Tím je prokázáno, že řídící terminál přijatou zprávu po CANu zpracoval.

Čísla jsou v řídícím terminálu převedena do formátu MSB – LSB. A poté následně uložena do dvou proměnných (*unsigned long RTD1* a *unsigned long RTD2,* 32bit čísla, součást struktury merici_ustredna). Ve zprávě posílané po CANu je to ve formátu first LSB. Podrobnější informace o zasílaných zprávách jsou uvedené v Příloze A.



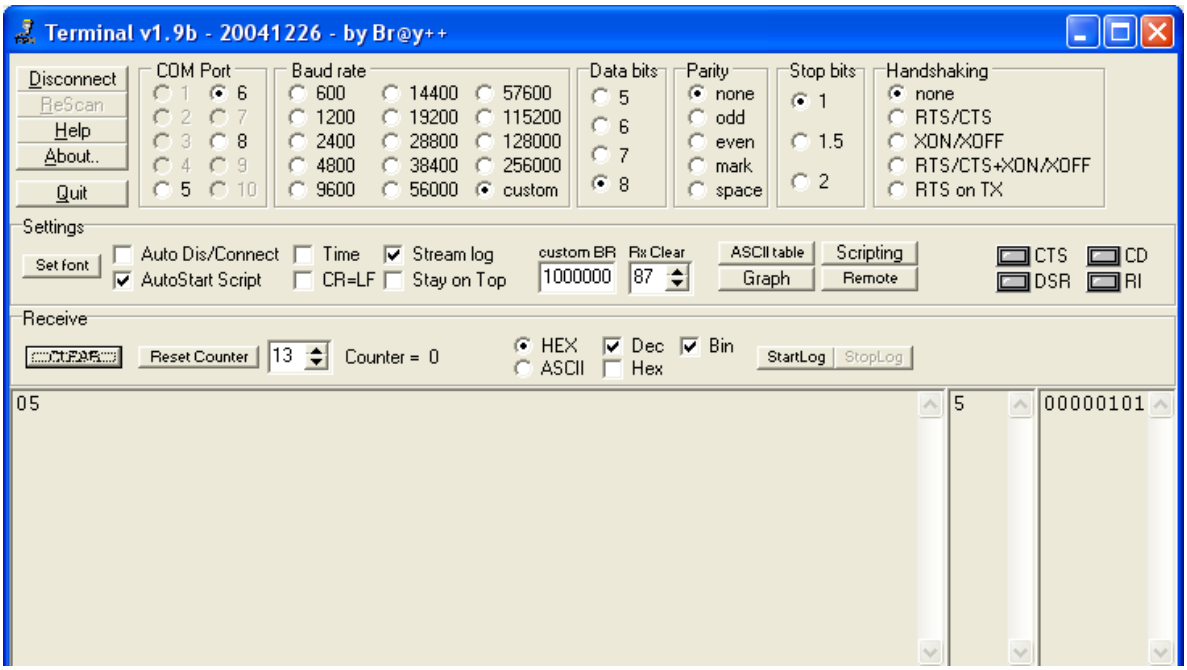
Obrázek 4-10 stav RTD1



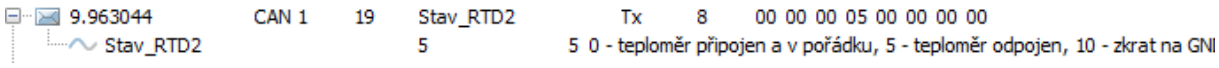
Obrázek 4-11 Výřez z programu Vector CANalyzer, stav RTD1

Pokud se podíváme na obrázek 4-10 a obrázek 4-11, tak zjistíme, že odesílaná data v odeslané zprávě a v přijatých bajtech po sériové lince (program Terminal pro sledování komunikace na sériové lince v PC) se shodují. Tím je prokázáno, že řídící terminál přijatou zprávu po CANu zpracoval.

Řídící terminál zpracovává čtvrtý zaslaný datový bajt v přijaté zprávě. Číslo se ukládá do proměnné *unsigned char stav_vstupu_RTD1* (8bit číslo, součást struktury merici_ustredna).



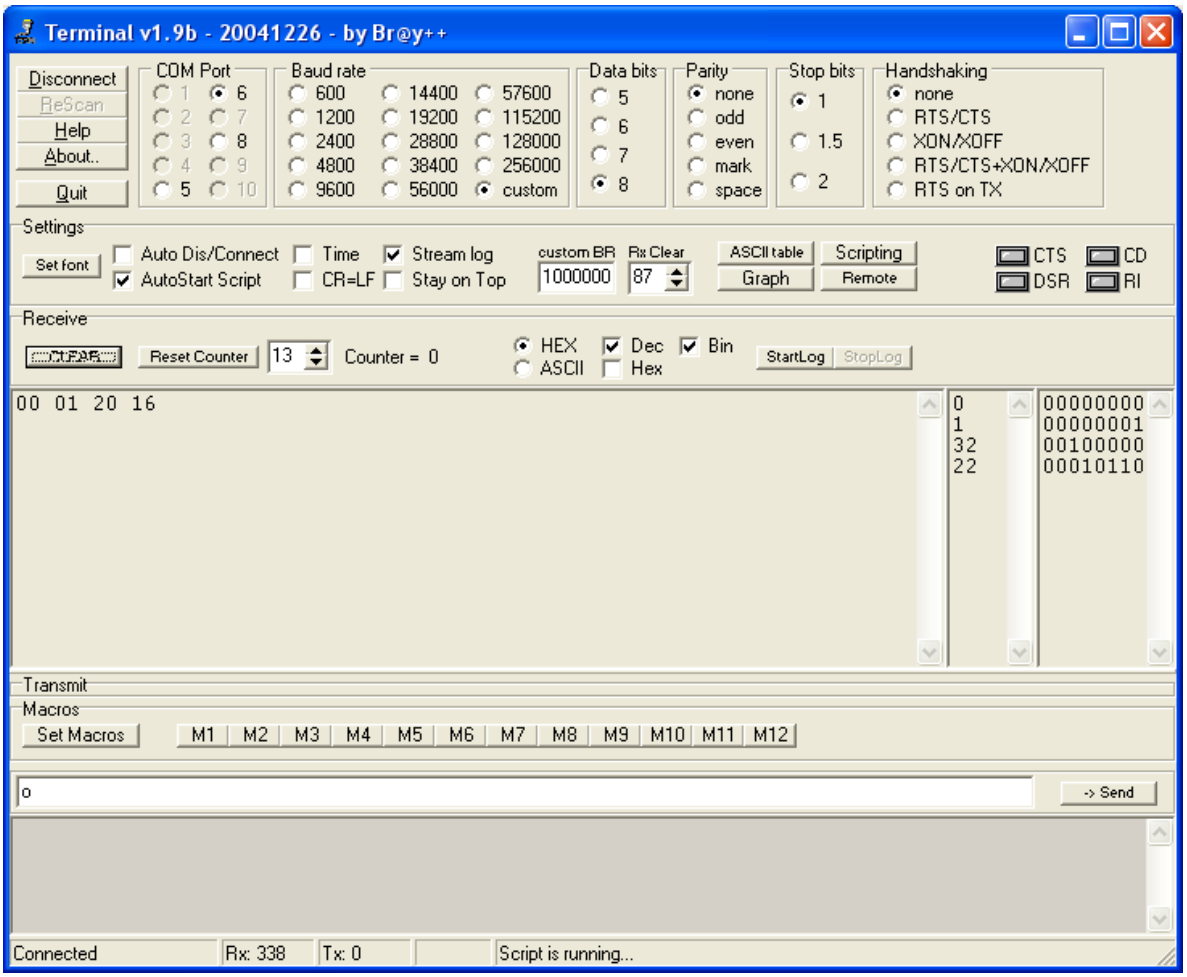
Obrázek 4-12 stav RTD2



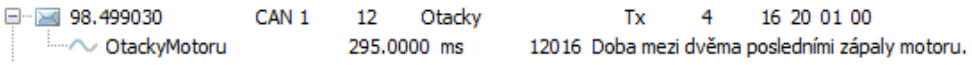
Obrázek 4-13 Výřez z programu Vector CANalyzer, stav RTD2

Pokud se podíváme na obrázek 4-12 a obrázek 4-13, tak zjistíme, že odesílaná data v odeslané zprávě a v přijatých bajtech po sériové lince (program Terminal pro sledování komunikace na sériové lince v PC) se shodují. Tím je prokázáno, že řídící terminál přijatou zprávu po CANu zpracoval.

Řídící terminál zpracovává čtvrtý zaslaný datový bajt v přijaté zprávě. Číslo se ukládá do proměnné *unsigned char stav_vstupu_RTD2* (8bit číslo, součást struktury merici_ustredna).



