ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ

Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

2005 Petr SUCHÁNEK

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra měření

Jednotka sběru dat s procesorem ARM7-TDMI

Diplomová práce

Vypracoval: Petr Suchánek
Vedoucí diplomové práce: Ing. Jan Fischer, CSc.
Praha 2005

Tady bude originál zadání ??

Anotace:

Cílem diplomové práce je návrh a realizace jednotky sběru dat pro měření parametrů signálů. Výsledné zařízení je koncipováno jako samostatný měřicí modul, který spolupracuje s nadřazeným systémem. Hlavním požadavkem je vysoký výpočetní výkon modulu a malé rozměry.

Dále je provedena analýza vhodných procesorů pro jednotku sběru a zpracování dat. V rámci analýzy je popsán obvodový návrh a realizován modul, na kterém byly prozkoumány možnosti použití RISC procesorů ARM. Na základě zjištěných poznatků byl realizován nový modul s vylepšenými parametry. Analyzovány byly také dostupné vývojové nástroje pro tvorbu aplikací.

V rámci diplomové práce bylo vytvořeno programové vybavení pro výpočty parametrů signálů a generování definovaných průběhů.

Anotation:

This graduation thesis describes design and realization of a data-acquisition module for signals' parameters measurement. The final device uses the „piggyback-module“ concept for cooperation with a supervisory application. The main features of the module are high computing performance and small dimensions.

The analysis of a suitable processor for data-acquisition module is conducted in the thesis. A prototype module was designed and basic features of systems with ARM processors were studied. Based on the gained findings a new improved module is designed. The availability of development tools for the ARM processors was also analyzed.

Software for measurement of signals' parameters and user-defined signal generation is developed and described.

P r o h l á š e n í

Prohlašuji, že jsem diplomovou práci *Jednotka sběru dat s procesorem ARM7-TDMI* vypracoval samostatně a použil podklady (literaturu, projekty, SW) uvedené v seznamu přiloženém k práci.

Nemám námitky proti půjčování, zveřejnění a dalšímu využití této práce ve smyslu § 60 zákona č.121/2000 Sb., o právu autorském a právech souvisejících s právem autorským, pokud s tím bude souhlasit katedra měření.

V Praze dne 28.1.2005 .....

Petr Suchánek

P o d ě k o v á n í:

Na tomto místě bych rád poděkoval vedoucímu diplomové práce Ing. Janu Fischerovi, CSc., který cennými připomínkami směroval mé úsilí k jejímu zdárnému dokončení, a spolupracovníkům ve firmě AMiT, spol. s.r.o., kteří mi vždy a ochotně podali pomocnou ruku, zejména Tomášovi Novákovi za přečtení a korekturu práce.

Velké poděkování patří mým rodičům a přítelkyni Daniele, kteří mě po celou dobu studia podporovali.

V neposlední řadě bych rád poděkoval i svému kolegovi Petrovi Česákovi za spolupráci při osvojování problematiky ARM procesorů.

Obsah

Seznam obrázků

Kapitola 1

Úvod a cíl práce

V průmyslových měřicích a řídících systémech je velmi často nutné měřit parametry signálů. Předmětem zájmu v této oblasti jsou stejnosměrné nebo střídavé signály o síťovém kmitočtu. Jejich průběhy však bývají zatíženy superponovaným šumem, který nepříznivě ovlivňuje přesnost měřených parametrů. Nejdůležitějšími parametry jsou efektivní hodnota průběhu, elektrický výkon, perioda a fázový rozdíl dvou průběhů. Soustavy, ve kterých se tyto parametry měří, jsou soustavy jednofázové nebo vícefázové. Méně častý, ale výpočetně mnohem náročnější, je požadavek na výpočet spektra signálu. Z hlediska zarušení měřených průběhů a vzrůstající potřeby měřit s vysokou přesností je nutné konstruovat takové měřiče, které nejsou rušivými signály ovlivňovány. Měřiče bývají použity např. pro měření spotřeby v elektronických elektroměrech, měření výkonu, fázování generátorů elektrické energie na rozvodnou síť, měření spektální čistoty napájecí sítě, řízení jednoduchých strojů, monitoring různých systémů, systémy řízení výroby, vzduchotechnické a vytápěcí jednotky, jednotky sběru dat a mnoho dalších aplikací.

V minulosti bylo k měření parametrů signálu používáno analogových převodníků a násobiček. Jejich nevýhodou byla nestálost elektrických parametrů pasivních součástek (např. vlivem teplotní závislosti). Analogová zapojení nejsou snadno modifikovatelná, a tak při požadavku na zlepšení přesnosti měření vyžadovala zpravidla změnu zapojení. Pro měření výkonu lze použít jednoúčelové obvody (např. obvod ADE7757 od firmy Analog Devices [33]), nebo komplexní systémy na bázi PC a zásuvných měřicích karet[41]. V dnešní době se problematika měření parametrů signálů, zejména efektivní hodnoty a výkonu, ubírá směrem číslicového zpracování vzorkovaných průběhů. Číslicové zpracování může být provedeno např. mikroprocesorovým systémem. Tím se systém stává velmi snadno modifikovatelný pouhou změnou programu. Mikroprocesorový měřicí systém může provádět nejen sběr dat z různých senzorů a jejich nastavování, ale navíc může naměřená data zpracovávat (např. korigovat nelinearity použitých senzorů) a současně zprostředkovávat komunikaci s nadřazeným systémem po sběrnicích různého typu.

V systémech, kde jsou měřená data používána k dalšímu řízení určitého procesu, je zpravidla kvůli pořizovací ceně systému požadováno, aby se výpočtům věnoval hlavní procesor. Ten již může být dostatečně vytížen řízením procesu. Tato koncepce vyžaduje výkonné a drahé procesory, které vedou na složité systémy. Možným řešením je separace výpočetně náročných úkolů do jiného modulu, který provádí měření a výpočty, a na dotaz poskytuje dílčí výsledky. Pro takový modul se ujalo označení *„měřicí koprocesor“*. Při volbě vhodného mikroprocesoru pro měřicí koprocesor existuje celé spektrum možností, kde na straně jedné mohou být výkonné signálové procesory (DSP), a na straně druhé výkonově slabší procesory na bázi velmi známých mikroprocesorů řady 8051. *Signálové procesory* jsou velmi vhodné pro výpočetní aplikace orientované na matematické operace s daty. Nejsou ale snadno programovatelné a program vytvořený pro jeden typ procesoru není přenesitelný na jiný typ kvůli hardwarovým odlišnostem jednotlivých procesorů. Zpravidla bývají také dražší a vyžadují externí paměť programu. Na druhé straně, procesory založené na bázi velmi oblíbené řady s jádrem 8051 jsou levnější, velmi rozšířené, snáze programovatelné, běžně mají paměť dat, programu, někdy i EEPROM na čipu, ale zdaleka nedosahují takového výpočetního výkonu. Vhodným kompromisem se jeví mikroprocesory typu *ARM*, které pracují s výkony až stovek MIPS (million instructions per second) nativně ve 32 bitech. V dnešní době jsou již tyto

mikroprocesory osazeny pamětí RAM a Flash, analogově-číslicovým a číslicově-analogovými převodníky s vhodným rozlišením přímo na čipu, a proto jsou vítaným prostředkem pro konstrukci právě zmíněných modulů s funkcí měřicího koprocesoru.

Inspirací pro technickou realizaci systému vyvinutého v rámci diplomové práce byl systém DV456 vyvinutý firmou AMiT, spol. s.r.o. Jednotka umožňuje vícekanálové měření a je osazena 16bitovým procesorem 80C166, který pracuje na hranici svých možností a další změna SW je prakticky nemožná.

Diplomová práce se věnuje možnostem návrhu jednotky měřicího koprocesoru a implementaci známých metod číslicového zpracování signálů. Nabízí odvození a shrnutí potřebných požadavků na součástkovou základnu číslicového modulu a programové vybavení *měřicího koprocesoru*. Nejdůležitějším cílem práce je **ověření použitelnosti a vhodnosti procesorů s jádry ARM** pro aplikace v jednotkách s matematickým zpracováním dat a **analýze rychlostí provedených výpočtů** dosažitelných s mikroprocesorovým modulem osazeným procesorem řady ARM. V práci je proveden návrh a realizována vzorová jednotka ve formě zásuvného modulu („piggyback“). Jednotlivé body práce lze stručně shrnout do těchto požadavků:

- Provést analýzu požadavků na návrh jednotky měřicího koprocesoru,
- ověřit použitelnost a vhodnost procesorů s jádry ARM pro měřicí koprocesor,
- a provést implementaci známých metod číslicového zpracování měřených signálů.
- Navrhnou a realizovat prototypového modulu.
- Na prototypovém modulu analyzovat dosažitelné rychlostí výpočtů.

Diplomová práce je doplněna přílohami se schématy jednotlivých modulů, seznamy součástek a slovníčkem použitých pojmů. K práci je přiložen CD ROM s doplňkovými informacemi z oblasti ARM procesorů, katalogovými listy vybraných součástek, zdrojových programů, schémat a výkresů desek plošných spojů.

Kapitola 2

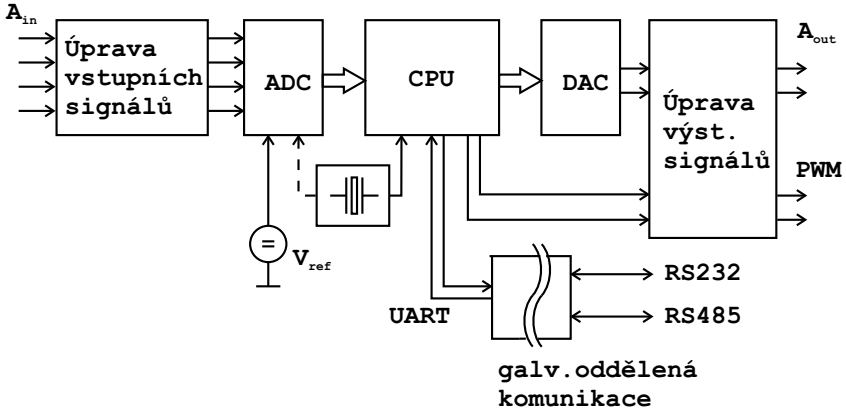
Rozbor požadavků kladených na jednotku sběru dat

Pro jednotku sběru dat, která bude provádět výpočet parametrů měřených průběhů, jsou důležité dvě stránky návrhu, součástková základna (tj. použité převodníky a procesor) a výpočetní výkon použitého procesoru včetně znalosti výpočetní náročnosti jednotlivých parametrů signálu. Obě oblasti jsou provázané, např. požadavkem na dostatečný počet vzorků a tedy volbu vhodné vzorkovací frekvence.

Co by však taková jednotka měla obsahovat? Po jednoduché úvaze mohou být základní části, které by měl nebo musí měřicí koprocesor obsahovat, přehledně shrnuty jako

- vícekanálový A/D převodník pro odběr vzorků z měřených průběhů,
- D/A převodník (vícekanálového) pro generaci měřicího průběhu,
- analogové i číslicové výstupy pro řízení senzorů,
- popř. PWM výstupy pro řízení výkonových obvodů.

Blokové schéma takového systému je na obr. 2.1. V bloku „úprava vstupních signálů“ se nachází např. obvody impedančního přizpůsobení, úprava vstupních napěťových úrovní (vstupní zesilovače) a antialiasingový filtr. Modul může obsahovat další, systémové obvody, které nejsou nezbytně nutné pro vlastní měření, ale jsou žádané, pokud je modul použit jako samostatná, tzv. „stand-alone“ aplikace[1]. Těmito obvody mohou být např. obvod reálného času RTC včetně zálohovací baterie, konfigurační EEPROM a zobrazovací jednotka (pokud je požadována) včetně příslušného ovladače.



Obrázek 2.1: Blokové schéma jednotky měřicího koprocesoru

Jednotka měřicího koprocesoru může komunikovat s nadřazeným systémem přes galvanicky oddělenou sériovou linku. Důvodem je zamezení rušivému vlivu proudů protékajících

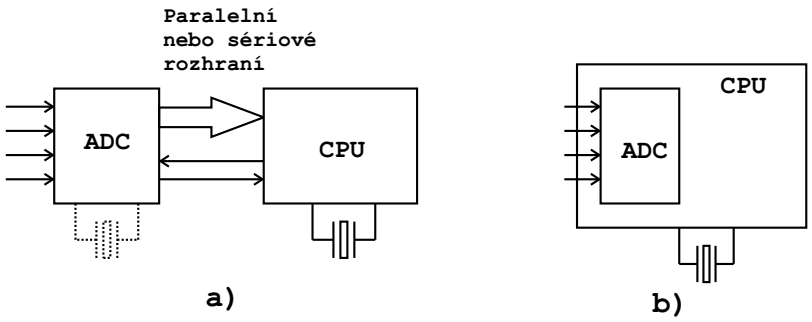
[1] Jednotka měřicího koprocesoru není připojená k nadřízenému systému, ale sama zobrazuje naměřené výsledky nebo rozhoduje o dalším řízení procesu.

společným zemnícím vodičem, anebo pokud jsou obě zemní svorky obou jednotek připojeny na rozdílné potenciály[2].

2.1 Požadavky na A/D převodník

Jaké jsou požadavky na vstupní část jednotky, která zprostředkovává styk s vnějším prostředím? Nejdůležitějšími parametry při výběru vhodného analogově-číslicového převodníku jsou rozlišovací schopnost, maximální vzorkovací frekvence, počet kanálů, struktura převodníku, šířka pásma vstupní části převodníku a způsob jeho připojení k procesoru. Důležitými parametry jsou samozřejmě i chyby převodníků, zejména integrální a diferenciální nelinearita a počet efektivních bitů (ten je však komplexního charakteristikou celého zapojení analogové části včetně kvality provedení desky plošných spojů). Efektivních počet bitů určuje přesnost měření.

Možnosti umístění vzorkovacího A/D převodníku jsou interní a externí převodník, jak je naznačeno v blokovém schématu na obr. 2.2. *Interní A/D převodník* má tu výhodu, že nezabírá místo na desce plošných spojů (nevznikají problémy s fyzickým propojením obou součástek), a komunikační interface mezi převodníkem a procesorem je již implementován v procesoru. Také je možnost použít kanál přímého přístupu do paměti DMA, pokud jím je procesor vybaven, pro zápis odebraných vzorků do datové paměti aplikace. Jejich nevýhodou ovšem můžou být horší parametry. *Externí A/D převodník* je k procesoru připojen buď paralelně, nebo prostřednictvím sériového rozhraní, většinou SPI nebo SPORT. Většinou je výhodné, hlavně z hlediska komunikace s připojeným převodníkem a velikostí zabraného místa na desce plošných spojů, pokud je převodník umístěn přímo na čipu procesoru, viz. obr. 2.2 b). Stejná je situace u D/A převodníku, pokud je umístěn přímo na čipu, je to výhodnější. V obou případech musí převodníky splňovat požadavky zadání.



Obrázek 2.2: Připojení A/D převodníku k procesoru: a) externí převodník — sériová nebo paralelní sběrnice, b) interní převodník na čipu procesoru

Protože skutečný průběh měřených napětí je v praxi značně zarušen, je nutno volit velký počet vzorků na periodu. Podrobný rozbor je proveden v kap. 5.2. Velikost vzorkovací frekvece vyplývá z požadovaného počtu vzorků na periodu maximální harmonické složky, kterou je nutno měřit. Např. pro 50. harmonickou a 200 vzorků na periodu je potřebná vzorkovací frekvence $50 * 50Hz * 200 = 500kSa/s$.

Dalším z parametrů, který rozhoduje o výběru převodníku, je možnost použití externího zdroje referenčního napětí, kdy při použití kvalitních monolitických referencí je možné dosáhnout lepší přesnosti a stability, než s referencemi umístěnými na čipu převodníku.

