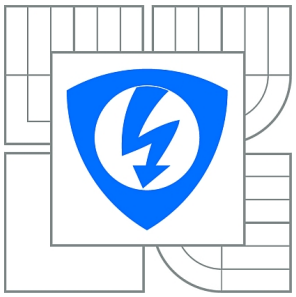
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOMBINOVANÁ V/V KARTA S ROZHRANÍM ETHERNET

I/O CARD EQUIPPED BY ETHERNET

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. PETR MASLÁK

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. ZDENĚK BRADÁČ, Ph.D.

SUPERVISOR

BRNO 2013

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Petr Maslák
ID: 115227
Ročník: 2
Akademický rok: 2012/13

NÁZEV TÉMATU:

Kombinovaná V/V karta s rozhraním Ethernet

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte kombinovanou V/V kartu s rozhraním Ethernet. Kartu vybavte digitálními a analogovými V/V. Systém postavte na základě mikrokontroléru s komunikačním rozhraním Ethernet. Navrhněte a realizujte HW, oživte a otestujte. Vytvořte programové vybavení pro mikrokontrolér a pro nadřízené PC.

DOPORUČENÁ LITERATURA:

Pavel Herout: Učebnice jazyka C, KOPP, 2004, IV. přepracované vydání, ISBN 80-7232-220-6
Dle pokynů vedoucího práce.

Termín zadání: 11.2.2013
Termín odevzdání: 20.5.2013

Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práce se zpočátku zabývá hardwarovým návrhem kombinované karty s modulem Rabbit3200 vybaveným rozhraním ethernet. Softwarová část byla vytvářena v programovacích jazycích assembler, C a C#. Na závěr byla karta otestována generátorem, multimetrem a osciloskopem.

KLÍČOVÁ SLOVA

Kombinovaná karta, AD převodník, DA převodník, modul RCM3200, DPS, UDP protokol, C#, Dynamic C

ABSTRACT

This thesis is at the beginning about designing hardware of I/O card with module - Rabbit3200 equipped by ethernet. Programs were created in programming languages such as assembler, C and C#. At the end card was tested by generator, multimeter and oscilloscope.

KEYWORDS

I/O card, AD converter, DA converter, module RCM3200, DPS, UDP protokol, C#, Dynamic C

MASLÁK, Petr *Kombinovaná V/V karta s rozhraním Ethernet*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2013. 84 s. Vedoucí práce byl doc. Ing. Zdeněk Bradáč, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Kombinovaná V/V karta s rozhraním Ethernet“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Tato diplomová práce se zpočátku ve stručnosti zabývá přehledem měřicích karet běžně dostupných na trhu. Tento úsek je důležitý pro získání základního přehledu, jaké parametry by mohla navržená karta mít s ohledem na cenu. Nakonec byly vybrány komponenty, které bohatě stačí na splnění cíle této práce. Analogové vstupní i výstupní linky byly navrženy tak, aby se jejich vzorkovací rychlost pohybovala okolo sto tisíc vzorků za sekundu. Základní komponentu celé multifunkční karty tvoří již zakoupený modul RCM3200 s procesorem Rabbit. Multifunkční karta se skládá celkem ze dvou desek plošných spojů a již zmíněného modulu. Tato karta bude komunikovat s počítačem prostřednictvím ethernetové sběrnice, která je už implementovaná na modulu. Karta se bude lišit oproti ostatním kartám zabudovaným displejem, maticovou klávesnicí a tlačítky.

Pokud nebude karta připojena k počítači přes ethernetový kabel, bude možné i přesto generovat základní přednastavené signály, u nichž bude možnost nastavovat parametry jako je frekvence nebo amplituda. Snímaní v offline režimu bude také možné, ale v omezené míře. Veškeré tyto funkce bude možné ovládat v menu díky maticové klávesnici a tlačítek. Součástí karty bude také osm digitálních linek, proto název diplomové práce nenese jednoduchý název „Měřicí karta“, ale „Kombinovaná V/V karta“. V textu se setkáme i s názvem "multifunkční karta".

Cílem této práce není vyprojektovat konkurence schopnou kartu, a tudíž veškeré návrhy nebudou brát zřetel na možnost sériové výroby, cenu, pracovní podmínky atd. V určitých pasážích se však zmíníme o tom, jak lze kartu navrhnout jiným způsobem, aby byla například levnější nebo aby se deska plošných spojů dala pájet vlnou.

V diplomové práci budou vysvětleny jednotlivé problémy, s nimiž jsem se potýkal během oživování. Jedním z těchto problémů byla rychlost modulu: zdrojové kódy proto nebyly programovány jenom v jazyce C, ale i v assembleru. Ve stručnosti se zmíníme o jazyku C# v programovacím prostředí Visual studio, kde bude implementován uživatelský program na PC.

Ve finální fázi vyhodnotím přesnost karty, možnosti budoucího vylepšení a provedu celkové zhodnocení projektu.

1 MĚŘICÍ KARTY

V technické praxi se měřicí karty využívají více než 15 let. Jejich rozmanitost se liší v závislosti na použití. Multifunkční karty jsou vybaveny jak analogovými, tak digitálními linkami a některé mají i čítače. Díky klesající ceně se staly dostupné i pro běžného uživatele.

1.1 Rozhodující vlastnosti

Mezi základní parametry analogových vstupů se řadí počet kanálů, vzorkovací frekvence, kvantizační rozlišení, napěťový rozsah. Počet kanálů se zvyšuje předřazením analogového multiplexoru před AD převodník. Tímto se však snižuje maximální vzorkovací frekvence na jeden kanál. Většina karet má na vstupu filtr dolní propusti. Napěťový rozsah opět hraje velkou roli, co se týče kvantizačního šumu. U některých typů se napěťový rozsah může softwarově měnit. Kritérií na výběr vhodné karty existuje spousta, například odolnost vůči přepětí, pádu, prašnosti či vlhkosti prostředí. Následující odstavce se zabývají jen těmi nejdůležitějšími parametry pro získání základního přehledu.

Analogové výstupy mají nejčastěji rozsah napětí ± 10 V, karty s proudovým výstupem pak 0 až 20 mA nebo 4 až 20 mA. Analogové výstupy mají v současnosti obvyklé rozlišení od 12 do 16 bitů. Osmibitové převodníky se vyskytuji pouze u extra rychlých zařízení, jako jsou osciloskopy. Dvaceti-čtyř bitové převodníky se aplikují pouze u audio karet.

Digitální linky, které jsou galvanicky oddělené, se používají buď jako vstupy nebo výstupy. Směr dané linky se mění již softwarově. Počet digitálních linek bývá vesměs násobek osmi (1 bajt). U digitálních linek se rozlišuje i napěťová úroveň logické jedničky.

Co se týče softwaru, tato oblast nesmí být také samozřejmě opomíjena. Karta s nejlepšími vlastnostmi, ale bez dostupného ovladače a kvalitního softwaru, ztrácí výrazně svoji hodnotu. Výborně propracovaný software mají zařízení od firmy National Instruments. Zde se ale za software platí zvlášť.

Nejrychlejší měřicí karty se připojují přes PCIe rozhraní. Nejnovější rozhraní PCI-e x16 3.0 dosahuje rychlosti 32 GB/s, což je oproti klasickému USB několikanásobně větší datová propustnost. Další rozhraní mohou být GPIB, RS 232, FireWire. Pro vzdálená připojení k zařízení se používá rozhraní ethernet, které je implementováno u naší realizované karty. Rozhraní určuje kromě rychlosti i přenositelnost měřicí karty. Žádná rychlejší karta se však neobejde bez vyrovnávací paměť, do níž se dočasně ukládají vzorky určené pro následný přenos do PC.

1.2 Přehled několika měřicích karet

V dnešní době existuje široká škála měřicích karet, které se dají aplikovat na běžná i laboratorní měření. Pro základní přehled byly vybrány pouze multifunkční karty s rozdílnými parametry od různých výrobců.

1.2.1 USB 6008

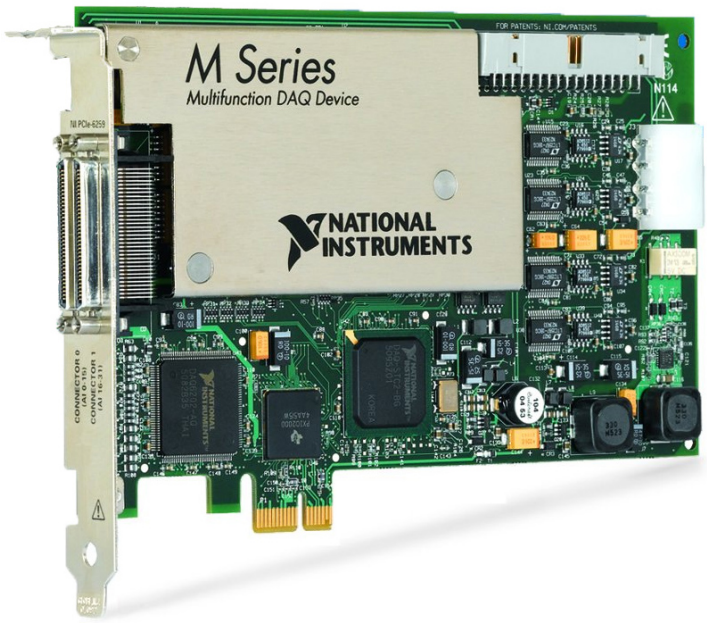
Tato měřicí karta od Firmy National instrument komunikuje s počítačem přes rozhraní USB. Karta má v sobě zabudovaných 8 analogových vstupů. Maximální vzorkovací frekvence se pohybuje okolo 10 kS/s. Při použití více kanálů se vzorkovací frekvence dělí. Karta disponuje pouze 2 výstupními kanály, kdy maximální obnovovací frekvence dosahuje pouze 150 S/s, a dvanácti obousměrnými digitálními linkami. Díky nízké ceně (cca 4000 Kč) a omezeným možnostem se tato karta upotřebí spíše ve výuce.

Obr. 1.1: Měřicí karta USB 6008 od NI [10]

1.2.2 NI PCIe-6259

Opět multifunkční karta od National Instrument, jež má ale k dispozici až 32 analogových vstupů. Rozlišení analogových kanálů udává datasheet 16 bitů. Tato karta může měřit mnohem větší vzorkovací frekvencí než USB 6008, a to rychlostí 1.25 MS/s. Vzorkovací frekvence 2.86 MS/s se již přibližuje vzorkovací frekvenci

známého generátoru Agilent 33120, jenž má maximální výstupní vzorkovací frekvenci 20 MS/s. Vyšší možnosti karty zvyšují i její cenu, a tudíž cena 37 000 Kč bez příslušenství je odpovídající.



Obr. 1.2: PCIe-6259 od NI [11]

1.2.3 PCI-1711

Tento typ karty vyrábí firma Advantech. Již z názvu vyplývá, že je karta určena do PCI slotu. K výbavě karty patří 16 analogových vstupů s 12bitovým rozlišením a vzorkovací frekvencí 100 kS/s. Vzorkovací frekvenci výstupních kanálů výrobce neudává. Karta disponuje také digitálními vstup-výstupy, jež dohromady tvoří 32 linek. I když karta nedosahuje takových možností jako PCIe-6259, tak vzhledem k nízké ceny má dobré parametry.

Obr. 1.3: PCI-1711 od Advantechu [12]

1.2.4 U2352A

Firma HP také nabízí multifunkční měřicí karty. Karta U2352A byla navržena se 16 analogovými vstupy o rozlišení 16 bitů a se slušnou vzorkovací frekvencí 250 KS/s. Za samozřejmost můžeme u této karty brát i digitální linky, jejíž celkový počet čítá 24. Zajímavostí u této karty je kompatibilita se softwarem LabView, které produkuje firma NI. Toto zařízení se hodí i mimo laboratorní prostory, neboť komunikace probíhá přes USB. Na obrázku 1.4 si i všimněme celkového designu zařízení, které je díky pogumování uzpůsobené do terénu.

Obr. 1.4: U2352A od firmy HP [13]

1.2.5 Závěrečné porovnání vybraných měřicích karet

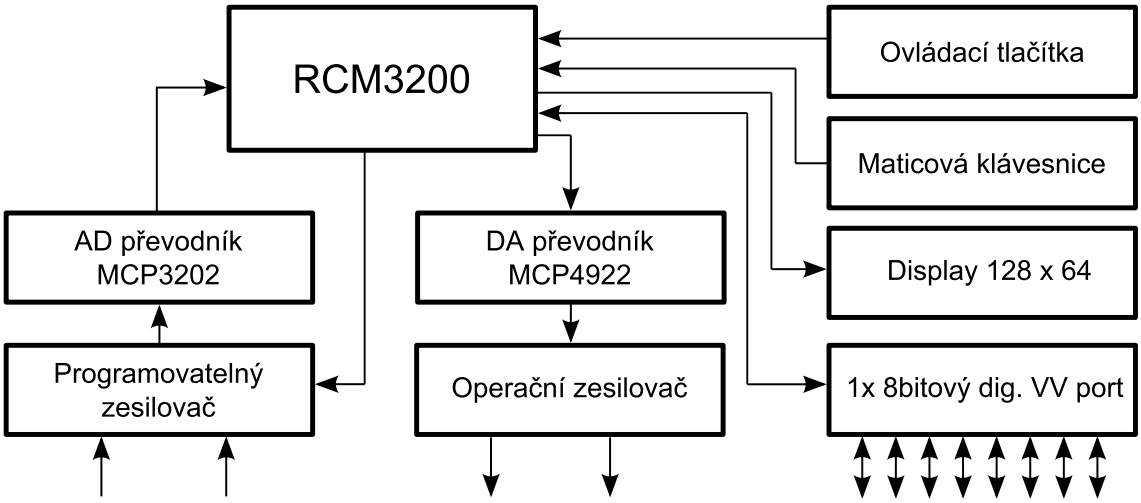
Díky velkému množství multifunkčních měřicích karet máme na trhu opravdu z čeho vybírat. Platí zde však známé pravidlo: „Čím víc peněz, tím víc muziky.“ V následující přehledové tabulce jsou uvedeny a popsány jen některé vybrané měřicí karty.

Měřicí karta	Firma	Počet AI	Rozlišení [bits]	Frekvence Vzorkování AI	Počet AO	Rozlišení[bits]	Frekvence obnovování AO	Počet dig. linek	Rozhraní	Cena
USB 6008	NI	8	12	10 kS/s	2	12	150S/s	12	USB 2.0	4 070 Kč
Usb 6009	NI	8	14	48 kS/s	2	12	150S/s	12	USB 2.0	6 690 Kč
PCIe-6259	NI	32	16	1,25 MS/s	4	16	2,86MS/s	48	PCIe	36 900 Kč
PCI-1711	Advantech	16	12	100 kS/s	2	12	38 kS/s	32	PCI	6 770 Kč
U2352A	HP	16	16	250 kS/s	-	-	-	24	USB 2.0	22 689 Kč
PCIe-7852R	NI	8	16	750kS/s	8	16	1MS/s	96	PCIe	100 900 Kč

Tab. 1.1: Porovanání vybraných multifunkčních karet běžně dostupných na trhu

2 HARDWAROVÝ NÁVRH

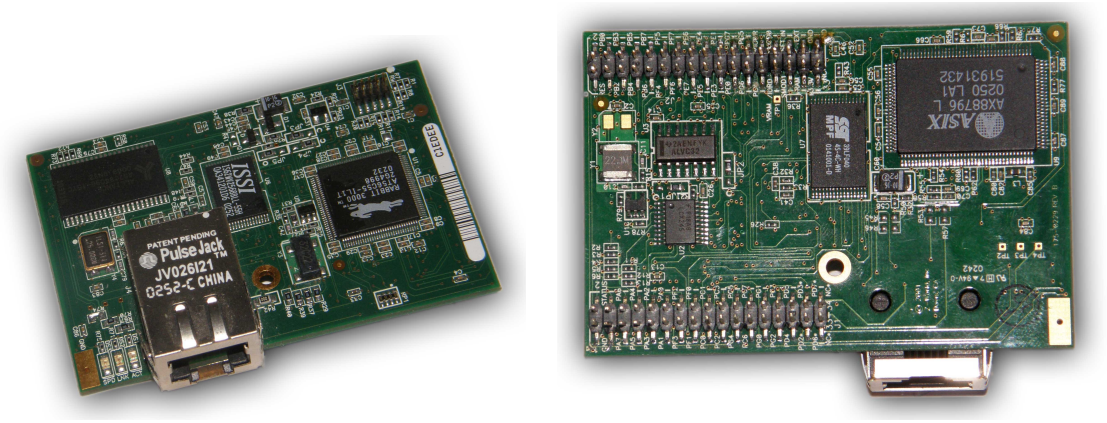
Kombinovaná karta je navržena z několika komponent, z nichž některé budou dále rozebírány. Celá karta je řízena modulem RCM3200 s procesorem Rabbit. V následujících kapitolách je popsán způsob výběru jednotlivých komponent, schéma a nakonec navržená deska plošných spojů (DPS).



Obr. 2.1: Blokové schéma kombinované karty

2.1 Modul RCM3200

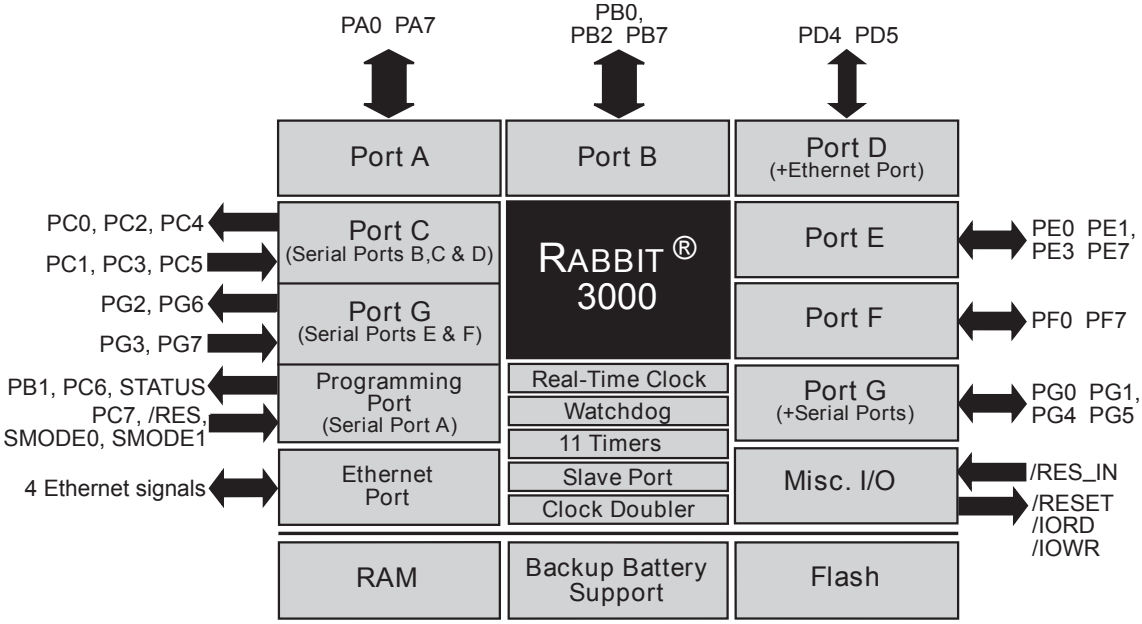
Díky tomuto modulu se již nemusí implementovat DPS pro komponenty, jako je například krystal nebo Ethernet. Moduly jsou tedy výhodné z časového i ekonomického hlediska a využívají se pro embedded systémy. Jiné moduly mohou mít i Bluetooth, Wi-fi nebo GPRS/GSM. Napájecí napětí výrobce uvádí v rozmezí od 1,8 V do 3,6 V, primární určení je stanoveno pro 3,3 V. Tento modul pracuje na frekvenci 44,2 MHz s osmi bitovou šířkou slova. Pracovní teploty se pohybují od -40 °C do 70 °C. Počet priorit v přerušení se dá nastavit až na 4 úrovně, my však využijeme pouze dvě přerušení: pro tlačítka spolu s maticovou klávesnici a pro časovač. Napájecí proud mikroprocesoru výrobce udává 2 mA/MHz při 3,3 V, u frekvence 44,2 MHz zhruba 88,4 mA, avšak napájecí proud celého modulu činí až 255 mA. Tolerance napětí na vstupních portech je stanovena maximálně na 5,5 V.



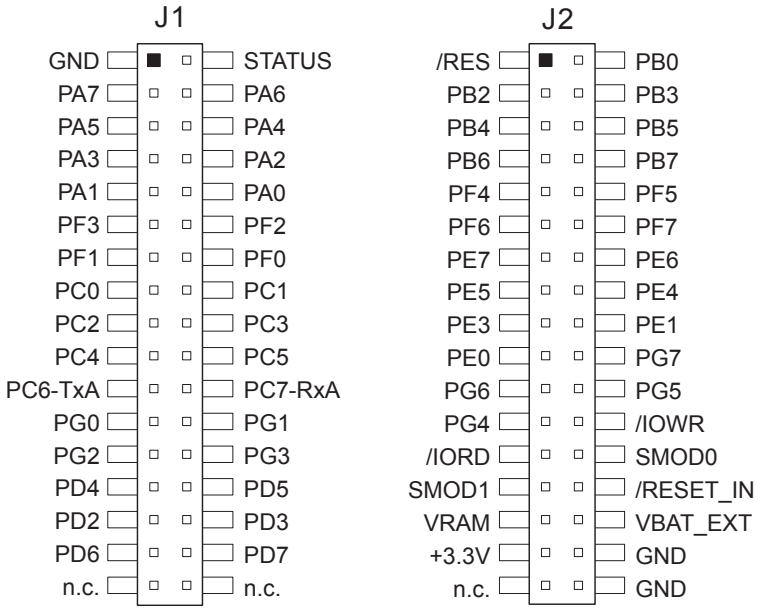
(a) horní strana (b) spodní strana

Obr. 2.2: Modul RCM3200

Z blokového schématu na obrázku 2.3 můžeme vidět, že mikroprocesor disponuje šesti obousměrnými osmibitovými porty A - G. Na mikroprocesoru je možné inicializovat až šest sériových linek, jedna z nich se však používá pouze pro naprogramování mikroprocesoru a ladění. Na obrázku 2.4 jsou přehledně zobrazeny jednotlivé piny na modulu.



Obr. 2.3: Blokové schéma modulu RCM3200 [17]



Obr. 2.4: Piny modulu RCM3200 [17]

2.2 Napájecí obvody

Multifunkční karta bude napájena spínaným adaptérem s výstupním napětím 24 V. Vstupní obvod je chráněn transilem proti přepětí a Schotkyho diodou proti přepólování. Na DPS jsou použity součástky s různými napájecími úrovněmi. Modul RCM3200 pracuje na úrovni 3,3 V. Grafický displej, AD a DA převodníky vyžadují 5 V. Převodníky mohou být napájeny i s úrovní 3,3 V, ale potom by se jejich vzorkovací frekvence snížila přibližně na polovinu, tedy 50 kS/s, což by již nevyhovovalo našemu požadavku 100 kS/s. Z obrázku 2.5 vyplývá, že snížení napětí 24 V je řešeno pomocí dvou spínaných zdrojů NDL2415 a NDL2405, pro něž je příznačná jejich nízkošumová charakteristika a možnost vstupního napětí v rozmezí od 18 V do 36 V. Za těmito spínanými zdroji jsou umístěny kondenzátory, které plní funkci jak filtrační, tak i blokovací. AD a DA převodníky potřebuji přesné referenční napětí, neboť na výstupech spínaných zdrojů není přesné požadované napětí. Spínané zdroje jsou závislé i na teplotě. Pro tyto účely existuje součástka, jež má na výstupu referenční napětí. Tuto funkci plní součástka s názvem LT1021N, která disponuje opět nízkým šumem (výrobce udává 0,1 Hz až 10 Hz) a přesností $\pm\, 0,05$ % (na 5 V: 2,5 mV). Referenční zdroj se napájí 15 V, neboť minimum pro napájení je stanoveno okolo 8 V. Pro napájení modulu RCM3200 byl zvolen klasický stabilizátor dimenzovaný pro snížení napětí z 5 V na 3,3 V. Nyní je potřeba spočítat, kolik energie bude přibližně stabilizátor pohlcovat.

$$P = U \cdot I \tag{2.1}$$

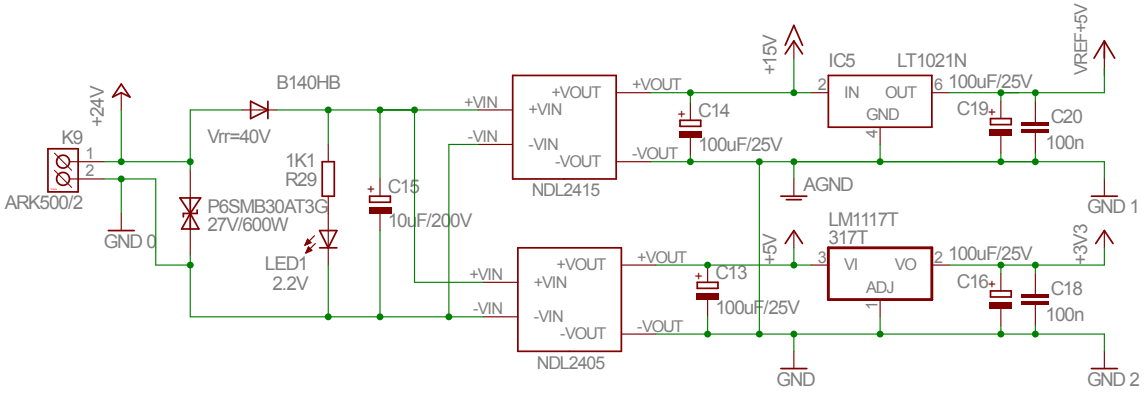
Přibližný součet proudů všech součástek včetně modulu činí přibližně 400 mA. Napěťový úbytek na stabilizátoru:

$$U_{vstup} - U_{vystup} = 5 - 3{,}3 = 1{,}77\ V \tag{2.2}$$

a tedy dle rovnice 2.1:

$$P = 1{,}77 \cdot 0{,}4 = 0{,}7\ W \tag{2.3}$$

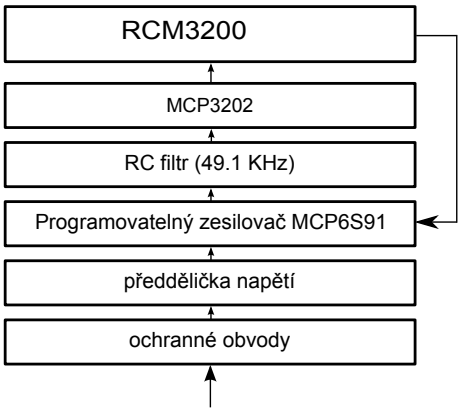
Při tomto výkonu je potřeba teplo ze stabilizátoru účinně odebírat, jinak by mohlo dojít k přehřátí nejen samotného stabilizátoru, ale hlavně by se ohřívala samotná DPS, což by mohlo mít neblahý vliv například na referenční napětí. Proto se kolem stabilizátoru nachází tolik volného prostoru ke chlazení (viz příloha B.1).



Obr. 2.5: Schéma napájecího obvodu

2.3 Vstupní analogové obvody

Vstupní analogový obvod se skládá z ochrany proti přepětí, ochrany proti přepólování, z předděličky, programovatelného zesilovače, RC filtru jako dolní propusti a nakonec samotného AD převodníku MCP3202. Popis vstupní analogové části je zřejmý z obrázku 2.6 a přílohy A.2.



Obr. 2.6: Blokové schéma vstupní analogové části

2.3.1 Ochranné obvody, předdělička a progr. zesilovač

Kombinovaná karta je dimenzována pro měření od 0 do 10 V, proto je na vstupu zapojena přepěťová ochrana v podobě transilu reagující na napětí vyšší než 12 V. Jako ochrana proti zápornému napětí zde slouží Schottkyho dioda, která reaguje již při úrovni napětí -0,3 V. Rozsah vstupního měřeného signálu bude možno softwarově nastavovat pro zvýšení přesnosti. Předdělička slouží ke snížení vstupního napětí na polovinu. Programovatelný zesilovač dle zdroje [20] zesiluje až 32krát, přesnost referenčního zdroje se pohybuje okolo 220 μV, dále známe rozlišení AD převodníku, proto nám bude bohatě postačovat zesílení až desetkrát. Pokud bude nastavené maximální vstupní napětí na 0,5 V, využijeme celý rozsah AD převodníku. Jako nevýhodu u tohoto operačního zesilovače můžeme vnímat dle [20] jeho minimální hodnotu napětí na výstupu při nulovém napětí na vstupu, hodnota tohoto napětí může činit až 4,5 mV, měřené napětí do této velikosti bude vykazovat velké relativní chyby. Operační zesilovač opět také nemusí na výstupu dosáhnout přesně takového napětí, jakým je napájen, proto by se mohly chyby vyskytnout i při vysokých napětích. Tento problém se však může kompenzovat snížením napětí pomocí předděličky a softwarového doladění. Za tuto cenu se však sníží využitý rozsah na AD převodníku.