Obrázek 4-14 otáčky

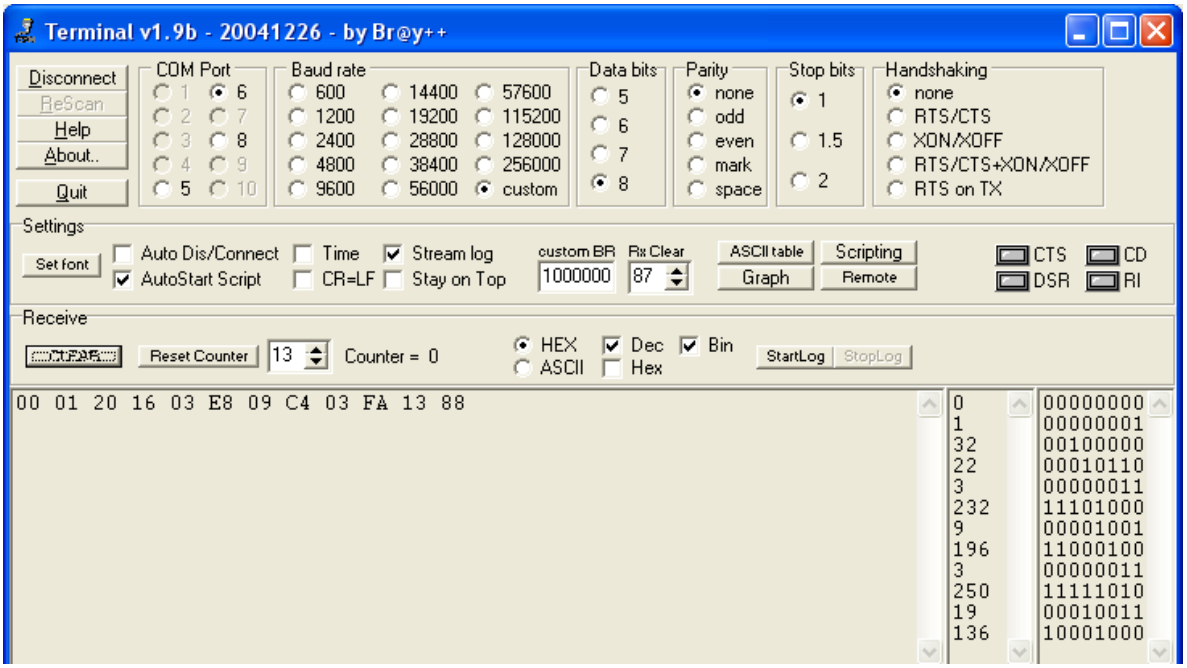


Obrázek 4-15 Výřez z programu Vector CANalyzer, otáčky

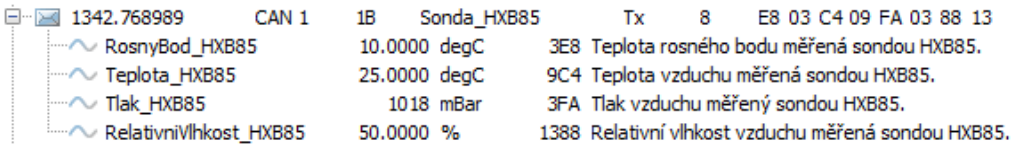
Pokud se podíváme na obrázek 4-14 a obrázek 4-15, tak zjistíme, že odesílaná data v odeslané zprávě a v přijatých bajtech po sériové lince (program Terminal pro sledování komunikace na sériové lince v PC) se shodují. Tím je prokázáno, že řídící terminál přijatou zprávu po CANu zpracoval.

Číslo je v řídícím terminálu převedena do formátu MSB – LSB. Ve zprávě posílané po CANu je to ve formátu first LSB.

Číslo se ukládá do proměnné *unsigned long int otacky_motor_CAN* (32bit číslo, součást struktury merici_ustredna).



Obrázek 4-16 Sonda HXB-85



Obrázek 4-17 Výřez z programu Vector CANalyzer, Sonda HXB-85

Pokud se podíváme na obrázek 4-16 a obrázek 4-17, tak zjistíme, že odesílaná data v odeslané zprávě a v přijatých bajtech po sériové lince (program Terminal pro sledování komunikace na sériové lince v PC) se shodují. Tím je prokázáno, že řídící terminál přijatou zprávu po CANu zpracoval.

Čísla (datový typ signed int, 16bit) jsou v řídícím terminálu převedena do formátu MSB – LSB. A poté následně uložena do pole, ve kterém jsou ukládány hodnoty ze sondy HXB-85 viz tabulka 4-4. Ve zprávě posílané po CANu je to ve formátu first LSB.

Tabulka 4-4 Formát pole (merici_ustredna.sonda_HXB_85_vstupy)

rosný bod(16 bit)	teplota	tlak	relativní vlhkost

4.3. Ověření funkce snímače momentu

Činnost snímače momentu byla prověřena tak, že tenzometrický snímač byl demontován z brzdy a postupně bylo na něj zavěšováno závaží. Změřené údaje byly vyčítány z programu OSVD.

Tabulka 4-5 Naměřené údaje ze snímače momentu

zátěž	terminál (MC9S12XEP100)	Amit (80C166)
	AD_moment [Nm]	AD_moment [Nm]
baterie PB	33,4	33,6
kanystr s olejem	19,2	19,3
autotransformátor	110,2	110,3

Naměřené údaje ze snímače momentu, změřené novým terminálem se shodují s naměřenými hodnotami terminálu na platformě Amit. Tím je prokázána funkce snímače momentu.

4.4. Ověření funkce regulátorů

Program obsahuje regulátory proudu, momentu a otáček. Regulátor otáček je nadřazen všem regulátorům. Regulátor proudu je naopak podřízen všem a jeho výstupní hodnota slouží pro nastavení střídy generované PWM, podrobněji viz obrázek 4-18 .

Z tohoto důvodu je vhodné prověřit funkci regulátoru proudu a až pak následně ostatních regulátorů.

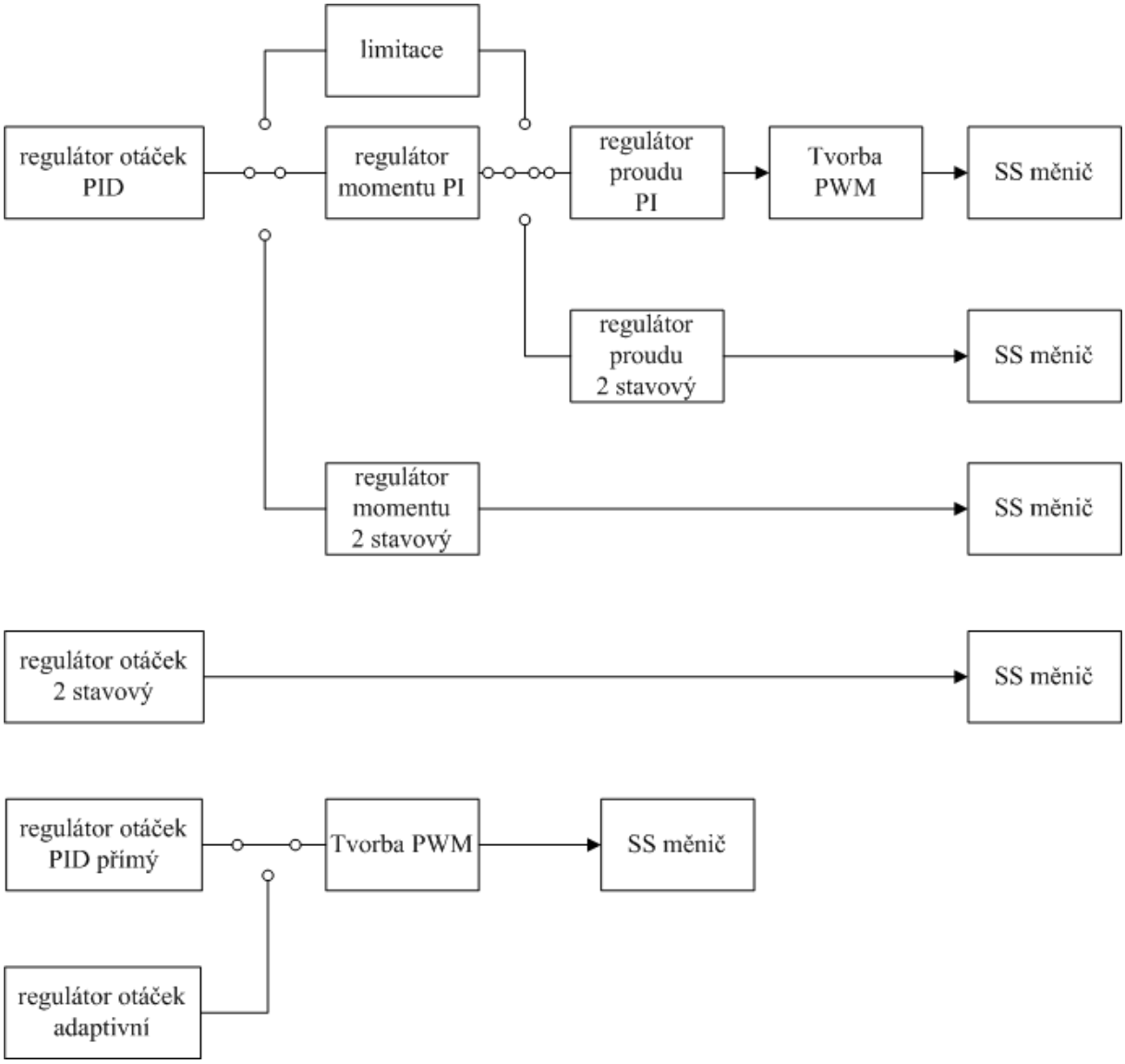
Dle požadavku vedoucího práce byly vyzkoušeny pouze funkce regulátorů – regulátor otáček PID, regulátor momentu PI a regulátor proudu PI .