[2]K tomu dochází v případech, kdy jsou obě jednotky uzemněny, ale je mezi nimi velká vzdálenost. Pak na společném (zemním) vodiči vzniká úbytek napětí, pokud zemí mezi jednotkymi prochází velké rušivé proudy.

2.2 Požadavky kladené na procesor

Náročné matematické operace bylo v historii možné provádět pouze na počítačích typu PC vybavených zásuvnými měřicími deskami. Zásuvné měřicí desky byly osazeny vstupním zesilovačem s programovatelným zesílením (zesilovač nebyl vždy u starších levných desek), vzorkovacím obvodem (často součástí A/D převodníku), A/D převodníkem a vyrovnávací pamětí. Veškeré výpočty obstarává v takové situaci procesor PC. Pomocí těchto zásuvných desek je možné měřit fázový rozdíl s nejistotou desetin stupně, DC a AC napětí s přesností na desetiny až setiny procenta a výkon na setiny až desetiny procenta podle velikosti účiníku $\cos\varphi$ [4].

V této kapitole je odhlédnuto od možnosti použití zásuvných měřicích desek, a je proveden rozbor výpočetní náročnosti požadovaných parametrů. Ty jsou obecně výpočetně různě náročné, jedná se o sumační vzorce, které sčítají hodnoty vzorků (z jedné periody) jak je tomu při výpočtu např. střední hodnoty, nebo násobí vzorky, jak je tomu při výpočtu efektivní hodnoty a výkonu. Vzorce, podle kterých jsou počítány některé důležité parametry signálu jsou

$$U_{DC} = \frac{1}{N}\sum_{i=0}^{N-1} u(t_i), \qquad U_{RMS} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} u^2(t_i)}, \qquad P = \frac{1}{N}\sum_{i=0}^{N-1} u(t_i)\, i(t_i),$$

kde N je počet vzorků na periodu, t_i jednotlivé diskrétní časové okamžiky odběru vzorků, $u(t_i)$ vzorky příslušející průběhu napětí a $i(t_i)$ vzorky příslušející průběhu proudu. Výpočty periody a fázového rozdílu jsou méně náročné na výpočetní výkon procesoru, proto je v této analýze naznačena výpočetní náročnost výpočtu elektrického výkonu podle vztahu $P = \frac{1}{N}\sum_{i=0}^{N-1} u_i\, i_i$.

Pro další úvahy předpokládáme procesor s „load/store“ architekturou (aritmeticko-logické operace jsou prováděny nad registry a ne v paměti). Násobené vzorky jsou nejprve načteny z paměti do registrů, vynásobeny a mezivýsledky uloženy zpět do paměti. Ukládání mezivýsledků je třeba pouze v případě, že je k dispozici málo registrů a je třeba uchovávat velké množství mezivýsledků.

Jednotlivé operace, které jsou vykonány při výpočtu hodnoty elektrického výkonu jsou pro N vzorků:

- $2N$-krát načtení hodnot z paměti (R Read),
- N-krát výpočet součinu (M Multiply),
- N-krát načtení mezivýsledku (RSR Read SubResult),
- N-krát přičtení aktuálního mezivýsledku k uložené hodnotě mezivýsledku (A Add),
- N-krát uložení mezivýsledku do paměti (SSR Store SubResult),
- jednou dělení hodnotou N (D Divide).
- V případě efektivní hodnoty jednou provedeme výpočet odmocniny.

Sečtením jednotlivých fází výpočtu lze ukázat, že čas potřebný pro výpočet elektrického výkonu z N vzorků je celková doba výpočtu výkonu T_{vypocP} vyjádřena jako

$$T_{vypocP} = N \cdot (2R + M + RSR + A + SSR + K) + D \doteq N \cdot T_i,$$

kde K je konstanta daná instrukcemi, které se přímo nepodílejí na výpočtu parametrů a $T_i = 2R + M + RSR + A + SSR + K$ je doba výpočtu jednoho mezivýsledku. Konstantu je nutné předem zjistit a v dalších výpočtech s ní počítat. Další nežádoucí prodloužení doby výpočtu může být způsobeno vlastním přístupem k jednotlivým odebraným vzorkům průběhu a jeho parametrům. Naměřená data a parametry jsou uloženy buď v různých proměnných,

nebo jako datové struktury. Přístup pomocí datových struktur je sice programátorsky pohodlnější, ale při výpočtu zdlouhavější. Příspěvky jednotlivých fází výpočtu k celkové době lze minimalizovat volbou vhodné programovací metody. V případě tzv. real-time programování, kdy je po každém odběru vzorku proveden výpočet jeho příspěvku k měřené veličině, budou dominovat zejména časy RSR a SSR. Naopak, v případě použití metod „off-line“, kdy je nejprve odebráno dostatečné množství vzorků a teprve poté proveden výpočet, jsou časy RSR a SSR nulové, neboť mezivýsledky jsou uchovány v registrech. Pokud známe dobu trvání jednotlivých instrukcí, můžeme provést alespoň odhad celkové doby trvání výpočtu.

Provedeme úvahu pro zjištění nutného výpočetního výkonu modulu. Pokud je třeba vzorkovat signály s kmitočtem napětí rozvodné sítě až např. do 50. harmonické s 200 vzorky na periodu, je třeba k získání efektivní hodnoty přibližně $50 \cdot 50Hz \cdot 200 = 500$ výpočtů za sekundu. Pokud je výpočet jednoho mezivýsledku proveden 10 instrukcemi a průměrná doba na vykonání jedné instrukce 3,5 strojových cyklů, je potřebná doba $T_i = 35$ strojových cyklů. Potom požadovaný výpočetní výkon procesoru může být vyjádřen jako $500k \cdot 10 = 5MIPS$. Takový výkon bez problémů dosahují DSP procesory a téměř jich nelze dosáhnout s jednočipovými procesory řady x51.

Jedním z dalších parametrů, který rozhoduje při volbě procesoru, je velikost paměti dat RAM a programu (např. FLASH) přímo na čipu procesoru a možnost připojení externích pamětí obou typů. Velikosti paměti určuje, jak rozsáhlé aplikace je možné procesorem obsluhovat. Dalšími, pro výpočetní aplikace velmi důležitými parametry, jsou počet bitů procesoru, a osazení násobičky[3], která zkracuje dobu potřebnou k vynásobení dvou čísel. Potom není nutné programovat jednotlivé rutiny pro násobení.

Procesory ARM jsou v poslední době osazovány integrovanou pamětí dat a programu a tak se stávají náhradou malých jednočipovým procesorů pro embedded aplikace. Intregrovaná paměť programu je velkou výhodou v porovnání s procesory DSP.

Dalším hlediskem je počet bitů, se kterými procesor standardně pracuje. Procesory *ARM jsou plně 32-bitové*. Díky tomu má procesor velmi výkonnou aritmeticko-logickou jednotku, může pracovat s vyšším rozlišením, a rychleji přistupovat do paměti (4-krát větší datová šířka oproti 8bitovým procesorům[4]).

Procesory ARM se v tomto ohledu jeví jako vhodné řešení, které ve spojení s interními převodníky, integrovanou pamětí dat i programu a hardwarovou násobičkou umožní realizovat modul s vysokým výpočetním výkonem zabírajícím na desce plošných spojů minimální místo.

Procesor by měl být vybaven těmito bloky:

- integrovaná paměť dat a programu,
- komunikační rozhraní UART,
- vhodný interface pro připojení A/D a D/A převodníků, pokud nejsou integrovány na čipu
- blok PWM pro řízení výkonových obvodů, atd.

2.3 Požadavky kladené na strukturu programu

Pokud jsou již převodník a procesor vybrány, je vhodné zvážit strukturu vlastního programu, který je v jednotce spuštěn. Při jeho tvorbě je obecně několik možností, jak provádět výpočet. V prvním případě se jednotlivé výpočty provádějí po každém odběru vzorků, kdy se počítá

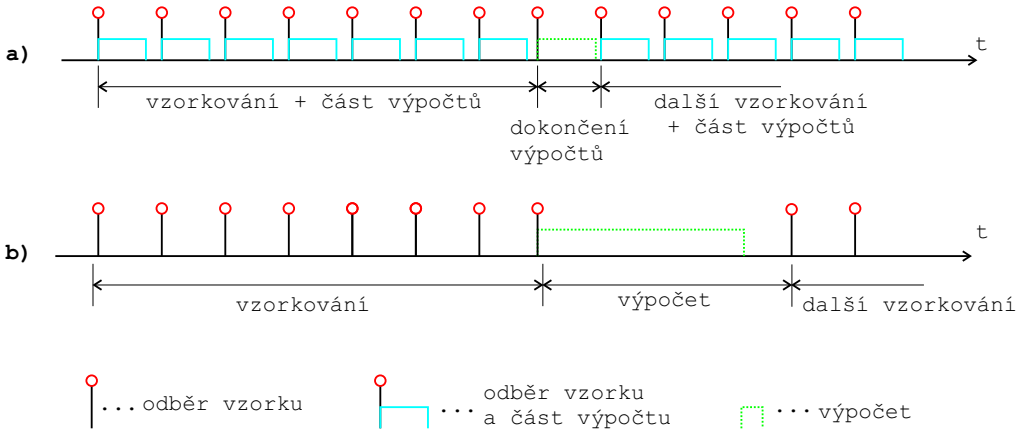
[3]Násobička, neboli „hardwarová násobička“, je speciální obvod umístěn v procesoru mimo aritmeticko-logickou jednotku ALU, který provádí pouze násobení. To má výhodu, že při psaní programu není nutné programovat rutiny pro násobení, ale jednoduše použít instrukci pro násobení.

[4]Při jednom přístupu do paměti je vybráno 32 bitů dat, tedy 4 Byte, což se efektivně jeví jako čtyřnásobné urychlení oproti 8bitovým procesorům.

částečný příspěvek aktuálně odebraného vzorku k měřeným veličinám. Po odebrání požadovaného počtu vzorků se provede dělení tímto počtem, popř. další výpočty (např. odmocnina u efektivní hodnoty). Je zřejmé, že perioda vzorkovací frekvence je limitována délkou doby výpočtu jednotlivých příspěvků všech parametrů. Tento styl se někdy nazývá *programování v reálném čase* a je naznačen na obr. 2.3 a). Výhodou této metody je, že není potřeba tak velké paměti[5], aby bylo možné uchovat všechny naměřené vzorky, ale stačí taková velikost paměti, kde lze uchovat všechny měřené parametry.

Druhou možností je rozdělení postupu na dvě části, nejprve je proveden odběr požadovaného počtu vzorků a teprve poté výpočet všech parametrů tak, jak je naznačeno na obr. 2.3 b). Je zřejmé, že touto metodou lze dosáhnout vyšších vzorkovacích frekvencí. Nevýhodné naopak je, že vlastní výpočet zdržuje začátek odběru další skupiny vzorků a je potřeba alokovat velkou část paměti pro požadovaný počet odebíraných vzorků. Tento styl bývá označován *off-line* zpracování dat.

U obou metod platí, že v mezičase, kdy výsledek aktuálně prováděného výpočtu není ještě platný, je pro nadřazenou aplikaci k dispozici výsledek předchozího výpočtu. Při komunikaci po sériové lince by mohlo u druhé možnosti dojít k překročení doby povolené délkou vzorkovací periody na obsluhu sériového kanálu. Proto tato varianta vyžaduje použití vyrovnávací paměti v UARTu sériové linky.



Obrázek 2.3: Struktura programu z hlediska okamžiku výpočtu parametrů: a) výpočet v reálném čase, b) výpočet off-line

Pokud doba výpočtu při off-line metodě nepřekračuje výrazně dobu vzorkovací periody, která je daná dobou výpočtu při zpracování dat v reálném čase, je možné považovat obě metody za výpočetně stejně náročné.

[5]pokud není prováděn výpočet FFT, který musí být všechny vzorky k dispozici najednou

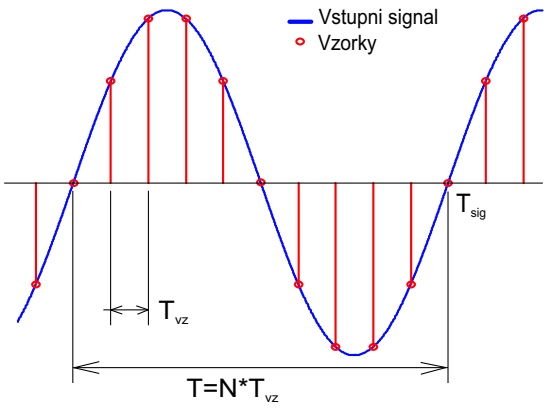
Kapitola 3

Popis vzorkovacích metod použitelných pro jednotku sběru dat

Vstupní data jsou pro potřeby číslicového zpracování získávána vzorkováním. Vzorkování přiřazuje vstupnímu, v čase spojitému signálu $x(t)$ časovou posloupnost vzorků $x_k = x(kT_{vz})$, kde T_{vz} je vzorkovací perioda, k celé číslo. Aby vzorky obsahovaly stejnou informaci jako původní spojitý signál, je třeba splnit vzorkovací teorém, že $f_{vz} > 2f_{max}$, kde f_{vz} je vzorkovací frekvence $f_{vz} = 1/T_{vz}$, f_{max} je maximální frekvence obsažená ve spektru vstupního signálu. Rozlišuje se mezi synchronním[6] a asynchronním vzorkováním podle počtu vzorků v periodě měřeného signálu. V případě vícekanálových se jedná o simultánní a sekvenční vzorkování. V případě následného zpracování vzorků, zejména pro spektrální analýzu vyžadující vzorkování celistvého počtu period signálu a rovnoměrné rozdělení vzorků, není vhodné používat metody náhodného vzorkování [7].

3.1 Synchronní vzorkování

V ideálním případě je vzorkovací frekvence synchronní s frekvencí vstupního signálu, viz obr. 3.1. Zajištění synchronnosti vyžaduje přídavné obvody s dostatečnou frekvenční stabilitou, což situaci většinou komplikuje. Typické je použití fázového závěsu. Počet vzorků odebraných z jedné periody signálu je potom celé číslo, a je určeno násobícím poměrem obvodu pro násobení frekvence. Vzorkovací frekvence je pak celočíselným násobkem základní frekvence signálu.



Obrázek 3.1: Synchronní vzorkování

Obecná měřená veličina X z funkce $f(x_i, y_i)$, kde jednotlivé vzorky jsou měřeny pomocí číslicové integrace, je sumacemi vyjádřena vztahem

$$X = \sum_{i=1}^{N} f(x_i, y_i) \tag{3.1}$$

[6] Synchronní vzorkování, někdy také nazývané *koherentní*.

3.2 Asynchronní vzorkování

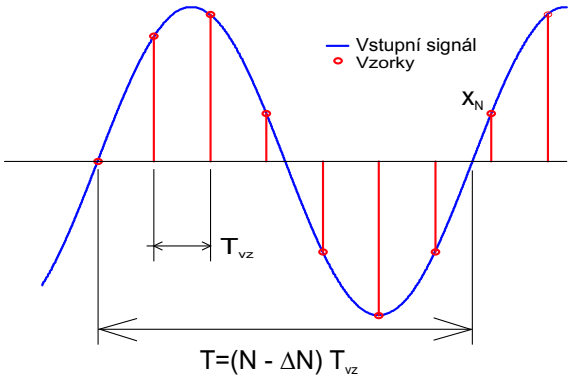
Při asynchronním vzorkování není počet vzorků odebraných z periody měřeného signálu celým číslem, ale platí, že $NT_{vz} \neq T$, což lze dále upravit zavedením korekčního doplňku ΔN. Periodu lze potom vyjádřit jako

$$(N - \Delta N) \cdot T_{vz} = T \tag{3.2}$$

kde N je poměr T/T_{vz} zaokrouhlené na celé číslo nahoru, T_{vz} vzorkovací frekvence a ΔN je korekce počtu vzorků na skutečnou periodu. Synchronní vzorkování je speciální případ asynchronního, kdy platí $\Delta N = 0$. Při výpočtu obecné veličiny X funkce $f(x_i, y_i)$ přejde rovnice (3.1) do tvaru

$$X = \sum_{i=1}^{N} f(x_i, y_i) - \Delta N f(x_N, y_N) \tag{3.3}$$

kde $\Delta N f(x_N, y_N)$ je korekční faktor [2].