Před AD převodník je navržen jednoduchý filtrační RC článek. Dle Schannon-Kotelnikova teorému musí být vzorkovací frekvence dvakrát vyšší, než je frekvence signálu na vstupu. U RC článku se vypočítá časová konstanta (zdroj [1], str 77) jako:

$$\tau = R \cdot C = \frac{1}{\omega_{mez}} = \frac{1}{2\pi f_{mez}} \tag{2.4}$$

$$f_{mez} = \frac{1}{2\pi RC} \tag{2.5}$$

mezní frekvence f_{mez} RC článku je tedy dle vzorkovacího teorému a znalosti vzorkovací frekvence AD převodníku rovna

$$f_{mez} = \frac{f_{AD}}{2} = \frac{10^5}{2} = 50\ kHz \tag{2.6}$$

Časová konstanta z rovnice 2.4 tedy bude

$$\tau = \frac{1}{2\pi \cdot 5 \cdot 10^5} = 3{,}18 \cdot 10^{-7}\ s \tag{2.7}$$

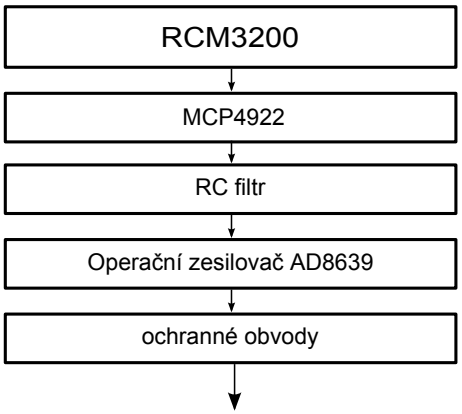
Z toho byly odvozeny přibližné hodnoty součástek dle dostupnosti na trhu 10 nF a 324 Ω. Vstupní odpor AD převodníku je dobré volit co nejmenší, neboť vzorkovač obsahující kondenzátor k udržení napěťové hodnoty by měl větší časovou konstantu. Mezní frekvence vybraných součástek bude:

$$f_{mez} = \frac{1}{2\pi \cdot R \cdot C} = \frac{1}{2\pi \cdot 324 \cdot 10^{-8}} = 49\ 121\ Hz \tag{2.8}$$

Na vstupu AD převodníku se nachází dva kanály, které jsou zapojeny přes multiplexer, proto by se maximální rychlost při použití obou kanálů dělila dvěma. Z tohoto důvodu byly použity dva AD převodníky MPC3202, aby se zachovala požadovaná vzorkovací frekvence 100 kS/s . Nevyužité kanály byly zapojeny pro měření napětí zdrojů.

2.4 Výstupní analogové obvody

Výstupní analogové zapojení popisuje obrázek 2.7 a příloha A.3. Tyto výstupní kanály se skládají z DA převodníku MCP4922, operačního zesilovače AD8639 a ochranného transilu proti přepětí.



Obr. 2.7: Blokové schéma výstupní analogové části

2.4.1 MCP4922

Tento DA převodník je opět 12bitový. Komunikace probíhá přes SPI rozhraní. Referenční napětí je pevně dané na 5 V, proto není možné měnit pomocí softwaru rozsah, a tím i přesnost. Referenční napětí by bylo možné měnit pomoci programovatelného zesilovače, jenž by měnil velikost referenčního napětí. Bylo by však zapotřebí dalších nepřesných součástek, jako je předdělička, což by zhoršovalo stabilitu referenčního napětí. DA převodník MCP4922 je schopen obnovovat napětí na obou výstupech současně rychlostí 100 kS/s.

2.4.2 RC filtr

RC filtr byl navržen opět dle rovnice 2.4. Mezní frekvence byla stanovena na polovinu vzorkovací frekvence MCP4922, tedy na 50 kHz. Hodnoty rezistoru a kondenzátoru dle dostupných parametrů byly zvoleny 150 Ω a 20 nF. Z rovnice 2.5 se vypočítá finální mezní frekvence

$$f_{mez} = \frac{1}{2\pi \cdot 150 \cdot 2 \cdot 10^{-8}} = 53\ 052\ Hz \tag{2.9}$$

2.4.3 Operační zesilovač

Operační zesilovač byl vybrán s předem definovanými požadavky. Jeden z hlavních požadavku je například rychlost přeběhu (Slew Rate - SR). Výpočet potřebné rychlosti přeběhu se je čerpán ze zdroje [23]. K tomuto výpočtu je zapotřebí znát rozdíl minimálního a maximálního výstupního napětí V_{pk} v absolutní hodnotě a požadované časové rozmezí přeběhu

$$SR = max \left| \frac{dv_{out(t)}}{dt} \right| \tag{2.10}$$

Pro náš případ je dt = 10 μs a maximální změna napětí za tuto dobu je 10 V. Potom:

$$SR = \frac{10}{10 \cdot 10^{-6}} = 1\ V/\mu s \tag{2.11}$$

Operační zesilovač AD8639 má dle dokumentace [22] rychlost přeběhu 2 $V/\mu s$. Operační zesilovač disponuje pásmovou propustností 1,5 MHz, která opět bohatě postačuje. Při nulové hodnotě na vstupu operačního zesilovače se může na výstupu objevit napětí řádově několika milivoltů, což je způsobeno nesymetrickým napájením.

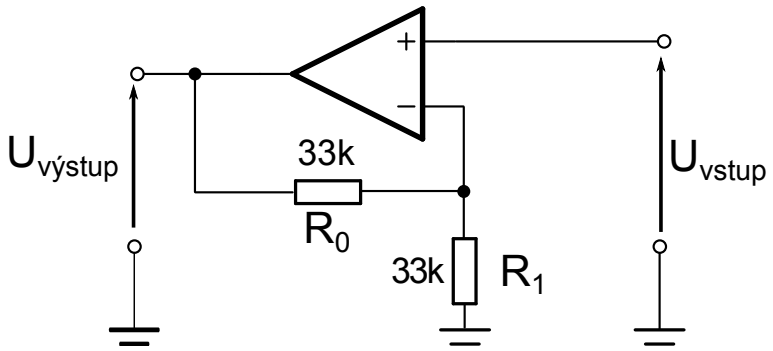
Operační zesilovač je zapojen v neinvertujícím zapojení se zesílením dva. Výpočet zesílení byl převzat z pramene [2].

$$K = \frac{U_0}{U_1} = \frac{R_1 + R_0}{R_1} = 1 + \frac{R_0}{R_1} \tag{2.12}$$

Proud procházející rezistory by neměl být příliš velký ani příliš malý kvůli šumu. Proto byly hodnoty rezistorů zvoleny zhruba 24 $k\Omega$ a 33 $k\Omega$. Zesílení tedy bude dle rovnice 2.12:

$$K = 1 + \frac{R_0}{R_1} = 1 + \frac{33 \cdot 10^3}{33 \cdot 10^3} = 2 \tag{2.13}$$

R_0 je na obrázku 2.8 ve skutečnosti rezistor o velikosti 24 $k\Omega$ zapojený sériově s trimrem, neboť díky nízké toleranci rezistorů bychom nemuseli získat přesné dvojnásobné zesílení (podrobné schéma v příloze A.3)



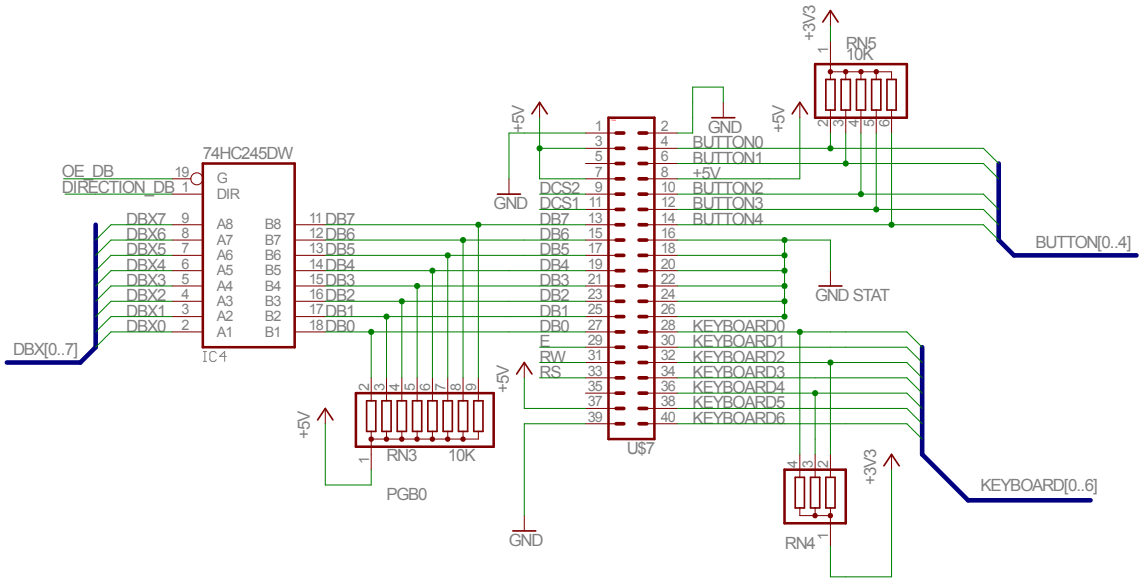
Obr. 2.8: Zapojení neinvertujícího zesilovače

2.5 Digitální VV port

Multifunkční karta nebude pouze generovat a snímat analogové signály, ale může generovat a přijímat digitální signál pomoci až osmi linek. Pokud karta generuje digitální signál, je potřeba zajistit, aby dané piny procesoru nebyly proudově přetíženy, proto zde byl zařazen budič sběrnice. Signalizující LED zobrazují aktuální napěťovou úroveň na dané lince. Digitální linky jsou chráněny jak proti přepětí, tak proti výskytu záporného napětí z vnějšího zdroje.

2.6 Uživatelské periferie

Uživatel bude moci nastavit pomocí maticové klávesnice a tlačítek parametry generovaného signálu, frekvenci atd. Veškeré tyto možnosti se budou zobrazovat na grafickém displeji, jenž bude vykreslovat naprogramované menu, jednoduchý graf či nastavené hodnoty. Maticová klávesnice i tlačítka jsou připojeny na NAND hradlo, odkud je výstup veden na pin modulu, na kterém lze nastavit přerušení reagující na změnu napěťové úrovně, a tím pádem dojde k detekci stisknutého tlačítka.



Obr. 2.9: Schéma zapojení sběrnice pro periferie

2.7 DPS

Počet desek plošných spojů byl stanoven na 2 z důvodu fixace displeje, tlačítek a maticové klávesnice. Aby se obě desky mohly k sobě snadno přimontovat, byla velikost DPS stanovena na šířku shodně 160 mm. Na výšku se již obě DPS liší, neboť je potřeba na svorkovnici umístěnou na spodní DPS přišroubovat jednotlivé signálové vodiče. Horní deska má na výšku 85 mm a spodní 94 mm. Spodní DPS by bylo samozřejmě možné navrhnout s menšími rozměry, avšak rozměry musely být přizpůsobeny dle horní desky, kde největší plochu zabírá displej s maticovou klávesnicí. Obě DPS byly navrženy v programu Eagle 5.9. Za nevýhodu tohoto softwaru můžeme pokládat rozmísťování zemí. V našem návrhu se nachází zem digitální, analogová, referenční a napájecí. Ve schématech jsou všechny tyto země rozlišeny podle značky země. Ve skutečnosti jsou tyto země na vhodném místě spojeny. Při návrhu samotné DPS bylo tedy potřeba dávat pozor, která zem se má kam vést. Některé součástky pro Eagle byly staženy z webové stránky http://paja-trb.cz/eagle/index.html.

2.7.1 Návrhová pravidla

Během návrhu DPS bylo dodržováno několik základních pravidel, z nichž většina byla čerpána z knihy [3]. Napájecí analogové a digitální součástky jsou od sebe odděleny . Napájecí spoje mají větší šířku než spoje signálové. Blokovací kondenzátory se nachází v těsné blízkosti dané součástky, aby nevznikaly velké proudové smyčky.

Oba napájecí spoje se vedou společně v těsné blízkosti, aby opět nedocházelo k vyzařování elektromagnetického pole do okolí. Veškeré spoje se lámou pod úhlem 45°, aby nedocházelo, hlavně u rychlých signálových spojů, k odrazům. DPS není určená k sériové výrobě, kde se součástky pájí nejčastěji vlnou, neboť některé piny jsou napojeny na plošný spoj z několika stran, a tak by zde docházelo k velkému tepelnému úbytku a součástka by se nemusela správně zapájet. Pro sériovou výrobu by se DPS musela poupravit. Kolem DPS jsou vedeny pásy, které mají chránit kartu před statickou elektřinou.

2.7.2 Oprava DPS

Některé piny modulu byly špatně zapojeny v návrhovém programu. Obousměrná digitální linka PGA7 byla zapojena na výstupní pin modulu s názvem PC7. Pin modulu PC7 je však jednosměrná vstupní linka a navíc se nachází na SPI lince A, jež slouží pro nahrávaní programu a také k ladění. Tvorba programu bez ladění by byla podstatně náročnější. Linka PGA7 byla tudíž přeložena na obousměrný digitální pin PG1, kde se původně nacházel hodinový signál E pro displej. Linka E byla přesměrována na výstupní pin PC4, neboť hodinový signál je pouze jednosměrný. Linka E se nachází na pinu od SPI linky B sloužící pro komunikaci s DA převodníkem MCP4922. Vyřešením této situace je správné využití CS. Další problém představovala linka BUTTON4, která vede od tlačítka, takže je vůči modulu jako vstupní linka, ale byla zapojena na PIN PC6, který funguje pouze jako výstupní pin a slouží pro nahrávání programu, jak již bylo uvedeno. Proto byla linka BUTTON4 přepájena na vstupní pin PC5. Pin PC5 je sice součástí SPI linky B, ale nevyužívá se, protože pro řízení DA převodníku stačí výstupní pin MOSI, DA převodník MCP4922 totiž nevysílá žádná potvrzovací data.

Opravená verze linek napojených na modul je znázorněna v příloze A.4. Obrázek původní verze není potřeba, poněvadž je zde pouhý rozdíl v zapojení tří linek.

Během zprovozňování osmi digitálních linek bylo potřeba odstranit pull - up rezistory, neboť LED dioda spolu s rezistorem splňuje funkčnost pull - down rezistoru, a tak tyto dvě varianty zároveň způsobovaly napětí v zakázaném pásmu.

Pří návrhu bylo opomenuto zapojení pinů $\overline{SHDN}$ a $\overline{LDAC}$ od MPC4922. Pin $\overline{SHDN}$ byl tedy dle kapitoly 2.4.1 připájen napevno k +5 V a $\overline{LDAC}$ ke GND.

Další problém nastal u operačního zesilovače. Původní operační zesilovač LT1352 (zdroj [21]) nebyl tzv. Rail to Rail, což znamená, že nebyl určen pro nesymetrické napájení, neboť minimální výstupní napětí bylo podstatně vyšší než záporné napájecí napětí. V našem případě je operační zesilovač napájen nesymetricky a tento operační zesilovač při nízkých hodnotách vstupního napětí vykazoval obrovskou chybu v podobě nelinearity. Operační zesilovač zesiloval vstupní signál správně až od hodnoty

0,5 V na vstupu, což bylo velmi nežádoucí. Proto byl tento operační zesilovač nahrazen jiným typem AD8639 [22], který je již Rail to Rail. Rozsah kladného napětí je od 5 do 16 V a slew rate udává výrobce 2 V/μs.

Poslední úpravou byla změna velikostí odporů v předděličce u programovatelného operačního zesilovače (OZ) MCP6S91. Původně zde byl celkový vstupní odpor 2 $M\Omega$. Takto vysoký odpor však způsoboval vysoký šum, jak se později ukázalo, a proto byla hodnota rezistorů zmenšena přibližně na polovinu.

U analogových částí bylo potřeba co nejlépe pájet součástky, jelikož v průběhu softwarového oživování se projevoval výrazný šum. Například cín na jednom rezistoru nebyl dostatečně prohřátý, a tak zde docházelo k nežádoucímu šumu. Po správném prohřátí se již šum do jisté míry eliminoval.

3 DYNAMIC C A RCM3200

3.1 Příprava pro nahrání programu do modulu

Modul RCM3200 je naprogramován ve vývojovém prostředí Dynamic C verze 9.62. Součástí modulu je také programovací kabel, na kterém probíhá komunikace přes sériovou linku, která se již vytrácí z počítačů, a proto byl použit převodník ze sériové linky na USB. Na druhé straně programovacího kabelu jsou dva konektory. K modulu se připojuje konektor s názvem „PROG“ a to tak, že červená linka kabelu směřuje k bližšímu okraji modulu (viz zdroj [18] na straně 12.). Poté již stačí pouze nastavit rychlost sériové linky (v našem případě to bylo 115 200 kb/s) a port sériové linky. Nyní je vše připraveno pro první nahrání programu do modulu. Stačí napsat funkci main, zkompilovat a přenést do modulu. Pokud nechceme aplikaci ladit, postačí po nahrání programu odpojit programovací kabel a resetovat procesor.

3.2 Syntaxe Dynamic C

Programovací prostředí je založeno na klasickém programovacím jazyce C až na pár výjimek, které budou také částečně popsány. Změna registrů probíhá přes funkci WrPortI() a není tudíž možné přidávat hodnotu registru pomocí rovnítka; například registr PEDR se vynuluje takto: WrPortI (PEDR, &PEDRShadow, 0x00). Pro čtení funguje analogický princip. V tomto případě představuje proměnná PEDRShadow aktuální stav portu PEDR. Při rozložení na assemblerovský si můžeme všimnout, že při zápisu do registru PEDR se okamžitě uloží daná hodnota i do dané proměnné. Pokud chceme zjistit hodnotu PEDR registru, zjistíme ji jedině právě z proměnné PEDRShadow.

Dynamic C nikdy nepoužívá funkci malloc pro dynamické proměnné, ale má opět vlastní funkce xalloc, xsetint a xgetint. Modul RCM3200 je vybaven 1 MB pamětí, což činí adresaci jednoho bajtu v paměti pomocí dvaceti bitů. Funkci xalloc se předá velikost paměti, která má být rezervována, xalloc vrací jako návratovou hodnotu počáteční fyzickou adresu. Tato počáteční fyzická adresa se využívá pro funkci xsetint (uložení int hodnoty) a xgetint (načtení int hodnoty). Při rozložení na assemblerovský kód bychom zjistili, že k této operaci slouží pouze instrukce ldp, kde registr A určuje hodnotu 4 nejvyšších bitů a zbývajících 16 bitů je určeno registry dle instrukce (IX, HL, IY).

Dynamic C je také specifický, co se týče knihoven. Běžný program se dělí na hlavičkový soubor a soubor se zdrojovým kódem. V Dynamic C vyskytují pouze zdrojové kódy s koncovkou lib a tyto soubory se musí nacházet ve složce

C:\\DCRABBIT9.62 \\Lib mezi ostatními knihovnami. V našem případě jsou veškeré knihovny uložené ve složce \MyLib. Pro načtení knihovny do programu je potřeba použít syntaxi #use nazev.lib. Za povšimnutí také stojí deklarace funkcí pro prekompiler. Aby prekompiler detekoval všechny funkce použité mimo knihovnu správně, je potřeba v knihovně na začátku použít syntaxi /*** BeginHeader NazevFunkce1, NazevFunkce2 */. Viz zdrojové kódy v příloze I.1.

3.3 Možnosti ladění zdrojového kódu

Jakmile je program nahrán do paměti v procesoru, potřebujeme ve většině případů sledovat, ve které části zdrojového kódu se procesor modulu právě nachází, jaké jsou hodnoty proměnných či registrů nebo kde dochází k chybě. Po přehrání strojového kódu do paměti procesoru stačí ponechat zapojený programovací kabel, který komunikuje s procesorem přes SPI rozhraní A (procesor má ještě linky pro SPI rozhraní B, C, D). V programu můžeme běžně krokovat, zjistit aktuální hodnoty jednotlivých proměnných, hodnoty v zásobníkové paměti, hodnoty jednotlivých registrů nebo příznakových flagů. Možnost ladění strojového kódu je nezbytný předpoklad pro rychlý vývoj softwarové části. Všech pět ladících nástrojů je zobrazeno v příloze C.1.

Pro ladění funkcí je potřeba před danou funkci vložit klíčové slovo debug, neboť Dynamic C poté přidá svoji vlastní instrukci „rst 0x28“ za každou instrukci zdrojového kódu. Tato nově vložená instrukce zajistí, aby procesor nepokračoval na další instrukci a čekalo se na příkaz od programátora. Slovo debug se tedy hodí pro ladění, v ostatních případech se používá klíčové slovo nodebug, kdy se program sice nedá ladit, ale na druhou stranu to zrychlí běh provádění operací, protože každá nově vložená instrukce pro ladění stojí několik hodinových cyklů.

3.4 Assembler Dynamic C

V Dynamic C se může samozřejmě programovat přímo ve strojovém kódu, který výrazně zrychlí běh programu, ale na druhou stranu značně ztíží samotné programování. V programu pro multifunkční kartu se programovaly v assembleru nejkritičtější úseky, kde bylo potřeba dodržet operaci za určitý čas. Typickým příkladem je načtení a odeslání dat do DA převodníku, přijmutí a uložení hodnot z AD převodníku pomocí SPI rozhraní. Před programováním v assembleru je nejdříve potřeba nastudovat architekturu procesoru: jaké má k dispozici registry, základní instrukce, jako jsou operace s paměti či registry, jak se vynásobí dvě čísla typu int a tak dále.

Obrovská výhoda assemblerovského kódu spočívá v přesně definovaných hodinových cyklech na instrukci. Procesor modulu běží na frekvenci 44 MHz, a tak

za jednu μs se na procesoru stihne vygenerovat 44 hodinových cyklů. Tato výhoda byla využita v „delay“ funkci, která se nachází v souboru Delay.lib.

Pokud programátor netuší, jak by se daná operace provedla v assemblerovském kódu, potom stačí napsat kód v jazyce C, odeslat do modulu a pomocí ladícího nástroje „Disassembled Code“, příloha č. (C.1), si zjistit jednotlivé instrukce.

3.4.1 Ukázka strojového kódu v Dynamic C

Níže je ukázán jednoduchý strojový kód, jenž má za úkol porovnat, zda se hodnota v zásobníkové paměti, na kterou ukazuje SP, nerovná více než 100.

Do registru hl se načte hodnota ze zásobníkové paměti, na jejíž adresu ukazuje SP (proměnná value). Poté se prohodí pomocí instrukce ex registry hl a de. Do registru hl se uloží hodnota 100. Instrukce „or a;“ způsobí vynulování carry flagu. Instrukce „sbc hl,de;“ zapříčiní odečtení registru de a carry flagu od hl a uložení do registru hl. Pokud hodnota registru de byla větší než hl, potom se nastaví carry flag do jedničky. Tím pádem instrukce „sbc hl, hl“ uloží do registru hl nenulovou hodnotu. Instrukce „bool hl“ vyhodnotí, zda je registr hl nenulový a podle toho i nastaví jak hl, tak i zero flag. Podle toho jakou hodnotu má zero flag, dojde ke skoku na dané navěští nebo se bude pokračovat na dalším řádku strojového kódu. Zde vidíme, jak složitě se dá napsat ve strojovém kódu jedna požadovaná operace ve tvaru: if (value>100) goto ...

Příklad strojového kódu:

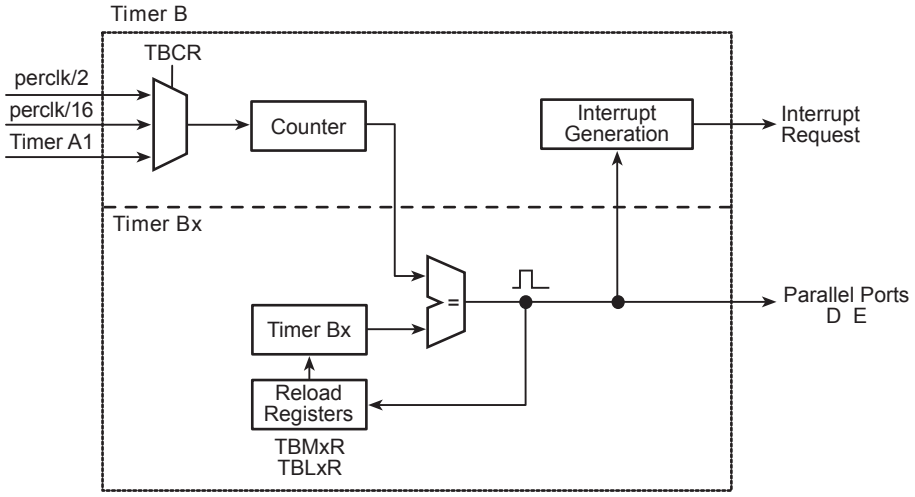
```
ld      hl, (SP);
ex      de, hl;
ld      hl, 100;
or      a;
sbc     hl, de;
sbc     hl, hl;
bool    hl;
jp      z, next;
```

4 VLASTNÍ KNIHOVNY V DYNAMIC C

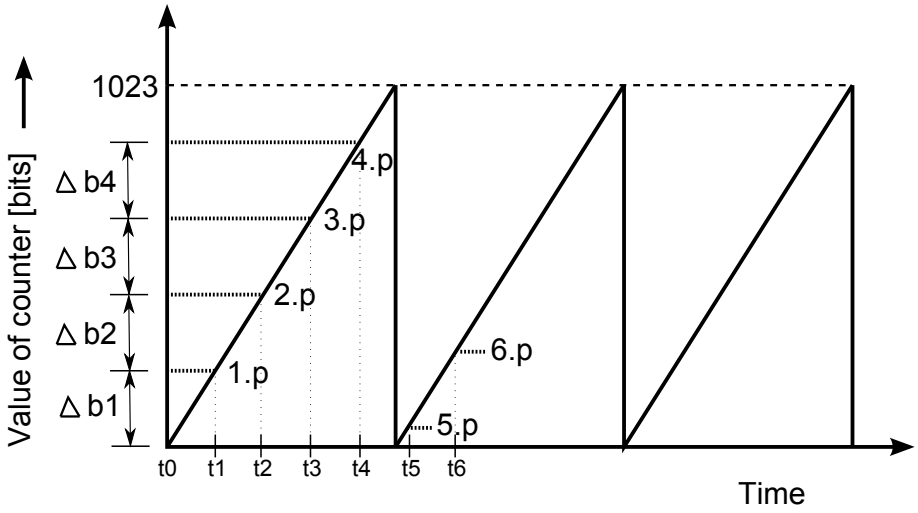
Pro jednotlivé hardwarové komponenty bylo potřeba vytvořit knihovny, pomocí nichž by modul komunikoval s periferiemi. Pro komunikaci s periferiemi byl základní předpoklad nastudovat jednotlivé dokumentace, které obsahují veškeré nezbytné informace ohledně osazení do DPS a komunikace s procesorem. Tato kapitola se zabývá nejen hardwarovými periferiemi, ale i samotnými knihovnami pro procesor Rabbit. Zde tedy bude popsán časovač, generování přerušení reagující na změnu hrany na pinu. Také zde bude ve zkratce uveden zjednodušený popis vytváření menu pro displej, komunikace přes ethernetový kabel a ošetření chyb. Veškeré tyto knihovny jsou k dispozici u přiloženého CD v příloze I.1.