4.4.1. Nastavené konstanty regulátorů

regulátor proudu PI Kp = 4, Ti = 250 ms

regulátor momentu PI Kp = 6, Ti = 500 ms

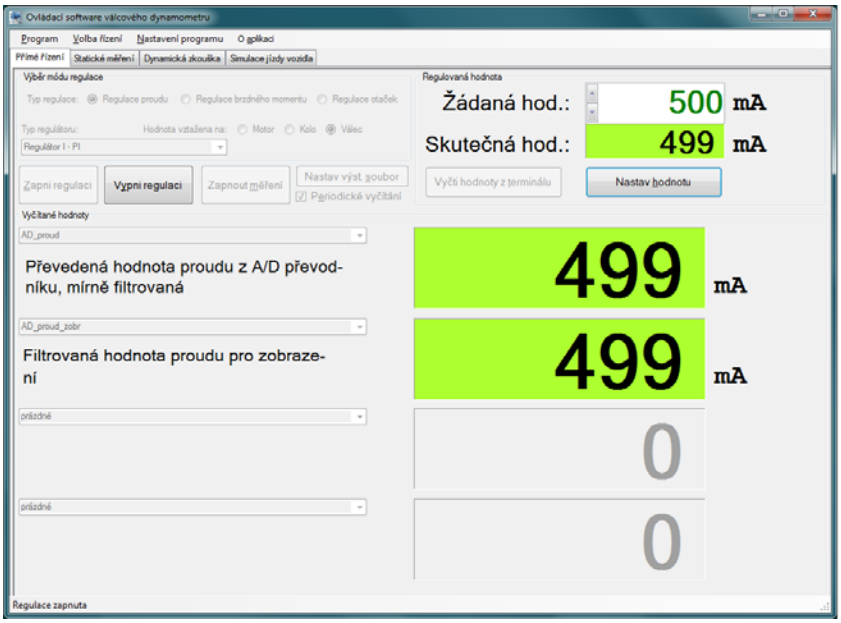
regulátor otáček PID Kp = 8, Ti = 3000 ms, Td = 600 ms



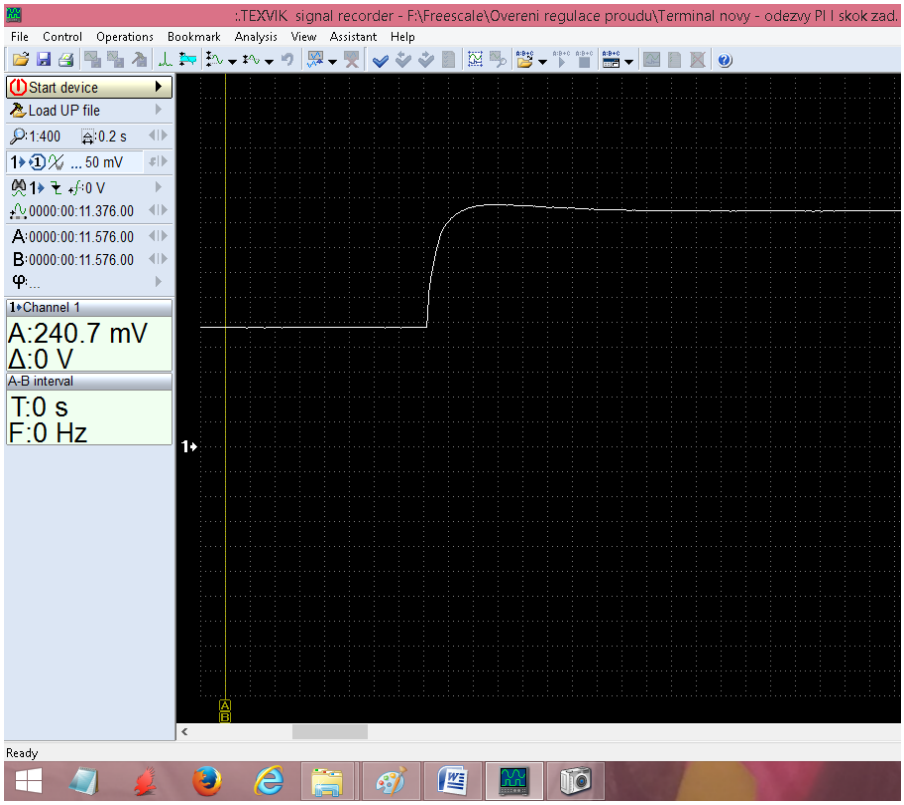
Obrázek 4-18 kaskáda regulátorů, regulace otáček

Pro podrobnější tok informací viz literatura [3].

4.4.2. PI regulátor proudu



Obrázek 4-19 Výpis z programu OSVD



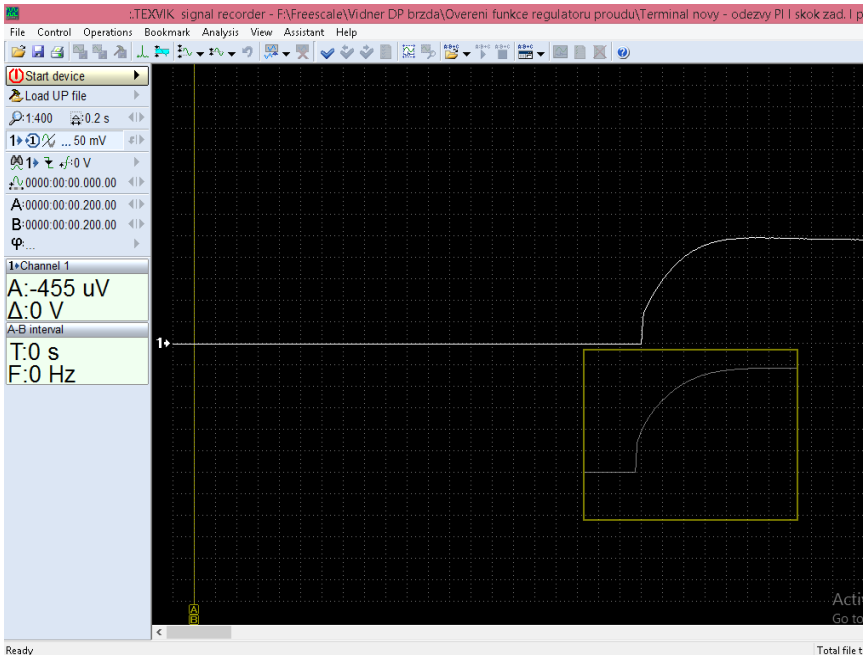
Obrázek 4-20 Záznam průběhu proudu (rozsah 100mA/dílek)

Proud nastavený v OSVD se shoduje s naměřeným proudem protékajícím budícím vinutím. Tím je prokázána funkce a možnost dálkového ovládání regulátoru proudu z nadřazeného PC.