Obrázek 3.2: Asynchronní vzorkování

V rámci technického řešení jednotky sběru dat je použito právě asynchronního vzorkování z důvodů jeho jednoduchosti. Při dostatečném počtu vzorků se snižuje vliv necelého počtu vzorků na periodu.

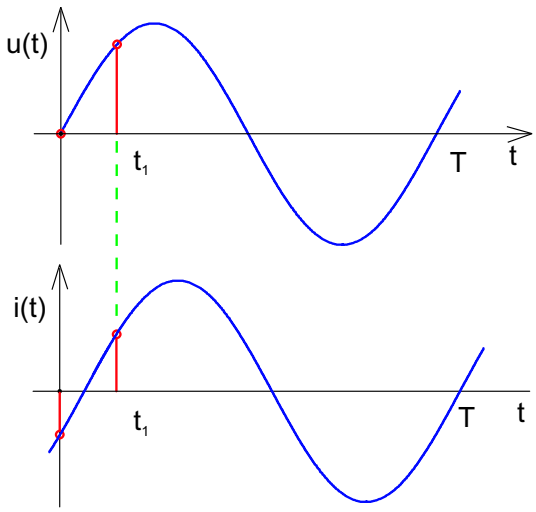
3.3 Simultánní vzorkování

Podstatou metody simultánního vzorkování je vzorkování všech měřených kanálů ve stejném časovém okamžiku, jak je to naznačeno na obr. 3.3 a). Prakticky toho lze dosáhnout několika způsoby, použitím více převodníků taktovaných z jediného zdroje vzorkovacího kmitočtu, nebo více paralelně řízených vzorkovačů pro každý kanál a analogové paměti (kapacitor s kvalitním dielektrikem).

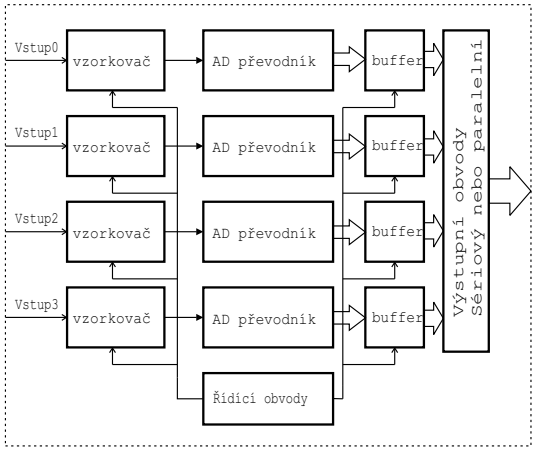
V anglické literatuře jsou převodníky pro simultánní vzorkování označovány *dual* pro 2 kanály (např. ADS8361), *quad* pro 4 kanály (např. AD9229) nebo *hex* šestikanálový AD73360. Možná struktura vnitřního zapojení převodníku pro simultánní vzorkování je naznačena na obr. 3.3 b). Z podstaty činnosti je zřejmé, že nemohou být levné, neboť se jedná o více stejných součástek v jednom pouzdře.

3.4 Sekvenční vzorkování

Při sekvenčním vzorkování dochází v měřených kanálech k odběru vzorků v různých okamžicích, časově posunutých vůči referenčnímu (prvnímu) kanálu, viz obr. 3.4 a). Vzorky jsou odebírány po sobě (sekvenčně), což vnáší do výpočtu chybu. Pokud je vzorek v referenčním



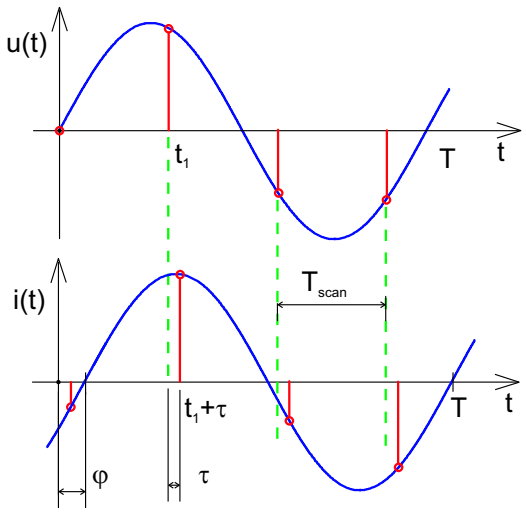
(a) Odběry vzorků ve stejnou dobu



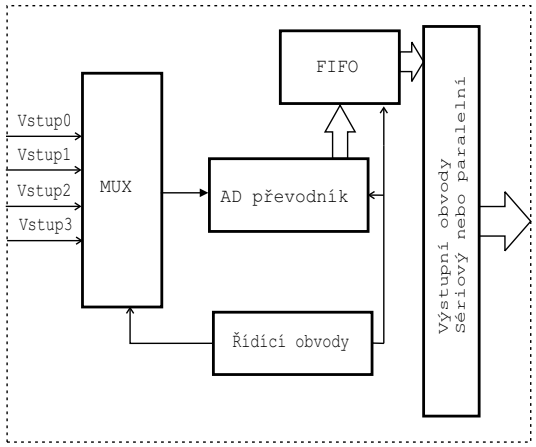
(b) Struktura A/D převodníku pro simultánní vzorkování

Obrázek 3.3: Simultánní vzorkování

kanále odebrán v čase t, pak vzorek v následujícím kanále v čase $t + \tau$, v dalším kanále v okamžiku $t + 2\tau$ atd.



(a) Okamžik vzorkování



(b) Struktura A/D převodníku pro sekvenční vzorkování

Obrázek 3.4: Sekvenční vzorkování

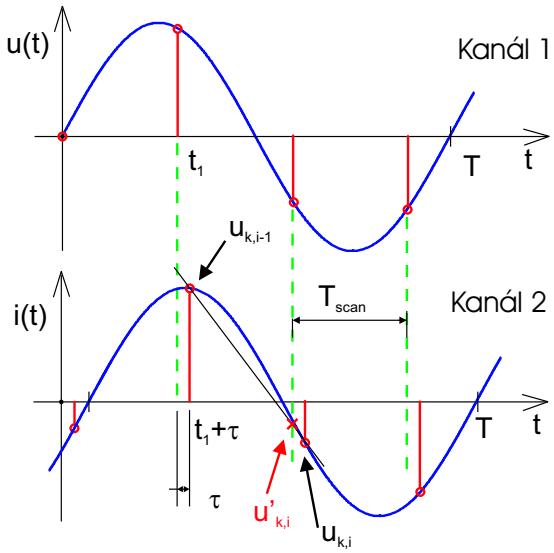
V systémech, kde je přepínání kanálů řízeno obslužným programem, je doba zpoždění τ dána prodlevou programu od dokončení předchozího převodu, přepnutí měřeného kanálu (u sériových převodníků dáno rychlostí komunikace a počtem přenesených bitů) a samozřejmě vlastní dobou převodu převodníku. Tato doba přepnutí nemusí být konstantní. Tím se vnáší nejistota odběru vzorku (jitter) v dalších kanálech ve formě jitteru vzorkovací frekvence. Pokud převodník umožňuje funkci tzv. *autochannel sweep* (např. TLC2578) nebo *sequencer* (např. AD7927) je τ dáno pouze dobou převodu (*conversion time*). Tím je zajištěno, že jitter doby τ je minimální.

Částečnou korekci lze provést lineární interpolací a dopočítat hodnotu měřeného průběhu v okamžiku vzorkování v referenčním kanále ze znalosti předchozího vzorku v měřeném

kanále, jak je naznačeno na obr. 3.5.

$$u'_{k,i} = u_{k,i} - (u_{k,i} - u_{k,i-1}) \frac{\tau}{T_{scan}} (k-1) \tag{3.4}$$

kde $u'_{k,i}$ je dopočítaná hodnota pro okamžik vzorkování v referenčním kanálu, k pořadové číslo kanálu vzhledem k referenčnímu, pro který $k = 1$, T_{scan} perioda scanovací frekvence.



Obrázek 3.5: Detail sekvenčního vzorkování

Kapitola 4

Popis číslicových metod výpočtu parametrů signálů a generování průběhů

Poté, co již víme, jaké metody získání dat (vzorkování) lze použít, mohou být popsány algoritmy výpočtu jednotlivých parametrů signálů a metoda generování průběhů přímou číslicovou syntézou DDS. Výpočty vychází z definičních integrálů, které jsou definovány v časové oblasti. V číslicové oblasti je výpočet integrálů aproximován sumačními vzorci. Tak lze v číslicové oblasti numerickými metodami spočítat potřebné parametry: střední hodnotu, efektivní hodnotu, periodu, fázový posun a činný i jalový výkon. Frekvenční analýza může být provedena algoritmem FFT.

4.1 Měření periody

Při výpočtu periody je každý vzorek časové posloupnosti porovnáván s referenční úrovní. Je-li při měření použit analogově-číslicový převodník s bipolárním rozsahem, je možné za referenční úroveň považovat nulovou hodnotu. Pro unipolární rozsah převodníku je za takovou hodnotu vhodné považovat polovinu vstupního rozsahu A/D převodníku.

Přesnost určení periody je klíčová pro určení parametrů signálu v časové oblasti, neboť se vyskytuje ve všech vzorcích (počet vzorků N vyjadřuje délku trvání periody). Pokud není vstupní signál zatížen šumem, má harmonický signál pouze dva průchody referenční úrovní za periodu. V opačném případě je vhodné signál před zpracováním filtrovat, například klouzavým průměrem z p vzorků. Pokud je pro hledání periody použita jako referenční úroveň nulová hodnota, stačí sledovat změnu znaménka vzorků. Je-li použito synchronní vzorkování a počet vzorků je N, pak při znalosti vzorkovací periody T_{vz} je perioda signálu určena jako $T = N \cdot T_{vz}$.

Pro zarušené nebo částečně zkreslené signály však takto určené průchody nulovou úrovní neodpovídají průchodům touto úrovní jejich základních harmonických. Určení skutečné periody pak může být zpřesnit lineární interpolace nebo regrese [8]. *Lineární interpolace* aproximuje průběh mezi dvěma body přímkou. Pro dva časové okamžiky a příslušné hodnoty $[t_1; x_1]$ a $[t_2; x_2]$ platí

$$t_0 = \frac{x_2 t_1 + x_1 t_2}{x_2 + x_1} \tag{4.1}$$

Pro bipolární rozsah převodníku je x_1 záporné. Pro další zpřesnění okamžiku průchodu referenční úrovní je možné proložit posloupnost vzorků *regresní přímkou* z k vzorků před očekávaným průchodem signálu referenční úrovní a k po něm tak, aby součet kvadrátů odchylek jednotlivých vzorků od této přímky byl minimální. Pro okamžik průchodu t_0 pak platí

$$t_0 = \bar{t} + (x_{ref} - \overline{x}\,) \cdot \frac{\sum_{i=1}^{m}(t_i - \bar{t})^2}{\sum_{i=1}^{m}(t_i - \bar{t})x_i} \tag{4.2}$$

kde m je počet bodů, ze kterých se regresní přímka počítá, x_{ref} je referenční úroveň (pokud

není rovna nule) a pro $\bar{t}$ a $\overline{x}$ platí

$$\bar{t} = \frac{1}{m}\sum_{i=1}^{m} t_i,\ \overline{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

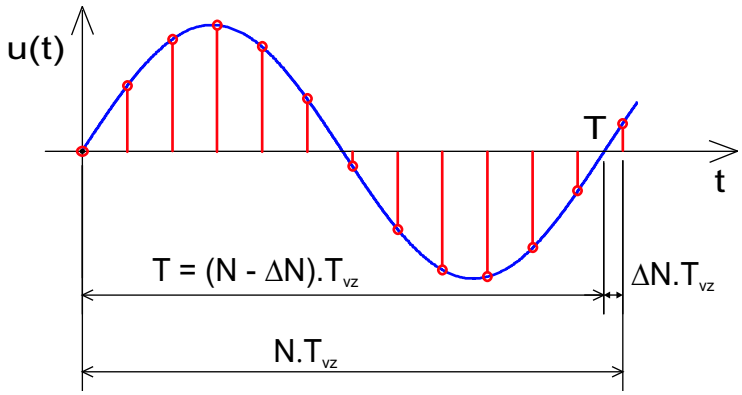
Perioda signálu je vyjádřena výpočtem dvou časů průchodů ref. úrovní t_{01} a t_{02} se stejným znaménkem derivace a jejich odečtením

$$T = t_{02} - t_{01} \tag{4.3}$$

Přídavný signál s efektivní hodnotou X_{RMS}, který způsobuje více průchodů měřeného signálu referenční úrovní je možné odfiltrovat s klouzavým průměrem. Po upřesnění výpočtu regresní přímkou lze vyjádřit chybu určení periody vztahem [1]

$$\Delta T = \frac{6\,X_{RMS}}{\sqrt{mp}} \left(\frac{\mathrm{d}u}{\mathrm{d}t}\right)^{-1} \tag{4.4}$$

kde m je počet bodů, ze kterých se regresní přímka počítá, p je počet bodů pro výpočet klouzavého průměru, X_{RMS} efektivní hodnota rušivého signálu a $\mathrm{d}u/\mathrm{d}t$ strmost měřeného signálu v oblasti uvažovaného průchodu referenční úrovní.



Obrázek 4.1: Měření periody signálu

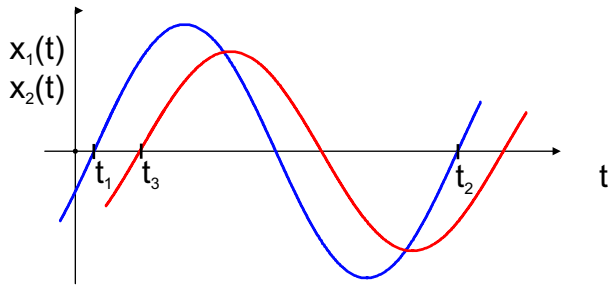
V případě velmi zarušených signálů lze periodu určit pomocí autokorelace nebo metodou spektrální analýzy (kap. 4.6).

4.2 Měření fázového rozdílu

Fázový rozdíl dvou signálů je důležitým parametrem při měření výkonu na obecné zátěži. V praxi je nutné vzít v úvahu zkreslení obou signálů rušivým napětím. K rušivému vlivu tohoto napětí je nutné přihlédnout při výběru vhodné metody měření fázového rozdílu.

Nejjednodušší metodou je *přímé měření časového intervalu* mezi průchody měřených signálů referenční úrovní. pro referenční průběh najdeme časy t_1 a t_2, pro druhý průběh najdeme průchod referenční úrovní se stejným znaménkem derivace t_3 jako v případě prvního průběhu, viz obr. 4.2. Pak pro fázový rozdíl platí

$$\varphi = 2\pi\,\frac{t_3 - t_1}{t_2 - t_1} \tag{4.5}$$



Obrázek 4.2: Měření fázového rozdílu signálů

Další metodou je úprava předchozí metody přímého měření. Výpočty jednotlivých časů t_1, t_2 a t_3 lze zpřesnit lineární interpolací (4.1) nebo regresní přímkou (4.2) a fázový rozdíl vypočteme opět dle (4.5).