4.1 TimerB.lib

Dle schématu na obrázku 4.1 vidíme základní funkčnost časovače. Do registru TBCR se ukládá do nejnižších dvou bitů priorita přerušení, další dva bity registru TBCR nastavují multiplexor určující zdroj pulsů, jak je znatelné ze schématu. Tento registr tedy určuje, s jakou rychlostí se bude inkrementovat čítač. Díky tomuto registru se tedy mění strmost nástupních hran pilovitého signálu na obrázku 4.2. Čím vyšší strmost, tím rychleji přeteče čítač. Z tohoto obrázku je také patrné, že při přetečení čítače, jenž má deseti bitovou hodnotu (registry TBCMR a TBCLR), se automaticky registry vynulují a čítač pokračuje dále. Hodnoty čítače se mohou pouze číst a nikoli zapisovat. Z toho plyne, že není možné čítač vynulovat dle své potřeby. Pokud je povoleno přerušení od časovače B, potom registry TBM1R a TBM2R (tzv. match registry) slouží jako hodnoty pro komparátor, bude-li se hodnota těchto dvou registrů rovnat hodnotě čítače, vygeneruje se přerušení. Při přerušení je potřeba změnit hodnoty registrů TBM1R a TBM2R, neboť časově rozmezí mezi dvěma přerušeními by pak bylo ovlivněno pouze hodnotou registru TBCR. Podrobný popis jednotlivých registrů se nachází v dokumentaci [5] od strany 115. Na obrázku 4.2 znamená písmeno p přerušení (komparátor v tu chvíli určil, že hodnoty match registrů a registrů od čítače se rovnají). Všimněme si, že časový interval mezi dvěma přerušeními je vždy stejný, díky zajištění: $\Delta b1 = \Delta b2 = \Delta b3$ ...



Obr. 4.1: Blokové schéma časovače Timer B [5]



Obr. 4.2: Graf znázorňující princip šesti přerušení v pravidelných intervalech

Část implementované funkce pro obsluhu přerušení byla převzata ze zdroje [4]. Nejdříve se uloží hodnoty veškerých registrů do zásobníkové paměti, odkud se opět na konci funkce uloží zpět do registrů. Pokud by se tento krok vynechal, mohlo by dojít k selhání programu, jelikož by se změnily hodnoty registrů funkce, ve které došlo k přerušení. Například by došlo k zápisu do náhodné paměti. Poté proběhne aktualizace „match“ registrů. Proměnná count_timer slouží v programu jako pomocná dělička. Obsluha tohoto přerušení potřebuje na zpracování 100 hodinových cyklů, tudíž se přerušení pro 10 μs nevyužívá, ale v programu je názorně ukázáno, jak by vypadalo nastavení jednotlivých registrů, včetně proměnné count_timer. Pro zjednodušení bylo předem definováno, aby periodu jednoho vzorku (ať už při snímání

nebo generování) bylo možno nastavit v rozsahu 10 μs až 10 s, kdy další hodnoty k výběru periody jsou násobky deseti: 100 μs, 1 ms, ...

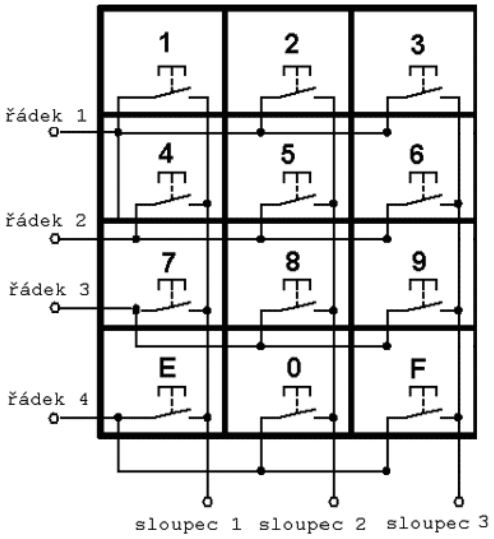
4.2 IntKeypressed.lib, vstup od uživatele

Zkratka Int v tomto případě znamená interruption, tudíž tato knihovna obsluhuje přerušení při stisknutí tlačítka nebo klávesnice uživatelem. Výstupy veškerých tlačítek jsou vedeny dle schématu v příloze A.4 do Nand členu, jehož výstup je veden do modulu na pin INT0, kde se detekuje změna napěťové úrovně. V registru I0CR se nastavuje v nejnižších dvou bitech priorita přerušení. V dalších dvou bitech se pak určí na jakou hranu bude přerušení reagovat. Uživatel tedy stiskne libovolné tlačítko, následně se vygeneruje přerušení a dojde k zavolání funkce int0_isr, kde se na začátku uloží jednotlivé registry do zásobníkové paměti, stejně jako to bylo u přerušení od časovače. Poté se zkontroluje výstupní hodnota všech tlačítek, včetně maticové klávesnice, a do proměnné btn_value se uloží zakódovaná hodnota tlačítka. Nakonec se zpátky načtou hodnoty registrů ze zásobníkové paměti.

4.2.1 Keyboard.lib, maticová klávesnice

Tato knihovna je využita pouze v knihovně IntKeypressed, proto je tato kapitola podpodkapitolou. Tato knihovna mohla být součástí knihovny IntKeypressed, ale pro přehlednost jí byl přidělen vlastní soubor. Funkce keyboard_read vrací zakódovanou hodnotu stisknutého tlačítka (kódování je určeno pomocí #define).

Maticová klávesnice se skládá ze 7 pinů, z nichž čtyři slouží jako vstup/výstupy řádků a tři jako vstup/výstupy sloupců. Význam jednotlivých pinů, které přísluší danému sloupci či řádku, je patrný z obrázku 4.3. V implementované knihovně jsou sloupce brány jakou výstupní piny, řádky jakou vstupní piny. Pokud není na začátku stisknuté žádné tlačítko, potom veškeré sloupce díky pull-up rezistorům vykazují logickou jedničku. Na vstupech (řádcích) jsou celou dobu samé nuly. Jakmile se zmáčkne jedno tlačítko, potom na výstupu jednoho sloupce detekuje procesor logickou nulu. Víme sice sloupec, ale nevíme řádek. Proto funkce keyboard_read postupně otestuje řádek po řádku, jinak řečeno vždy bude jeden vstup nulový a ostatní vstupy jedničkové. V okamžiku, kdy máme nulový pouze jeden vstup a detekujeme na výstupu také nulu, jsme schopni zjistit o jaké tlačítko se jedná. Bude-li zmáčknuto více tlačítek najednou, detekuje se první shora a zleva, ostatní jsou ignorována.



Číslo pinu	SYMBOL
1	Sloupec 2
2	Řádek 1
3	Sloupec 1
4	Řádek 4
5	Sloupec 3
6	Řádek 3
7	Řádek 2

Obr. 4.3: Princip maticové klávesnice a popisy pinů [15], [16]

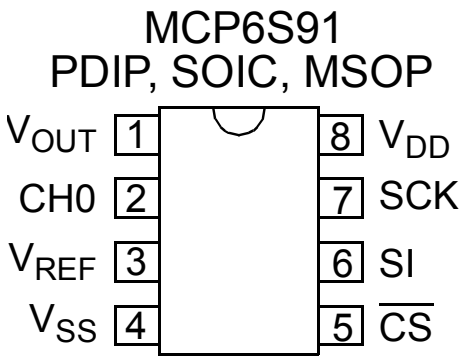
4.3 AD.lib

Knihovna obsahuje zdrojové kódy pro celý analogový vstup: programovatelný zesilovač a AD převodník. Než se začne snímat napětí na vstupu, je potřeba nastavit zesílení programovatelného zesilovače a teprve potom zaměstnat samotný AD převodník. U samotného získávání naměřených dat a následného ukládání s frekvencí vzorkování 100 KS/s již vyvstal problém. Původní koncepce byla snímat oběma kanály zároveň. Nejdříve byl celý kód naprogramován v jazyce C pouze pro jeden kanál s nastavením maximální rychlosti na SPI lince. Dle naměřených dat z předem známého signálu na vstupu se rychlost snímání pohybovala okolo 2 KS/s. Nezbývalo tedy nic víc než zdrojový kód pro tuto náročnou smyčku naprogramovat v assembleru. Jak již bylo řečeno v kapitole 3.4, v assemblerovském jazyce je možnost spočítat časovou náročnost jednotlivých instrukcí jako celku. Tudíž po naprogramování a úspěšném snímaní byla analyzována časová náročnost této smyčky. Odeslání příkazu pro snímaní AD převodníku, získání naměřených hodnot, uložení do fyzické paměti, kontrola, zda uživatel nezrušil snímání (stisk libovolného tlačítka), inkrementace ukazatelů a reset CS od AD převodníku trvá zhruba 400 hodinových cyklů, my však máme k dispozici pouze 440 hodinových cyklů na 10 μs. Z toho bylo tedy usouzeno, že není reálné snímat při maximální vzorkovací periodě 100 KS/s na více kanálech zároveň. Totéž platí i pro DA převodník. Pro snímání napětí byla vytvořena funkce pro každý kanál zvlášť, neboť vytvářet rozhodování (if-else) v kritické smyčce, například jaký CS aktivovat nebo jakou SPI linku použít, by výrazně zvý-

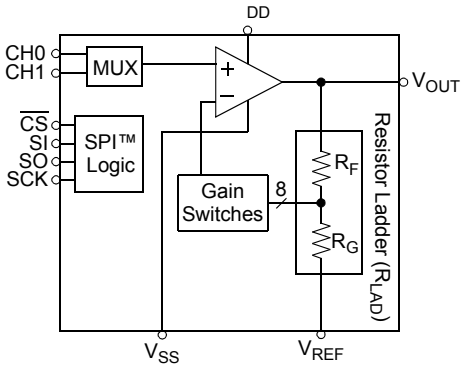
šilo časovou náročnost. Proto jsou tyto dvě funkce totožné až na pár hodnot. Během snímání (stejně tak generování) se nekontrolují příchozí data od PC, protože by tato operace zabrala další instrukce a tím pádem drahocenný čas.

4.3.1 Programovatelný zesilovač MCP6S91

Úkol tohoto čipu je zesílit napětí na vstupu multifunkční karty na požadovanou hodnotu z důvodu zvýšení přesnosti převodu na AD převodníku. Pro komunikaci přes SPI linku byly vyňaty zdrojové kódy z již implementované knihovny spi.lib od Dynamic C.



Obr. 4.4: Piny MCP6S91 [20]



Obr. 4.5: Schéma MCP6S91 [20]

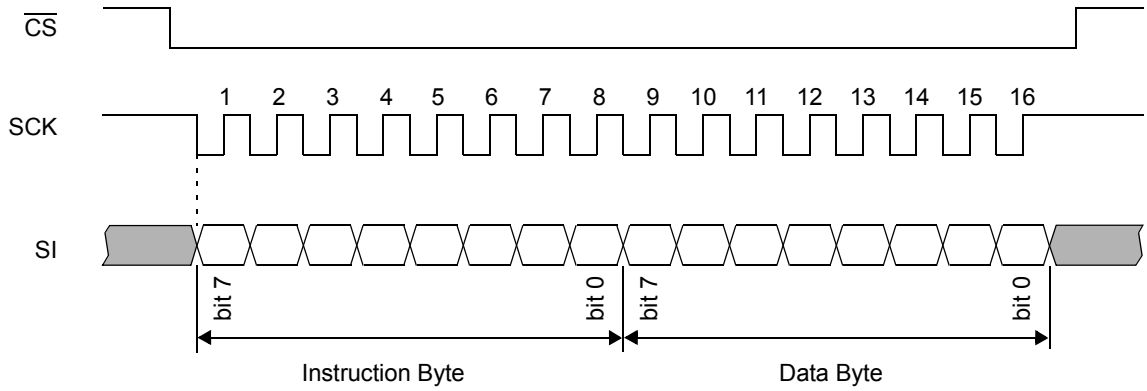
Vysvětlení pinů:

V_{out}	výstupní kanál zesíleného vstupního signálu
CH0	vstupní kanál napěťového signálu
V_{ref}	Externí referenční pin (= V_{ss} = GND)
V_{ss}	Záporné napájení (GND)
$\overline{CS}$	Chip Select (de/aktivace komunikace přes SPI)
SI	vstupní linka od SPI
SCK	hodinové impulsy
V_{dd}	Kladné napájení (+5 V)

Programovatelný zesilovač slouží ke zvýšení přesnosti měření menších napětí. Pokud tedy uživatel bude dopředu znát maximální měřenou hodnotu na vstupu, pak může signál díky této informaci patřičně zesílit a výsledek díky menšímu kvantizačnímu šumu zpřesnit. Každý vstupní signál je na vstupu kvůli vstupní děličce zmenšen na polovinu. Pokud by tedy uživatel věděl, že se napětí na analogovém

vstupu bude pohybovat v rozmezí do 1 V, může nastavit zesílení deset, takže na výstupu programovatelného zesilovače dostane maximální žádanou hodnotu do 5 V (maximální možné napětí na vstupu AD převodníku).

Komunikace probíhá přes rozhraní SPI. V našem případě se jedná o jednostrannou komunikaci, kdy procesor odesílá pouze 2 bajty a žádné nepřijímá, jak vypovídá obrázek 4.6



Obr. 4.6: Komunikace s MCP6S91 přes SPI [20]

První bajt se nazývá instruction byte a vypadá:

M2	M1	M0	-	-	-	-	A0
bit 7							bit 0

bity 7-5: **M2-M0**
001 = vypnutí zesilovače
010 = zápis do registru bytu, který bude následovat
Ostatní kombinace: zatím nemají využití
bity 4-1: zatím nemají využití
bit 0: zde se adresuje kanál, pro náš jednokanálový typ však nemá využití

Po instruction bytu následuje bajt pro gain register, který určuje celkové zesílení pomocí tří bitů. V našem programu se využívá rozsah zesílení 1 až 10 (rozsah 10 V - 1 V maximálního napětí na vstupu).

-	-	-	-	-	G2	G1	G0
bit 7							bit 0

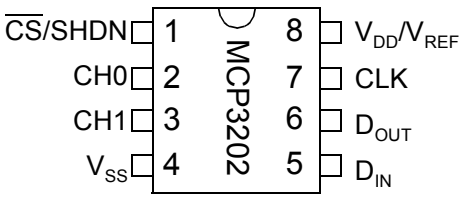
bity 7-3: zatím nemají využití

bity 2-0: **G2-G0:** bity určující velikost zesílení

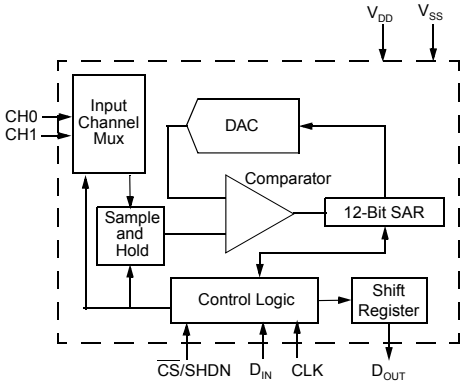
000 = zesílení +1
001 = zesílení +2
010 = zesílení +4
011 = zesílení +5
100 = zesílení +8
101 = zesílení +10
110 = zesílení +16
111 = zesílení +32

Před použitím samotného AD převodníku tak musí uživatel specifikovat maximální velikost vstupního napětí, bez specifikace je nastavení zesílení +1 (tedy pro rozsah 10 V). První odeslaný bajt od procesoru má vždy hodnotu 0x40, tudíž se zapisuje jenom do registru. Další bajt je dle volby uživatele.

4.3.2 převodník MCP3202



Obr. 4.7: Piny MCP3202 [28]



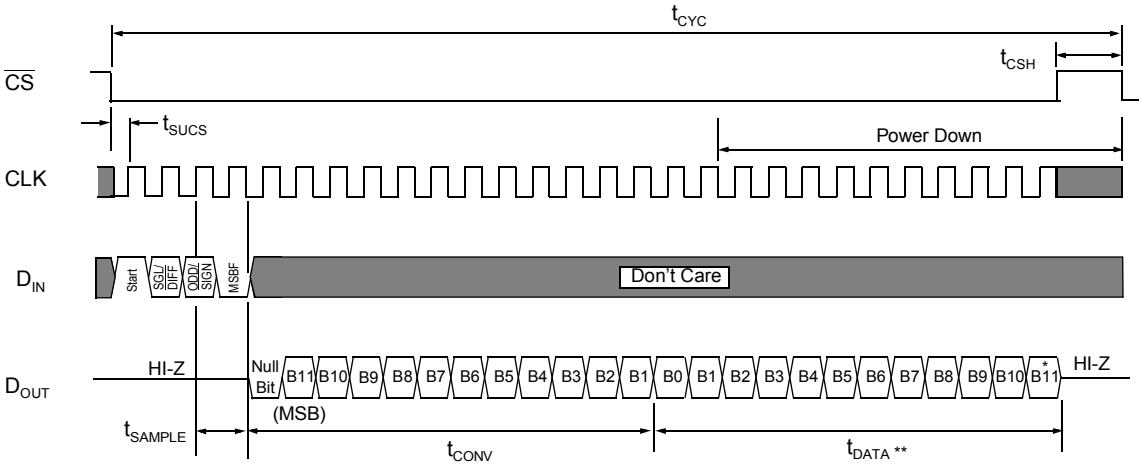
Obr. 4.8: Schéma MCP3202 [28]

Vysvětlení pinů:

V_{dd}/V_{ref}	Napájecí a zároveň referenční napětí
CH0	1. vstupní kanál napěťového signálu
CH1	2. vstupní kanál napěťového signálu
CLK	hodinové impulsy pro SPI
D_{in}	Vstupní linka pro SPI
D_{out}	Výstupní linka pro SPI
V_{ss}	Záporné napájení (GND)

$\overline{CS}/SHDN$ Chip Select (de/aktivace komunikace přes SPI)/vypnutí vstupů

Na obrázcích 4.7 a 4.8 vidíme, že pro komunikaci s procesorem pomocí SPI rozhraní nám bohatě stačí linka $\overline{CS}$ a linky CLK, D_{out}, D_{in}. Pokud chceme procesorem vyčíst aktuální napěťovou hodnotu na vstupu z jednoho kanálu, potřebujeme nejdříve úroveň $\overline{CS}$ od AD převodníku nastavit na nulu (aktivace AD převodníku) a poté odeslat jeden bajt od procesoru (master). Při každé vyčtené hodnotě je potřeba deaktivovat a aktivovat $\overline{CS}$ AD převodníku pro zahájení další komunikace. V prvním bajtu odesílá procesor dle obrázku 4.9 a tabulky 4.1 čtyři nastavovací bity, které určují, v jakém zapojení má DA převodník měřit (bit SGL/ $\overline{DIFF}$), zda v diferenciálním nebo vůči zemi. V našem případě se jedná o zapojení vůči nule, tudíž další bit ODD/$\overline{DIFF}$ specifikuje, z jakého kanálu se mají data načíst (hodnota v nule je kanál č.0, hodnota v jedničce je kanál č.1). Poslední bit se týká pořadí odeslaných bitů z AD převodníku, mají-li se odesílat jako MSB (bit MSBF = 1) nebo jako LSB (bit MSBF = 0). AD převodník je 12bitový, tudíž se data z převodníku odesílají ve dvou bajtech, kde 4 bity jsou nepodstatné.



Obr. 4.9: Komunikace s MCP3202 přes SPI[28]

-	-	-	-	Start bit	SGL/$\overline{\text{DIFF}}$	ODD/$\overline{\text{SIGN}}$	MSBF
bit 7							bit 0

Tab. 4.1: Bajt určený k odeslání pro MCP3202 (příkaz k měření)

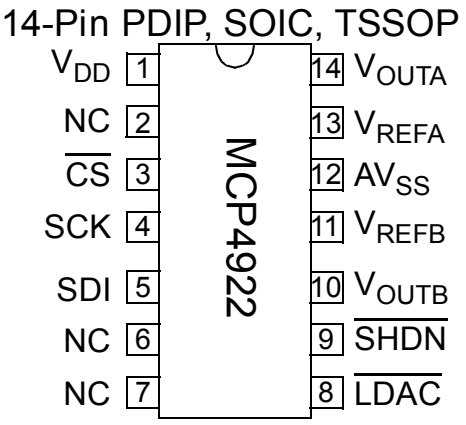
bity 7-4: zatím nemají využití
bit 3: **Start bit:** vždy hodnota 1
bit 2: **SGL/ $\overline{\text{DIFF}}$:** nastavení měření buď vůči zemi (1) nebo v dif. zapojení (0)
bit 1: **ODD/ $\overline{\text{SIGN}}$:** výběr kanálu (0 = 1. kanál, 1 = 2. kanál)
bit 0: **MSBF:** určení pořadí příchozích bitů od AD

1.byte	Null bit	B11	B10	B9	B8	B7	B6	B5
2.byte	B4	B3	B2	B1	B0	-	-	-
	bit 7							Bit 0

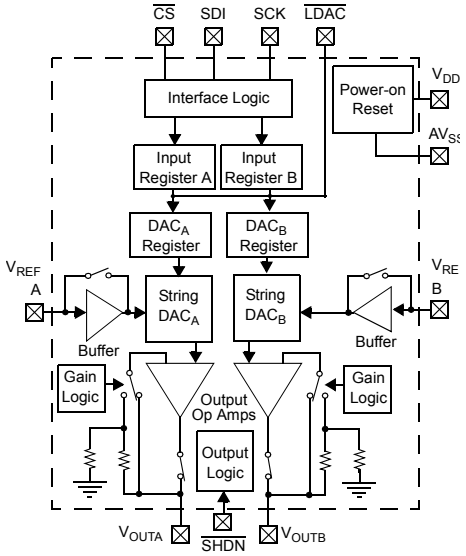
Tab. 4.2: Přijaté bajty naměřených dat od MCP3202

4.4 DA.lib, převodník MCP4922

Slouží ke generování výstupního napětí v rozsahu od 0 V do 5 V.



Obr. 4.10: Piny MCP4922 [29]

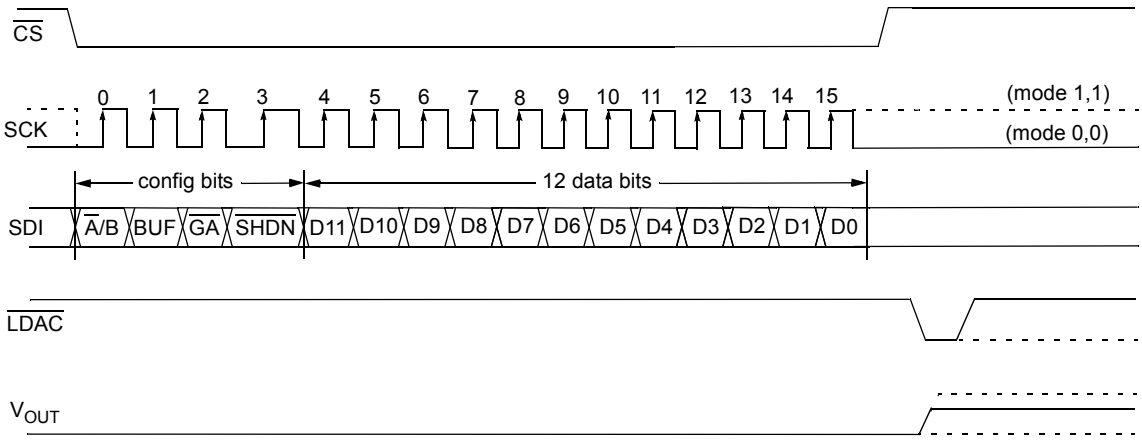


Obr. 4.11: Schéma MCP4922 [29]

Pin	**Interpretace**
V_{dd}	Napájecí napětí
$\overline{CS}$	Chip Select (de/aktivace komunikace přes SPI)
SCK	Hodinové impulsy pro SPI
SDI	Vstupní linka pro SPI
$\overline{LDAC}$	Slouží pro synchronizaci aktualizace registrů
$\overline{SHDN}$	Vypnutí převodníku
V_{OUTB}	Výstupní linka kanálu B
V_{REFB}	Referenční napětí pro pro DA převodník kanálu B
V_{OUTA}	Výstupní linka kanálu A
V_{REFA}	Referenční napětí pro pro DA převodník kanálu A
NC	Není využit (not connected)
AV_{ss}	Záporné napájení (GND)

Tab. 4.3: Vysvětlení pinů od MCP4922

Pro nastavení MCP4922 nám opět stačí jednosměrná komunikace. Jelikož je AD převodník 12bitový, potřebujeme odeslat přes SPI rozhraní minimálně 2 bajty. V prvním bajtu se nachází čtyři horní bity pro AD převod spolu se čtyřmi nastavovacími bity, další bajt obsahuje zbývajících 8 bitů pro AD převod, jak vyplývá z obrázku 4.12. V našem případě bylo $\overline{LDAC}$ připojeno k nule, neboť není potřebná synchronizace. K pinu $\overline{SHDN}$ je natrvalo připojena logická jednička, jelikož chceme vždy DA převodník zapnutý, v opačném případě by nám linka zabírala místo na portu modulu.



Obr. 4.12: Komunikace s MCP4922 přes SPI[29]

Popis bitů na obrázku 4.12

bit 15: $\overline{A}/B$: Výběr kanálu (0 = kanál A, 1 = kanál B)

bit 14: BUF: nastavení sledovače před vstupem ref. napětí (1 = nastaven)

bit 13: $\overline{GA}$: zesílení výstupního napětí(0 = žádné zesílení,1 = 2x zesílení)

bit 12: $\overline{SHDN}$: aktivace výstupu (0 = výstup odpojen, 1 = výstup aktivní)

bity 11-0: D11-D0: bity určující hodnotu na výstupu DA převodníku

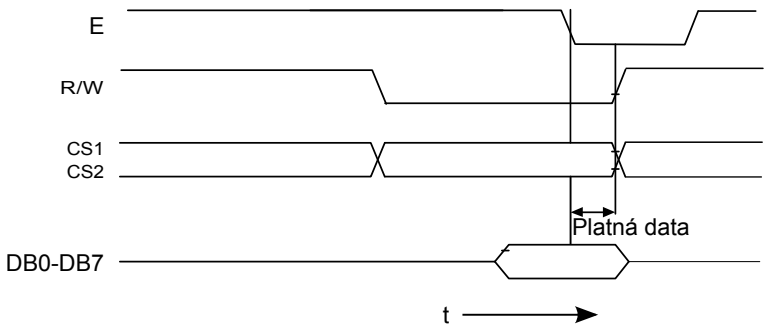
V naší knihovně je bit BUF vždy nastaven do nuly, neboť si nepřejeme, aby byl zapojen sledovač na referenčním napětí. Sledovač totiž snižuje maximální výstupní napětí a zvětšuje minimální výstupní napětí, na obě strany to činí 0.04 V. Sledovač se hodí v případě, pokud chceme vysoký vstupní odpor pro referenční napětí. I bez sledovače ale vstupní odpor zůstává 165 kΩ, což pro náš případ postačuje. Bit $\overline{GA}$ je vždy nastaven do jedničky, neboť referenční napětí dosahuje hodnot napájecího napětí DA převodníku, a tudíž není potřeba zesílení. Bit $\overline{SHDN}$ je také nastaven do logické jedničky, poněvadž chceme vždy aktivní výstup.

4.4.1 Display.lib, displej s řadičem KS108

Displej má rozlišení 128 x 64 pixelů a vstup čítá 20 pinů. Pin RST je připojen k nule, to znamená neaktivní reset, díky tomu byl ušetřen jeden pin na portu modulu. Místo resetu byla implementována funkce, která provede zapnutí a vypnutí displeje, jeho nastavení a následné vymazání RAM obsahující aktuální hodnotu jednotlivých pixelů. Výstupní napětí V_{EE} slouží jako zdroj záporného napětí, neboť pro nastavení kontrastu se využívá napětí záporné vůči zemi. Změřené V_{EE} se pohybovalo okolo -9.5 V a pomocí děliče se na vstup V_0 přivádělo napětí okolo -5.3 V (aktuální nastavení optimálního kontrastu). Dle obrázku 4.13 vidíme, že data se stávají aktivní od příchodu sestupné hrany na pinu E, poté musí být data alespoň 20 ns platná pro načtení dat ze sběrnice řadičem. Pin RS definuje, zda se bude pracovat s pamětí RAM nebo s instrukčními registry (hodnota nula znamená práci s instrukčními registry). Pin R/W značí, jestli se bude z řadiče číst nebo zapisovat.