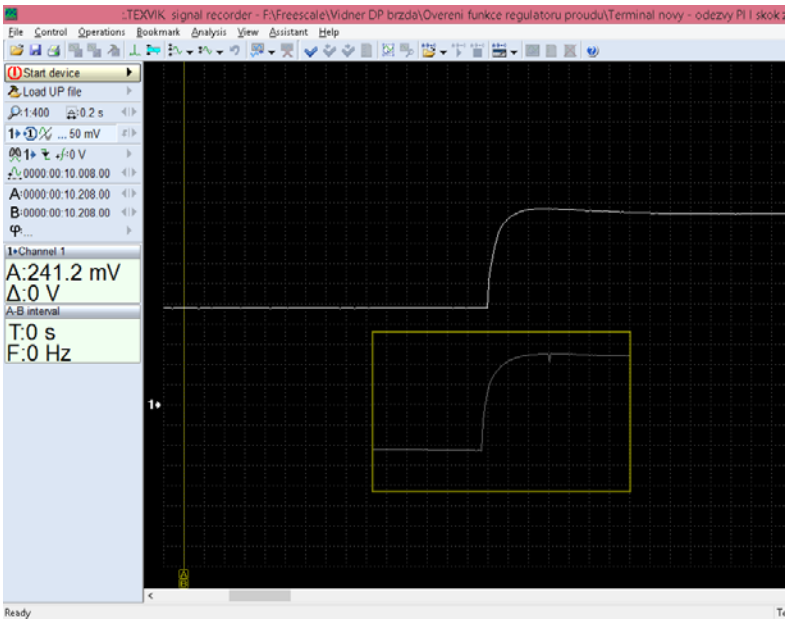
4.4.3. Odezva regulátoru proudu na skok žádané hodnoty

Nastavené konstanty regulátoru Kp = 4 a Ti = 250 ms.

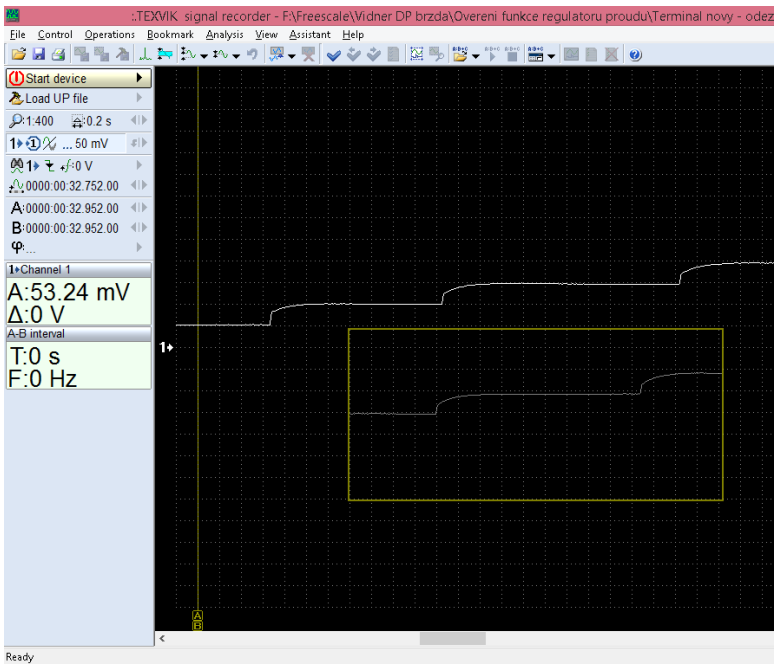
Na následujících obrázkách jsou odezvy zobrazeny podle následujícího pravidla. Horní průběh patří odezvě regulátoru proudu nového terminálu, průběh dole – odezva regulátoru proudu původního terminálu (terminál na platformě Amit).



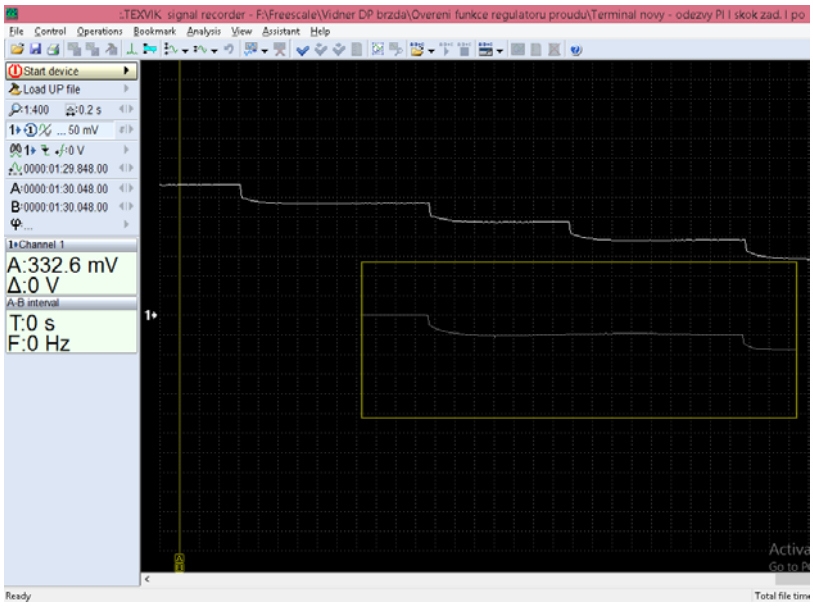
Obrázek 4-21 Porovnání odezev regulátorů proudu na skok žádané hodnoty z 0A na 0,5A (100mA/dílek, časová osa 0,2s)



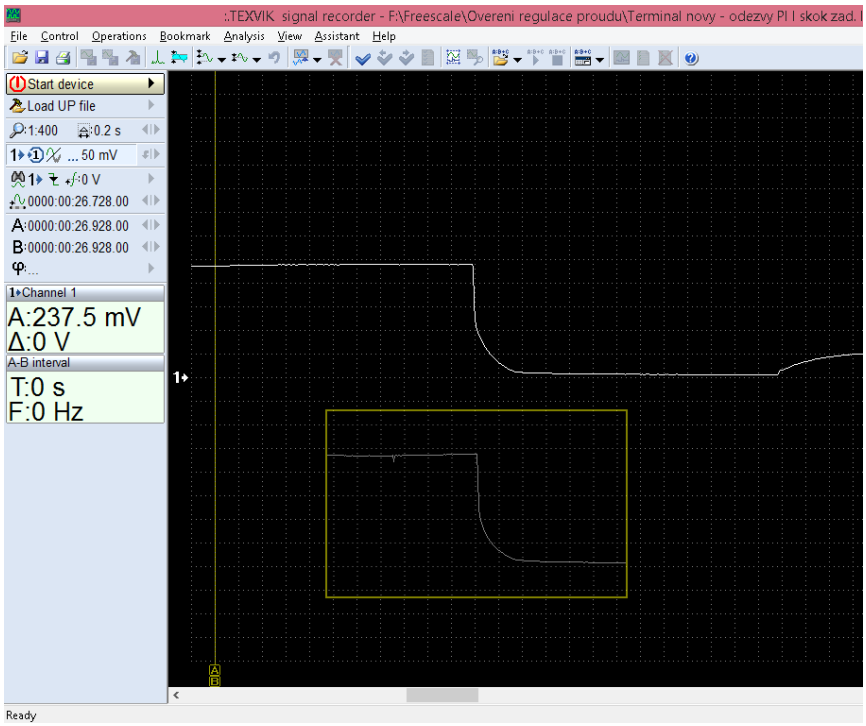
Obrázek 4-22 Porovnání odezev regulátorů proudu na skok žádané hodnoty z 0,5A na 1A (100mA/dílek, časová osa 0,2s)



Obrázek 4-23 Porovnání odezev regulátorů proudu na skok žádané hodnoty po krocích 0,1A (100mA/dílek, časová osa 0,2s).



Obrázek 4-24 Porovnání odezev regulátorů proudu na skok žádané hodnoty po krocích 0,1A (100mA/dílek, časová osa 0,2s).



Obrázek 4-25 Porovnání odezev regulátorů proudu na skok žádané hodnoty z 0,5A na 0A (100mA/dílek, časová osa 0,2s).

Odezvy regulátoru proudu (nový terminál, mikrokontrolér MC9S12XEP100) souhlasí s odezvami regulátoru proudu (platforma Amit, mikrokontrolér 80C167). Tím je prokázána funkčnost regulátoru.