Pro značně zarušené průběhy je možné použít časově náročnější metodu výpočtu ve frekvenční oblasti a využít Fourierovy analýzy. Fázi základní harmonické vzorkovaného signálu x_1 lze vyjádřit vztahem

$$\varphi_{x_1} = \text{arctg}(\frac{b_{1,x_1}}{a_{1,x_1}}) \tag{4.6}$$

kde b_{1,x_1} a a_{1,x_1} jsou komplexní koeficienty základní harmonické složky získané Fourierovou transformací (kap. 4.6) vstupního signálu, odpovídající základní harmonické složce vstupního signálu. Stejnou metodou určíme tyto koeficienty pro druhý signál x_2. Fázový rozdíl mezi x_1 a x_2 je pak dán vztahem

$$\varphi = \varphi_{x_2} - \varphi_{x_1} \tag{4.7}$$

4.3 Měření střední hodnoty

Výpočet střední hodnoty signálu je ve spojité oblasti definován vztahem

$$U_{DC} = \frac{1}{T}\int_T u(t)\mathrm{d}t \tag{4.8}$$

kde T je perioda spojitého signálu $u(t)$. Při výpočtu je integrál nahrazen sumou N vzorků na periodu a rovnice (4.8) přejde do tvaru

$$U_{DC}^{cisl} = \frac{1}{N}\sum_{i=0}^{N-1} u(t_i) \tag{4.9}$$

což je číslicová podoba vztahu (4.8).

4.4 Měření efektivní hodnoty

Efektivní hodnota je základní parametr harmonických signálů, který je velmi často třeba měřit. Matematická definice efektivní hodnoty je

$$U = \sqrt{\frac{1}{T}\int_T u^2(t)\mathrm{d}t} \tag{4.10}$$

kde T je perioda spojitého signálu $u(t)$. Při výpočtu je integrál (4.10) nahrazen sumou N vzorků.

Konkrétní algoritmy pro výpočet efektivní hodnoty rozlišují, je-li použito synchronní nebo asynchronní vzorkování.

4.4.1 Efektivní hodnota při synchronním vzorkování

Při synchronním vzorkování (kap. 3.2) připadá celistvý počet vzorků na jednu periodu vstupního signálu. Rovnice (4.10) pak přejde do tvaru

$$U_{RMS}^{sch} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} u^2(t_i)} \tag{4.11}$$

kde U_{RMS}^{sch} je aproximace výpočtu efektivní hodnoty schodovitou funkcí. Je-li pro aproximaci použita lineární interpolace (sousední vzorky spojeny úsečkami), platí pro výpočet vztah

$$U_{RMS}^{lin} = \sqrt{\frac{1}{3N}\sum_{i=0}^{N-1}\Big[u^2(t_i) + u(t_i)u(t_{i+1}) + u^2(t_{i+1})\Big]} \tag{4.12}$$

Pro méně než zhruba 500 vzorků je schodovitá aproximace výpočtu integrálu přesnější než lineární, jejíž výpočet je navíc časově náročnější [3].

Pokud je nejvyšší harmonická složka obsažená ve spektru vstupního signálu řádu s, potom lze pro harmonický průběh a celistvý počet vzorků $N > s$ dokázat, že chyba určení efektivní hodnoty (stejně tak i výkonu) je rovna nule. Pokud tato podmínka není splněna, dochází k chybě, jejíž velikost je určena vztahem [9]

$$\delta U = \frac{2,5}{N^2} + \frac{0,25}{N^3} \tag{4.13}$$

Poznámka: V systému řízeném mikroprocesorem lze synchronní vzorkování zajistit přerušením od interního časovače procesoru, nebo je-li procesor vybaven, záchytnými obvody CAP/COM (capture-compare obvody)[7]. V přerušení je spouštěn odběr vzorku A/D převodníkem externím signálem převodníku.

4.4.2 Efektivní hodnota při asynchronním vzorkování

Při asynchronním vzorkování (kap. 3.1) není počet vzorků na periodu celistvé číslo, a proto je třeba provést korekci odečtením příspěvku od vzorku u_N, který je již mimo měřenou periodu.

$$U_{RMS} = \sqrt{\frac{1}{N - \Delta N}\left(\sum_{i=1}^{N} u_i^2 - \Delta N u_N^2\right)} \tag{4.14}$$

kde $0 < \Delta N < 1$ je korekční člen doby periody definovaný v kapitole 3.2.

Při výpočtu U_{RMS} dochází k chybě způsobené vlivem ΔT, jejíž velikost je

$$\delta_{RMS} \doteq \frac{\Delta T}{2T}\left[1 - \left(\frac{u_N}{U_{RMS}}\right)^2\right] < \frac{\Delta T}{2T} \tag{4.15}$$

[7] Jedná se o periferní obvody procesoru. Obsah COM registrů je neustále porovnáván s aktuální hodnotou čítače a v případě jejich rovnosti se generuje událost, např. přerušení. Naopak, změna úrovně vstupního signálu může zachytit aktuální hodnotu čítače v CAP registrech.

4.5 Měření výkonu

Výkon je další velmi důležitý parametr signálů. V praxi je jeho výpočet velmi často požadován po měřicích jednotkách. Definiční integrál pro výpočet činného výkonu ve spojité časové oblasti je

$$P = \frac{1}{T}\int_{T} u(t)\, i(t)\, \mathrm{d}t \tag{4.16}$$

V případě sekvenčního vzorkování je nutné provést korekci podle vztahu (3.4). Potom je definiční integrál nahrazen sumou vzorků. Pokud je vzorkováno synchronně s N vzorky na periodu, je výkon vypočten podle vztahu

$$P = \frac{1}{N}\sum_{i=0}^{N-1} u_i\, i_i \tag{4.17}$$

Jak již bylo naznačeno v kap. 4.4.1 je chyba výpočtu výkonu nulová, pokud je počet vzorků (synchronní vzorkování) na periodu větší, než řád nejvyšší harmonické složky, obsažené ve spektru vstupního signálu.

Pokud byly vzorky získány asynchronně, je vztah (4.17) korigován a platí vztah

$$P = \frac{1}{N - \Delta N}\left(\sum_{i=1}^{N} u_i\, i_i - \Delta N u_N\, i_N\right) \tag{4.18}$$

a chyba výpočtu výkonu je ovlivněna chybou určení periody

$$\delta_P \doteq \frac{\Delta T}{T}\left[1 - \frac{u_N\, i_N}{P}\right] < \frac{\Delta T}{T} \tag{4.19}$$

Pro činný a jalový výkon platí $P = UI\cos\varphi$, $Q = UI\sin\varphi$, kde $P = UI$ je zdánlivý výkon jako součin efektivních hodnot napětí a proudu a $\cos\varphi$ tzv. *účiník*. Jalový výkon (nekoná žádnou práci) lze určit výpočtem

$$Q = \sqrt{(U_{RMS}\, I_{RMS})^2 - P^2} \tag{4.20}$$

4.5.1 Příklad měření výkonu

Pro jednoduchost je uvažována jednofázová síť, kde jsou měřeny 2 kanály, napětí a proud, obecně s fázovým posunutím průběhů φ daném impedancí připojené zátěže. Ze vzorků odebraných v obou kanálech je počítán okamžitý výkon dle vztahu

$$p(t) = u(t)i(t) \tag{4.21}$$

Pro případ vícefázové soustavy, na které je prováděno simultánní vzorkování nezáleží na pořadí připojení jednotlivých dvojic napětí-proud na vstupní svorky a jednotlivé kanály A/D převodníku. V případě sekvenčního vzorkování může chyba způsobená korekcí dle vztahu (3.4) způsobit nepřesnosti při měření výkonu. Proto je důležitá správná volba připojení dvojic napětí-proud zejména při nízkých vzorkovacích kmitočtech.

Výpočet výkonu při simultánním vzorkování

Navzorkované průběhy odpovídají harmonickému napětí $u(t)$ a proudu $i(t)$, obecně s fázovým posunem φ.

$$u(t) = U_m \sin \omega t \tag{4.22}$$
$$i(t) = I_m \sin(\omega t + \varphi) \tag{4.23}$$

Při simultánním vzorkování je okamžik odběru vzorků v obou kanálech ve stejném okamžiku t_1, viz obr. 3.3. Okamžitý výkon $p(t)$ v okamžiku t_1 bude $p(t_1) = u(t_1)i(t_1)$ a jeho hodnota bude

$$p(t_1) = u(t_1)i(t_1) = U_m I_m \sin \omega t_1 \sin(\omega t_1 + \varphi) = \frac{U_m I_m}{2} \Big(\cos \varphi - \cos(2\omega t_1 + \varphi)\Big)$$

jehož přepsáním dostaneme

$$p(t_1) = \frac{U_m I_m}{2} \Big[(1 - \cos 2\omega t_1)\, \cos \varphi + \sin 2\omega t_1 \sin \varphi\Big] \tag{4.24}$$

Výpočet výkonu při sekvenčním vzorkování

Pokud je vzorek napětí odebrán v okamžiku t_1 a vzorek napětí odpovídajícího proudu v okamžiku $t_1+\tau$, kde τ je zpoždění, je hodnota okamžitého výkonu $p_1 = u(t_1)i(t_1+\tau)$. Po dosazení do vzorce pro výpočet okamžitého výkonu (4.21) a po úpravě pomocí goniometrických vztahů postupně dostáváme

$$p\,(t_1) = u(t_1)\, i(t_1 + \tau) = U_m I_m \sin \omega t_1 \sin\Big(\omega(t_1 + \tau) + \varphi\Big)$$

V praxi je zpoždění způsobeno konečnou dobou přepnutí multiplexeru na vstupu vícekanálového převodníku.

$$p\,(t_1) = \frac{U_m I_m}{2} \Big[(1 - \cos 2\omega t_1)\, \cos(\omega\tau + \varphi) + \sin \omega t_1 \sin(\omega\tau + \varphi)\Big] \tag{4.25}$$

Ve vzorci (4.25) je zřejmá podobnost se vztahem (4.24). Časové zpoždění přepnutí kanálů τ mezi odměry obou vzorků se projevuje jako aditivní fázový rozdíl vzorkovaných průběhů o velikosti $\Omega\tau$.

4.6 Výpočet periody a efektivní hodnoty ve frekvenční oblasti

Až doposud byly zmíněny pouze metody výpočtu v časové oblasti. Periodu a efektivní hodnotu periodického signálu je možné vypočítat i ve frekvenční oblasti [6]. Základem těchto metod je rychlé a přesné určení spektra signálu. Pro výpočet spektra se dnes takřka výhradně používá algoritmu FFT (rychlá Fourierova transformace). Pokud je průběh měřeného napětí periodický, lze vyjádřit efektivní hodnotu součtem efektivních hodnot jednotlivých harmonických složek podle vztahu

$$U_{RMS} = \sqrt{\sum_{k=0}^{N} U_{kRMS}^2} \tag{4.26}$$

kde U_{kRMS} jsou efektivní hodnoty k-té harmonické. Vyšší složky od řádu N jsou zanedbatelné.

Ve vzorci (4.26) nevystupuje podmínka synchronního vzorkování. Tato podmínka je skryta v metodě výpočtu FFT, kde je většinou algoritmů požadováno $2^m = N$, m je celé číslo. Chyba,

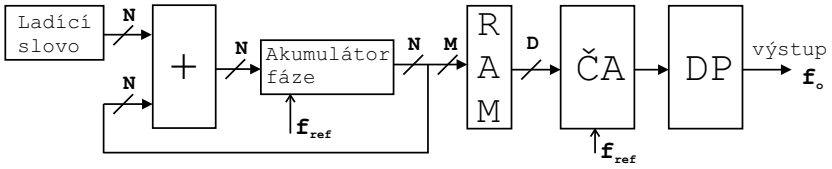
která při nedodržení této podmínky vzniká, se nazývá „*leakage*“. Existují metody, které se snaží potlačit vliv nesplnění podmínky. (např. použití oken nebo převzorkování signálu [5]).

4.7 Generace průběhů pomocí DDS

Často je třeba generovat průběhy o definovaném kmitočtu. K tomu lze v číslicových systémech použít metodu *přímé číslicové syntézy* (DDS). Tak lze generovat signály o požadované frekvenci, popřípadě i frekvenčně a fázově modulovaného signálu ze zdroje konstantního a stabilního referenčního kmitočtu. Přímá číslicová syntéza má následující výhody:

- Ladění výstupní frekvence s mikrohertzovým rozlišením s číslicovým řízením.
- Extrémně rychlé skokové změny frekvence bez překmitů kolem požadované hodnoty.
- U DDS nejsou potíže s teplotním driftem a s tím spojené nutné dolaďování jako u analogových systémů.

Vzorky jedné periody generovaného průběhu jsou předem vypočítány a uloženy do paměti. Pro adresaci paměti je použito horních M bitů z N bitového „*akumulátoru fáze*“, jak je naznačeno v blokovém schématu na obr. 4.3, kde blok *ČA* značí číslicově-analogový převodník a *DP* dolní propust.



Obrázek 4.3: Blokové schéma generátoru signálu na principu DDS

Pro výstupní frekvenci f_o platí vztah

$$f_o = M\, f_{ref}/2^N \tag{4.27}$$

kde f_{ref} je referenční zdroj frekvence, M počet adresních bitů paměti a N počet bitů fázového akumulátoru. Výstupem fázového akumulátoru je lineární průběh, proto je třeba použít tabulku pro převod fáze signálu na amplitudu. Následuje číslicově-analogový převodník a výstupní filtr.

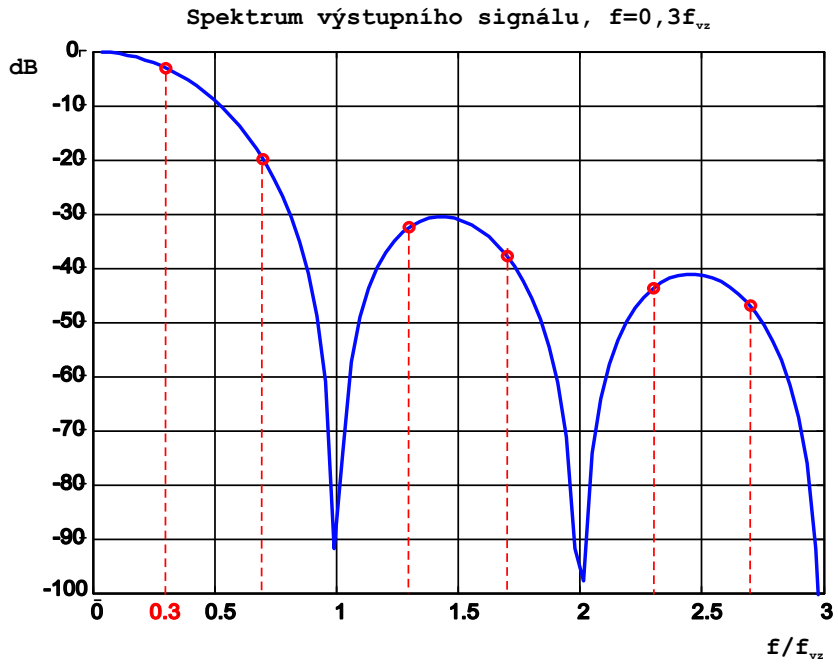
Při generaci průběhu o frekvenci f_o vzniká na výstupu teoreticky nekonečný počet spektrálních čar s amplitudami klesajícími podle funkce $\sin(\pi f_o/f_{ref})/(\pi f_o/f_{ref})$ na frekvencích $f_{ref} - f_o$, $f_{ref} + f_o$, $2f_{ref} - f_o$, $2f_{ref} + f_o$, $3f_{ref} - f_o$, atd. Tento efekt lze částečně odstranit číslicovou filtrací funkcí inverzní k $(\sin x/x)$. Pro odstranění nežádoucí části výstupního spektra se používají analogové dolní propusti. Výstupní spektrum DDS generátoru pro výstupní frekvenci $0,3f_{ref}$ je naznačeno na obr. 4.4

Minimální generovatelná frekvence pro N-bitový akumulátor je $f_{o,min} = f_{ref}/2^N$ (např. pro $N = 32$ platí $f_{o,min} = f_{ref}/2^{32} = 0,23 \cdot 10^{-9} f_{ref}$). Maximální hodnota obsahu střadače je 2^M a odpovídá jedné periodě nosného signálu. Protože úhlové rozlišení je velké $(2\pi/2^M)$, je možné generovat extrémně pomalé signály.