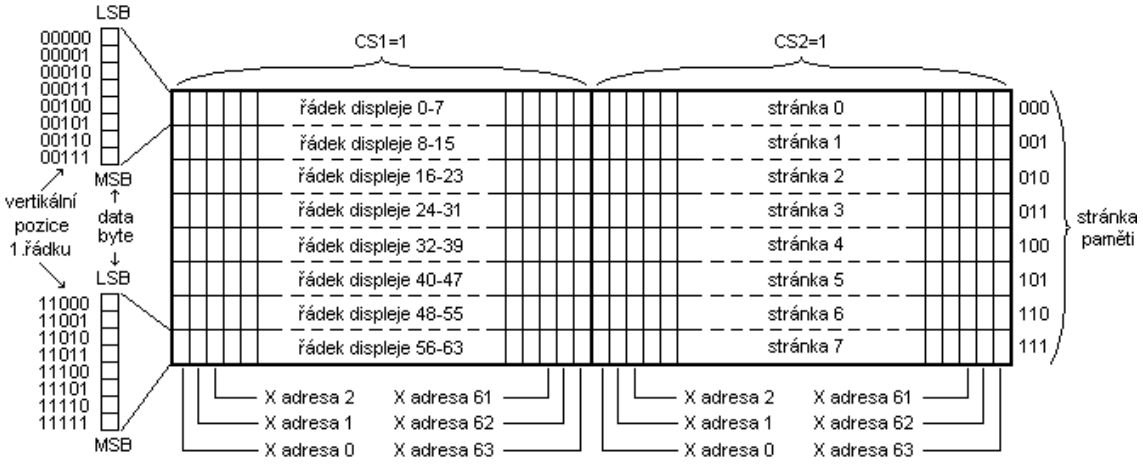
PIN	Symbol	Interpretace
1	V_{SS}	Záporné napájecí napětí (GND)
2	V_{DD}	Kladné napájecí napětí (+5 V)
3	V_0	Vstupní napětí určující kontrast displeje (záporné vůči zemi)
4	RS	Aktivace instrukčního registru nebo datové paměti (RAM)
5	R/W	Zápis nebo čtení do registru nebo paměti
6	E	Hodinový signál: aktivace platných dat na sestupnou hranu
7-14	DB0-DB7	Datová sběrnice
15	CS1	Chip Select pro levou polovinu displeje
16	CS2	Chip Select pro pravou polovinu displeje
17	RST	Reset displeje
18	V_{EE}	Záporné výstupní napětí pro další použití
19	K	Napájecí napětí pro katodu podsvícení (+5 V)
20	A	Napájecí napětí pro anodu podsvícení (GND)

Tab. 4.4: Vysvětlení pinů od displeje s řadičem KS108



Obr. 4.13: Zápis do 2. řadiče od displeje

Displej je rozdělen na 2 poloviny, jak můžeme vidět na obr. 4.14, přičemž každá polovina má svůj CS, vlastní instrukční registr a svoji paměť RAM, která dosahuje velikosti 4096 bitů (64 x 64 pixelů), neboť jeden pixel vždy odpovídá jednomu bitu v RAM. Hodnoty v RAM tedy odpovídají aktuálním hodnotám vykreslených pixelů. Displej má zabudovaný 1 čítač (=X čítač) a 2 registry (Y a Z registry), které můžeme měnit. Jedna stránka se skládá z osmi hodnot, což odpovídá osmi řádkům. Pří zápisu nebo čtení z RAM musíme tedy měnit vždy 8 pixelů, což činí jeden bajt. Registr Z určuje vertikální posunutí zobrazení pixelů vůči RAM. Pokud tedy hodnota registru Z bude nulová, potom souřadnice daného bitu v RAM odpovídá souřadnici pixelu na displeji. Pokud ovšem bude hodnota registru Z například 10, potom 1. řádek v paměti RAM odpovídá 10. řádku na displeji. Tento registr Z se tedy hodí pro rolování menu.



Obr. 4.14: Uspořádání RAM v řadiči KS0108 [14]

Na začátku je potřeba displej inicializovat. V inicializační části jsou oba CS aktivní. Displej se nechá převést do vypnutého režimu, poté se opět zapne, což provádíme kombinací hodnot pinů dle prvního řádku v tabulce 4.5. Následně se vynulují X čítač a Y registr a dochází k zápisu samých nul do paměti RAM na základě hodinových impulsů od pinu E. Inkrementace registru Y (změna stránky) se musí provádět v procesoru, viz 3. řádek v tab.4.5. Po vymazání celé RAM paměti se opět vynuluje X čítač a Y registr a displej je připraven pro zápis dat. Mezi jednotlivými zápisy nebo čteními do registrů nebo paměti RAM je potřeba dodržovat buď časový interval, nebo cyklicky vyčítat příznakový registr do doby, kdy je Busy flag v logické jedničce.

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Vysvětlení
Zapnout nebo vypnout displej	L	L	L	L	H	H	H	H	H	L/H	Zapnutí nebo vypnutí displeje dle hodnoty DB0 L:OFF H:ON
Nastav adresu (sloupec)	L	L	L	H	Adresa (sloupec 0 - 63)						Nastaví v čítači pro sloupce (=X čítač) danou hodnotu
Nastav adresu stránky	L	L	H	L	H	H	H	Stránka (0 - 7)			Nastaví v registru pro stránky (=Y registr) danou hodnotu
Nastav 1.řádek displeje z Ram (Z adresa)	L	L	H	H	Adresa (posun pixelů 0 - 63)						Data z RAM se na displeji zobrazí posunuté o daný počet pixelů směrem dolů
Čtení příznakového registru	L	H	Busy	L	On / Off	Reset	L	L	L	L	Možnosti příznakového registru: BUSY L: připravenost řadiče H: řadič je zaměstnán ON/OFF L: displej je zapnut H: displej je vypnut RESET L: normalni mód H: resetovací mód
Zapiš data	H	L	Hodnota dat určených k zápisu								Zapíše data (DB0:7) do daného řádku (8 pixelů) a do daného sloupce v RAM paměťi
Čti data	H	H	Hodnota dat určených ke čtení								Přečte data (DB0:7) z daného řádku (8 pixelů) a z daného sloupce v RAM paměťi

Tab. 4.5: Přehledová tabulka veškerých instrukcí pro displej

Nastavit hodnotu jednoho pixelu na displeji není realizovatelné, neboť se vždy zapisuje po osmi bitech. Pro nastavení hodnoty jednoho pixelu slouží funkce LCD_set-_pixel, které se předají jako parametry hodnoty souřadnic x a y. Tato funkce řeší celý problém takovým způsobem, že nejdříve přečte hodnoty v příslušném sloupci a řádku (RS = high , R/W = high, funkce LCD_read_data), poté pomocí bitové operace pozmění daný bit. Jako další krok je potřeba vrátit kurzor o jeden sloupec zpět (při čtení nebo zápisu se automaticky posouvá, jak již bylo řečeno) a následně zapsat pozměněný bajt do paměti displeje. Další možností by bylo udržovat aktuální hodnotu pixelů od displeje v paměti modulu, ale tím pádem bychom zbytečně ztratili paměťové místo.

4.4.2 DisplayMenu.lib, menu na displeji

Jakmile byla knihovna pro displej naprogramována, bylo potřeba dobře rozvrhnout, jak bude vypadat menu a jakým způsobem bude vypadat knihovna. Na začátku byly stanoveny požadavky, mezi něž patří výběr nabídky, možnost zadávat hodnoty z maticové klávesnice, zadávat znaky, které se nenachází na klávesnici (pro digitální

výstup A-F), možnost mazat znaky a v neposlední řadě vykonat danou operaci při stisku na daný výběr z menu (například Generate signal).

V tomto případě by se hodilo objektově orientované programování, neboť máme jedno menu, které se skládá z jednotlivých stránek a každá stránka obsahuje další komponenty, jako je například vkládání textu od uživatele do tzv. TextBoxu. Celá tato problematika byla do určité míry vyřešena pomocí struktur a funkcí, které měly jako vstupní parametr právě proměnnou typu struktura.

Jak již bylo řečeno, struktura „menu“ obsahuje veškeré stránky (např. Main menu). Jednotlivé stránky mohou být ve zdrojovém kódu volány pomocí indexů, což je s rozrůstajícím programem těžko udržovatelné, nebo pomocí názvů jednotlivých stránek. Tato 2. koncepce byla inspirací v programovém prostředí C#. Programátor si tedy nemusí pamatovat jednotlivé indexy, dále může v libovolném pořadí přidávat do menu jednotlivé stránky. Jak jednoduše se vytváří menu, můžeme vypozorovat z funkce Menu_init. Při větším menu již docházelo k problémům s pamětí, protože každá stránka zabírá paměťový prostor zhruba několik stovek kilobajtů. Alokace paměti probíhá již v prostředí Dynamic C, jelikož se do modulu zapisuje pouze strojový kód, kde veškeré globální proměnné mají již pevně stanovenou adresu. Program při spuštění vždy hned spadl kvůli tomu, že se paměťový prostor proměnných překrýval. Celý tento problém byl vyřešen manuálně, a to alokací paměti pro všechny stránky v SRAM. Proměnná menu obsahuje vždy jednu stránku. Pokud dojde k přepnutí na jinou stránku, veškeré nastavení se uloží do předem definované paměti v SRAM a z této paměti se zároveň načte stránka do proměnné ve struktuře Menu. Tento algoritmus zkomplikoval celé programování nabídky, ale na druhou stranu se od té doby neobjevovaly chyby kvůli náhodnému přepisování paměti.

Každá stránka obsahuje komponenty. Jedna z nich se nazývá TextBox. TextBox je opět samozřejmě struktura zahrnující proměnné. V našem případě uživatel zadává pouze čísla z klávesnice (perioda vzorkování, amplituda, ...). Pouze u jednoho TextBoxu bylo potřeba zadat hodnoty hexa čísla. V tomto případě byla využita tlačítka s šipkou nahoru a dolů, kdy při zmáčknutí daného tlačítka došlo k inkrementaci či dekrementaci v rozpětí od nuly do „F“. V TextBoxu se mohou znaky mazat pomocí klávesy „*“ reprezentující běžně známou klávesu delete.

K další nezbytné komponentě patří bezesporu ComboBox, který je znám v běžném programu jako rozevírací nabídka. Zde se však nerozevírá, ale pomocí tlačítek nahoru a dolů si uživatel vybírá předdefinovanou možnost, jako je perioda jednoho vzorku, neboť z kapitoly 4.1 víme, že tyto hodnoty nemůžou být náhodné číslo.

Jako poslední součástí stránky je tzv. Button, jenž umožňuje přepínání mezi jednotlivými stránkami. Pokud je daný Button zrovna vybrán a uživatel stiskne tlačítko OK, potom se menu změní na nabídku, na niž tento Button právě ukazuje. Aktivací Buttonu se však nemusí změnit nabídka, ale může proběhnout patřičný úkon. Mezi

tyto úkony se řadí měření napětí, generování signálu, odeslání hodnoty na digitální port, čtení z digitálního portu a neposledně navázání spojení přes ethernetový kabel s PC.

Při měření napěťového signálu na vstupu se uloží přesně 128 naměřených hodnot, což odpovídá horizontálnímu rozlišení displeje, a tyto hodnoty se zobrazí do grafu s popisem dole.

Vykonávání jednotlivých operací, jako je měření signálu, je obsluhováno funkcí DoWork. Ke vstupním parametrům patří právě proměnná typu úloha. Tato funkce byla vložena právě do této knihovny, neboť jednotlivé stránky od menu obsahují hodnoty potřebné k vykonání úlohy (amplituda, počet vzorků,...).

4.5 Ethernet.lib

Pro správnou funkčnost je zapotřebí použití křížového ethernetového kabelu, nastavení IP adres a portů.

4.5.1 Vyzkoušení vzorového příkladu

Dynamic C se může chlubit bohatostí vzorových příkladů, nacházejících se ve složce Samples, jež je součástí instalace. Jako jeden z jednoduchých vzorových příkladů byl vyzkoušen „pingme.c“. V PC stačí nastavit IP adresu na 10.10.6.101 a masku na 255.255.255.0, nahrát program do modulu, poté spustit příkazový řádek a zadat „ping 10.10.6.100“. Pokud je vše v pořádku, do příkazového řádku se vypíše, že pakety byly úspěšně přijaty.

4.5.2 Popis knihovny

Z předchozí kapitoly je zřejmé, že programátor nemusí zbytečně programovat žádný protokol od začátku. V této knihovně Ethernet.lib nebylo však využito TCP-IP spojení, ale UDP. Po vyzkoušení obou protokolů byla s velkým rozdílem naměřena větší datová propust u UDP protokolu, jehož zprávy se nejmenují pakety, ale datagramy.

Pro komunikaci pomocí UDP protokolu je potřeba do knihovny přilinkovat knihovnu od Dynamic C dcrtcp.lib. Následně se musí deklarovat daná makra, jako je například IP adresa, hodnota portu. Pro komunikaci bohatě postačí čtyři funkce: sock_init (inicializace soketu), udp_open (zpřístupnění daného portu), udp_recv (přijetí paketu) a udp_send (odeslání paketu). Nastavení a funkčnost ze strany PC je rozebráno v kapitole 5.3.1.

Nejnižší vrstvy komunikačního protokolu již máme naprogramovány, nyní přichází na řadu aplikační vrstva, jež určuje, jakým způsobem si budou programy vyměňovat svá data. Nejdříve je dobré si uvědomit, že se modul chová jako slave a program v PC jako master určující operace. Tabulka 4.6 naznačuje způsob komunikace. Program v PC odesílá vždy jako první bajt informaci určující, jaká data jsou obsažena v následujících bajtech, kolik datagramů má být ještě přijato, co se má s nimi udělat, popřípadě jaká data bude program v PC očekávat nazpět. Multifunkční karta si sama spočítá, kolik má celkem přijmout datagramů, neboť maximální délka datagramu je stanovena na 1460 bajtů a od PC získá v prvních bajtech informaci o počtu vzorků. Zařízení vrací vždy pouze buď potvrzovací data, nebo naměřená data.

Přečti hodnotu na digitálním vstupu a přepošli ji zpět:

10xxxxxx
1 1

Odešli hodnotu na digitální výstup:

11xxxxxx	hodnota
1	2

Generuj signál na výstup 1. anal. kanálu:

011xxxxx	vzorkovací perioda	Počet vzorků/perioda	Počet period	vzorky
1	2	3	5	7 x

Generuj signál na výstup 2. anal. kanálu:

0110xxxx	vzorkovací perioda	Počet vzorků/perioda	Počet period	vzorky
1	2	3	5	7 x

Snímej data z 1. anal. kanálu (AAA = zesílení signálu):

0010xAAA	vzorkovací perioda	Celkový počet vzorků
1	2	3 4

Snímej data z 2. anal. kanálu (AAA = zesílení signálu):

0011xAAA	vzorkovací perioda	Celkový počet vzorků
1	2	3 4

Odešli libovolný bajt zpět (kontrola komunikace):

0000xxxx
1 1

Tab. 4.6: Rozbor jednotlivých bajtů odeslaných ze strany PC

4.6 ErrorHandler.lib, způsoby obsloužení chyby

Každý program by měl mít ošetřené chyby, jako je například dělení nulou nebo zápis do náhodné paměti. V jazyce C je potřeba veškeré chyby ošetřit manuálně. Existuje několik způsobů, jak chybu detekovat a jak ji obsloužit.

Tato knihovna naznačuje právě jeden z nich. Při vygenerování chyby se uloží text chyby do proměnné message ve struktuře typu errorHandler a do další proměnné value této struktury se uloží hodnota true oznamující, že vyvstala chyba. Chybu je potřeba ošetřit v nejnižší vrstvě, co se týče hierarchie funkcí. Ohlášení nebo zpracování chyby se může detekovat ve vyšších vrstvách.

Další možností, a zároveň možností nejvíce používanou, je využití návratové hodnoty funkce. V jazyce C by se měly veškeré hodnoty předávat pomocí ukazatelů, a návratové hodnoty se tedy nemusí vůbec používat. Toho se dá využít k ošetření chyb. Pokud je funkce správně vykonána, potom vrátí návratovou hodnotu 0x01, v jiném případě 0x00. Funkce, která tuto funkci s návratovou hodnotou zavolala, díky tomuto způsobu chybu detekuje a učiní patřičné kroky. Jako příkladová funkce byla zvolena LCD_drawLine v knihovně Display.lib. Úkol této funkce spočívá ve vykreslení čáry na displej, kde vstupní parametry představují souřadnice dvou bodů. Pokud se však alespoň jeden bod nachází mimo rozlišení displeje, detekuje se chyba, vykreslení čáry neproběhne a funkce vrátí hodnotu false, tedy 0x00.

Poslední způsob obsloužení chyby tkví v okamžitém zpracování, díky čemuž nedochází k tzv. „probublávání“ do vyšších vrstev. Jako příklad byla zvolena funkce tb_init v knihovně TimerB.lib. Jestliže vstupní parametr není v požadovaném rozmezí, potom funkce vykoná předdefinované obsloužení chyby a zároveň vypíše na monitor PC, že došlo k chybě, poté program pokračuje dále. K vypsání hlášení na monitor PC je potřeba se nacházet v ladících módu (programovací kabel je k modulu připojen) a dále použít nativně zabudovanou funkci printf, jež zapřičiní vypsání hlášení v prostředí Dynamic C.

V programu byly využity všechny tři metody, ale doporučuje se vždy pouze jedna, neboť zdrojový kód je poté více jasnější. V programu nejsou ošetřeny návratové hodnoty ze všech funkcí, jenom ty nejdůležitější a nejčastější.

4.7 Main.c

V tomto souboru se díky preprocesoru vloží zdrojové kódy jednotlivých knihoven. Ve funkci main dochází k inicializaci jednotlivých hardwarových komponent, k inicializaci proměnných a také k alokaci paměti pro proměnnou menu (pomocí funkce xalloc). Poté následuje supersmyčka, která vždy čeká na vstup od uživatele. Proměnná DeviceState do určité míry nahrazuje stavový automat.

5 VÝVOJ APLIKACE PRO PC V C#

Existuje spoustu různých vývojových prostředí, ve kterých by se snadno dal vytvořit náš požadovaný program. Dokonce by bylo možné použít LabView od firmy National Instrument. Nakonec byl vybrán jazyk C# z prostředí Visual Studio 2012. S licencí nebyl žádný problém, neboť studenti mají tento software zdarma. Jazyk C# byl vybrán z jednoho prostého důvodů, většina aplikací se dnes vyvíjí v Javě, které je C# svojí syntaxí velmi podobný.

Za velkou výhodu tohoto jazyka je považováno vyřešení špatně alokované paměti způsobované v důsledku chyby programátora. Zde se totiž vůbec nepracuje s ukazateli jako v jazyce C, ale s instancemi. Jestliže se daná instance dlouho nebo vůbec nepoužívá, potom tzv. „Garbage collector“ tuto paměť automaticky uvolní.

Nejdříve ve zkratce naznačíme, co znamená WPF, poté popíšeme pomocný nástroj ClassDiagram pro vytváření tříd a nakonec ukážeme vybrané zajímavé pasáže. Syntaxi jazyka C# vynecháme z důvodů velké obsáhlosti. Přiblížíme však jen některé funkce, ať už vlastní nebo integrované od Visual studia.

5.1 WPF

Zkratka je odvozena od slov Windows Presentation Foundation, jedná se o způsob programovaní grafického vzhledu aplikace. Běžná starší aplikace běží v klasickém formulářovém stylu (Windows Forms Application). Pro nově vznikající programy se již preferuje právě WPF.

Výhoda WPF spočívá bezpochybně v oddělení grafického návrhu a logiky programu. Jednotlivé komponenty (TextBox, ComboBox) se přidávají do okna programu pomocí jazyka XAML (čte se jako zaml). Díky tomuto jazyku se může okno chovat dynamicky bez složitějších algoritmů. Spuštěním programu MyApp.exe se přesvědčíme, že při změně velikosti okna se dynamicky mění i velikost nebo poloha jednotlivých komponent, a okno tudíž nemusí být po celý běh procesu statické. Díky tomu můžeme bez většího úsilí nastavit velikost grafu dle velikosti hlavní okna. Syntaxe pro XAML a vše tykající se kolem WPF najdeme v literatuře [6]. V našem programu byla využita ještě jedna užitečná vlastnost WPF, a to jsou styly. Chceme-li například, aby všechna tlačítka měla modrou barvu, stačí nastavit globální styl pro tlačítka, jako je naznačeno ve zdrojovém kódu níže.

```
<Style TargetType="Button">
    <Setter Property="Background" Value="Blue">
    </Setter>
</Style>
```

5.2 classDiagram

V C# se pracuje vždy s objekty. Na začátku je potřeba si správně rozvrhnout jednotlivé třídy. ClassDiagram, jakožto podprogram Visual Studia, nám tuto počáteční práci zjednodušuje. Náš vytvořený program byl zde také navržen (příloha D.1). Slovo Fields znamená proměnné od dané třídy, tyto proměnné by měly být private (použitelné pouze v rámci třídy). K proměnným se přistupuje totiž přes tzv. Properties. Neodmyslitelnou součástí každé třídy jsou metody. Díky classDiagramu získáme větší přehled nad jednotlivými třídami.

5.3 Implementovaný program

Třída RCM3200 komunikuje přímo se zařízením pomocí UDP protokolu. Třídy rcmDA a rcmAD obsahují v sobě aktuální nastavení z menu. Jestliže uživatel změní cokoli, co se týče aplikace (například amplitudu výstupního napětí), tato hodnota se okamžitě uloží do proměnné „amplitude“ ve třídě rcmDA. Před odesíláním hodnot multifunkční kartě se tedy hodnoty nenačítají přímo z komponent v menu, ale vezmou se aktuálně uložené hodnoty v dané třídě.

Program neustále kontroluje, zda je karta připojena. Pro kontrolu bylo využito další vlákno procesoru, protože při používaní jednoho vlákna v důsledku čekání na odezvu zařízení se program „zasekával“. Použití dalšího vlákna je velmi jednoduché, neboť je vlákno deklarováno opět jako třída. Nejdříve je nutné specifikovat jmenný prostor System.Threading. Při vytváření instance je potřeba jako vstupní parametr zadat funkci, kterou má vlákno vykonat, poté stačí zavolat funkci pro zahájení vykonání. Viz příklad níže:

```
Thread t = new Thread(NazevFunkce);
t.start();
```

Opět bylo nutno detekovat chyby. K tomu účelu slouží blok try-catch. V programu byly opět ošetřeny nejnutnější chyby, které se týkaly zejména komunikace přes ethernet. Další chyby mohly vzniknout v důsledku nevhodně zadané hodnoty od uživatele. Tyto chyby byly také eliminovány, takže uživatel není například schopen vložit písmeno do TextBoxu od amplitudy. Další chyba nastala, když uživatel zmáčknul tlačítko generuj signál a přitom nebylo navázáno žádné spojení s kartou. Proto tlačítka generuj a měř při neaktivním spojení s kartou jsou deaktivované.

5.3.1 Ukázka odeslání bajtů pomocí UDP protokolu

Pro odeslání jednoho bajtu stačí opět pár řádků, jak můžeme vidět v ukázkovém zdrojovém kódu dole. Tento kód byl zestručněn, neboť zde ještě musí být try-catch blok pro „vychytávání“ chyb. Abychom mohli příslušné třídy, jako je například UdpClient použít, je zapotřebí vložit jmenný prostor System.Net, System.Net.Sockets. Stejně jako u modulu je zde nezbytné nastavit jak IP adresu zařízení, tak IP adresu počítače.

```
static public string IPaddressComputer = "10.10.6.101";
private IPEndPoint ip = new IPEndPoint(IPAddress.Parse("10.10.6.100"), 1234);
private UdpClient udpClient;

private void  sendData(byte[] data)
{
    udpClient.Connect("10.10.6.100", 1234);
    this.udpClient.Send(data, data.Length);
}
```

6 UŽIVATELSKÝ POHLED

6.1 Ovládaní přímo na multifunkční kartě

Multifunkční karta nemusí být vždy připojena k PC, v tomto případě pak uživatel veškerá nastavení aplikuje přímo na zařízení pomocí displeje a tlačítek. Každá aktuálně zobrazená stránka má v pravém horním rohu vypsaný název, aby se uživatel neztratil. Dále každá stránka, kromě hlavní nabídky, vždy dole nabízí možnost vrátit se o nabídku výše (tlačítko back). Pokud uživatel zadá z maticové klávesnice hodnotu, která nesplňuje určitý rozsah (např. amplituda výstupního napětí 20 000 mV), automaticky se zadaná hodnota změní na nejbližší povolené číslo. Během generování nebo měření signálu smí uživatel jakýmkoliv tlačítkem tuto operaci zrušit a vrátit se do nabídky. Vzhled aplikace na displeji se nachází pro ilustraci v příloze číslo E.1.

6.2 Popis programu pro PC

Aby karta vůbec navázala spojení s PC, musí uživatel nejdříve zadat přímo na multifunkční kartě spojení s PC (Ethernet connection->Connect). Ve zbývající části kapitoly se již budeme zabývat pouze uživatelským programem v PC.

Uživatel se v programu zorientuje během několika sekund. Celé ovládání je intuitivní, a tudíž nebylo zapotřebí vytvářet manuál. Grafický vzhled programu zachycují přílohy F.1 a F.2

Program vpravo nahoře indikuje aktuální stav připojení ke kartě. Mohou nastat dohromady tři situace. Nemá-li uživatel nastavenou IP adresu na 10.10.6.101, hláška mu doporučí, aby to napravil. Při splnění této podmínky se zobrazí buď, že je multifunkční karta správně připojena, nebo naopak není. Kromě vlastní IP adresy tedy uživatel nemusí nic nastavovat, aby program úspěšně komunikoval se zařízením.

Před vygenerováním signálu vidí uživatel náhled díky grafu. Náhledový graf se průběžně mění s tím, jak uživatel nastavuje jednotlivé parametry signálu. Tytéž hodnoty v grafu jsou zobrazeny i v tabulce. Uživatel smí zadat svá data uložená v souboru programu Excel (pomocí tlačítka import). V průvodci si uživatel vybere, na kterém listě se data nacházejí. Musí však být splněna jediná podmínka, a to umístění dat pro napětí v prvním sloupci. Excelová tabulka tedy obsahuje pouze napěťové hodnoty. Interval mezi jednotlivými vzorky je totiž vždy stejný a nastavuje se přímo v programu (ComboBox time/Sample). Během generování signálu se zobrazí čas do konce. Uživatel může danou operaci kdykoliv zrušit díky tomu, že aplikace využívá více jader. Při zrušení generování se však vypíše hláška, aby uživatel

resetoval připojení na multifunkční kartě, neboť karta v průběhu generování nekontroluje příchozí datagramy z důvodů, které jsou nám známy z kapitoly 4.3, totiž že není časové možné při maximální periodě vzorkování provádět další operace. Tento postup pro zrušení operace platí stejně i u snímání signálu.

Po změření dat na analogovém vstupu se hodnoty zobrazí opět do grafu a do tabulky, z níž mohou být dále exportovány do Excelu. Výhoda grafu spočívá v přibližování: uživatel si může libovolná data přibližovat a oddalovat dle své potřeby. Naměřených hodnot může být až 65 000 (zaokrouhlený unsigned int).

Nastavení digitálního portu je velmi triviální, jak můžeme vidět z přílohy F.1(b). Zde snad pouze stojí za zmínku, že při změně směru digitální linky ze snímaní na generování se nejdříve zkontroluje, zda veškeré vstupní hodnoty jsou nulové. Pokud by však alespoň na jedné lince byla kladná hodnota (externí signál), potom se vypíše hlášení, že změna směru linky není z tohoto důvodu možná. Uživatel zadává vždy hexa hodnoty (0-F), ostatní znaky jsou ignorovány.

7 OTESTOVÁNÍ MULTIFUNKČNÍ KARTY

Vyrobenou multifunkční kartu s vytvořeným softwarem je potřeba otestovat, co se týče funkčnosti a přesnosti. Nejdříve byly testovány vstupní kanály pomocí generátoru Agilent 33220A, a poté výstupní kanály pomocí multimetru Agilent 34410A a osciloskopu MSO6014A. Digitální linky byly také podrobeny úspěšnému testování, ale nevěnujeme jim žádnou kapitolu, protože není třeba je složitě nastavovat, jak vyplývá z přílohy F.1 na obrázku F.1(b).

7.1 Vstupní analogové kanály

Pro oba vstupní kanály byly naměřeny hodnoty na všech rozsazích. Multifunkční karta byla propojena s PC přes ethernetový kabel pro jednodušší manipulaci. Na příslušný vstupní kanál byl přiveden stejnosměrný signál z generátoru Agilent 33220A. V programu v PC byl vždy požadavek na změření deseti tisíc hodnot. Tyto všechny hodnoty byly exportovány do excelové tabulky. Z těchto všech hodnot byl spočítán průměr dle vzorce 7.1, kde n = 10 000.

Obecný vzorec aritmetického průměru:

$$U_n = \bar{u} = \frac{1}{n}\sum_{i=1}^{n} u_i \tag{7.1}$$

Dále byla vypočítána absolutní chyba:

$$\Delta U = U_n - U_s \tag{7.2}$$

U_n = naměřené napětí (aritmetický průměr z 10 000 hodnot)
U_s = správná hodnota (od Agilentu 33220A)

Relativní chyba se vypočítá jako:

$$\delta = \frac{\Delta U}{U_s} 100 \quad [\%] \tag{7.3}$$

Všechny tyto vypočtené hodnoty byly vyneseny do grafů v přílohách od G.1 do G.6. V příloze F.2 je znázorněn naměřený harmonický signál z generátoru Agilent.