4.4.4. Statické měření

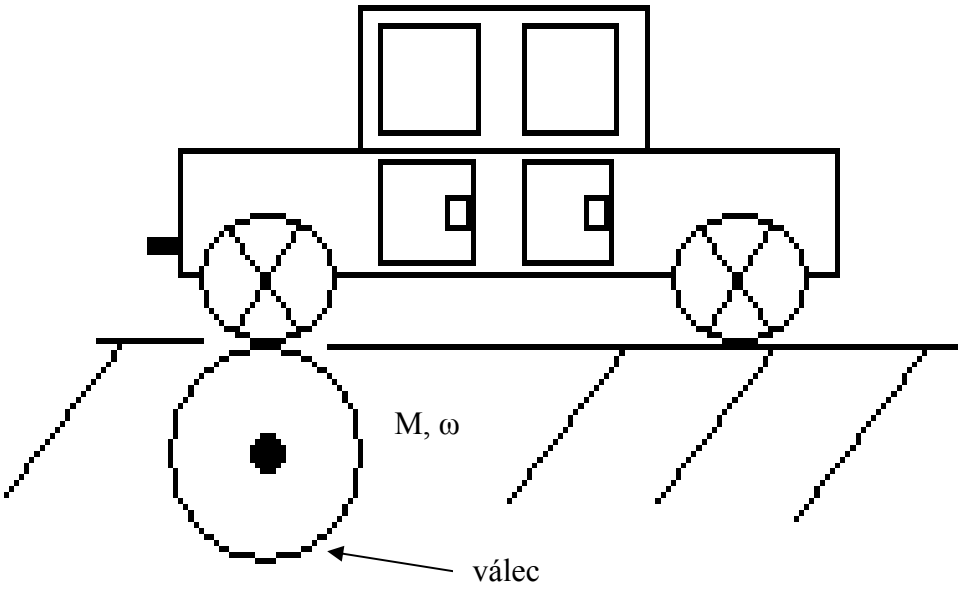
Cílem tohoto měření je určit dodávaný výkon motorem. Výkon motoru se určí podle vztahu 4-1 .

$$P = M \bullet \omega \ldots [W, Nm, \frac{rad}{s}] \qquad 4\text{-}1$$

P – výkon, M – moment, ω – úhlová rychlost

Pokud motor je již zamontovaný v daném vozidle, tak výkon se zjišťuje nepřímo, přes hnané kolo. Hnané kolo otáčí válcem. Na válci měříme jeho otáčky a moment, náčrt viz obrázek 4-26.

Zkouška se provádí tak, že při zadaných otáčkách měříme moment. Válec je nutno udržovat na konstantních otáčkách, tak aby bylo možno odečíst moment. K udržení konstantních otáček lze například použít zpětnovazební regulátor.



Obrázek 4-26 Statické měření

Ztráty momentu vznikají na soukolí motor – hnané kolo, přilnavost hnané kolo – válec a aj. Ztráty jsou závislé na otáčkách a nejsou konstantní. Ke zjištění ztrátového momentu se provádí výběhová zkouška.

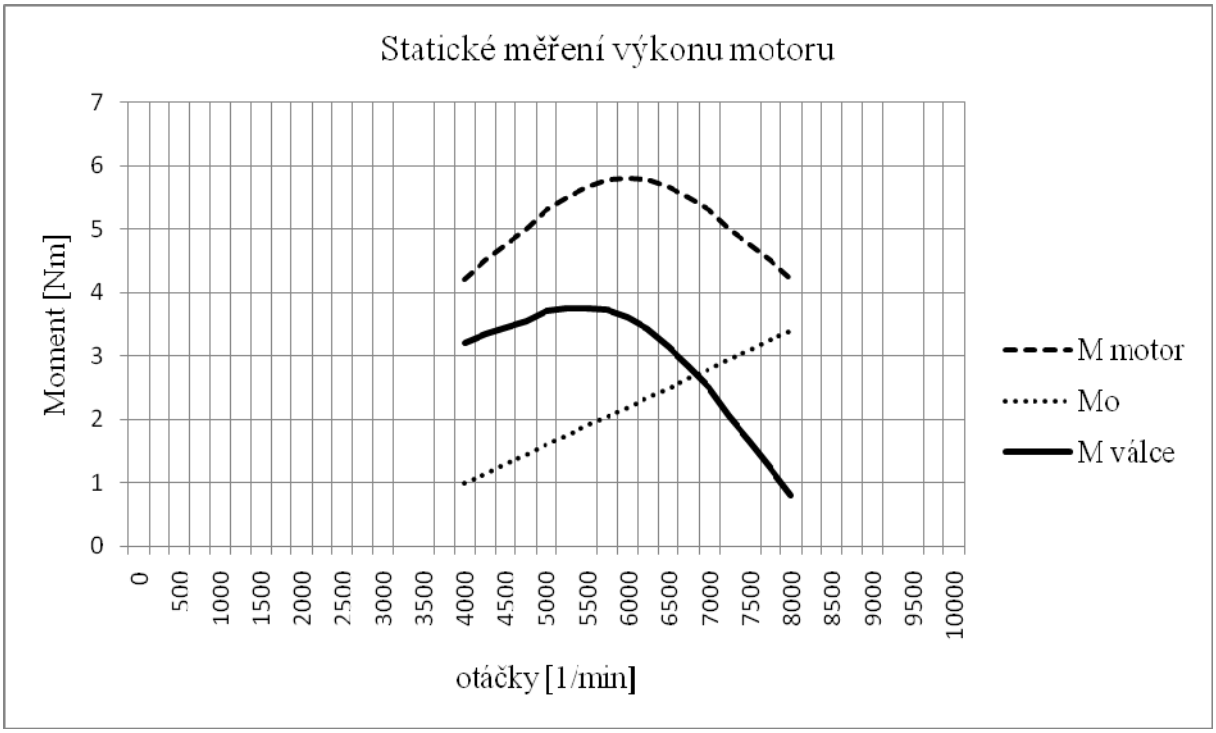
Při výběhové zkoušce se dané hnané kolo roztočí na požadované otáčky a pak se nechá doběhnout. Daný ztrátový moment se určí podle vztahu 4-2.

$$M_o = J \bullet \frac{\Delta\omega}{\Delta t} \ldots [\frac{kg}{m^2}, \frac{rad}{s}, s, Nm] \qquad 4\text{-}3$$

M_o ztrátový moment, J moment setrvačnosti, ω – úhlová rychlost, t - čas

Vzhledem, že se nesnadno zjišťují ztráty momentu na soukolí motor – hnané kolo, tak se tyto ztráty zanedbávají.

Následující graf udává grafické vyjádření vztahu ztrát a momentu motoru.

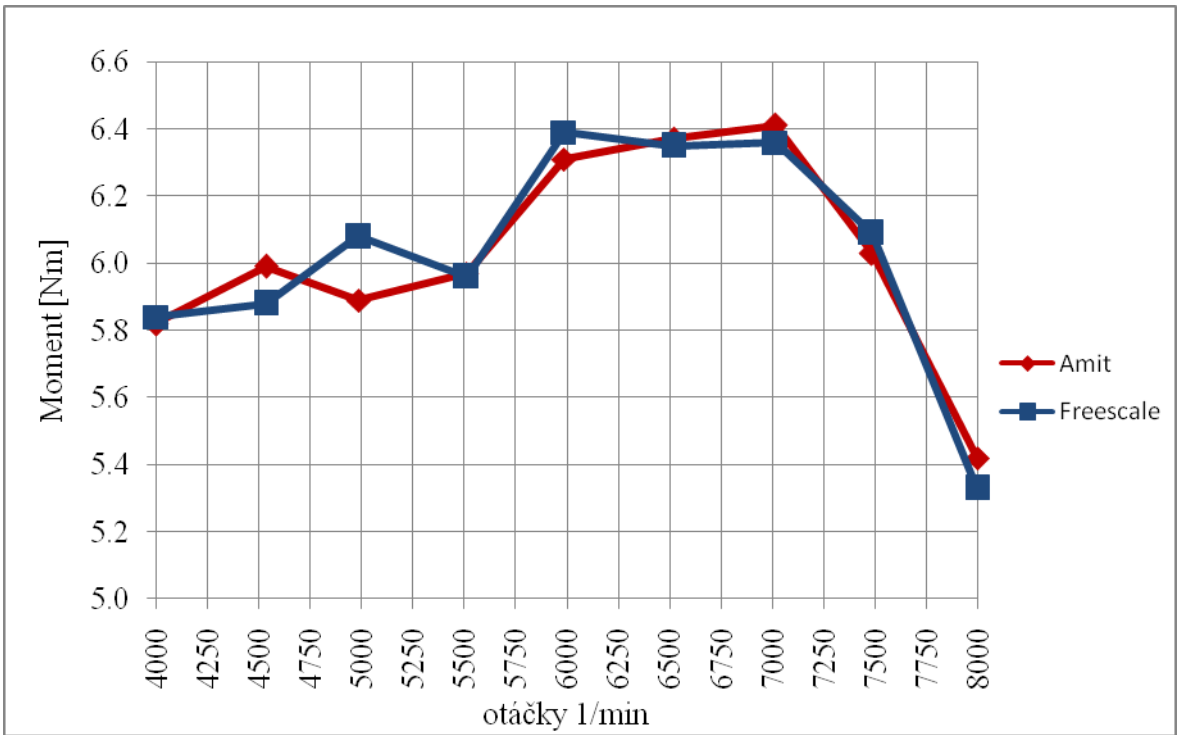


Obrázek 4-27 Statické měření výkonu motoru

Kde M motor – moment motoru, Mo – ztrátový moment, M válce – moment, který je změřen na válci.