Jednou z možných implementací je i řešení pomocí procesoru. Pak je fázový akumulátor realizován programově a zdrojem přesného kmitočtu f_{ref} je hodinový krystal, jehož kmitočet je dělen např. pomocí přerušení.

4.8 Výpočet frekvenčního spektra signálu

Při analýze signálů je možné na signály nahlížet v časové oblasti (angl. *time domain)* a nebo ve frekvenční oblasti (*frequency domain)*. Jakýkoliv signál může být plně popsán v obou oblastech. Každá z těchto oblastí je vhodná pro posuzování jiných parametrů, časová oblast je



Obrázek 4.4: Spektrum výstupního signálu generovaného metodou DDS

vhodnější pro hodnocení např. amplitudy, frekvence a činitelů tvaru signálu, zatímco frekvenční analýza je vhodná pro spektrální analýzu signálu, výpočet korelačních funkcí signálů, při měření spektrální čistoty rozvodné sítě nebo v diagnostice. *Fourierova transformace* vyjadřuje vzájemnou souvislost signálu a jeho spektra. Mění časovou závislost amplitudy na frekvenční závislost. Matematicky vyjadřuje fakt, že každý periodický průběh (splňující určité podmínky) lze nahradit součtem nekonečného počtu harmonických průběhů o frekvencích rovných celistvým násobkům frekvence původního průběhu. V praxi je počet harmonických složek vždy omezen. Fourierova řada je matematicky vyjádřena vztahem

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{n} (b_k \sin k\omega t + a_k \cos k\omega t)$$

kde n je teoreticky ∞, ale v praxi je vždy konečné. Jednotlivé koeficienty a_k a b_k jsou vyjádřeny vztahy

$$a_k = \frac{2}{T} \int_0^T f(t) \cos k\omega t dt, \qquad b_k = \frac{2}{T} \int_0^T f(t) \sin k\omega t dt$$

Předchozí vzorce platí pro spojitý signál. V případě diskrétního signálu používáme tzv. *Diskrétní Fourierovu transformaci*, DFT. Pro hodnoty jednotlivých spektrálních čar $X(k)$ platí

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \tag{4.28}$$

kde $0 \leq k \leq N-1$, $x(n)$ jsou vzorky vstupního signálu a $W_N = e^{-j\frac{2\pi}{N}}$ je otáčecí činitel (*twiddle factor*).

Výpočet DFT dle vztahu 4.28 je časově velmi náročný, v dnešní době je téměř vždy použit k výpočtu algoritmus FFT (rychlá Fourierova transformace). Urychlení spočívá v rozdělení vstupní posloupnosti na dílčí posloupnosti. V praxi jsou nejvíce používané algoritmy Radix-2

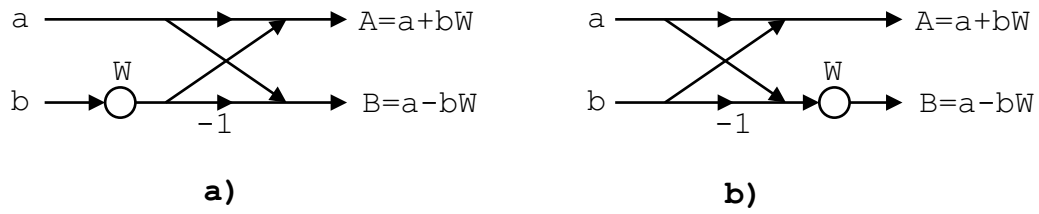
nebo Radix-4, podle počtu částí, do kterých jsou rozdělena vstupní data. Existují samozřejmě i další algoritmy [12]. Pokud je násobení otáčecím faktorem provedeno na straně časové oblasti výpočtu, jedná se o algoritmus s decimací v čase DIT (decimation-in-time)

$$X(k) = \sum_{m=0}^{N/2-1} x(2m) W_N^{2mk} + \sum_{m=0}^{N/2-1} x(2m+1) W_N^{(2m+1)k} \tag{4.29}$$

Pokud je násobení provedeno až na výstupní straně algoritmu, jedná se o algoritmus s decimací ve frekvenci DIF (decimation-in-frequency)

$$X(k) = \sum_{m=0}^{N/2-1} x(m) W_N^{mk} + \sum_{m=N/2}^{N} x(m) W_N^{mk} \tag{4.30}$$

Vhodným přerovnáním jednotlivých částí výpočtu podle (4.29) a (4.30) je získán tzv. motýlek (angl. *butterfly*), zobrazený na obr. 4.5.



Obrázek 4.5: Motýlek algoritmu FFT: a) DIT FFT, b) DIF FFT

Nutná podmínka pro aplikaci těchto algoritmů je celistvý počet vzorků na periodu měřeného signálu $N = 2^m$ nebo $N = 4^m$. Pokud není tato podmínka splněna, dochází k chybě tzv. rozmazáním spektra, angl. *leakage*. Tomu se lze vyhnout, použijeme-li metodu převzorkování signálu (vyžaduje dostatečně výkonný procesor). Vzorky získané asynchronním vzorkováním jsou přepočteny lineární interpolací na „nové“ vzorky tak, aby počet vzorků splňoval danou podmínku. Převzorkování lze provést pouze pro periodické signály. Metoda potlačuje vznik leakage (prosakování) ve spektru [5].

Vzhledem k velkému rozvoji této problematiky byly během posledního desetiletí vyvinuty algoritmy optimalizované nejen pro DSP procesory, ale díky velkému rozvoji dalších řad procesorů a jejich výkonu i pro další architektury. S tím je spojená otázka optimální algoritmizace výpočtu FFT pro danou architekturu.

Kapitola 5

Vlivy působící na přesnost numerických výpočtů

Na přesnost měření mají zásadní vliv vzorkovací frekvence, rozlišení A/D převodníku a v případě asynchronního vzorkování i chyba určení periody. Do vlastního procesu výpočtu pak vstupuje kvantizační šum a nelinearity A/D převodníku. V této kapitole jsou shrnuty některé z těchto vlivů. Jednotlivá data jsou získaná simulací v MATLABu 6.5. Zdrojové kódy jsou přiloženy na doprovodném CD ROM.

5.1 Vliv nedostatečného počtu vzorků na výpočet efektivní hodnoty při synchronním vzorkování

Aby byla chyba výpočtu efektivní hodnoty a výkonu při synchronním vzorkování nulová, je nutné, aby *počet vzorků* odebraných za jednu periodu vstupního signálu byl *vyšší*, než je *řád nejvyšší harmonické složky* ve spektru vstupního signálu. Je-li počet vzorků menší, dochází při výpočtu k chybě δ_{RMS} podle vztahu (4.13) vyčíslené v tab. 5.1. Velikost této chyby je závislá na tvaru vstupního signálu [9].

Počet vzorků N	3	4	5	6	8	10	16	32	64
$\delta_{RMS} \sim 1/n^2$ [%]	29	16	10	7	4	2,5	1	0,25	0,06

Tabulka 5.1: Relativní chyba U_{RMS} ovlivněna počtem vzorků podle (4.13)

Vysvětlení: Pokud je v tabulce uvedeno např. $N = 5$, znamená to, že maximální spektrální složka vstupního signálu je minimálně 6. řádu a dochází tedy k chybě.

5.2 Vliv necelistvého počtu vzorků při výpočtu efektivní hodnoty

K posouzení vlivu necelistvého počtu vzorků na periodu získaných při asynchronní vzorkování na výpočet efektivní hodnoty byla provedena simulace. Výpočet efektivní hodnoty vychází ze vztahu (4.14), který koriguje vliv vzorku, který již do periody nepatří[8].

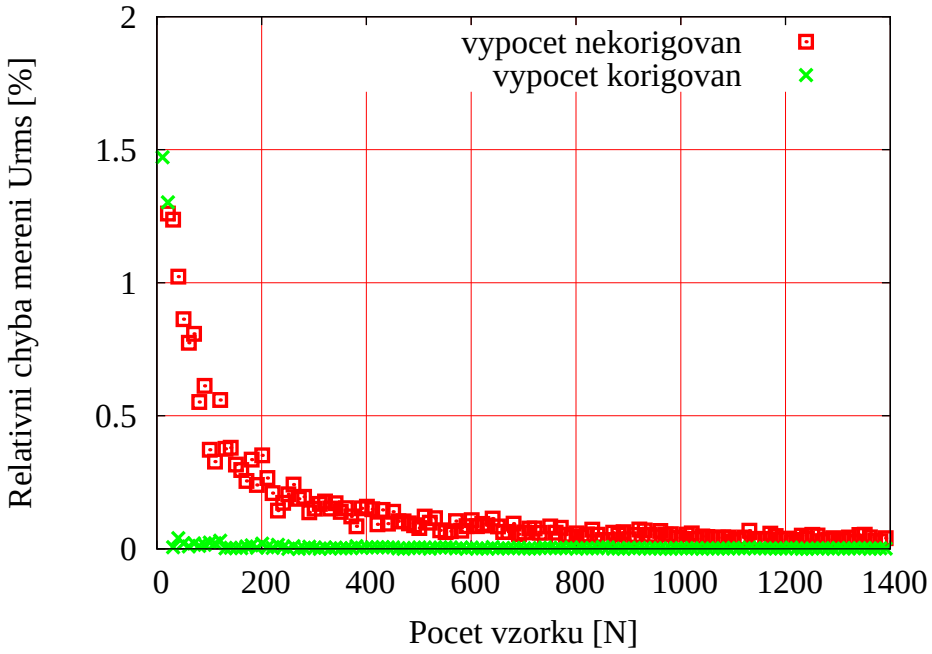
Pozice odběru vzorků v periodě je náhodná veličina. Efektivní hodnota vypočtená z těchto vzorků proto byla získána průměrováním z několika realizací, řádově z jedné desítky. Simulace byla provedena pro různé vzorkovací frekvence tak, aby platilo, že $\Delta N \neq 0$. V grafech jsou vyneseny průběhy bez korekce vypočtené podle vztahu (4.11) i s korekcí dle vztahu (4.14), aby bylo možné posoudit její vliv. V grafu na obr. 5.1 je průběh velikosti chyby při průměrování z 10 realizací. Velikost a průběh chyby není podstatně závislý na spektrálním složení signálu, tj. na amplitudách harmonických složek.

Skutečnosti zjištěné z grafu 5.1 je možné interpretovat takto:

- není-li prováděna korekce vlivu asynchronního vzorkování, je výsledná chyba určení efektivní hodnoty menší než 1 % až po odebrání více než 50 vzorků z jedné periody,

[8] Zdrojový kód simulace je v souboru sim_asynchro.m

- pokud byla provedena korekce podle rovnice (4.14), je chyba pro stejný počet vzorků menší než 0,1 %.
- V každém případě je z grafu vidět, že závislost chyby je řádově $\delta U_{RMS} \sim \frac{1}{N}$ až $\frac{1}{N^2}$.



Obrázek 5.1: Vliv asynchronního vzorkování na chybu určení efektivní hodnoty a korekce tohoto vlivu

5.3 Vznik a vliv kvantizačního šumu

Kvantizační šum vzniká při převodu analogového signálu do číselné formy A/D převodníkem s reálnými parametry. Příčinou kvantizačního šumu je kvantování signálu v úrovni, kdy vlivem konečné rozlišovací schopnosti převodníku a jeho nelinearit dochází k nepřesnému vyjádření číselné hodnoty vzorku. Při výpočtu je uvažováno s hodnotami rozsahu převodníku U_{ROZS} a rozkmitu vstupního signálu U_{pp}. Kvantizační krok převodníku je určen

$$q = U_{ROZS}\ 2^{-N} \tag{5.1}$$

kde U_{ROZS} je plný vstupní rozsah A/D převodníku a N počet bitů převodníku.

Pro modelování skutečného A/D převodníku je použit ideální převodník s nekonečným rozlišením a náhodný bílý šum, který se přičítá k jeho výstupu. Takový šum se označuje *kvantizační*. Odstup signál/šum lze popsat *SNR* činitelem. Kvantizační šum se jeví jako aditivní složka k užitečnému signálu a má rovnoměrnou hustoty rozdělení $p(e)$ v intervalu $< -q/2; q/2 >$. O vstupním signálu nelze obecně říci více, než že je složen z několika harmonických signálů o různých amplitudách, frekvencích a fázových posunech, je pro další zpracování použito statistických metod. Dále je předpokládáno, že vzorky kvantizačního šumu nejsou korelované mezi sebou, ani se vstupním signálem. Pro takové signály je střední hodnota nulová. Mnohem více informací lze získat ze střední kvadratické hodnoty kvantizačního šumu

$$\sigma_e^2 = \int_q e^2 p_e(e)\,\mathrm{d}e = \int_{-q/2}^{q/2} e^2 \frac{1}{q}\,\mathrm{d}e = \frac{1}{q}\frac{1}{3}[e^3]_{-q/2}^{q/2} = \frac{1}{3}\frac{q^3}{4} = \frac{q^2}{12} \tag{5.2}$$

kde $p_e(e)$ je hustota pravděpodobnosti kvantizačního šumu.

Do výsledku (5.2) můžeme dosadit za q z rovnice (5.1) a dostaneme kvadrát směrodatné odchylky, rozptyl šumu.

$$\sigma_e^2 = \frac{q^2}{12} = \frac{(U_{ROZS}\ 2^{-N})^2}{12} = \frac{U_{ROZS}^2\ 2^{-2N}}{12} \tag{5.3}$$

Nyní provedeme vyčíslení SNR pomocí střední kvadratické hodnoty kvantizačního šumu. Činitel SNR je poměr užitečného signálu a šumovému signálu, kterým je v našem případě kvantizační šum. Ve vzorci je užitečný signál reprezentován rozptylem.

$$SNR = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 10\log_{10}\left(\frac{12\ \sigma_x^2}{U_{ROZS}^2\ 2^{-2N}}\right) \tag{5.4}$$

Po vyjádření logaritmu součinu jako součtu logaritmů jednotlivých členů dostaneme tvar vhodnější pro další analýzu.

$$\begin{aligned} SNR &= 10\log_{10} 12 + N \cdot 20\log_{10} 2 - 20\log_{10}\frac{U_{ROZS}}{\sigma_x} \\ &= 10,8 + 6,02N - 20\log_{10}\frac{U_{ROZS}}{\sigma_x} \end{aligned} \tag{5.5}$$

Tento vztah můžeme dále ještě rozepsat, víme-li, že efektivní hodnota σ_x harmonického signálu a jeho rozkmit jsou ve vztahu $\sigma_x = U_{pp}/2\sqrt{2}$. Po dosazení do (5.5) dostaneme

$$SNR = 1,76 + 6,02N - 20\log_{10}\frac{U_{ROZS}}{U_{pp}} \tag{5.6}$$

Čím více se rozkmit vstupního signálu blíží k plnému vstupnímu rozsahu převodníku, tím dochází k menší chybě vlivem kvantovacího šumu. V dalším textu této kapitoly proto předpokládáme, že třetí člen v (5.6) je nulový, tedy $U_{pp} = U_{ROZS}$. Potom platí vzorec

$$SNR = 1,76 + 6,02N \tag{5.7}$$

Máme-li vzorkovat signál s definovanou maximální chybou(způsobenou kvantováním, ostatní chyby zatím neuvažujeme), je třeba tuto chybu vyjádřit pomocí SNR. Relativní chyba je vyjádřena jako $\delta = \Delta/S$, kde Δ je velikost chybového signálu, v našem případě šumu, a S je užitečný signál, neboli „správná hodnota“. Dle definice SNR (5.4) je možné psát

$$\begin{aligned} SNR_\delta &= 20\log_{10}\delta = -20\log_{10}\frac{S}{\Delta} \\ -20\log_{10}\delta &= 1,76 + 6,02N \end{aligned} \tag{5.8}$$

Nyní můžeme snadno najít potřebné rozlišení A/D převodníku pro měření s maximální přípustnou chybou. Např. pro měření s chybou menší než 0,1 % ($\delta = 0,001$) je potřeba

$$N = \frac{-20\ \log\delta - 1,76}{6,02} = \frac{-20\ \log\ 0,001 - 1,76}{6,02} = 9,7 \tag{5.9}$$

Nejbližší vyšší dostupné rozlišení je 10 bitů.