7.2 Výstupní analogové kanály

Přesnost kanálů byla zachycena v grafech přílohy G.7. Postup výpočtu chyb je stejný jako v kapitole 7.1 s tím rozdílem, že hodnota napětí na multimetru Agilent 34410A byla měřena pouze jednou a nebyla průměrována z více hodnot. U výstupních kanálů bylo také nutné otestovat, zda nedochází k výrazným nežádoucím přechodovým jevům, proto byly změřeny na osciloskopu MSO6014A různé průběhy signálů generované multifunkční kartou. Výsledky jsou zaznamenány v přílohách H.1 a H.2.

8 ZÁVĚR

V diplomové práci byly splněny všechny požadované body dle zadání. Nad rámec zadání byly implementovány další hardwarové komponenty, aby se multifunkční karta mohla ovládat bez ethernetového připojení k počítači. Pro ovládaní multifunkční karty byl tedy přidán displej a tlačítka.

Nejdříve jsme nastínili základní přehled multifunkčních karet dostupných na trhu a u každé z nich jsme zjednodušeně vypsali základní parametry a vlastnosti.

Před samotnou realizací bylo potřeba vybrat vhodně jednotlivé součástky, mezi něž patří především spínaný zdroj, AD převodník, DA převodník, programovatelný zesilovač. Modul RCM3200 byl vypůjčen od vedoucího diplomové práce, a proto nebyl použitý modul konfrontován.

Během oživování multifunkční karty se vyskytlo několik problémů, které byly následně odstraněny. Touto tématikou se zabývala kapitola 2.7.2.

Programování samotného modulu probíhalo v programovacím jazyce C ve vývojovém prostředí Dynamic C, kterému se věnuje jedna kapitola. Programování modulu bylo jednodušší díky možnému ladění, kterým modul spolu s vývojovým prostředím disponuje. Část programu byla realizována v assembleru, neboť rychlost pro snímání nebo generování s maximální periodou vzorkování sto tisíc vzorků za sekundu v klasickém C nebyla dosažitelná. V programu byly také ošetřeny chyby, které mohly nastat špatným vstupem od uživatele nebo připojením k neexistující IP adrese.

Program pro PC byl vytvořen v programovacím jazyce C# od Visual Studia. Aplikace se chová vzhledově dynamicky - komponenty okna se mění v závislosti na velikosti okna. Program pravidelně kontroluje připojení k multifunkční kartě. Při nefunkčním spojení ohlásí program chybu. Generování nebo snímání může uživatel kdykoliv zrušit díky použití více vláken.

Základní přehled z pohledu uživatele zachycuje kapitola 6, která díky jednoduchosti programu v PC a ovládání přímo na multifunkční kartě nahrazuje manuál.

Na závěr byla karta otestována z pohledu funkčnosti a přesnosti. Z grafů absolutních chyb u příloh G.1 až G.6 vyplývá, že absolutní chyba vstupních kanálů rostla lineárně s rostoucím napětím na vstupu. Tato příčina mohla být způsobena například malým vstupním odporem, jenž zvyšoval absolutní chybu. Z těchto grafů můžeme také vyčíst velkou chybu na začátku a konci daného měřeného rozsahu. Chyba na začátku je způsobena programovatelným zesilovačem, neboť nedokáže lineárně zesilovat velmi malá napětí. Chybu na konci ovlivňuje již samotný AD převodník, jehož napájení slouží zároveň jako referenční napětí. Z grafů u DA převodníků v příloze G.7 nastává chyba pouze na začátku v okolí nula voltů, protože operační zesilovač je napájen nesymetricky.

V této diplomové práci se nabízí spoustu dalších možností, jak lépe propracovat

hardwarovou i softwarovou část. U hardwarové části se mohou vylepšit přesnosti díky zvolení správného vstupního odporu analogových vstupů. V diplomové práci byly zbytečně použity duplicitní součástky MCP3202, neboť pro měření v daný okamžik se využívá právě jeden kanál. Původní záměr byl měřit na obou kanálech zároveň. Avšak, jak vyplývá z kapitoly 4.3, nebylo realizovatelné využívat při maximální vzorkovací periodě oba vstupní kanály. Stačil by tedy jeden AD převodník MCP3202 se dvěma kanály. Lepší charakteristiku napětí výstupního kanálu v oblasti nuly by vyřešilo symetrické napájení zesilovače. Vylepšení softwarové části se může týkat rozšíření na analogový simulátor soustavy nebo analogový regulátor.

LITERATURA

[1] PATOČKA, Miroslav. *Řídicí elektronika - pasivní obvody.* . Brno, 2004. Skripta. VUT.

[2] PATOČKA, Miroslav. *Řídicí elektronika - aktivní obvody.* . Brno, 2004. Skripta. VUT.

[3] ZÁHLAVA, Vít. *Návrh a konstrukce desek plošných spojů: Princip a pravidla praktického návrhu.* . 1. vyd. Praha: BEN, 2010. ISBN 9788073002664.

[4] HYDER, Kamal a Bob PERRIN. *Embedded Systems Design using the Rabbit 3000 Microprocessor: Interfacing, Networking, and Application Development* . [online]. United States of America, 2005 [cit. 2013-05-06]. ISBN 0-7506-7872-0. Dostupné z: <http://www.scribd.com/doc/69025518/Embedded-Systems-Design-Using-the-Rabbit-3000-Microprocessor>.

[5] *Rabbit 3000 Microprocessor: User's Manual* [online]. 2010. vyd. USA, [cit. 2013-05-06]. Dostupné z URL: <ftp://ftp1.digi.com/support/documentation/019-0108_W.pdf>.

[6] *Windows Presentation Foundation 4.5 Cookbook* [online]. [online]. UK: Packt Publishing, 2012 [cit. 2013-05-07]. ISBN 978-1-8496-8622-8. Dostupné z: <http://it-ebooks.info/book/1234/>.

[7] *PCIe-6259* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://sine.ni.com/nips/cds/view/p/lang/en/nid/201814>.

[8] *PCI-1711* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.bb-elec.com/bb-elec/literature/pci-1711.pdf>.

[9] *U2352A* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://cp.literature.agilent.com/litweb/pdf/5989-9923EN.pdf>.

[10] *Obrázek USB6008* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://techteach.no/tekdok/usb6008/>.

[11] *Obrázek PCIe-6259* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.pressebox.de/pressemeldungen/national-instruments-germany-gmbh/boxid/39702>.

[12] *Obrázek PCI-1711* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.vatgia.com/raovat/7470/3530010/can-ban-card-pci1711-advantech-gia-re.html>.

[13] *Obrázek U2352A* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.conrad.de/ce/de/product/128077/AGILENT-U2352A-USB-DATENERFASSUNGSMODUL>.

[14] *Obrázek Uspořádání RAM v řadiči KS0108* [online]. [cit. 2012-12-25]. Dostupné z URL: <http://www.conrad.de/ce/de/product/128077/AGILENT-U2352A-USB-DATENERFASSUNGSMODUL>.

[15] *Obrázek Maticová klávesnice* [online]. [cit. 2012-12-25]. Dostupné z URL: <http://www.arifblog.web.id/2012/10/program-assembly-keypad-4x4-pada-at89s51.html>.

[16] *Zapojení maticové klávesnice* [online]. [cit. 2012-12-25]. Dostupné z URL: <http://http://www.farnell.com/datasheets/42264.pdf>.

[17] *RCM3000* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.datasheetarchive.com/Rabbit-3000-datasheet.html>.

[18] *RCM3000 manual* [online]. [cit. 2012-04-11]. Dostupné z URL: <ftp://ftp1.digi.com/support/documentation/0190118_n.pdf>.

[19] *Datasheet - LT1021* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/42356.pdf>.

[20] *Datasheet - MCP6S91* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/60246.pdf>.

[21] *Datasheet - LT1352* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/82619.pdf>.

[22] *Datasheet - AD8639* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/679724.pdf>.

[23] Slew rate. *Wikipedia: the free encyclopedia* . [online]. San Francisco (CA): Wikimedia Foundation, 2012.02.01 [cit. 2012-04-17]. Dostupné z URL: <http://en.wikipedia.org/wiki/Slew_rate>.

[24] *Datasheet - LED 3mm,red* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/95144.pdf>.

[25] *Datasheet - LED SMD1206,red* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://cz.farnell.com/kingbright/kpt-3216sgc/led-green-1206-smd/dp/2099247>.

[26] *Datasheet - Display 128x64* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.datasheetarchive.com/KS0108%20128X64-datasheet.html>.

[27] *Datasheet - Napěťový regulátor LM1117* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.ti.com/lit/ds/symlink/lm1117-n.pdf>.

[28] *Datasheet - A/D převodník MCP3202* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/21034D.pdf>.

[29] *Datasheet - D/A převodník MCP4922* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/22250A.pdf>.

[30] *Datasheet - Spínané zdroje NDY2405 a NDY2415* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.murata-ps.com/data/power/ncl/kdc_ndy.pdf>.

[31] *Datasheet - Ochranná Schotkyho dioda B140HB* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.diodes.com/datasheets/ds30128.pdf>.

[32] *Datasheet - Ochranná Schotkyho dioda BAT54A* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/13109.pdf>.

[33] *Datasheet - Transil P6SMB30AT3G* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/12422.pdf>.

[34] *Datasheet - Transil SMBJ12CA* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/41770.pdf>.

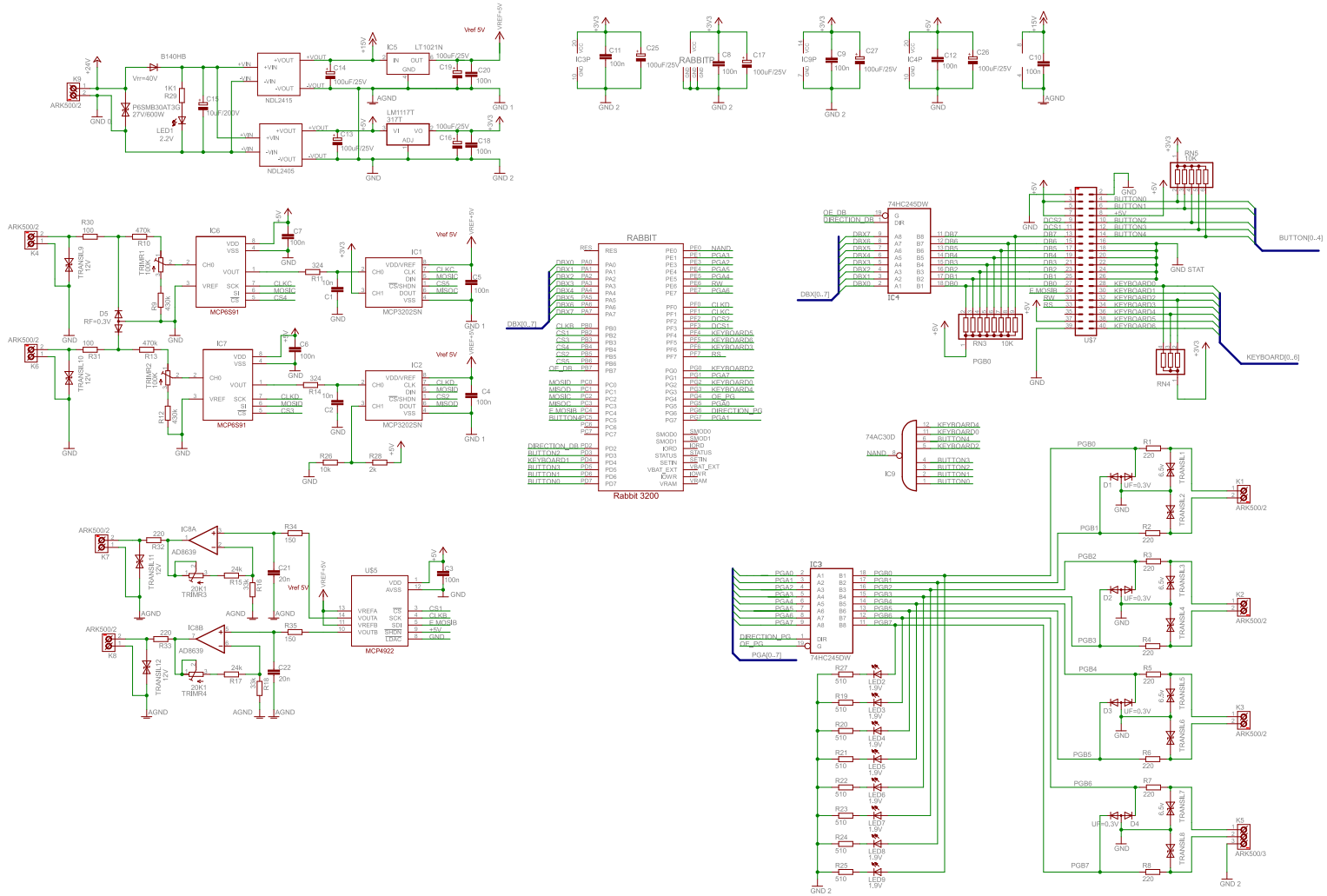
[35] *Datasheet - Transil SMBJ6.5CA-E3/52* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.vishay.com/docs/88392/88392.pdf>.

[36] *Datasheet - NAND: 8 inputs* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.farnell.com/datasheets/1389691.pdf>.

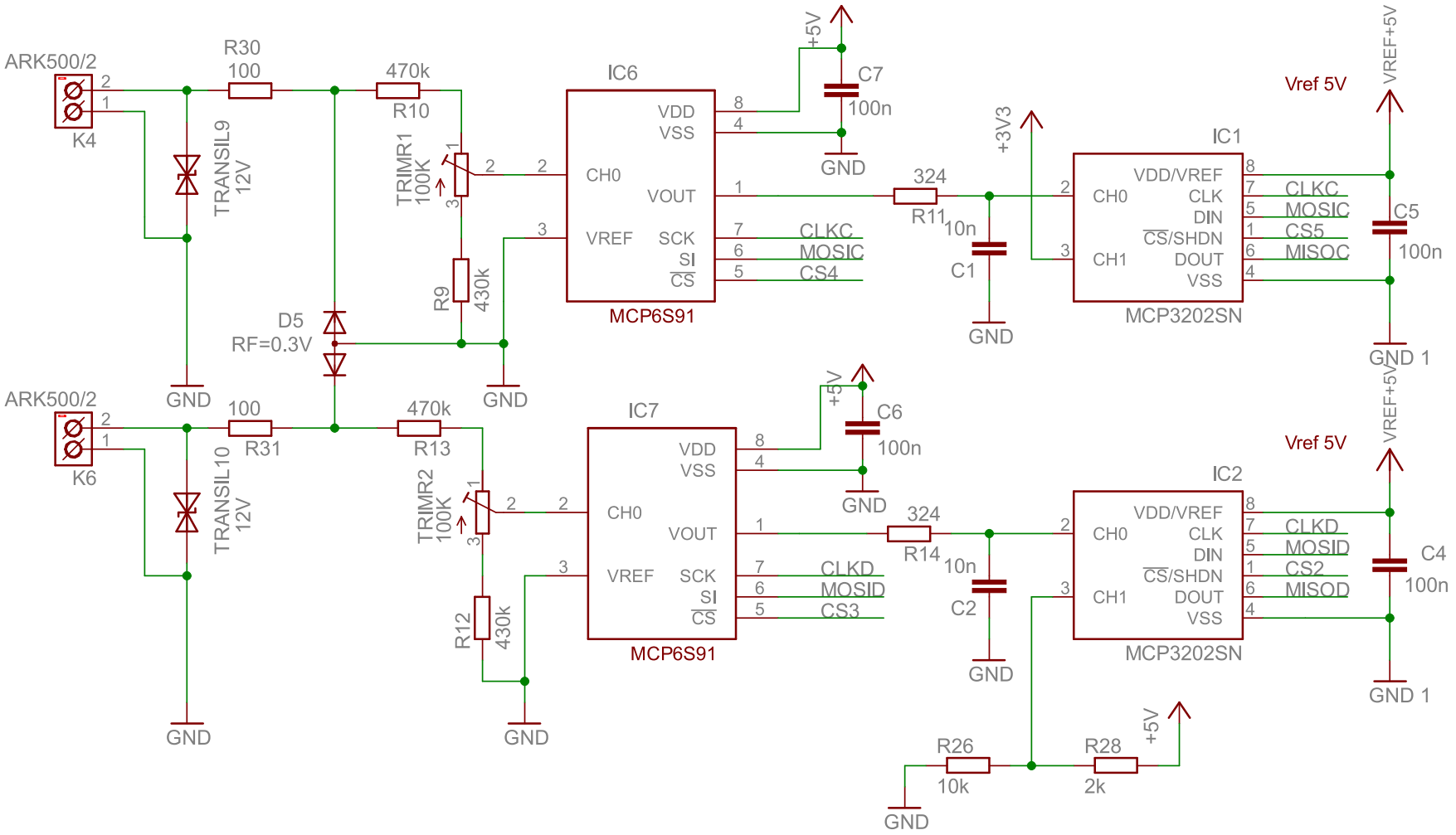
[37] *Datasheet - Octal bus tranceiver* [online]. [cit. 2012-04-11]. Dostupné z URL: <http://www.nxp.com/documents/data_sheet/74HC_HCT245.pdf>.

SEZNAM PŘÍLOH

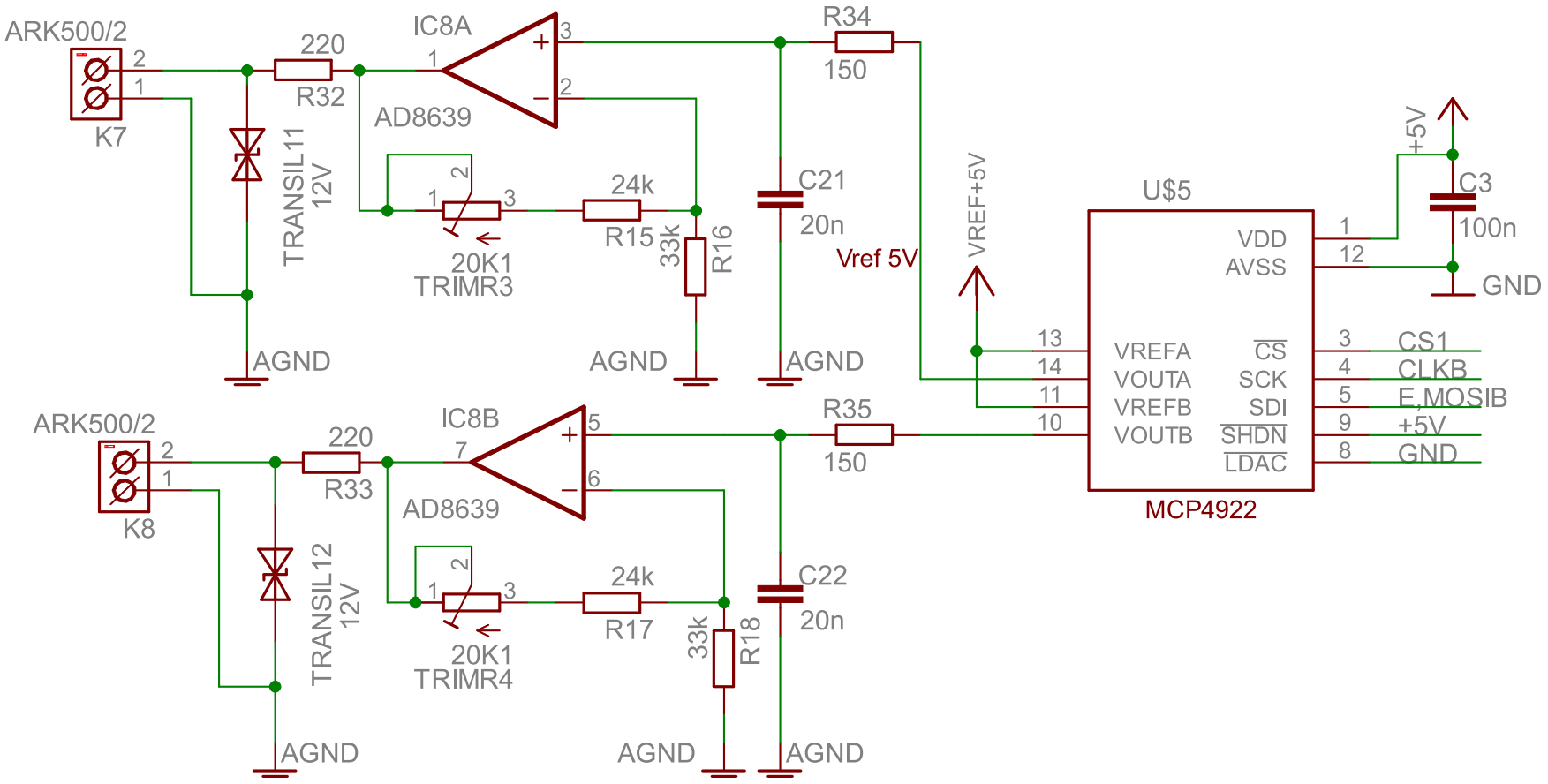
A.1 Schéma hlavní (spodní) DPS (určeno pro el. verzi)



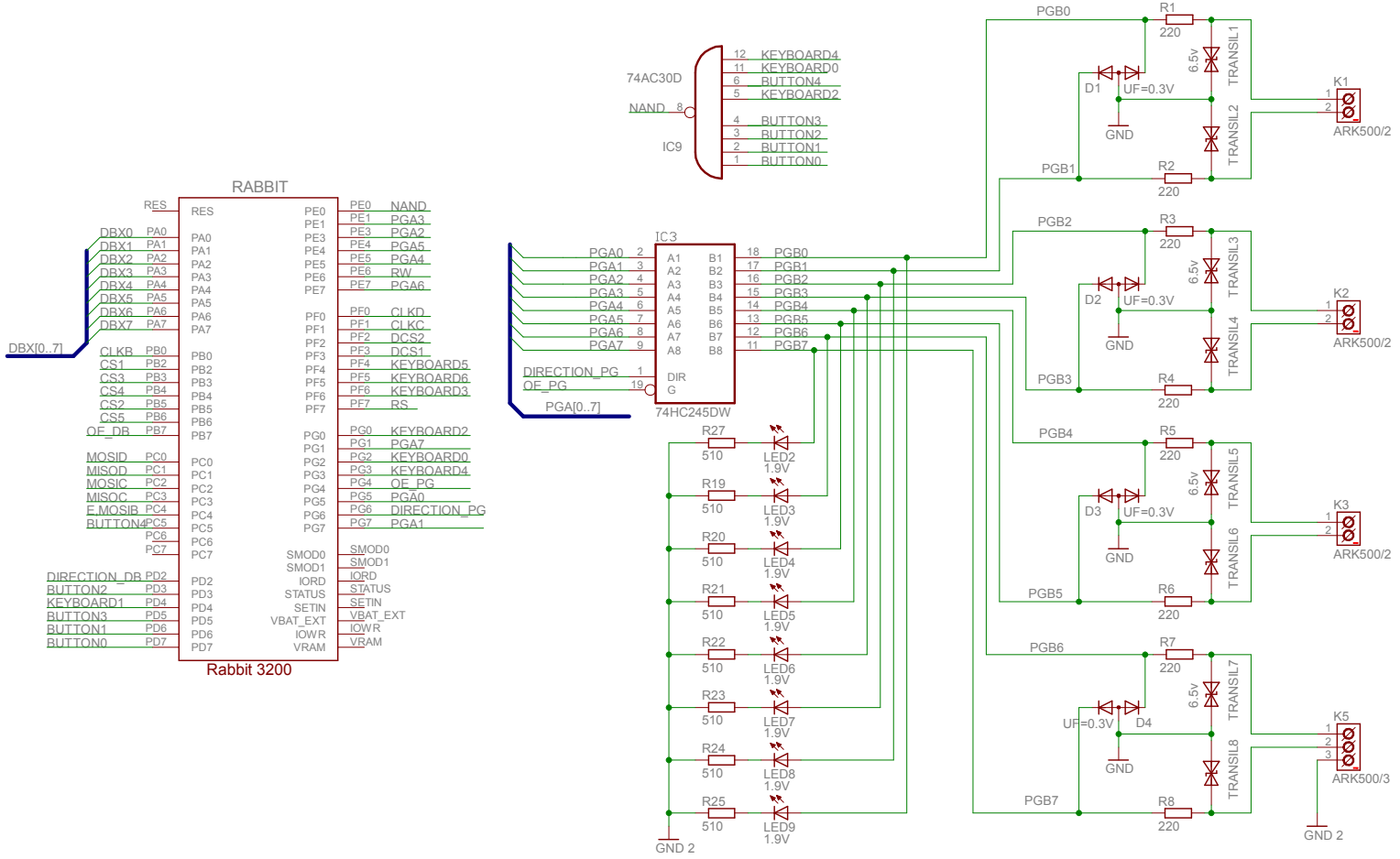
A.2 Schéma zapojení AD převodníků



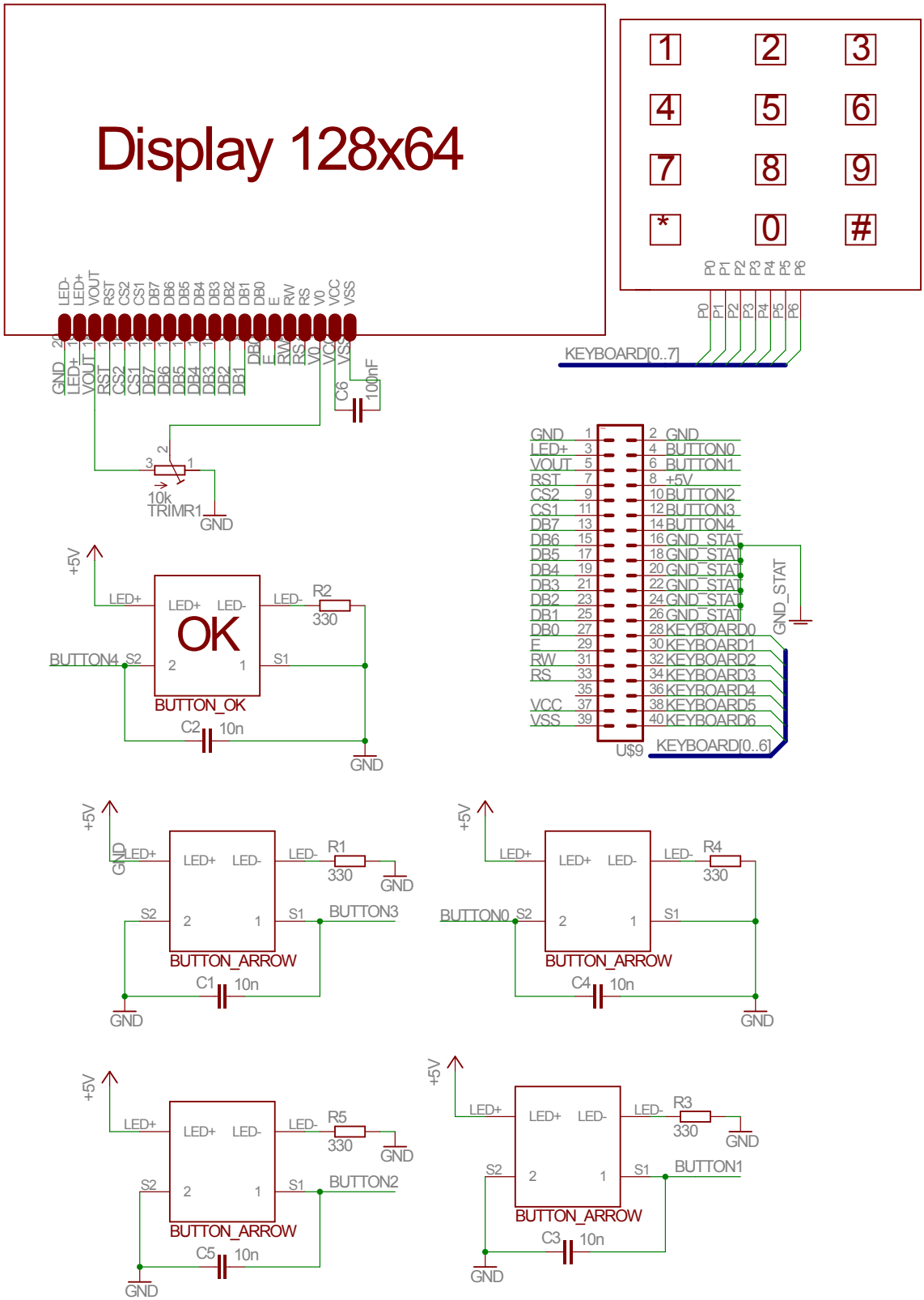
A.3 Schéma zapojení DA převodníků



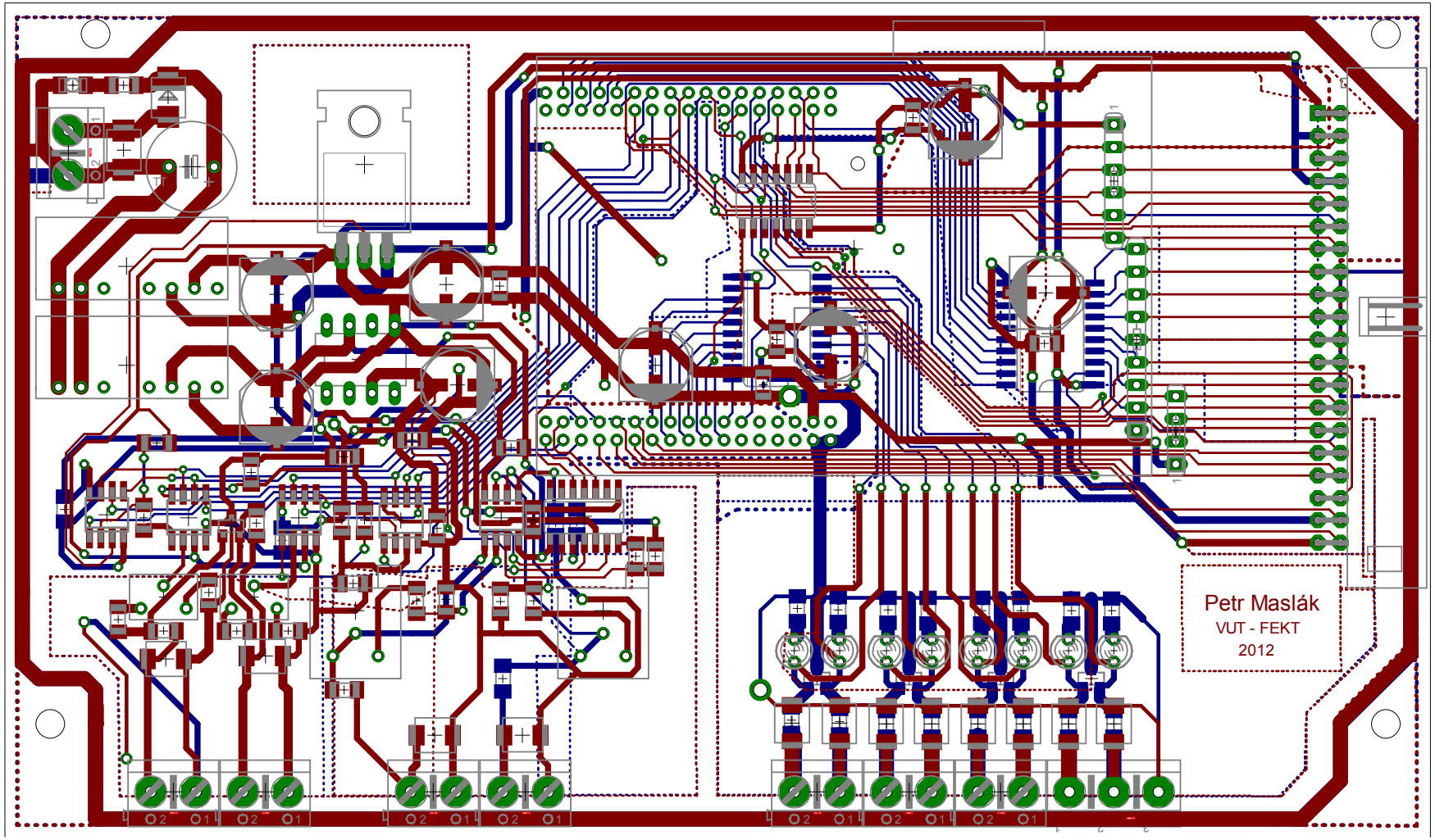
A.4 Schéma zapojení procesoru, digitálního IO portu