Pro M válce platí M celk. = M motor – Mo [Nm,Nm,Nm].

V následujících grafech je použit pro nový terminal název Freescale.

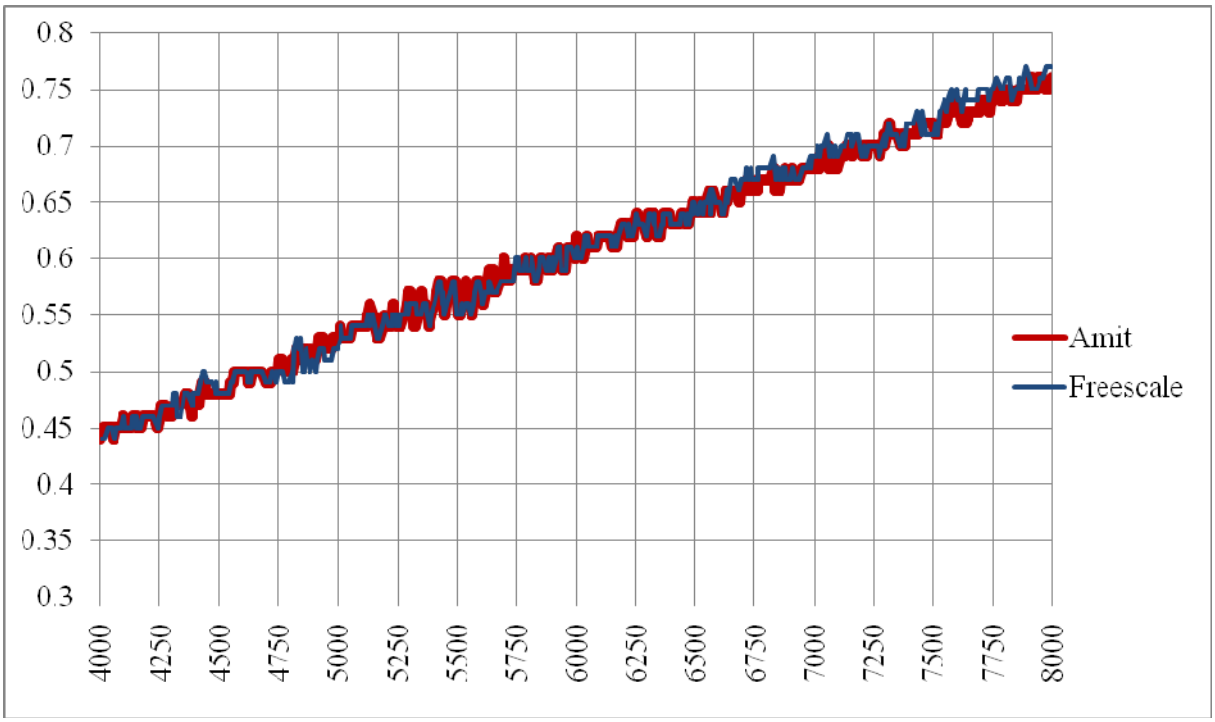


Obrázek 4-28 Statické měření výkonu motoru Amit vs Freescale

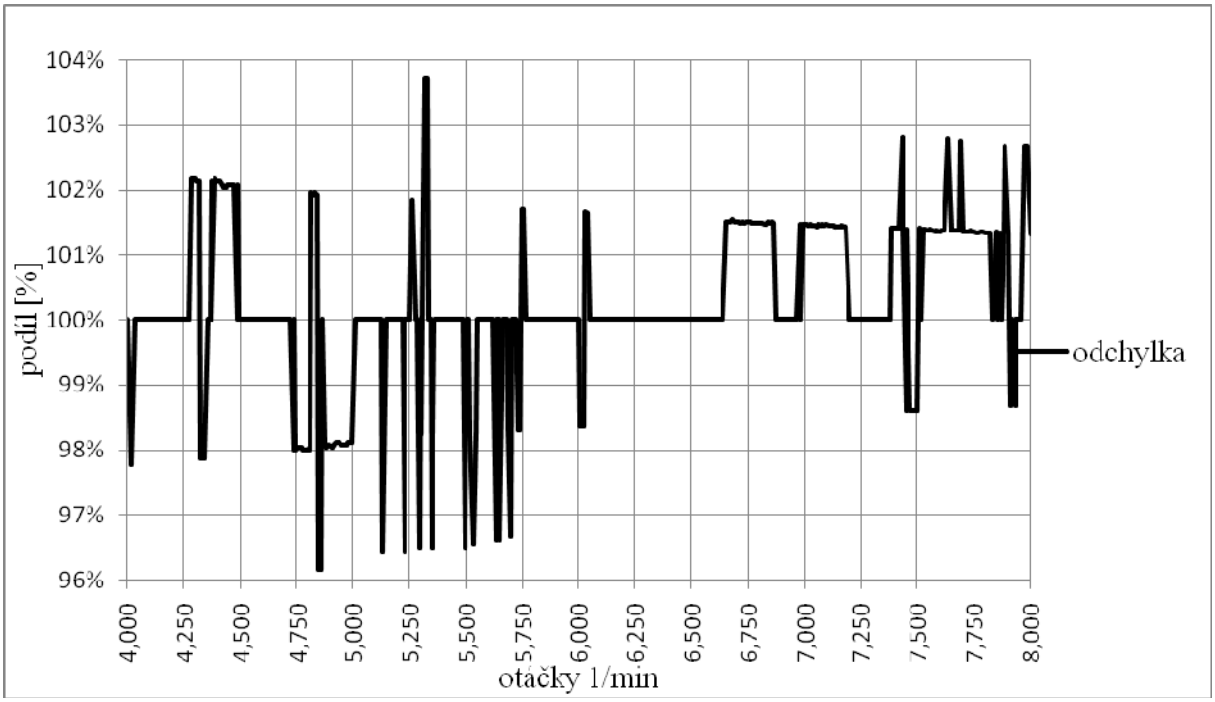
Obrázek 4-29 Odchylka naměřených hodnot (Statické měření)

Odchylka je vypočtena podle následujícího vzorce.

$$odchylka = \frac{hodnota\_změřená\_Freescale}{hodnota\_změřená\_Amit} \bullet 100 \ldots [Nm,Nm,\%] \qquad 4\text{-}4$$



Obrázek 4-30 Výběhová zkouška Amit vs Freescale



Obrázek 4-31 Odchylka naměřených hodnot (Výběhová zkouška)

Statické měření výkonu a výběhová zkouška, které byly měřeny novým terminálem (terminál běžící na mikrokontroléru MC9S12XEP100, Freescale) se shodují s naměřenými hodnotami, které poskytl terminál na platformě Amit. Tím je prokázána funkce nového terminálu.

5. BUDOUCÍ ZMĚNY SOFTWARU

5.1. XGATE

U mikrokontrolérů s jedním výpočetním jádrem je problém, že vykonávání programu je zdržováno obsluhou přerušení.

Možné řešení se naskýtá v použití dvou výpočetních jader. Jedno jádro bude vykonávat hlavní program a na další se přenese obsluha přerušení. Tím nebude činnost programu zdržována například obsluhou přerušení od sériové linky.

Mikrokontrolér MC9S12XEP100 je koncipován, tak že na čipu jsou přítomna dvě výpočetní jádra CPU a XGATE. XGATE je 16bit koprocesor s RISC architekturou a slouží pouze k obsluze přerušení. Podle výrobce může běžet až na dvojnásobné frekvenci vnitřní sběrnice.

V programu, kterým je naprogramován mikrokontrolér řídícího terminálu, tak využívá pouze jedno výpočetní jádro.

Vzhledem k tomu, že zprávy na sběrnici CAN jsou poměrně často zasílány (zhruba jedna zpráva za 2 ms), tak by bylo vhodné obsluhu přerušení přenést právě na XGATE. A dále by mohla být přesunuta na XGATE vyčítání naměřených dat po sériové lince do nadřazeného PC.

Tím by se docílilo výrazného odlehčení CPU, který by měl pouze na starosti činnost regulátorů, komunikaci s ovládacím software v PC a atd.

Výše uvedená změna vyžaduje změnu programu ohledně přístupu k obsluze přerušení a výbornou znalost mikrokontroléru MC9S12XEP100.

Nelze například použít k nastavení jednotlivých periferií PE.