5.4 Vliv nepřesného určení periody

V kapitole 4.1 byl uveden odhad chyby určení periody ΔT při asynchronním vzorkování, která ovlivní výpočty jednotlivých parametrů. Ze vztahu (4.4) vyplývá pro kmitočty 50Hz [2]

$$\Delta T = \frac{6\,X_{RMS}}{\sqrt{mp}} \left(\frac{\mathrm{d}u}{\mathrm{d}t}\right)^{-1} = \frac{6}{\omega\sqrt{2}} \frac{X_{RMS}}{U_{RMS}} \frac{1}{\sqrt{mp}} = 0,0078 \cdot \frac{X_{RMS}}{U_{RMS}} \tag{5.10}$$

kde X_{RMS} je efektivní hodnota rušivého signálu, např. kvantizačního šumu vyjádřeného prostřednictvím SNR. Pro výpočet je uvažováno $m = 3$ (lineární interpolaci), $p = 1$ (žádná filtrace). Hodnota X_{RMS}/U_{RMS} získaná ze vztahu (5.8) jako hodnota δ vyjadřuje velikost povoleného rušení pro dané rozlišení převodníku.

Vzniklá chyba nepřesného určení periody má vliv na přesnosti určení výkonu a efektivní hodnoty s chybami δ_{RMS} a δ_P odhadnutými podle vztahů

$$\delta_{RMS} \doteq \frac{\Delta T}{2T} > \frac{\Delta T}{2T}\left[1 - \left(\frac{u_N}{U_{RMS}}\right)^2\right], \qquad \delta_P \doteq \frac{\Delta T}{T} > \frac{\Delta T}{T}\left[1 - \left(\frac{u_N\, i_N}{P}\right)\right] \tag{5.11}$$

Počet bitů	8	10	12	14	16	24
SNR [dB]	49,92	61,96	74	86,04	98,08	146,24
X_{RMS}/U_{RMS} [%]	0,3	0,08	0,02	0,005	0,002	0,000005
ΔT [μs]	24,88	6,22	1,55	0,39	0,1	380 ps
δ_{RMS} [%]	0,06	0,015	0,004	0,001	2,5 ppm	0,01 ppm
δ_P [%]	0,12	0,03	0,008	0,002	5 ppm	0,02 ppm

Tabulka 5.2: Relativní chyba U_{RMS} a P vlivem ΔT způsobené rozlišením A/D převodníku

Kapitola 6

Popis architektury ARM

Jak vyplynulo z rozboru, jsou procesory ARM vhodné pro jednotku sběru a zpracování dat. Díky svému výpočetnímu výkonu, který dosahuje desítek MIPS, integrovanou pamětí dat a programu, HW-násobičkou a interními A/D a D/A převodníky nabízí tyto procesory komplexní řešení.

Procesory ARM jsou RISC procesory s tzv. Von Neumann architekturou založenou na *load/store* koncepci. To znamená, že data musí být nejprve načtena z paměti do registrů, zpracována a opět uložena zpět do paměti. Procesor využívá pipelining, sdílené registry, různé pracovní módy procesoru a jednotnou délku instrukcí. Technologie ARM je licencovatelná, což přináší záruku přenositelnosti kódu na úrovni kompilátoru a jisté další výhody. Zdrojem informací pro tuto kapitulu byly převážně originální dokumentace výrobců a webové stránky věnujících se dané problematice.

V dalším textu je myšlena architektura ARM obecně při použití pojmu „ARM“. Použití jiného pojmu značí konkrétní rodinu, např. ARM7, StrongARM nebo ARM920. Pro úvod si v několika bodech shrneme základní vlastnosti procesorů s jádrem ARM:

- jednoduchý a výkonný instrukční soubor, který umožňuje použití kompilátorů vyšších programovacích jazyků (jako např. jazyka C).
- Přístup do paměti je možný pouze instrukcemi LDR, STR (tzv. Load-Store Architecture), což výrazně zjednodušuje vlastní výkonnou jednotku procesoru, protože pouze tyto dvě instrukce přistupují do paměti, zatímco ostatní instrukce pracují pouze s registry. Další zvýšení výkonnosti při přístupu do paměti je možné použitím instrukcí vícenásobného přístupu do paměti LDM, STM. Tyto instrukce umožňují rychlé přepínání kontextu. Využívají sekvenční přístup do paměti.
- Instrukční sada má 44 základních instrukcí se stejnou šířkou 32 bitů (výhodné ve srovnání např. s i8086, neboť při jednom přístupu do paměti kódu, tzv. instruction fetch, dojde k vyčtení právě jedné instrukce a nemůže se stát, že se po jejím dekódování zjistí, že pro vykonání instrukce je třeba načíst z paměti další potřebný operand). Více v kap. 6.5.
- Všechny instrukce jsou podmíněně vykonávané, což je výhodné pro efektivní větvení programu (kap. 6.4.5).
- Procesor může pracovat v několika režimech, ve kterých používá částečně překrývající se registry. V každém režimu je programově přístupných pouze 16 z celkem 37 vnitřních registrů, ostatní jsou skryty (kap. 6.4.2). V případě přerušení pak nemusí dojít ke kompletnímu úložení obsahu registrů na zásobník procesoru, ale pouze těch, které jsou opravdu v rutině přerušení použity (kap. 6.4.1).
- Speciální režim procesoru *supervisor mode* je určen pro implementaci operačního systému. Je podobný chráněnému režimu u i8086. Do tohoto módu je procesor uveden při resetu nebo instrukcí `SWI`.
- Pro přístup do paměti lze použít několik adresovacích módů; bezprostřední adresování (*immediate*) s adresou zakódovanou přímo v instrukci, relativní adresování pomocí programového čítače (*relative*) a index-registrové (*indexed*), které využívá hodnoty umístěné v některém z registrů (kap. 6.4.9).
- Paměťově mapované periferie.

- Procesor podporuje dva základní typy přerušení, které se navzájem liší prioritou, latencí obsluhy a přístupnými registry (kap. 6.4.4).
- Některé rodiny současných procesorů jsou vybaveny dvěma instrukčními sadami, jednou základní 32bitovou ARM, a druhou tzv. Thumb sadou s komprimovanými, 16bitovými instrukcemi.
- Podpora debug módu přes JTAG rozhraní.
- Možnost implementace operačního systému.

6.1 Historický vývoj procesorů ARM

Procesory ARM (Acorn RISC Machine) byla vyvinuty britskou firmou Acorn, sídlící v Cambridge, která původně vyráběla osobní počítače. Poprvé byl na trh uveden procesor ARM Archimedes v roce 1988. V roce 1990 došlo k odloučení několika vývojových pracovníků ze společnosti Acorn a společně s firmou Apple a dalšími založili novou společnost se jménem ARM — Advanced RISC Machines, ze které se v roce 1998 stal dnešní ARM Limited (ARM Ltd.).

Pro představu chronologického vývoje procesorů ARM připomeňme několik významných milníků: v roce 1991 byla na trh uvedeno procesorové jádro ARM6, v roce 1996 ARM7, v roce 1997 ARM9TDMI, v r. 2000 ARM10 a v roce 2002 procesory ARM11. Počínaje rokem 1993, většina významných společností začíná licencovat ARM technologii. Prvními společnostmi byly Cirrus Logic a Texas Instruments v roce 1993. V současné době téměř neexistuje žádná významná firma vyrábějící procesory, která by nevyráběla čipy s ARM jádrem (viz kapitola Porovnání ARM procesorů různých výrobců).

Procesory nepodporovaly ze začátku 32bitové adresování, ale jen 26bitové. Od verze ARM6xx jsou procesory plně 32bitové.

Z výše uvedeného je vidět, že ARM není novinka a deriváty procesorů s jádrem ARM jsou na trhu již přes desítku let. Do nedávné doby však nebyly osazeny integrovanou pamětí dat a programu. To neumožňovalo implementaci těchto procesorů i do embedded aplikací, neboť to nevyžadovalo připojení externích pamětí. Současné procesory ARM mají kromě paměti na čipu také velké množství různých periferií. To je výhodné z hlediska minimální velikosti zařízení, nižší ceny a vyšší spolehlivostí celého zařízení.

Doposud nejvíce rozšířené je jádro ARM7TDMI, které bylo hojně používáno v mobilních telefonech. Další velmi známou je řada StrongARM vyvinutá společností DEC (procesory Alpha). Licenci na jejich výrobu později odkoupil Intel [16, 18].

6.2 Použité zkratky ve značení procesorových jader

T ... podpora 16bitových Thumb instrukcí
D ... podpora debug módu
M ... podpora násobení 32 × 32bitových čísel s 64bitovým výsledkem
I ... obsahuje EmbeddedICE logiku, umožňující Embedded System Debugging, využívá JTAG a ETM rozhraní
S ... jádro není uzavřené, ale lze jej dále parametrizovat (synthetisable)
J ... hardwarová podpora aplikací psaných v Javě, technologie Jazelle^{TM}
E ... rozšíření instrukční sady o DSP instrukce.

6.3 Přehled rodin ARM procesorů

Rodiny procesorů ARM se vyznačují některými společnými vlastnostmi, jako jsou podpora různých operačních systémů (včetně Real-Time) zahrnujících Windows CE, Palm OS, Symbian OS a Linux.

Pro procesory ARM existuje řada vývojových prostředků (cygnus, cygwin, Keil, Hitex) a podpora ladění aplikací zahrnující ETM (Embedded Trace Macrocell) interface (Hitex). V současné době (rok 2004) je pro jednoduché embedded aplikace nejrozšířenější jádro ARM7TDMI. Pro náročnější aplikace se používají jádra ARM9, ARM10 a v nedávné době uvedené na trh jádro ARM11 [19].

6.3.1 Rodina ARM7

- Kmitočet jádra je až do 130 MIPS, používá tříúrovňový pipelining.
- Instrukční sada ARMv4T. Strojový kód je dopředně kompatibilní s jádry ARM9, ARM9E, ARM10 a dokonce XScale od firmy Intel.
- Doporučovanou oblastí aplikací jsou tiskárny, pagery, kamery, MP3 přehrávače a PDA systémy.

6.3.2 Rodina ARM9

- Kmitočet jádra je až 300 MIPS, používá pětiúrovňový pipelining s výkonem až 1,1 MIPS/MHz.
- Obsahuje jednotku MMU pro podporu OS, obsahuje instrukční a datovou cache.
- Implementuje harvardskou architekturu s oddělenou datovou a programovou pamětí a sběrnicemi.
- Instrukční sada ARMv5T.
- Doporučovanou oblastí aplikací ARM9 jsou videotelefony, PDA, herní konsole, MP3 přehrávače, MPEG4 přehrávače, nebo informační automobilové systémy.
- Poprvé byla do některých jader (např. ARM926EJ-S) implementována hardwarová podpora výkonu Java instrukcí. Technologie je označována Jazelle^{TM}.
- Jádro ARM9 o 27 % výkonější oproti jádru ARM7 [26].

6.3.3 Rodina ARM10E

- Kmitočet jádra až 400MIPS (1,35 MIPS/MHz), šestiúrovňový pipelining využívá branch prediction.
- Umožňuje osazení koprocesoru pro operace v plovoucí řádové čárce.
- Interně využívá duální 64bitovou AMBA AHB interface se 64bitovou vnitřní sběrnicí.
- Obsahuje jednotku MMU pro podporu OS, dále instrukční a datovou cache, jednotku paralelního zápisu a čtení z paměti.
- Doporučovanou oblastí aplikací řady AMR10E jsou přenosné komunikátory, notebooky, herní konsole, laserové kamery, digitální kamery, jednotky Motor Managementu a automatických převodovek, řízení dodávek elektřiny.

6.3.4 Rodina ARM11

Novinkou jsou v poslední době jádra ARM11.

- Pracují s osmiúrovňovým pipeliningem podporující metody *Branch Prediction* a *Return Stack* známé z DSP. Oddělené pipeliningy pro načítání dat a vlastní zpracování.
- Na čipu osazena cache a DMA jednotka.
- Pracují s instrukční sadou ARMv6 a podporují zdokonalenou instrukční sadu *Thumb2*. Umožňují zpracování dat vhodné pro média technologií SIMD (Single Instruction Multiple Data).
- Doporučovanou oblastí aplikací ARM7 jsou tiskárny, bezdrátové komunikace, síťové prvky, mobilní telefony, PDA, navigační systémy a řídící jednotky.

6.4 Programátorský model procesorů ARM

6.4.1 Módy procesoru

Procesor umožňuje pracovat v několika režimech. Jsou to uživatelský mód (*User*), mód rychlé obsluhy přerušení (*FIQ*), mód obsluhy přerušení (*IRQ*), chráněný mód pro operační systém (*Supervisor*), mód selhání načtení dat nebo instrukce (*Abort*), systémový (*System*), a mód, do kterého se procesor přepne při vykonání nedefinované instrukce (*Undefined*). V režimu *Supervisor mode* je možné nastavovat velikosti zásobníků pro jednotlivé další režimy a globálně povolovat a zakazovat přerušení. Pro běžné programy je použit *User mode*. Mezi jednotlivými módy lze přepínat pouze v *Supervisor* módu pomocí instrukcí `MRS` a `MSR`.

6.4.2 Registry

Procesor ARM obsahuje 37 32bitových registrů, z toho 31 obecných a 6 stavových. Registry nejsou mapované do paměti a jsou rozděleny do částečně překrývajících se bank, každá banka je přístupná pouze v jednom režimu procesoru. Mezi jednotlivými bankami nelze programově přepínat. Každá banka obsahuje 16 registrů, programový čítač a 1 nebo 2 stavové registry. Některé registry mají speciální vyhrazenou funkci, R15 je program counter PC, R14 je link register LR a R13 je Stack Pointer SP. Registry R0 – R7 v každém módu sdílené, při každém použití adresujeme stejný fyzický registr. Registry R8 – R14 nejsou sdílené, v každém módu je adresován jiný fyzický registr stejného významu. Například v režimu User jsou viditelné registry R0 – R15. Dojde-li k přerušení, je procesor v módu přerušení IRQ, registry R13 – R15 budou zaměněny za registry R13_irq – R151_irq. Situace je naznačena na obr. 6.1.

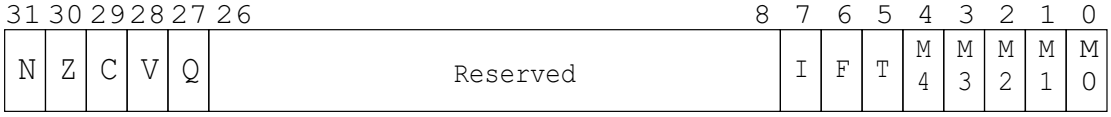
User, System	FIQ	IRQ	Supervisor	Abort	Undefined
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
r8	r8_fiq	r8	r8	r8	r8
r9	r9_fiq	r9	r9	r9	r9
r10	r10_fiq	r10	r10	r10	r10
r11	r11_fiq	r11	r11	r11	r11
r12	r12_fiq	r12	r12	r12	r12
r13 (SP)	r13_fiq	r13_irq	r13_svc	r13_abt	r13_und
r14 (LR)	r14_fiq	r14_irq	r14_svc	r14_abt	r14_und
r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_irq	SPSR_svc	SPSR_abt	SPSR_und

Obrázek 6.1: Přehled registrů procesoru ARM

6.4.3 Stavový registr programu

Stavový registr obsahuje příznakové bity, aktuální mód procesoru a globální povolení přerušení. Procesory ARM jsou vybaveny dvěma stavovými registry, *CPSR–Current Program Status Register*, který je přístupný v každém módu a určuje nastavení procesoru pro daný

mód. Každý přerušovací mód má navíc *SPSR–Saved Program Status Register*, který obsahuje stav procesoru v okamžiku před výskytem výjimky-přerušení.