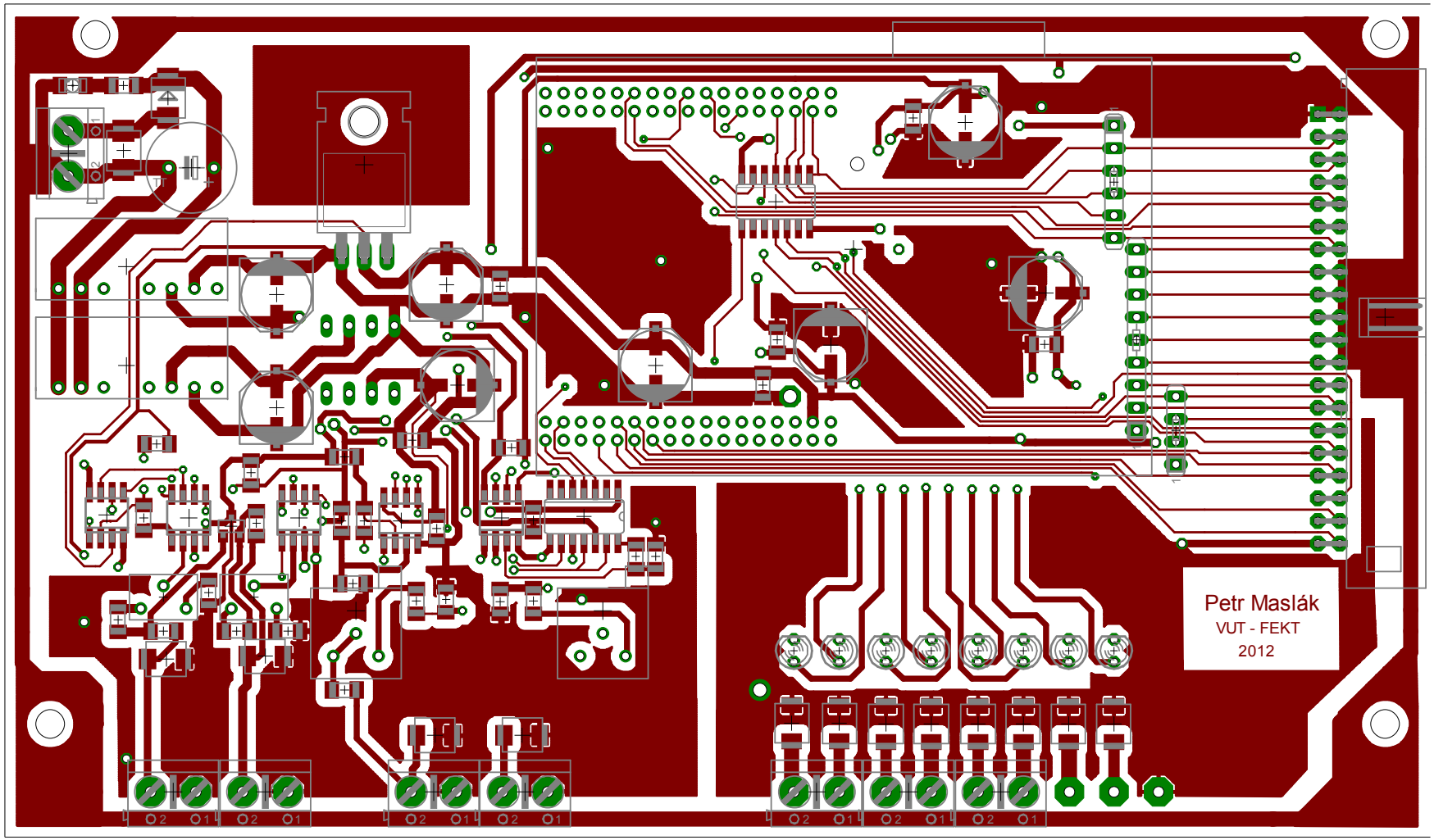
A.5 Schéma horní DPS



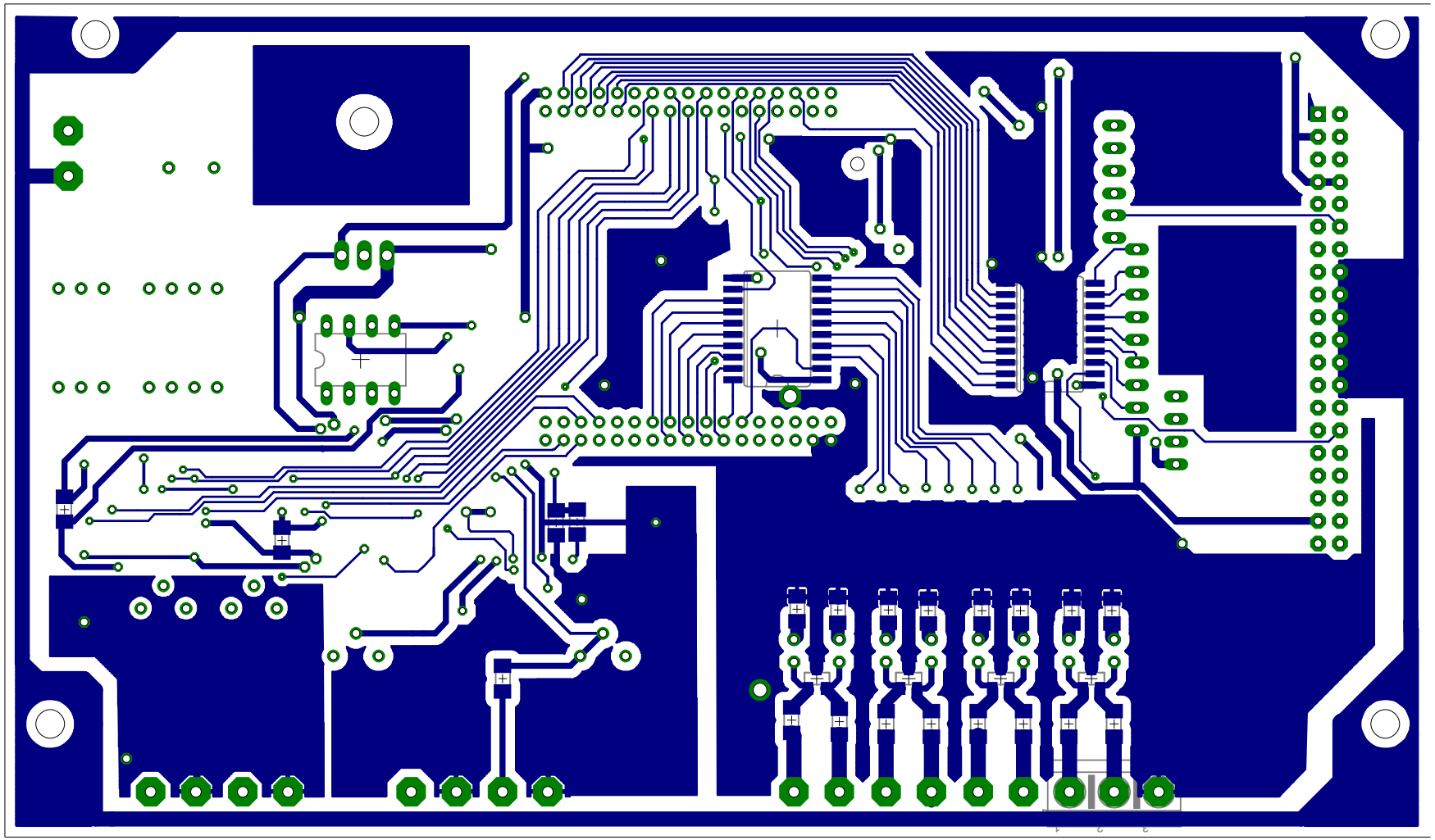
B.1 DPS hlavní desky (obě strany, již opravená verze)



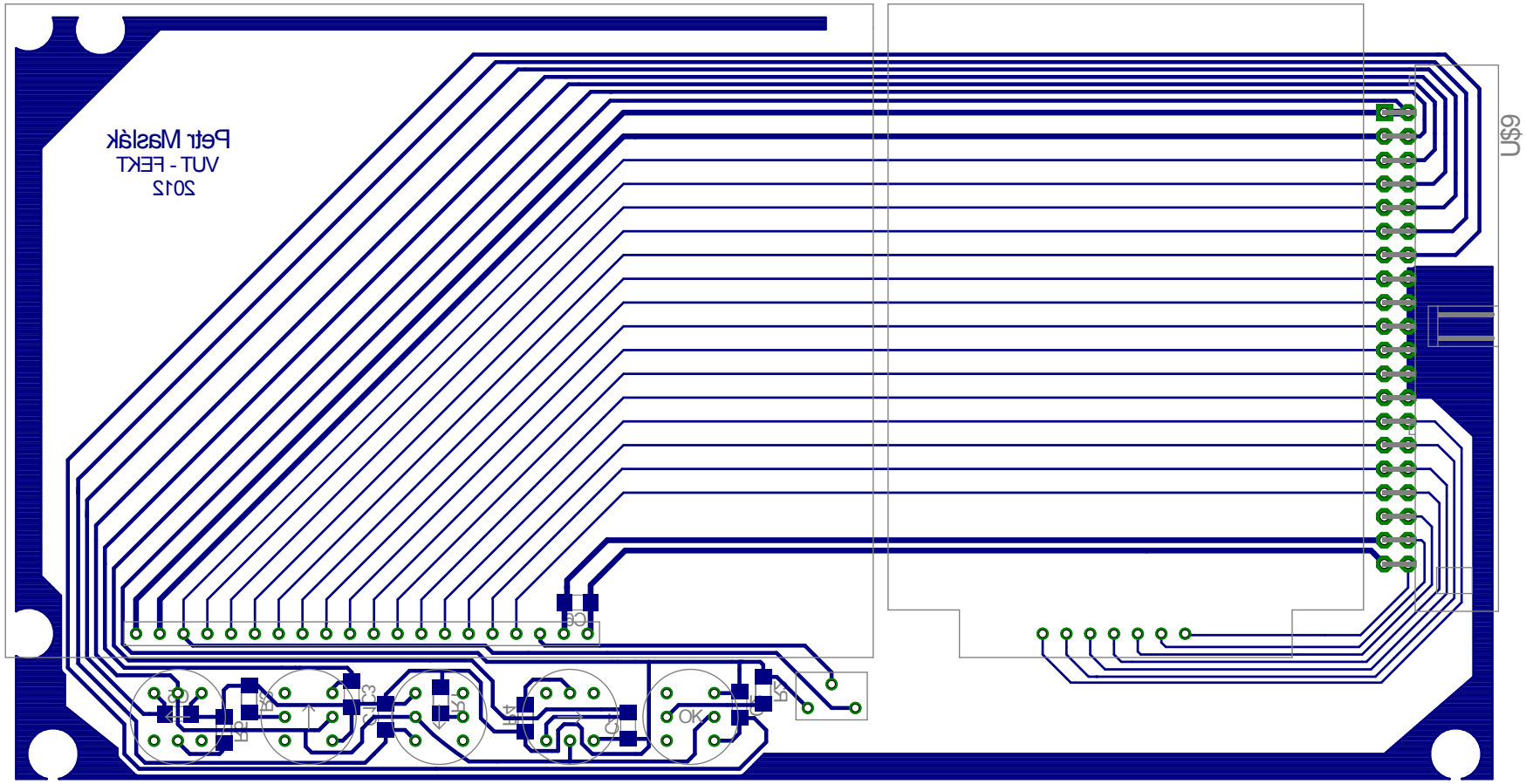
B.2 DPS hlavní desky (horní strana, již opravená verze)



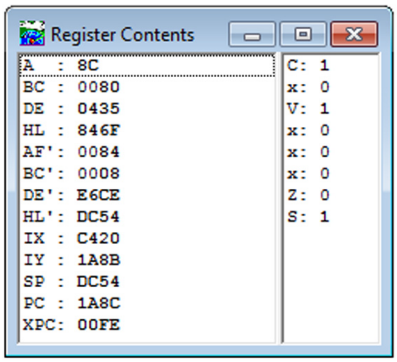
B.3 DPS hlavní desky (spodní strana, již opravená verze)

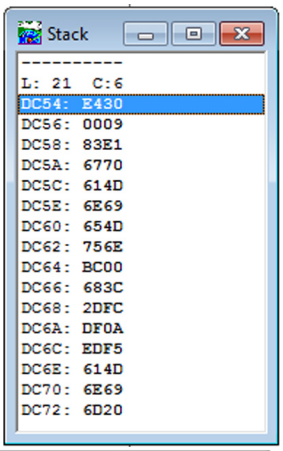


B.4 DPS horní desky (pohled shora)

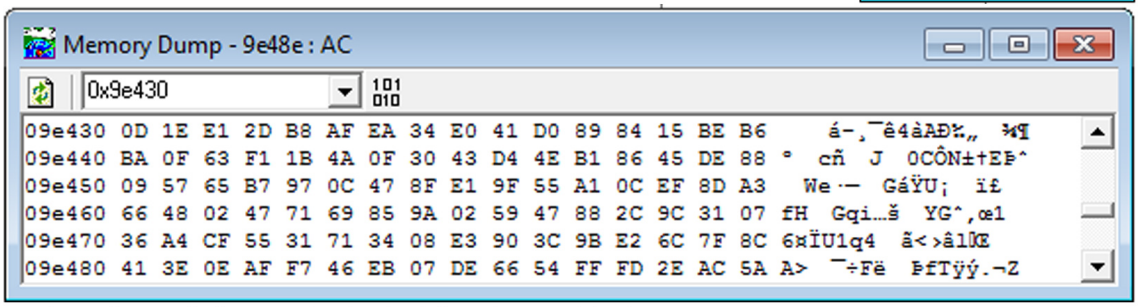


C.1 Nástroje pro ladění v Dynamic C

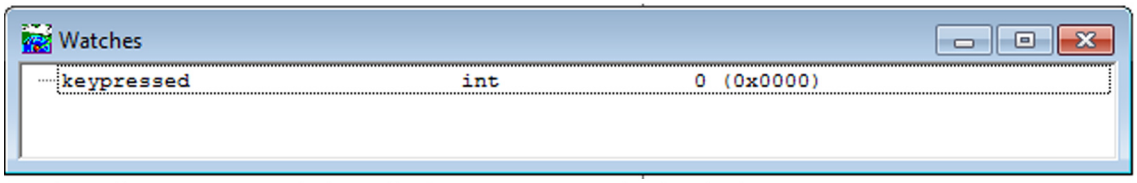




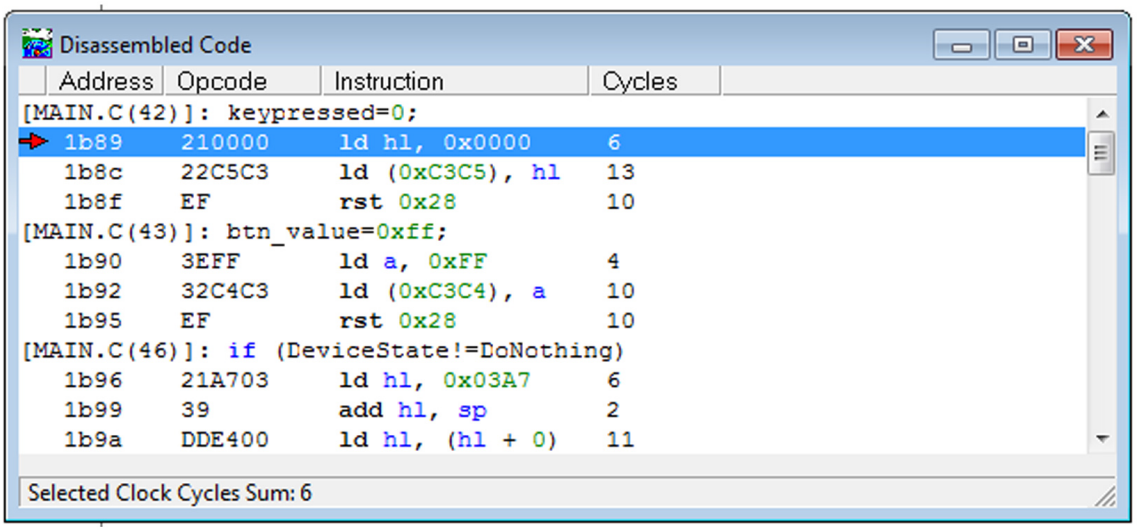
Obr. nahoře: sledování hodnot registrů a příznakových flagů
Obr. vlevo: sledování hodnot v zásobníkové paměťi



Obr. nahoře: sledování obsahu fyzické adresy od hodnoty 0x9e430

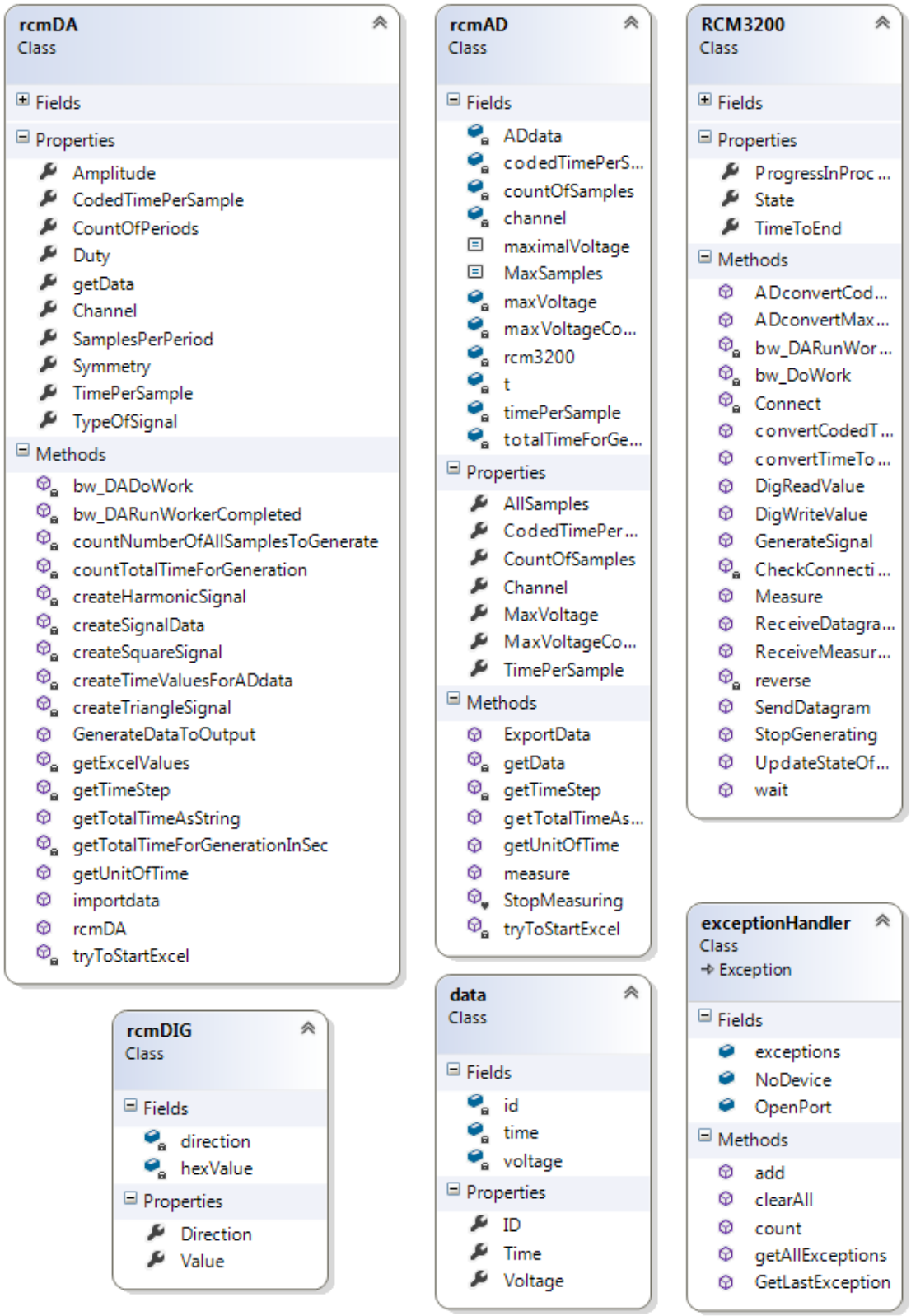


Obr. nahoře: možnost sledovat hodnoty proměnných, či celých výrazů



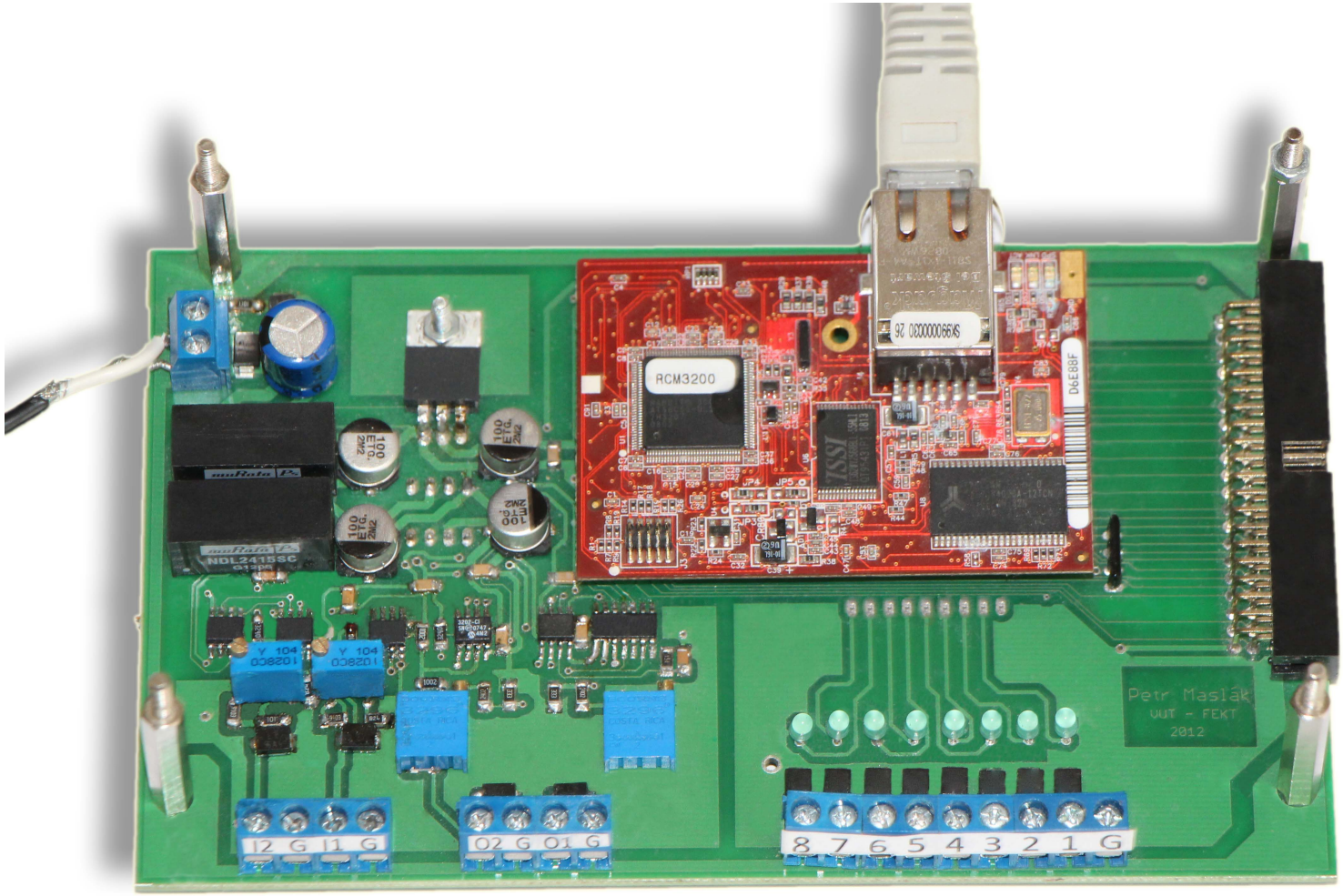
Obr. nahoře: možnost pozorování aktuálně zpracováváné instrukce