Vedoucí práce na mě nepožadoval, abych software přepsal, tak aby bylo využíváno i druhé výpočetní jádro.

5.2. Komunikační protokol

Zaslání vyčítaných dat do nadřazeného PC (data zasílána do řídícího terminálu měřící ústřednou). To by bylo vhodné provést následujícím způsobem.

Zasílány by jenom byly hodnoty proměnných, které se změnily. Musí být také zaslána informace o tom, jaké proměnné data obsahují.

Provede se to tak, že ke každé proměnné či datovému poli bude přidána na začátek značka. Značka bude udávat začátek přenosu. Dále bude následovat odeslání id. Id bude udávat, co bude přenášeno. Po id již mohou být zaslána vlastní data.

Již v tomto momentě se nabízí kontrola přijatých id. Pokud id nesedí se seznamem vysílaných id, tak se hned na začátku prohlásí daná data za neplatná a mikrokontrolér si může například vyžádat opakování přenosu.

Naskýtá se otázka jak realizovat začátek přenosu.

Začátek přenosu (značka) se realizuje nulovým bajtem s nulovým stop bitem. Nulový stop bit vyvolá v procesoru, který tuto zprávu přijímá chybu rámce. Chyba rámce je pouze při začátku, tedy jsme našli začátek přenášených dat.

To vyžaduje, že každý bajt musí být zakončen dvěma stop bity.

Vysílač vysílá bajty s dvěma stop bity. První bajt odešle s devíti bity, které jsou nastaveny na log. 0. Dále jsou zasílány bajty s osmi bity.

Přijímač je nastaven pouze na příjem osmi bitů s dvěma stop bity. Pokud příjme devátý nulový bajt, tak indikuje chybu rámce. Stop bity ukončí činnost přijímaného bajtu. Tím je přijímač schopen přijmout další bajty.

Tabulka 5-1 Formát přenášených dat

bajt	1.	2.	3. a až n
paket	značka	id	data

Tento způsob umožňuje vysílat zprávy s veličinami do nadřazeného PC, u kterých došlo ke změně jejich hodnoty.

Podle datasheetu výrobce mikrokontroléru MC9S12XEP100 umožňuje vysílání bajtu s obsahem devíti bitů po sériové lince.

ZÁVĚR

Cílem této práce bylo zprovoznit dálkové ovládání na řídícím terminálu, který běží na mikrokontroléru MC9S12XEP100. A doplnit možnost komunikace s ostatními moduly měřícího pracoviště válcového dynamometru po sběrnici CAN (měřící ústředna, modul ventilátoru).

Mimo nefunkčnosti dálkového ovládání bylo zjištěno, že nefunguje v souladu s terminálem Amit vyčítání naměřených hodnot ze snímačů proudu, momentu a otáček. Dále periodické zasílání parametrů do nadřazeného PC, periodické vykonávání regulátorů a ukládání dat do EEPROM.

Byla provedena mechanická úprava řídícího terminálu (umístění řídícího terminálu v použité PC šasi, výroba panelu s LCD displejem, výroba propojovacích kabelů).

Po úpravách a přidání možnosti komunikace s ostatními moduly po sběrnici CAN byl řídící terminál otestován na celkovou funkčnost.

Nakonec byla nastíněna další možná budoucí změna firmware řídícího terminálu. Konkrétně se jedná o využití druhého výpočetního jádra, které je součástí mikrokontroléru MC9S12XEP100 a návrh jednoduchého komunikačního protokolu (zasílání dat do nadřazeného PC po druhé sériové lince).

Výsledkem práce je řídící terminál, který je možno ovládat z nadřazeného PC. Dále je přidána komunikace s ostatními moduly měřícího pracoviště válcového dynamometru po sběrnici CAN.

Tím bylo dosaženo splnění všech úkolů práce.

6. LITERATURA

[1] VIDNER, Luboš. Modul měření otáček spalovacího motoru. Pardubice, 2015. Ročníkový projekt II. Univerzita Pardubice.

[2] MAREK, Jiří. Řídicí jednotka pro válcový dynamometr. Pardubice, 2012. Diplomová práce. Univerzita Pardubice.

[3] MAŠEK, Zdeněk. Programové řízení dynamometru pro zkoušení pohonu vozidel. Pardubice, 2005. Diplomová práce. Univerzita Pardubice.

[4] LELEK, Tomáš. Měřící ústředna k dynamometru s výstupem na sběrnici CAN. Pardubice, 2011. Bakalářská práce. Univerzita Pardubice.

[5] HLADÍK, Pavel. Modul řízení otáček ventilátoru. Pardubice, 2012. Bakalářská práce. Univerzita Pardubice.

[6] Přednášky z předmětu „Programování v jazyce C“

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHY DP

Příloha A

Seznam zpráv vysílaných po CAN (modul měřící ústředna)

Analogové vstupy

Vysílání hodnot napětí na analogových vstupech AI1 až AI8 je rozděleno do dvou zpráv.

CAN-ID	0x11
Perioda vysílání	8,192 ms
Význam dat	Hodnoty napětí na vstupech AI1 až AI4
Počet datových bajtů	8
Datový typ	UINT16
Pořadí bajtů	LSB first
Přepočet na fyzikální hodnotu	rozlišení 1 mV/bit, offset 0 mV
Rozsah	0 až 10000 mV

CAN-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x11	AI4		AI3		AI2		AI1	

CAN-ID	0x10
Perioda vysílání	8,192 ms
Význam dat	Hodnoty napětí na vstupech AI5 až AI8
Počet datových bajtů	8
Datový typ	UINT16
Pořadí bajtů	LSB first
Přepočet na fyzikální hodnotu	rozlišení 1 mV/bit, offset 0 mV
Rozsah	0 až 10000 mV

CAN-ID 0x10	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	AI8		AI7		AI6		AI5	

Termočlánkové vstupy

Vysílání hodnot teploty na termočlánkových vstupech TC1 až TC8 je rozděleno do dvou zpráv.

CAN-ID	0x13
Perioda vysílání	204,8 ms
Význam dat	Teploty na termočlánkových vstupechTC1 až TC4
Počet datových bajtů	8
Datový typ	UINT16
Pořadí bajtů	LSB first
Přepočet na fyzikální hodnotu	rozlišení 0,25 °C/bit, offset 0 °C
Rozsah	0 °C až 1023,75 °C

CAN-ID 0x13	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	TC1		TC2		TC3		TC4	

CAN-ID	0x13
Perioda vysílání	204,8 ms
Význam dat	Teploty na termočlánkových vstupech TC5 až TC8
Počet datových bajtů	8
Datový typ	UINT16
Pořadí bajtů	LSB first
Přepočet na fyzikální hodnotu	rozlišení 0,25 °C/bit, offset 0 °C
Rozsah	0 °C až 1023,75 °C

CAN-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x14	TC5		TC6		TC7		TC8	

CAN-ID	0x15
Perioda vysílání	204,8 ms
Význam dat	Stav vstupů TC1 až TC8
Počet datových bajtů	1
Datový typ	UINT8
Pořadí bajtů	-
Přepočet na fyzikální hodnotu	Každý bit v Byte 0 udává stav vstupu. 0 – vstup je obsazen 1 – vstup je volný Bit 0 náleží TC1 …. Bit 7 náleží TC8
Rozsah	0 až 255

CAN-ID	Byte 0
0x15	Stav vstupů TC1 až TC8

Platinové teploměry Pt100

Vysílají se hodnoty teplot z platinových teploměrů Pt100 na vstupech RTD1 a RTD2 a stav těchto vstupů.