Obrázek 6.2: Stavový registr procesoru ARM

Příznakové bity jsou nastavovány podle výsledku předchozí instrukce a jsou kódovány do horních čtyřech bitů stavového registru, viz obr. 6.2. V pořadí od MSB to jsou:

- **N** (Negative) ... pokud je výsledek typu *signed int* a výsledek je záporný, pak je příznak nastaven na 1. Jinak je nstaven na 0.
- **Z** (Zero) ... rovno 1 při nulovém výsledku předchozí aritmetické operace, jinak 0.
- **C** (Carry) ... při aritmetických operacích má význam „výpůjčky“ nebo „přetečení“ (unsigned overflow, underflow), při shift instrukcích ji C bit nastaven dle vysunutého bitu z registru operandu.
- **V** (Overflow) ... při aritm. operacích značí přetečení se znaménkem (signed overflow)
- **Q** (DSP Overflow, Saturation) ... je-li tento bit nastaven, pak procesor pracuje v režimu tzv. saturační aritmetiky. To znamená, že pokud se výsledek výpočtu nevejde do registru, nedojde k přetečení, ale registr se naplní maximální možnou hodnotou. Naopak bude-li výsledek příliš malý na uložení do registru, naplní se registr minimální možnou hodnotou. Toto je použito v DSP variantách ARM procesorů, v označení mají „E“.

Příznaky jsou nastavovány všemi aritmetickými instrukcemi. Jejich testování je možné pomocí instrukcí TST, TEQ, CMP a CMN, jak ukazuje příklad na obr. 6.3. Ostatní instrukce kromě aritmetických mohou tyto flagy nastavovat pouze mají-li příponu `-S`, např. `BLS`.

Význam řídících bitů I, F a T:

- **I** IRQ disable bit,
- **F** FIQ disable bit,
- **T** (Thumb) určuje, zda je procesor v THUMB módu.

Spodních 5 LSB CPSR[4:0] určují aktuální mód procesoru.

- 10000 - User Mode
- 10001 - FIQ mode
- 10010 - IRQ mode
- 10011 - Supervisor mode
- 10111 - Abort mode
- 11011 - Undefined instruction mode
- 11111 - Systém mode

```
zacatek:
    ldr r4, =0x0003     ; do R4 nahraje hodnotu 3
    tst r4, #0x0001     ; bit test, logicky AND, pokud nastaven, flag Z=1
    bne bit_nenastaven  ; pokud nenastaven, Z==0, skoc na bit_nenastaven
    ldr r
    f4, =0x0004     ; udelej kdyz je bit nastaven
bit_nenastaven:
    ldr r4, =0x0005     ; udelej když bit neni nastaven
```

Obrázek 6.3: Příklad použití instrukce `TST` k testu nastavení jednoho bitu v registru

6.4.4 Výjimky, neboli přerušení

Procesor má obecně několik zdrojů přerušení, každé z nich přísuší příslušný speciální mód. Test výskytu přerušení je proveden před vykonáním každé instrukce.

Při výskytu přerušení dojde k uložení aktuálního stavového registru (*CPSR – Current Program Status Register*), aby program mohl pokračovat po ukončení obsluhy přerušení v hlavní smyčce programu. Uložená adresa instrukce je současný ([PC]+4) nebo ([PC]+8) podle typu přerušení. Dojde ke změně módu procesoru (změna CPSR[4:0]). Do PC se nahraje adresa, kde je uložená první instrukce obsluhy příslušného přerušení. Automaticky dojde k zakázání dalšího přerušení bitem „I“ nebo „F“ v CPSR. Po zaplnění pipeliningu (2 strojové cykly) dojde k vykonávání obsluhy přerušení.

Při ukončení přerušení se nahraje obsah ([LR]-4) nebo ([LR]-8) do PC, zkopíruje uložené stavové slovo procesoru před příchodem přerušení do aktuálního stavu SPSR → CPSR a obnoví nastavení bitů i a F.

Pokud dojde k požadavku na dvě přerušení současně, je priorita jejich vykonání následující (od nejvyššího k nejnižší): Reset, Data Abort, FIQ, IRQ, Prefetch Abort, Undefined Instruction, SWI (software interrupt).

Při běžném programování jsou používány módy IRQ (interrupt request) a FIQ (fast interrupt request). Režim FIQ má vlastní registry R8 – R12 (viz obr. 6.1), proto nevyžaduje odkládání používaných parametrů na zásobník. Jeho přerušovací vektor je na konci tabulky přerušení. Obslužná rutina tedy může přímo následovat, nemusí zde být žádný skok, viz následující příklad:

```
Vectors:  LDR     PC, Reset_Addr
          LDR     PC, Undef_Addr
          LDR     PC, SWI_Addr
          LDR     PC, PAbt_Addr
          LDR     PC, DAbt_Addr
          NOP                       /* Reserved Vector */
          LDR     PC, IRQ_Addr
;         LDR     PC, FIQ_Addr      /* jump not needed, servicing */
                                    /* routine imediatelly follows */
obsluha_fiq:
          LDR ...
          atd.
          SUBS  PC, R14, #4         /* return from routine */
```

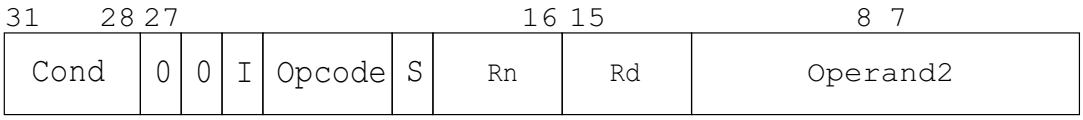
Minimální doba od výskytu přerušení k jeho obsluze jsou 4 strojové cykly, ale odhad maximální doby je složitější. Právě vykonávaná instrukce musí být dokončena a tou může být LDM s maximálním počtem parametrů, která trvá 25 cyklů. Latence tedy může dosáhnout až 27 strojových cyklů. Po vstupu do rutiny přerušení je nutné zálohovat použité registry a před ukončením je obnovit. To je záležitost programu, procesor se o toto nestará.

6.4.5 Instrukční sady

Procesory ARM, které mají v označení „T“ (např. ARMTDMI) jsou vybaveny dvěma instrukčními sadami. Jsou to 32bitová ARM sada a její komprimovaná verze, 16bitová Thumb™ sada. Thumb je tak jistým rozšířením ARM architektury.

Všechny instrukce mají stejnou délku, 32 bitů. Na obr. 6.4 je příklad aritmetické instrukce, např. `ADD Rd, Rn, <Oprnd2>`, která znamená `Rd:= Rn + <Operand2>`.

V následujícím textu popíšeme tvar instrukce v režimu ARM i THUMB. Thumb sada je zkomprimovaná instrukční sada ARM, pomocí které je možné zakódovat ARM program na 65 65 % jeho původní velikosti. Každá THUMB instrukce odpovídá jedné ARM instrukci se stejným výsledkem.



Obrázek 6.4: Příklad kódu instrukce pro aritmetické operace

6.4.6 Instrukce - ARM mód

Tento mód je základní. Instrukce jsou 32 bitů dlouhé.

6.4.7 Instrukce - Thumb mód

Poprvé byl implementován v jádrech ARM7TDMI. Jedná se o komprimovanou verzi instrukční sady ARM. Thumb mód obsahuje 36 instrukcí. Po jejich načtení z paměti jsou dekomprimovány na standartní 32bitové instrukce a jako takové vykonány. Instrukce neumožňují podmíněné vykonávání.

Tok instrukcí skrze pipelining je řízený hodinovým signálem. Jádra vybavena podporou Thumb instrukcí provádějí jejich dekódování ve stavu neaktivní hrany hodinového signálu. Díky tomu nedochází k žádnému taktu navíc, který by byl potřeba pro dekompresi instrukce. Dekomprese je částečně provedena pomocí převodní tabulky.

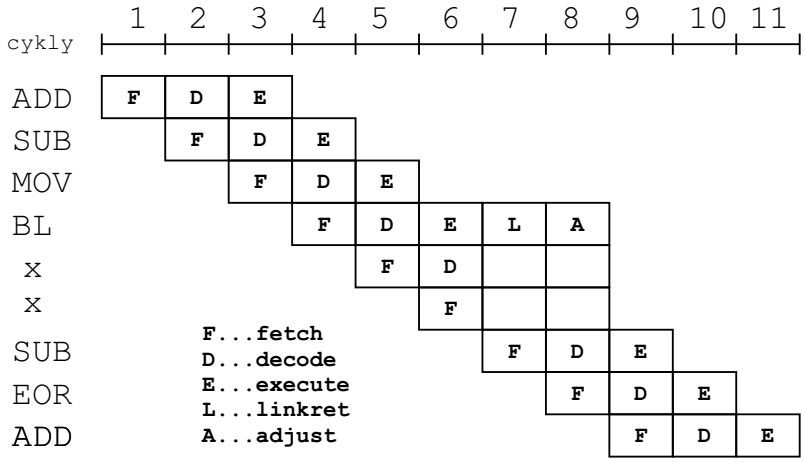
Pokud je kód napsaný v ARM sadě přepsán do Thumb sady, lze docílit úspory paměti až na 60 %. Běh programu je však o něco pomalejší. To umožňuje, aby časově kritické části programu byly napsány v ARM kódu a části u kterých je kritická velikost výsledného kódu v Thumb kódu.

K přepnutí mezi oběma režimy se používá instrukce `BX`. V Thumb módu lze pracovat pouze s registry od R0 – R7.

Podrobný popis módu Thumb je k dispozici v [27].

6.4.8 Pipelining

V procesorech s jádry ARM je implementován pipelining, neboli zřetězené vykonávání instrukcí. To znamená paralelní vykonání jedné instrukce, dekódování následující a čtení další instrukce z paměti. Jednotlivé cykly jsou *fetch*, *decode*, *execute*. V prvním cyklu je instrukce vyzvednuta z paměti kódu (flash), ve druhém je dekódována a nakonec, ve třetím cyklu dojde k jejímu vykonání, např. uložení hodnoty (obsahu registru) do paměti. V každé jednotlivé instrukci je navíc zakódována podmínka jejího vykonání (výhoda použité instrukční sady).



Obrázek 6.5: Vykonávání instrukcí v pipeliningu, přerušení pipeliningu instrukcí skoku

V obrázku 6.5 je $F\ldots$ načtení instrukce, $D\ldots$ její dekóování, $E\ldots$ vykonání, $L\ldots$ uložení návratové hodnoty, a $A\ldots$ zpětny zápis. Proto je zřejmé, že při instrukci skoku je pipelining přerušen a trvá další dva cykly, než dojde k jeho opětovnému zaplnění.

Pipelining je uplatněn pouze pro instrukce pracující s vnitřními registry, např. instrukce ADD, MUL a další. To samozřejmě neplatí pro instrukce přístupů do pamětí a periférií, typicky instrukce skoku a instrukce STR a LDR, kde jeden operand je nepřímo adresován. Skutečný průchod instrukcí procesorem s pipelingem je naznačen na obr. 6.5.

6.4.9 Adresovací módy procesorů ARM

ARM nabízí klasické adresování pro *load/store* archtekturu, ale má širokou paletu variant adresovacích módů. Ve stručnosti je jejich popis shrnut v tabulce 6.1 s ukázkami jejich použití v tabulce 6.2. Více je možno nalézt v literatuře [20].

nepřímé indexové	`LDR r0, [r1]`
nepřímé indexové s více registry	`LDR r0, [r1, -r2]`
bezprostřední	`LDR r0, [r1, #4]`

Tabulka 6.1: Adresovací módy ARM

Příklad kódu	**Efekt**
ADD r3, r2, r1, LSL #3	r3 = r2 + (r1 $\ll$ 3) = r2 + 8 $\cdot$ r1
ADD r4, r3, r2, LSL r1	r4 = r3 + (r2 $\ll$ r1)
LDR r5, [r2]	r5 = memory[r2]
STR r5, [r2]	memory[r2] = r5
LDR r4, [r3, #8]	r4 = memory[r3 + 8]
LDR r3, [r2], #12	r3 = memory[r2] r2 = r2 + 12
LDR r2, [r3, #8]!	r2 = memory[r3 + 8] r3 = r3 + 8
LDR r2, [r3], #8	r2 = memory[r3] r3 = r3 + 8

Tabulka 6.2: Ukázky použití adresovacích módů

6.5 Doba trvání vybraných instrukcí

Instrukční sada ARM procesoru je natolik výkonná a jeho popis by si zasloužil mnohem větší část textu. Pro podrobnější studium lze doporučit [17, 20, 15] nebo [21]. V této kapitole je uvedena přehledová tabulka doby trvání některých vybraných instrukcí.

Protože doba vykonávání některých instrukcí (např. `mul`) je závislá na velikosti operandů, je v tab. 6.3 uvedena maximální doba.

ARM instrukce jsou vykonávány kombinací S, N, i a C cyklů.

- S-cykl je sekvenční přístup do paměti, trvá jeden hodinový cyklus,
- N-cykl je nesekvenční přístup do paměti, trvá dva hodinové cykly,
- I-cykl je interní, který přistupuje pouze k registrům,
- C-cykl je použit při přenos slova mezi ARM a koprocesorem. Stejně jako I-cykl trvá jeden hodinový cyklus.

Instrukce	**Popis**	**Max. počet cyklů**
`movnv r0,#3`	nevykonaná, nv=never	S
`mov r0,#0x3` `cmp r0,r1`	hodnota přímo v instrukci `cmp,tst,teq`	S
`add r0, r1, r1, lsl #2`	hodnota vyjádřena rotací `r0 = r1+(r1*4) = r1*5`	I+S
`mul r0,r3,r1`	násobení, výsledek 32b `r0=r3*r1`	4I+S
`mla r0,r3,r1,r2`	násobení s akumulací, 32b `r0=(r3*r1)+r2`	5I+S

`mull r0,r3,r1,r2`	násobení, výsledek 64b
	`r0=(r1*r2)[31:0]`
	`r3=(r1*r2)[63:32]`
		5I+S

`mlal r0,r3,r1,r2`	násobení s akumulací, 64b
	`r0=(r1*r2)[31:0]+r0`
	`r3=(r1*r2)[63:32]+r3`
		6I+S

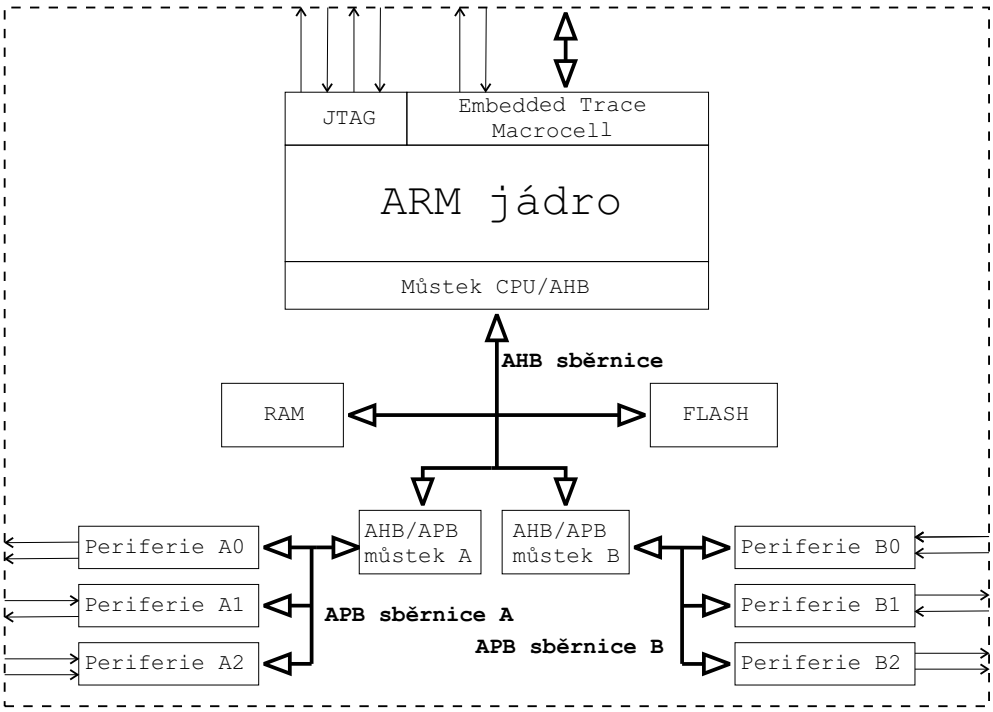
`b, bl`	„jmp“, „call“	N+2S
`ldr`	načtení hodnoty z paměti `ldr r0, =0x00000020`	N+I+S
`str`	uložení hodnoty do paměti `str r0, [r1]`	2N

Tabulka 6.3: Doba trvání základních instrukcí

6.6 Interní sběrnice architektury ARM

Interní architektura jádra ARM a jeho napojení na periferní moduly je velmi komplikovaná. Přesto bude v následujícím textu částečně vysvětlena její činnost, neboť z ní plynou jisté omezení při návrhu, se kterými musí konstruktér počítat.