D.1 ClassDiagram v C#: pomocník při rozvrhování programu

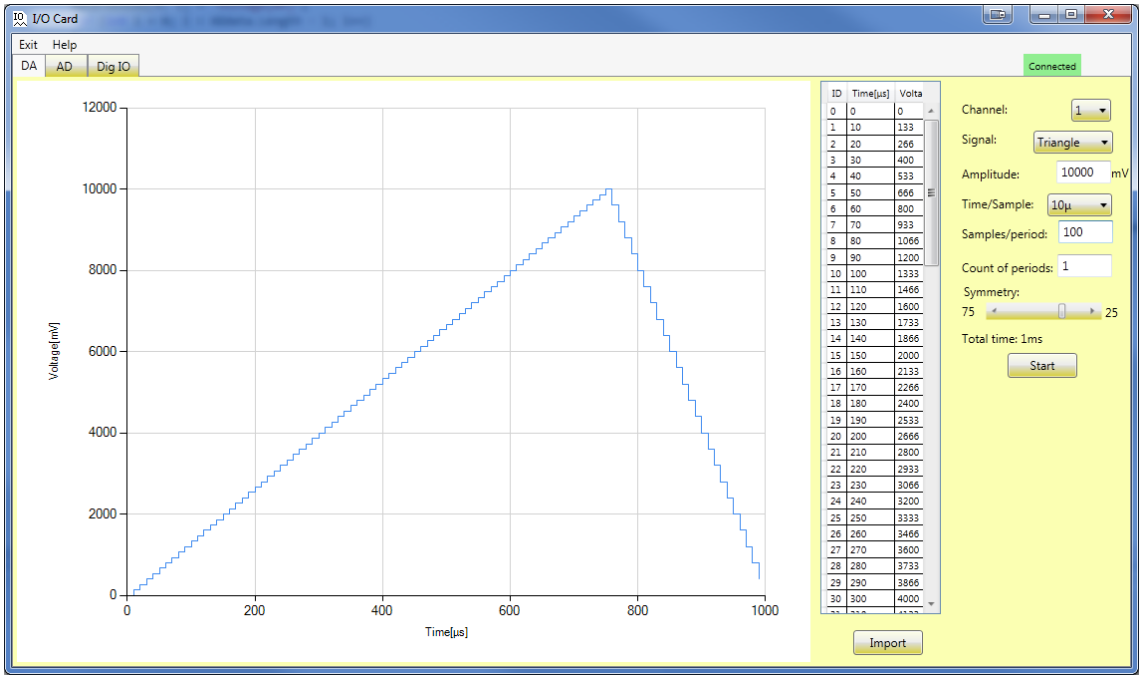


E.1 Fotka multifunkční karty

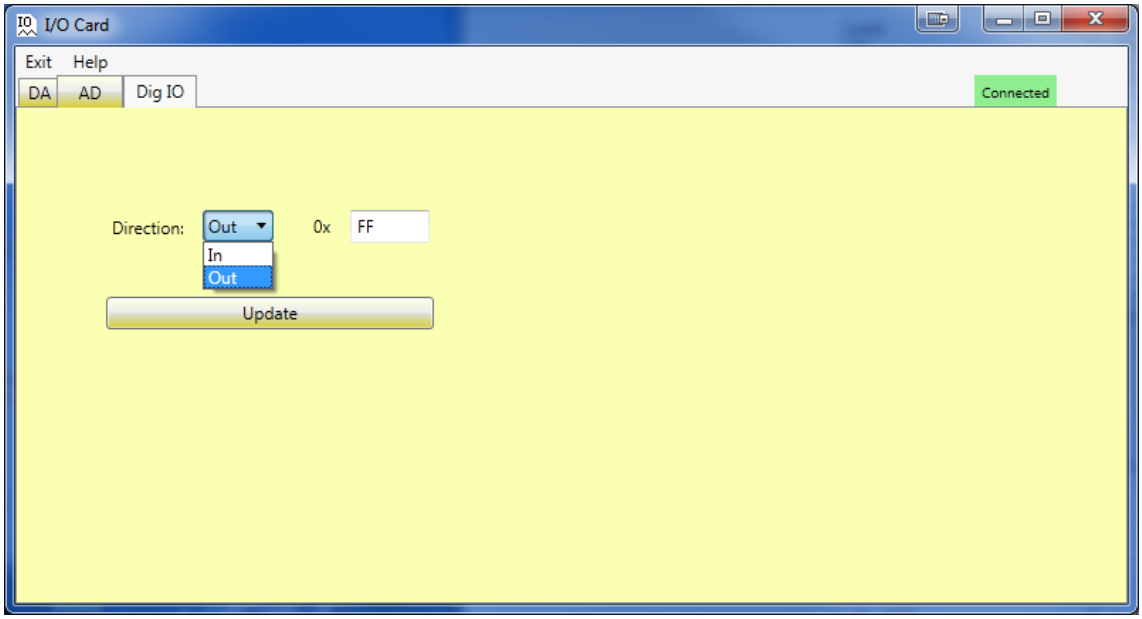
E.2 Fotka spodní DPS



F.1 Vzhled programu v PC - generátor a dig. signál

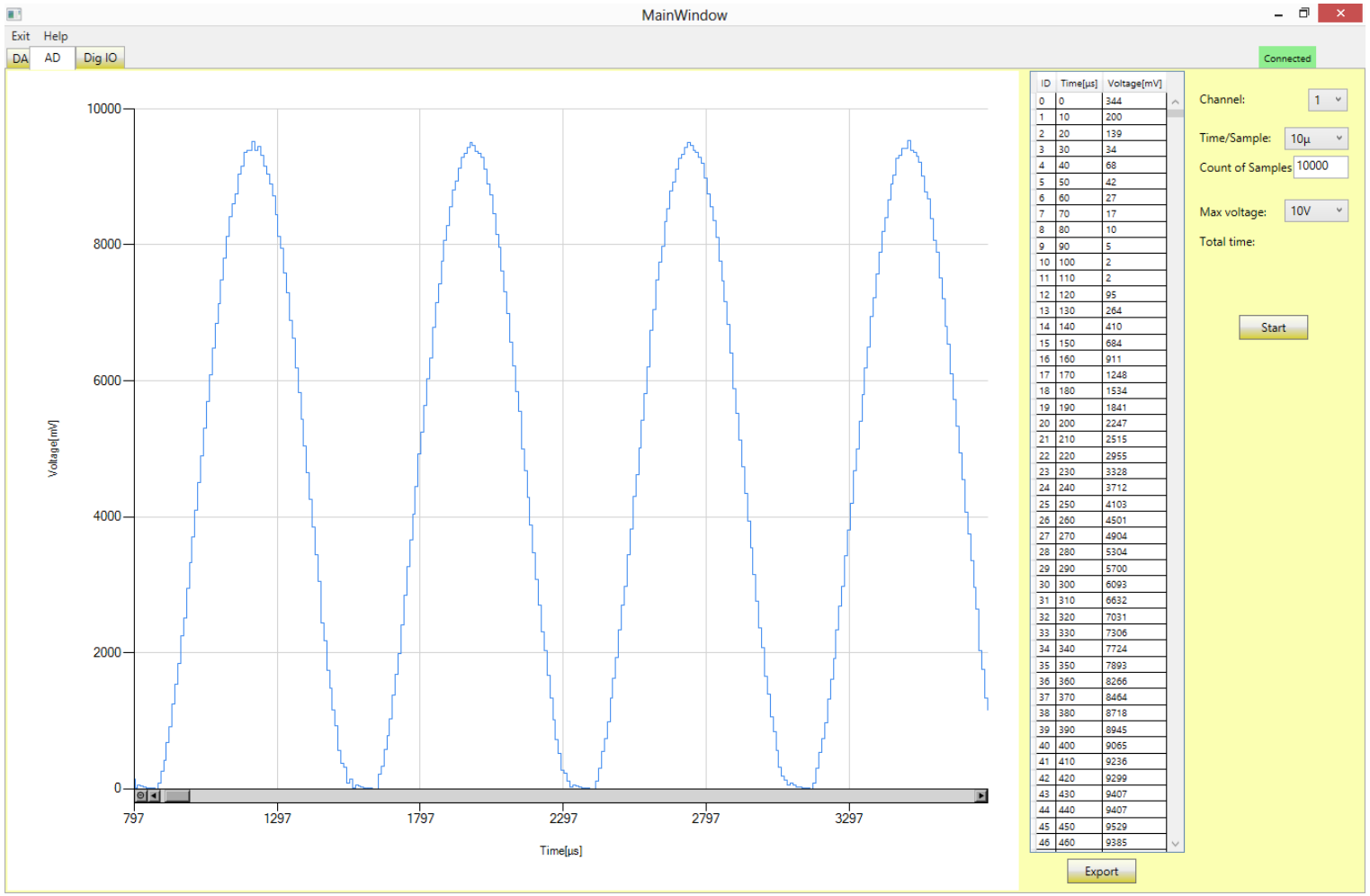


(a) Okno s náhledem na generovaný signál

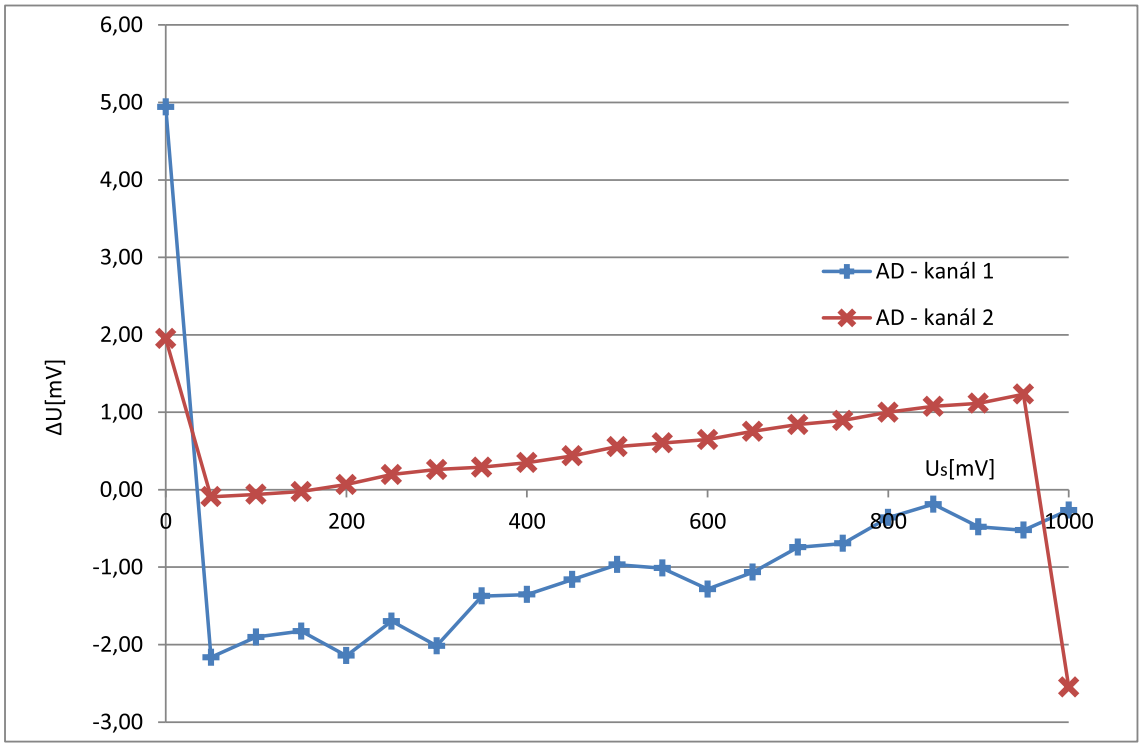


(b) Jednoduchá nabídka pro ovládání digitálních linek

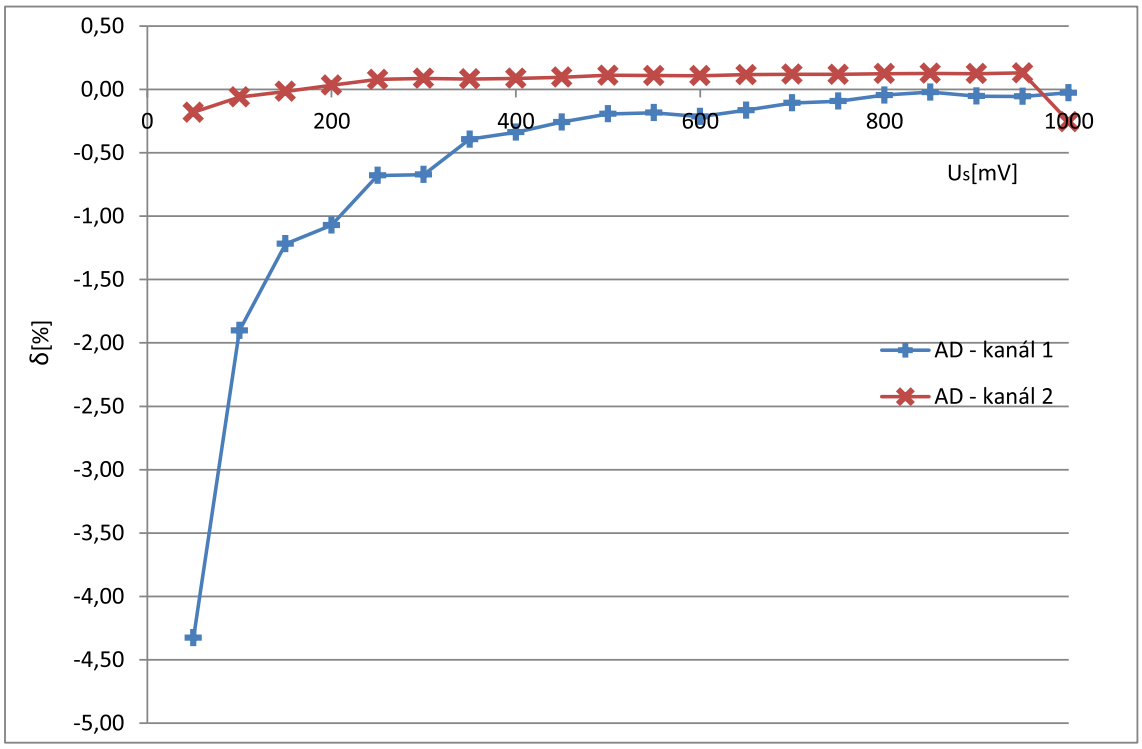
F.2 Vzhled programu v PC - naměřený signál