CAN-ID	0x16
Perioda vysílání	204,8 ms
Význam dat	Teploty na RTD vstupech RTD1 a RTD2
Počet datových bajtů	8
Datový typ	UINT32
Pořadí bajtů	LSB first

Přepočet na fyzikální hodnotu	rozlišení 1/1024 °C/bit, offset 0 °C
Rozsah	0 °C až 369,291 °C

CAN-ID 0x16	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	RTD1				RTD2			

CAN-ID	0x17
Perioda vysílání	204,8 ms
Význam dat	Hodnoty registrů obvodu XTR108 pro vstup RTD1
Počet datových bajtů	8
Datový typ	UINT8
Pořadí bajtů	Stav vstupu je uložen v bajtu č. 3
Přepočet na fyzikální hodnotu	0 – teploměr je zapojen a je v pořádku 5 – teploměr je odpojen 10 – zkrat na GND Další možnosti viz datasheet k XTR108
Rozsah	0 až 15

CAN-ID 0x17	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
				Stav vstupu RTD1				

CAN-ID	0x19
Perioda vysílání	204,8 ms
Význam dat	Hodnoty registrů obvodu XTR108 pro vstup RTD2
Počet datových bajtů	8
Datový typ	UINT8
Pořadí bajtů	Stav vstupu je uložen v bajtu č. 3
Přepočet na fyzikální hodnotu	0 – teploměr je zapojen a je v pořádku 5 – teploměr je odpojen 10 – zkrat na GND Další možnosti viz datasheet k XTR108
Rozsah	0 až 15

CAN-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x19				Stav vstupu RTD2				

Otáčky motoru

CAN-ID	0x12
Perioda vysílání	8,192 ms
Význam dat	Doba mezi dvěma posledními zápaly
Počet datových bajtů	4
Datový typ	UINT32
Pořadí bajtů	LSB first
Přepočet na fyzikální hodnotu	rozlišení 0,5 µs /bit, offset 0 µs
Rozsah	0 až UINT32_MAX

CAN-ID 0x12	Byte 0	Byte 1	Byte 2	Byte 3
	Doba mezi dvěma posledními zápaly			

Údaje ze sondy HXB-85

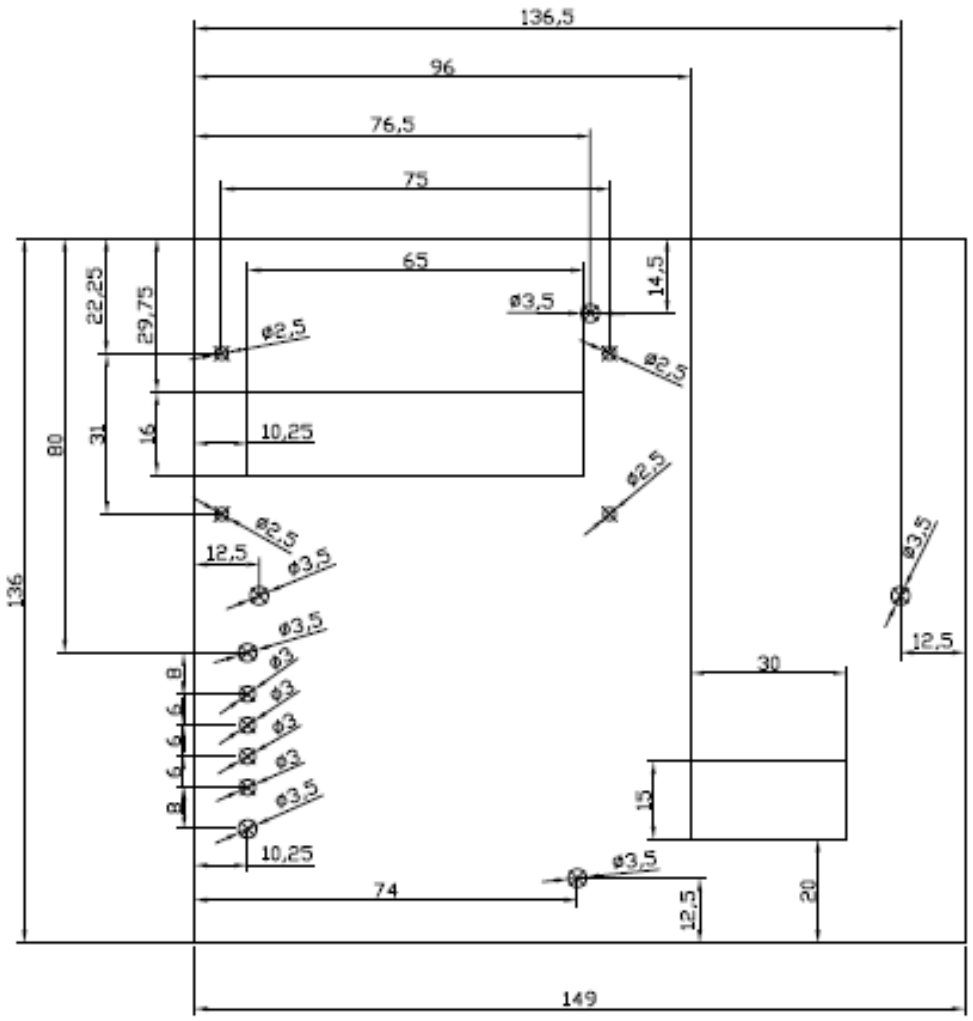
Jedná se o sondu, která měří teplotu, tlak, relativní vlhkost a rosný bod atmosférického vzduchu.

CAN-ID	0x1B
Perioda vysílání	Tak často, jak sonda HXB-85 posílá data (cca 1,5 s)
Význam dat	Rosný bod, teplota, tlak, relativní vlhkost vzduchu
Počet datových bajtů	8
Datový typ	INT16
Pořadí bajtů	LSB first
Přepočet na fyzikální hodnotu	Rosný bod: rozlišení 1/100 °C /bit, offset 0 °C Teplota:rozlišení 1/100 °C /bit, offset 0 °C Tlak: rozlišení 1 mBar/bit, offset 0 mBar Rel. vlhkost:rozlišení 1/100 %/bit, offset 0 %
Rozsah	Rosný bod: -60 °C až + 40 °C Teplota: -20 °C až + 80 °C Tlak: 750 mBar až 1100 mBar Rel. vlhkost: 0 % až 100 %

CAN-ID 0x1B	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	Rosný bod		Teplota		Tlak		Relativní vlhkost	

Příloha B

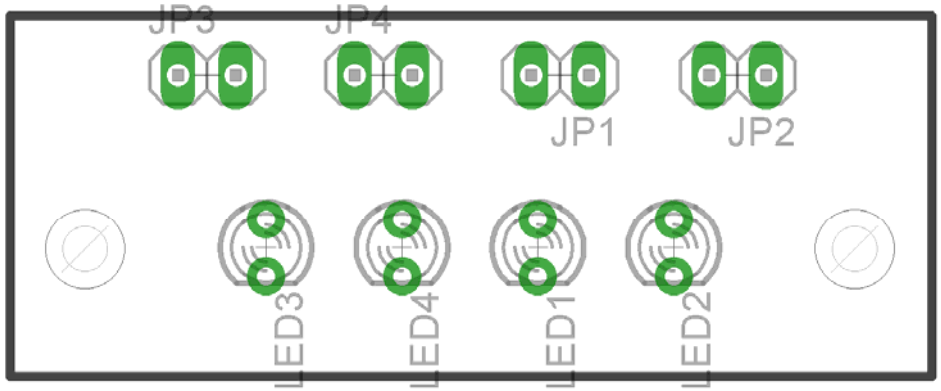
Čelní panel na LCD displaj



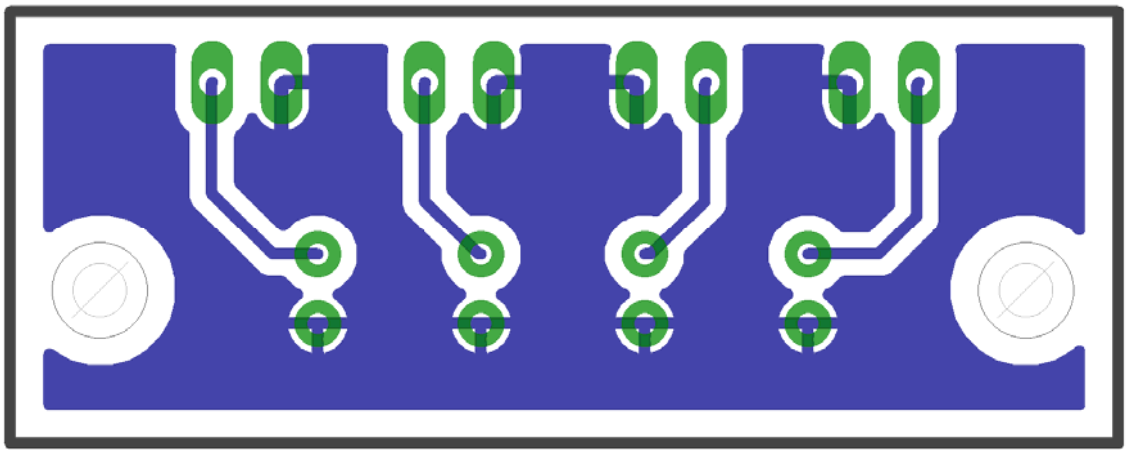
Obr. 0-1 Výkres čelního panelu

Příloha C

Panel signalizačních led diod



Obr. 2 Osazovací plán panel signalizačních LED



Obr. 3 DPS panelu signalizačních LED

Příloha D

Obr. 4 Foto převzaté DPS

Příloha E

Obr. 5 Čelní panel

Obr. 6 Foto konektorů řídícího terminálu