G.1 Přesnost měřicích kanálů na rozsahu do 1000mV

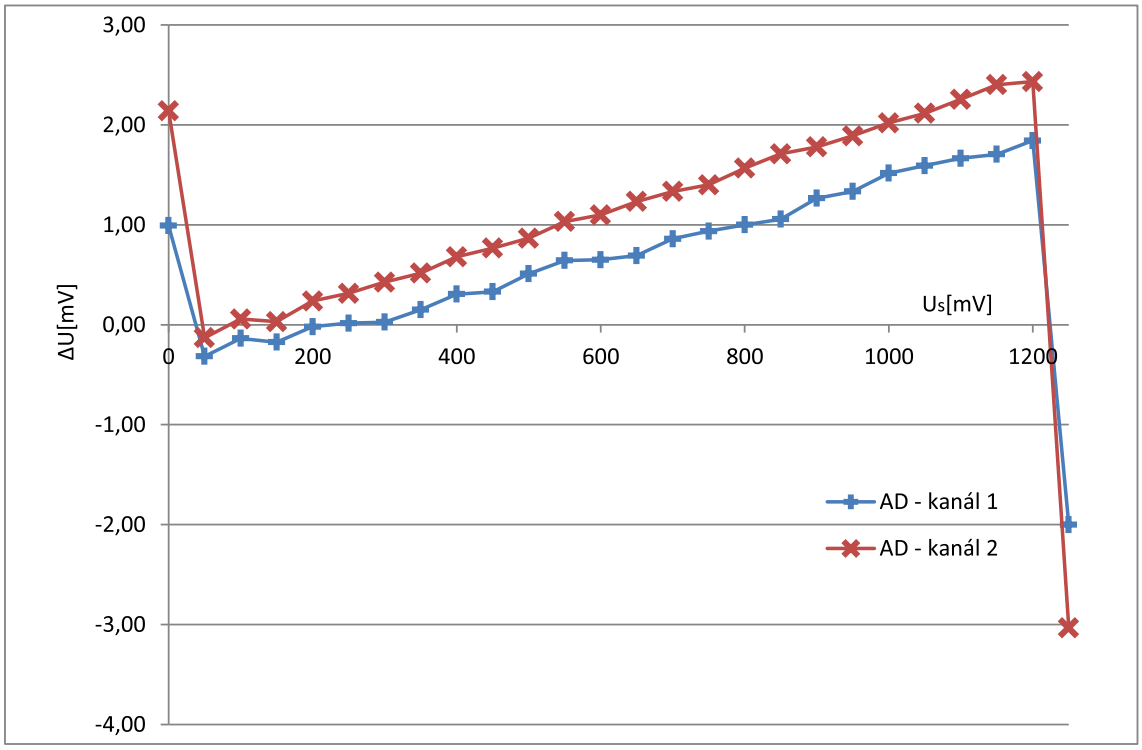


(a) Absolutní chyby naměřených hodnot

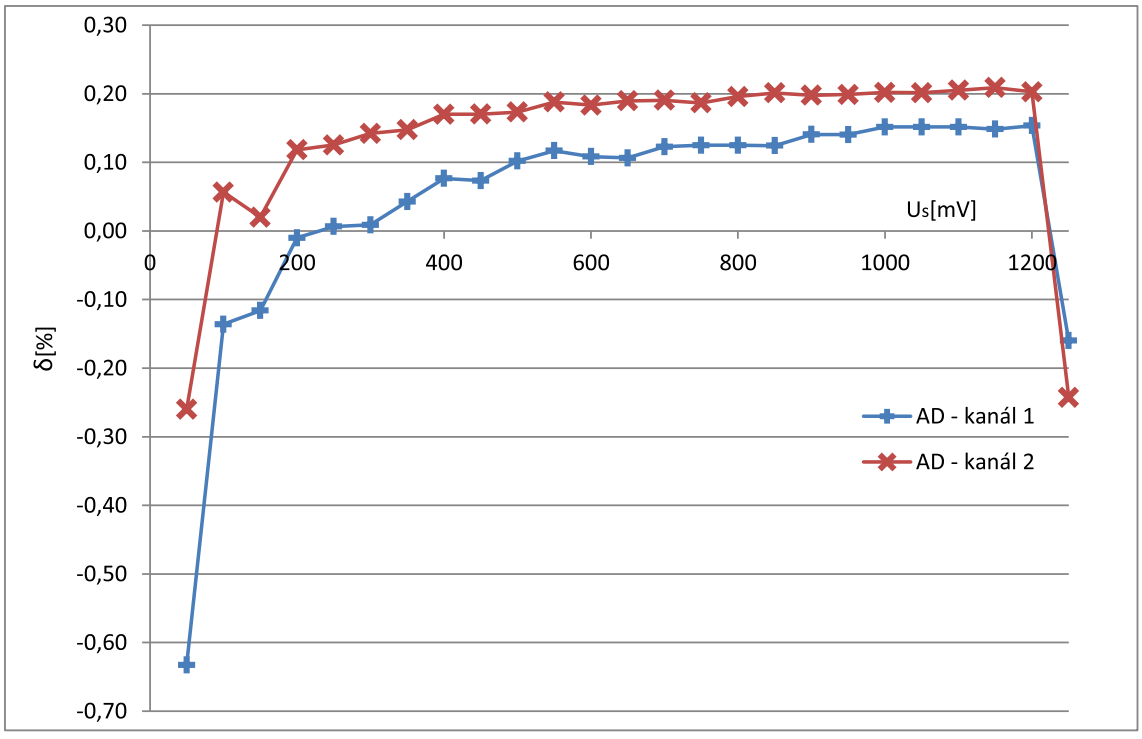


(b) Relativní chyby naměřených hodnot

G.2 Přesnost měřicích kanálů na rozsahu do 1250mV

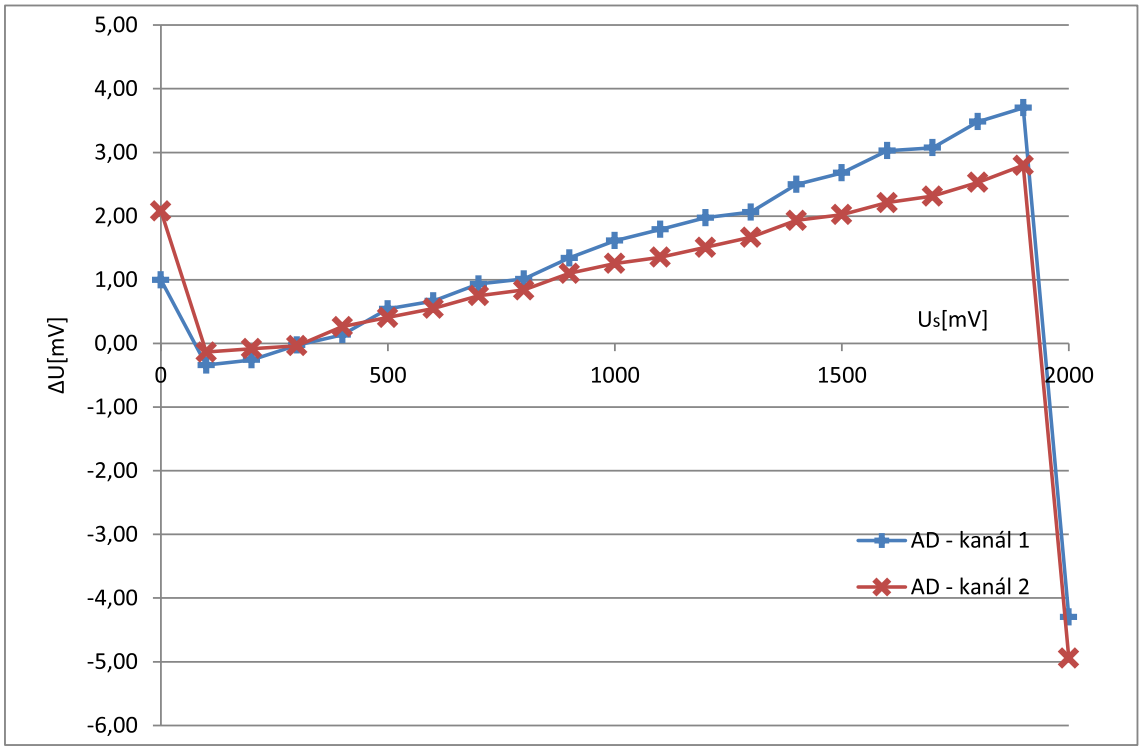


(a) Absolutní chyby naměřených hodnot

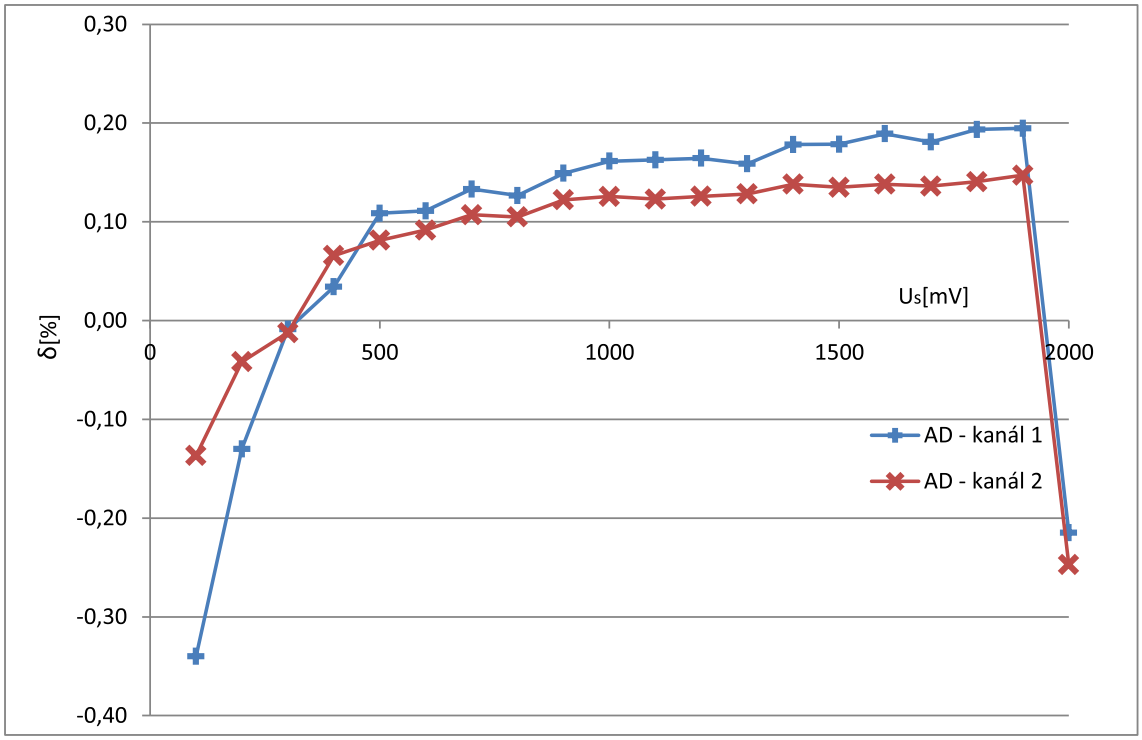


(b) Relativní chyby naměřených hodnot

G.3 Přesnost měřicích kanálů na rozsahu do 2000mV

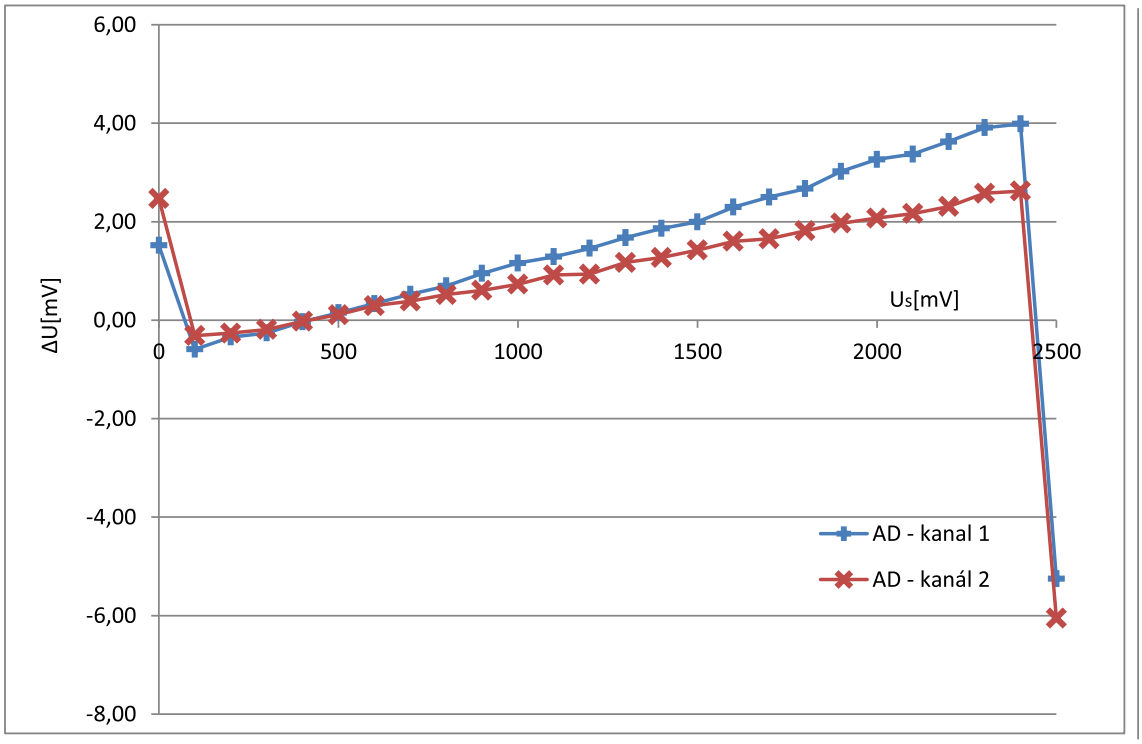


(a) Absolutní chyby naměřených hodnot

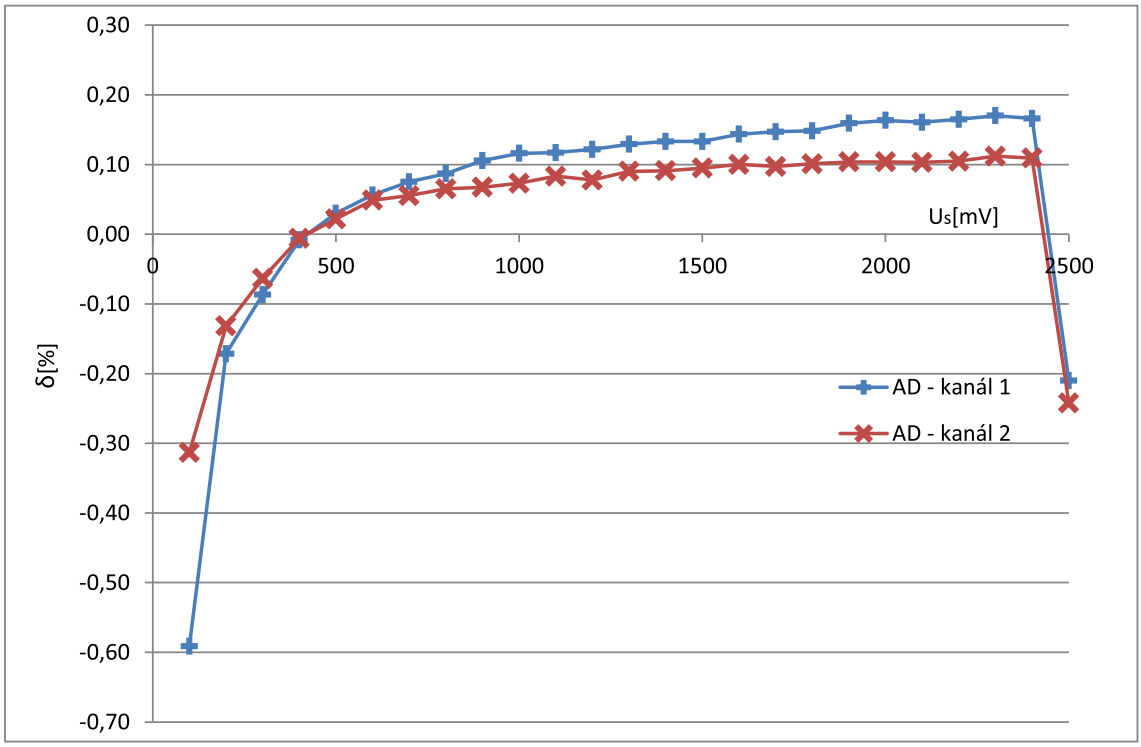


(b) Relativní chyby naměřených hodnot

G.4 Přesnost měřicích kanálů na rozsahu do 2500mV

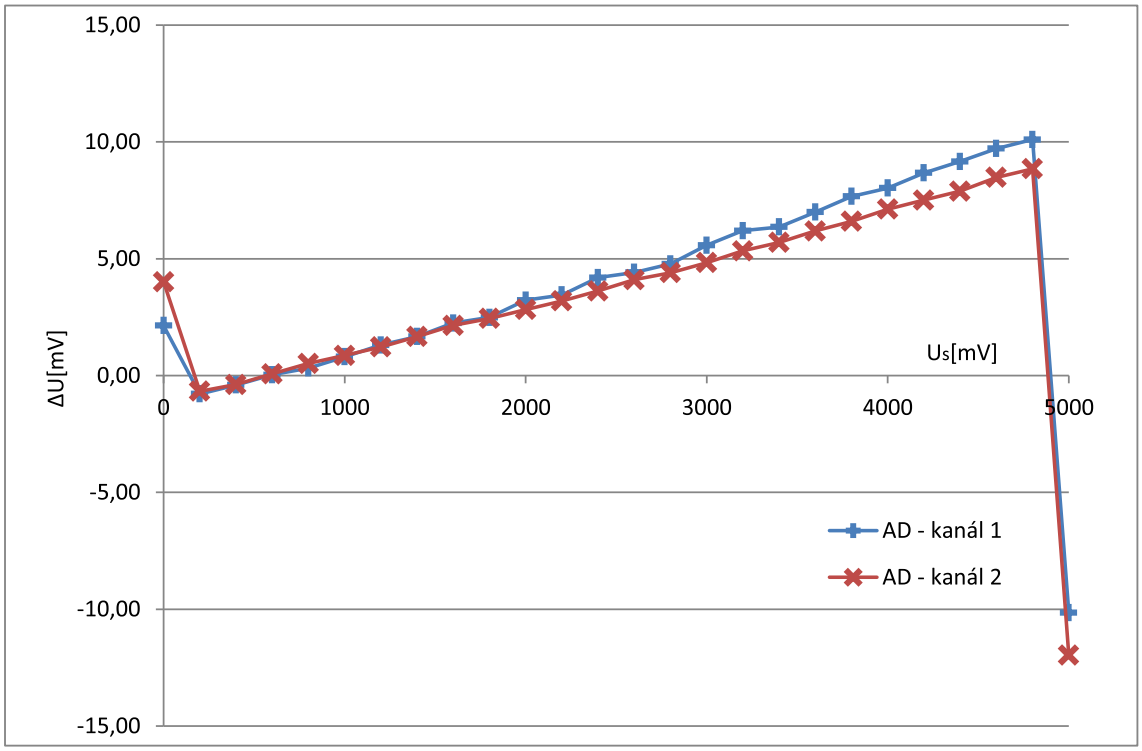


(a) Absolutní chyby naměřených hodnot

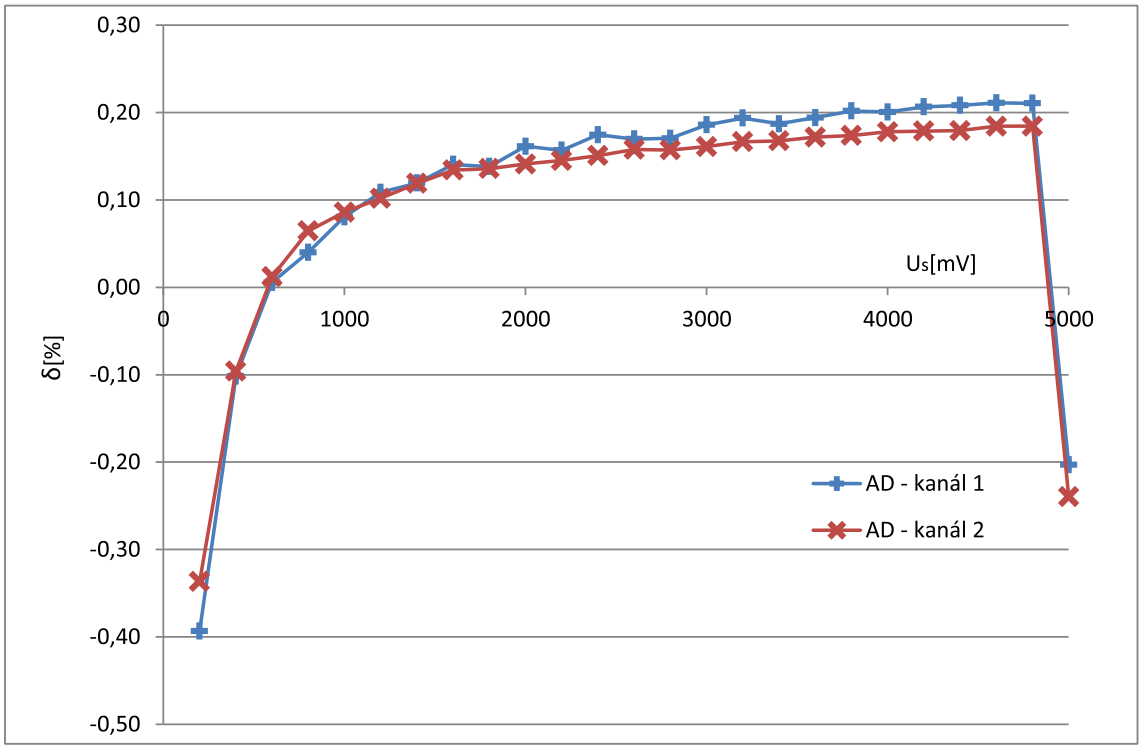


(b) Relativní chyby naměřených hodnot

G.5 Přesnost měřicích kanálů na rozsahu do 5000mV

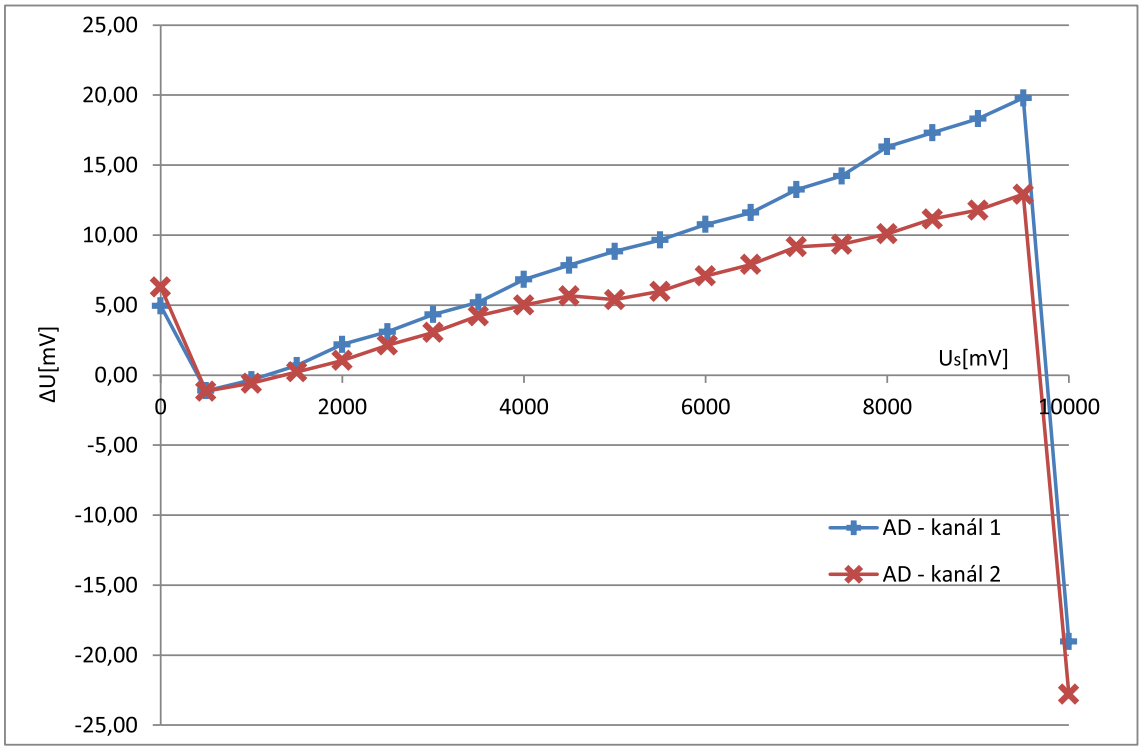


(a) Absolutní chyby naměřených hodnot

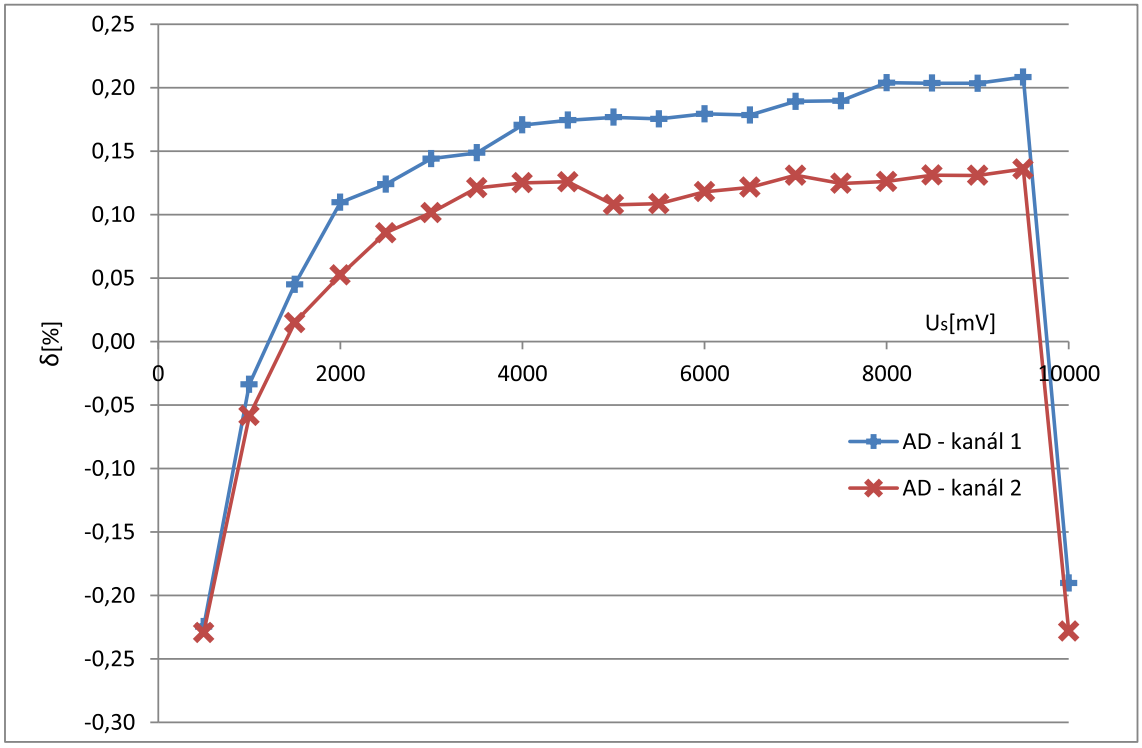


(b) Relativní chyby naměřených hodnot

G.6 Přesnost měřicích kanálů na rozsahu do 10000mV

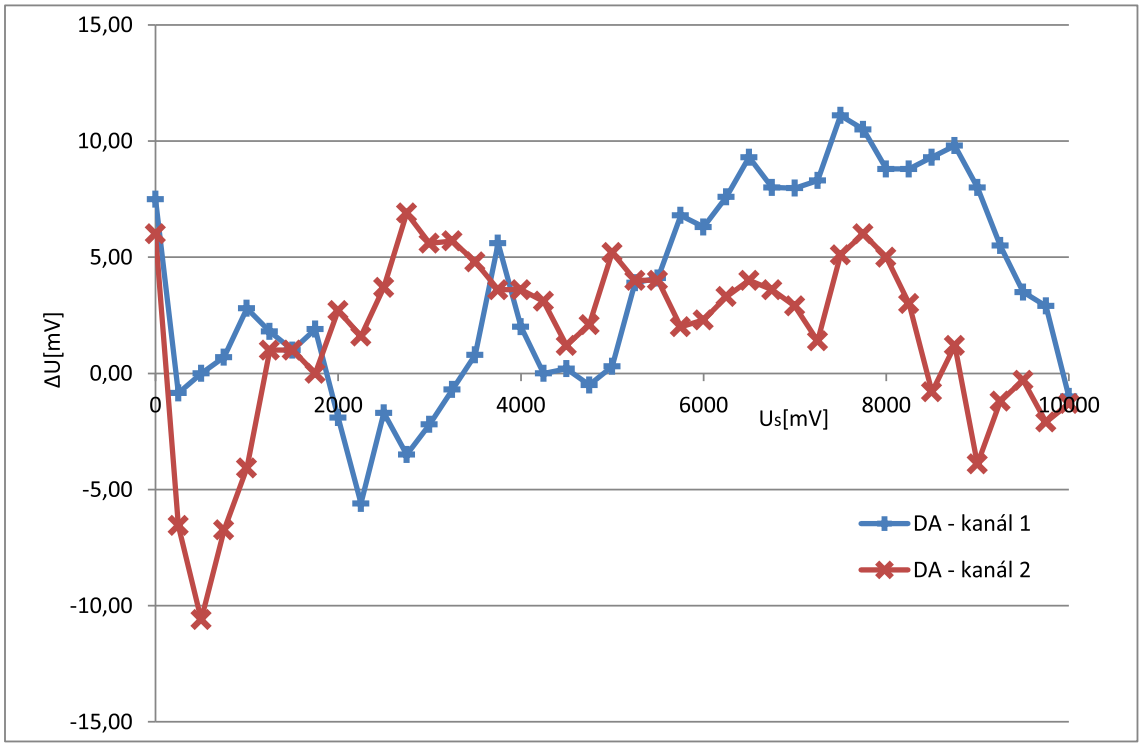


(a) Absolutní chyby naměřených hodnot

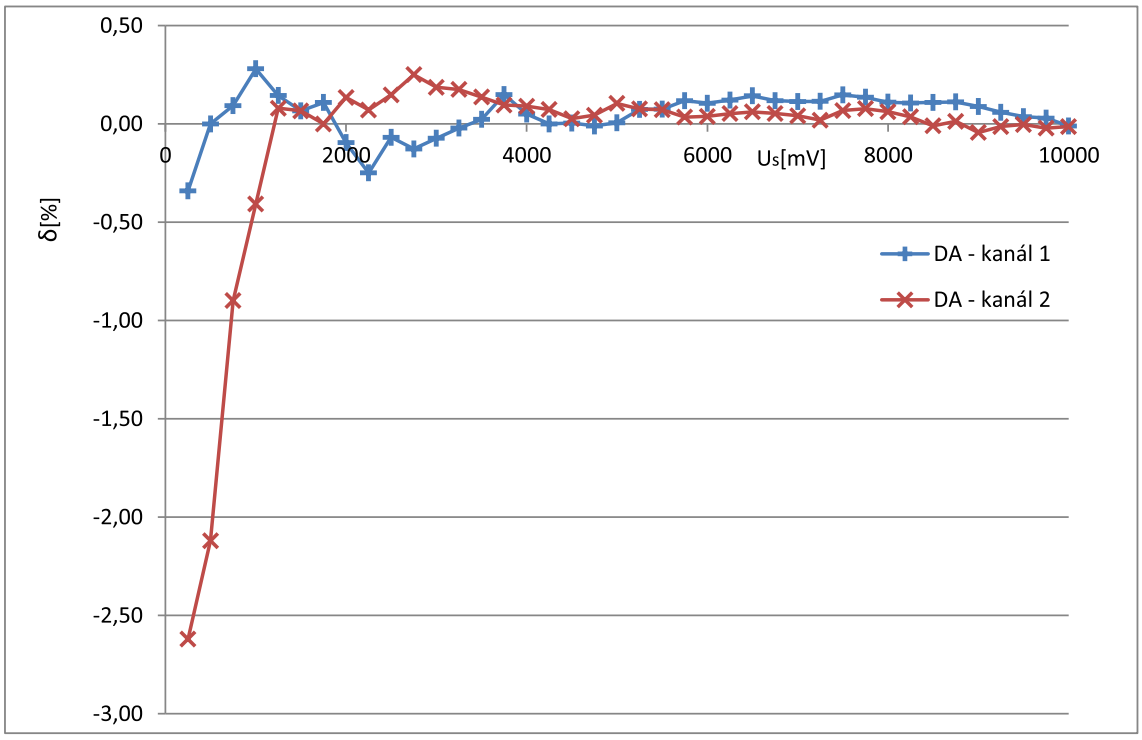


(b) Relativní chyby naměřených hodnot

G.7 Přesnost výstupních kanálů

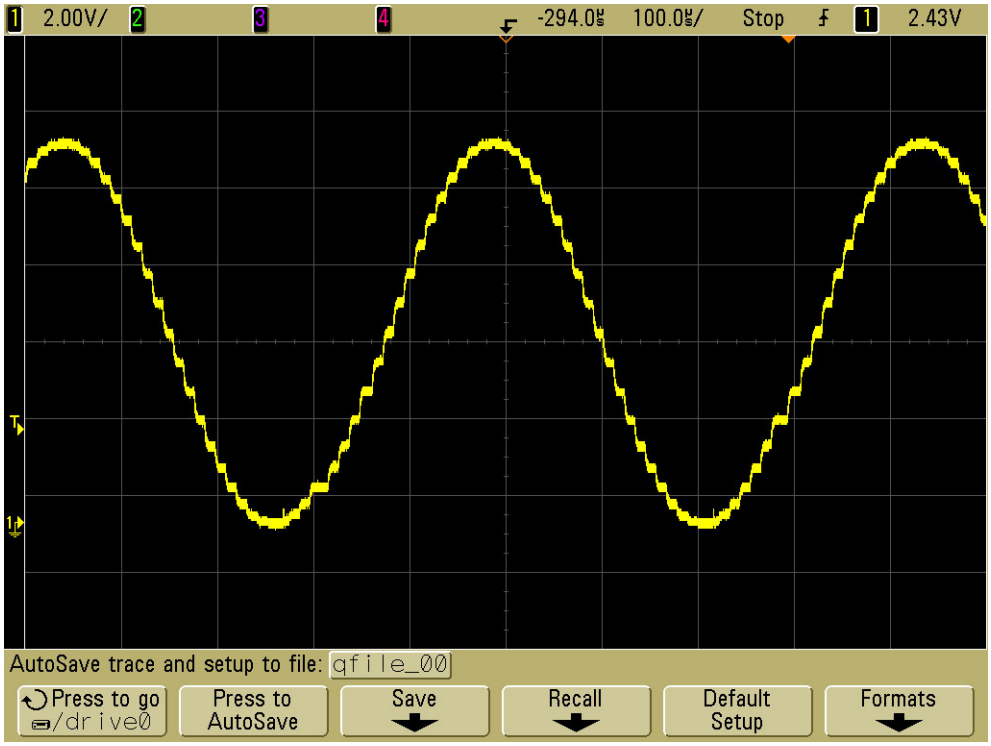


(a) Absolutní chyby naměřených hodnot

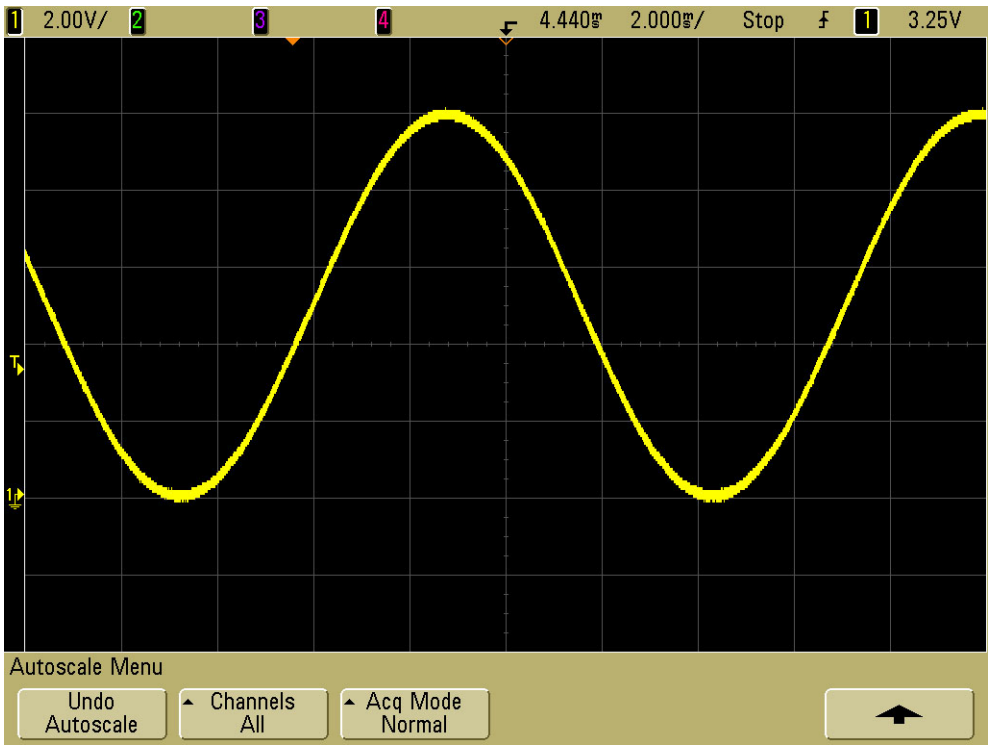


(b) Relativní chyby naměřených hodnot

H.1 Otestování anal. výstupu osciloskopem

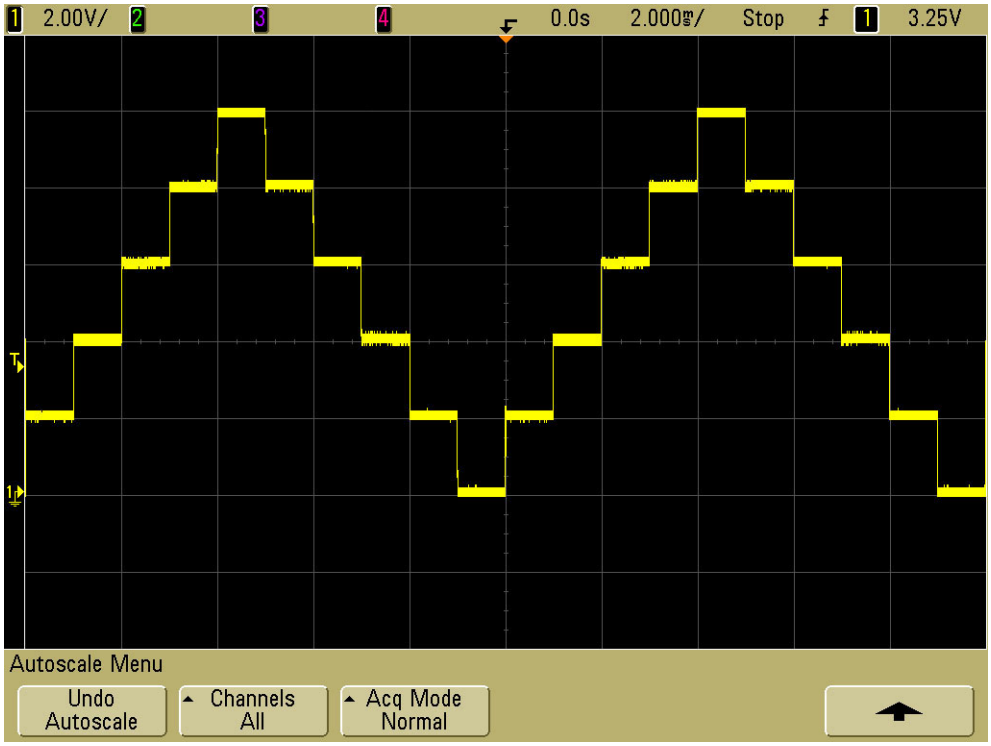


(a) Na výstup byl nastaven harmonický signál se 100 vzorky na periodu

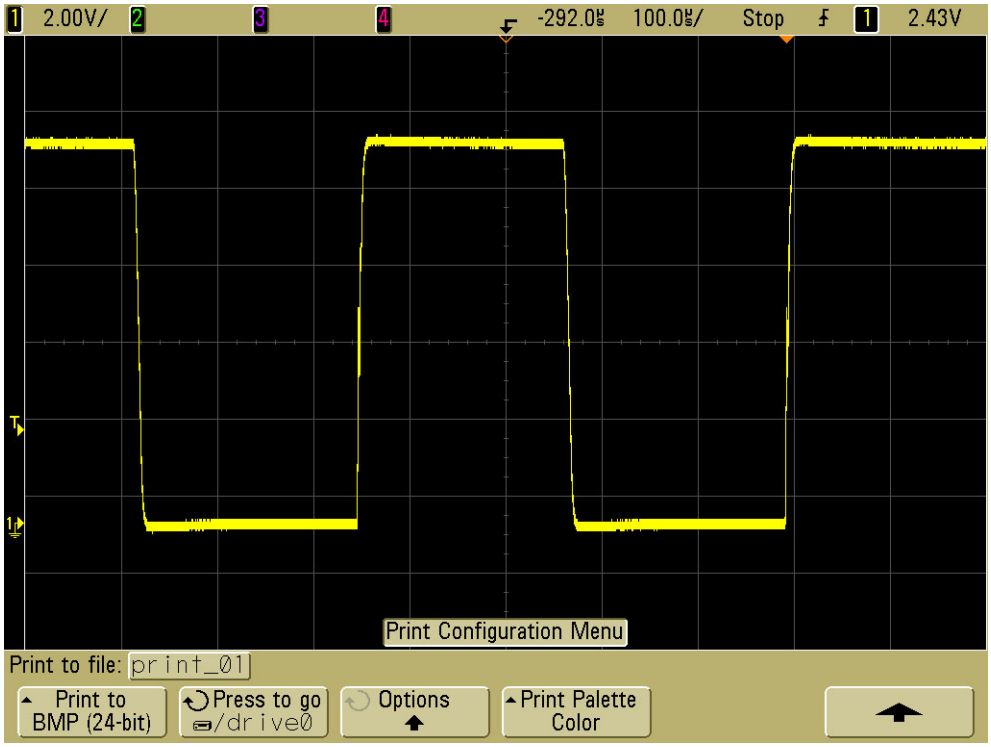


(b) Na výstup byl nastaven harmonický signál s 1000 vzorky na periodu

H.2 Otestování anal. výstupu osciloskopem



(a) Na výstup byl nastaven pilovitý signál o periodě 10ms



(b) Na výstup byl nastaven obdélníkový signál o periodě 500 μs

I.1 Obsah přiloženého CD

- **Složka Program_Module**
  Složka obsahuje všechny zdrojové kódy pro multifunkční kartu vytvořené v Dynamic C 9.62 .

- **Složka Program_PC**
  Složka obsahuje všechny zdrojové kódy pro program na PC ve Visual Studio 2012.

- **Složka DPS**
  Zde se nachází všechny podklady pro výrobu a osazení DPS navržené v Eaglu 5.9