Procesory s jádrem ARM používají sběrnicovou strukturu, a utváří tak základ architektury systému na čipu (System-On-Chip – SoC). Vnitřní architektura je definována standardem AMBA – Advanced Microcontroller Bus Architecture. Poskytovatelem tohoto standardu je ARM Ltd. Procesor s jádrem ARM se skládá minimálně ze dvou sběrnic — lokální paměťová sběrnice AHB a alespoň jedné VPB sběrnice. Jejich možné propojení je naznačeno na obr. 6.6.



Obrázek 6.6: Možné začlenění jádra ARM do SoC architektury

Sběrnicový systém *System-On-Chip* se skládá z těchto částí:

- AHB (*Advanced High-performance Bus*) tvoří rychlou páteřní sběrnici systému. Jedná se o sběrnici typu Master–Slave s vysokou propustností využívající pipelining. Umožňuje blokové přenosy. Jako slave jednotky mohou být připojeny řadič externí paměti (EMI), paměť RAM na čipu, AHB/APB můstek. Její rychlost dosahuje stovek MHz. Sběrnice umožňuje práci i v režimu multimaster pokud je na čipu další procesor (např. DSP), při použití řadiče DMA nebo v debug módu.
- APB (*Advanced Peripheral Bus*) je určena pro velký počet periférií, které typicky nepotřebují vysoký datový tok sběrnice AHB. APB má jednoduchý interface a je optimalizována pro nízkou spotřebu energie. APB můstek generuje sekvenční přístupy na APB sběrnici. To umožňuje jednodušší odhad statického chování sběrnice. Její rychlost je u většiny současných procesorů volitelná v několika krocích a dosahuje maximálně rychlosti lokání sběrnice AHB. Sběrnice pracuje s 32-bitovým přístupem. Přistupujeme-li tedy do registru některé periférie, který je 8bitový, přístup je 32bitový a požadovaný obsah je maskován až v procesoru při zpracování.

Můstek AHB/APB zachytává všechny signály určené pro periférie (adresy, data a řídící signály), provádí dekódování přístupů a generuje řídící signály („*slave select*“) pro příslušné periferie. Během každého přenosu může být aktivována pouze jedna podřízená periferie. Můstek AHB/APB se na APB sběrnici chová jako master (viz [14, 15]).

Zajímavé je, že v SoC architektuře je řadič přerušení umístěn jako periferie na celkem nevhodném místě (z hlediska časování a obsluhy) na APB sběrnici. Proto je při návrhu nutné počítat s jistou latencí obsluhy přerušení. Stejně tak, požadujeme-li např. čtení z paměti, kterému předcházelo čtení z některé periférie, je nutné počítat s jistým zdržením vyčtení dat z paměti dokud není dokončeno čtení z periférie. Procesory, které mají vysokokapacitní periferie (např. řadiče Ethernetu), mají i více APB sběrnic, a tyto náročné periferie jsou připojeny pouze na jednu z nich.

Z uvedeného popisu je vidět, že AMBA architektura má vlastnosti podobné jako PCI sběrnice v běžném PC, ovšem také se všemi jejími nectnostmi.

6.7 Porovnání mikroprocesorů s jádry ARM

V dnešní době licencuje procesory s jádrem ARM do svých procesorů snad každý velký výrobce integrovaných obvodů. V tab. 6.4 je uveden krátký přehled nejznámějších výrobců mikroprocesorů a jejich mikrokontrolérů založených na jádře ARM. Výrobci doplňují jádro různými periferiemi, buď svými nebo koupenými od společnosti ARM, a tak tvoří různé procesory založené na stejném jádře. Zajímavý přehled je uveden na adrese [24].

V tab. 6.4 znamená symbol „*“ přítomnost jednotky v daném procesoru a „–“ jeho absenci. Blok EMI symbolizuje External Memory Interface, blok připojení externí paměti.

Procesor Výrobce	**Jádro ARM**	**RAM FLASH**	**EMI**	**ADC**	**DAC**	**USB/CAN Ethernet**	**SPI/I²C PWM**	**Další perif.**
ADuC702x Analog Devices	7TDMI 45MHz	8k/64k	–	16/12b 1MSPS	–	–	1/2/6(3f)	PLA WD
LPC2214-2224 Philips	7TDMI-S 60 MHz	16k/256k	–	8/10b 400kSPS	–	–	2/1/6	RTC WD
LPC2119-2129 Philips	7TDMI-S 60 MHz	16k/128-256k	–	4/10b 400kSPS	–	–	2/1/6	RTC WD
LPC2114-2124 Philips	7TDMI-S 60 MHz	16k/128-256k	–	4/10b 400kSPS	–	–/2/–	2/1/6	RTC WD
STR720 STM	720 50 MHz	16k/– 4k bootROM	*	4/11b 0,95kSPS		1/2/–	2buff./–/–	DMA IDE
TMS320VC5470 Texas	7TDMIE 47,5/100 MHz	16k/–	*	–	–	–/–/–	1/1/–	DSPC54x IrDA,keyb
AT91SAM7A2 Atmel	7TDMI 30MHz	16k/–	*	2x8/10b	–	–/4/–	1/–/4	LIN
AT91M55800A Atmel	7TDMI 33MHz	8k/–	*	8/10b	2/10b 5 μs	–/–/–	1/–/–	
AT91RM9200 Atmel	920T 180 MHz	16k/128k	*	–	–	2/–/1	1/1/6	20DMA
AT75C221 Atmel	7TDMI 60 MHz	104k/– 1k bootROM	*	–	–	–/–/1	1/–/6	
PXA26x Intel	XScale 147 MHz	–/512k-1M 1k bootROM	*	–	–	–/1/1	1/–/–	DMA,LCD AC97,IrDA MCC,PCMCIA
EP9307 Cirrus	920T 200 MHz	–/– bootROM	*	–	–	1/–/1	1/1/–	DMA,LCD AC97,IrDA video
MAC7100 Motorola	7TDMI-S 50 MHz	48k/1M	*	2x16/10b 7 μs	–	–/4/–	4/1/–	DMA,LIN
LH75401 Sharp	7TDMI-S 90 MHz	32k/–	*	8/10b	–	–/1/–	1/–/3	4xDMA,LCD touchscreen

Tabulka 6.4: Přehled vlastností a periférií vybraných mikrokontrolérů s jádry ARM

Kapitola 7

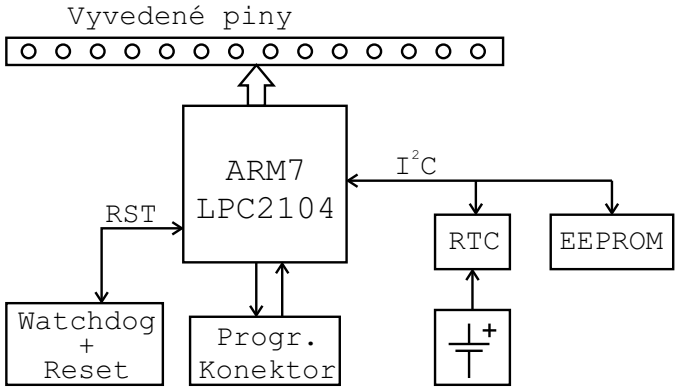
Prototypový modul s procesorem ARM – Procesorové jádro AMCLPC

7.1 Návrh prototypového modulu

Ve firmě AMiT, spol. s.r.o.[9], která se zabývá návrhem a výrobou modulárních řídících systémů, jednotek vzdáleného sběru dat a dalších zařízení pro průmyslovou automatizaci, ve které jsem v době řešení diplomové práce pracoval, vznikl požadavek na vývoj levného a výkonného procesorového modulu o malých rozměrech, který by mohl být použit v modulech pro sběr dat, které firma vyrábí. Řešení systému na bázi malého modulu zasunutelného do aplikační desky má jednu nepopíratelnou výhodu, a to že již jednou odladěný modul CPU jádra může být opakovaně použit v různých systémech. **Požadavky** na modul byly následující:

- napájecí napětí 3,3 V,
- sériová EEPROM pro zálohování nastavení včetně zálohovací baterie, RTC jako externí zdroj reálného času,
- čtyřvrstvá deska plošných spojů kvůli stabilitě a odolnosti proti rušení,
- dohlížecí obvod a obvod pro RESET mimo procesor,
- řady konektorů po obou stranách pro snadné zasunutí modulu do aplikační desky,
- vyvinout softwarové knihovny tak, aby mohl být modul efektivně použit.

Na základě těchto požadavků a v souladu s představami o praktickém použití modulu firmou AMiT, spol. s.r.o., byl proveden návrh modulu CPU jádra. Modul je označen AMCLPC. Lze ho zasunout do dutinkových konektorů desky aplikace. Modul jsem v prvním období práce používal převážně ke studiu základních vlastností a parametrů procesorů s jádry ARM, možností připojení různých periférií a komunikačních možností. V první fázi vývoje byl modul použit také k analýze dostupných programových prostředků pro vývoj sw aplikací, což je v případě „nového“ procesoru vždy velmi důležitá část vývojového úkolu. Blokové schéma modulu je na obr. 7.1.



Obrázek 7.1: Blokové schéma modulu AMCLPC

[9] AMiT, spol. s.r.o., – Aplikace Mikroprocesorové Techniky, `http://www.amit.cz`

Mikrokontrolér	RAM	FLASH
LPC2104	16kB	128kB
LPC2105	32kB	128kB
LPC2106	64kB	128kB

Tabulka 7.1: Procesory Philips použitelné pro modul AMCLPC

7.1.1 Volba procesoru pro AMCLPC

Podle požadavků na procesor uvedených v rozboru byl vybrán procesor Philips LPC2104. V období začátku vývoje[10] byl v České republice tento procesor téměř jediný snadno dostupný procesor ARM s integrovanou pamětí RAM a FLASH. Procesory jsou vybaveny těmito periferiemi:

- dvěma řadiči UART, rozhraním I^2C a SPI,
- dva 32bitové časovače s CAP/COM (capture/compare - záchytnými/komparačními) kanály,
- obvod reálného času RTC, watchdog WD,
- násobička hodin PLL,
- šest kanálů PWM a
- až 32 vstupně-výstupních pinů 5V tolerantních.

Procesor není osazen interním A/D převodníkem, což vedlo k anylýze možností použití externích převodníků s různými rozhraními.

7.2 Popis použitých součástek

7.2.1 Procesor LPC2104

Procesor má dvě napájecí napětí, napájení jádra procesoru je 1,8 V a pro periférie 3,3 V. Podle požadavků na velikost paměti RAM volíme procesor podle tab. 7.1. Z hlediska periferních bloků se jedná o procesor s minimálním počtem periférií vzhledem k nástupců této řady, např. řady LPC22xx nebo LPC231x.

7.2.2 Napájecí zdroj LP2951

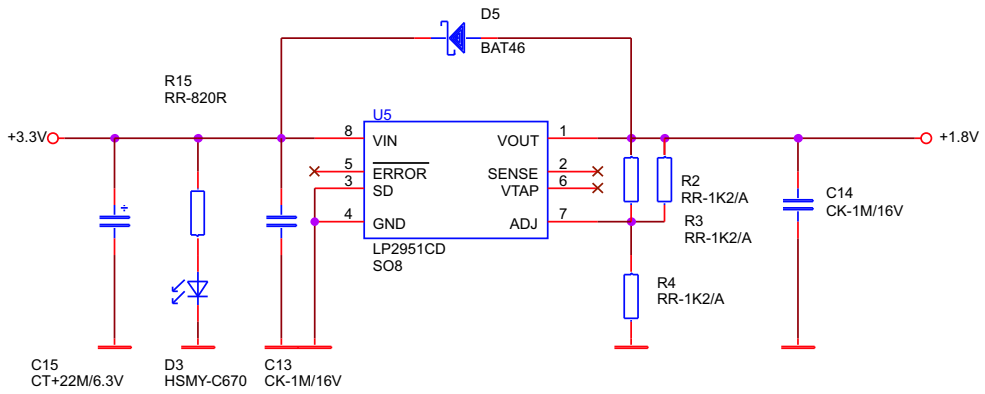
Napájení modulu bylo v návaznosti na další systémy firmy AMiT zvoleno 3,3 V. Na desce je pouze lineární stabilizátor 1,8 V pro napájení jádra. Podle dokumentace [22] je maximální proudový odběr jádra cca 30 mA. Zvolený stabilizátor LP2951 je typu bandgap s malým úbytkem napětí v propustném směru (max. 380 mV při 100 mA), maximálním výstupním proudem 100 mA a výstupním napětím 1,25 – 30 V s přesností 1 %. Stabilizátor má vestavěné teplotní a proudové omezení. Obvod je v pouzdře SO-8. Výstupní napětí 1,8 V je nastaveno odporovým děličem R2, R3, R4. Dioda D5 je ochranná dioda pro část s napětím +1,8 V.

7.2.3 Watchdog ADM706TAR, obvody resetu

Aby byl splněn požadavek na bezpečnost zařízení proti zacyklení programu, byl modul vybaven externím dohlížecím obvodem (watchdog), který současně provádí monitoring napájení. Byl zvolen obvod ADM706TAR (T ... resetovací úroveň při poklesu napětí 3,08 V, A ... industry provedení, R ... pouzdro SO-8). Obvod generuje signál RESET v těchto případech:

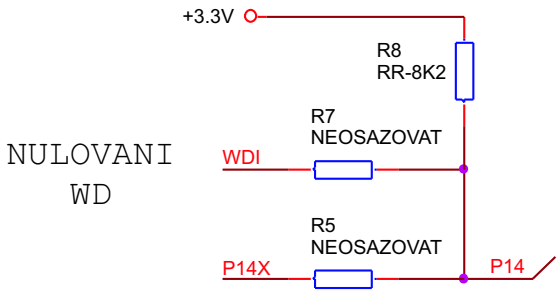
- při zapnutí napájení po dobu, než se napájecí napětí dostane na správnou úroveň,
- v případě poklesu napájecího napětí pod $3,08$ V,
- pokud dojde k zacyklení programu.

[10] podzim 2003



Obrázek 7.2: Zapojení napájecího zdroje

V tom případě je obvodem watchdog generován signál $\overline{RESET}$ na pinu 7, neboť správně se vykonávající program v pravidelných intervalech resetuje watchdog. Aktivace funkce watchdog je provedena osazením odporu R7, obr. 7.3.



Obrázek 7.3: Neosazený odpor R7, pro aktivaci WD nutno osadit

Po aktivaci funkce watchdog musí být obvod „resetován“[11] programem do doby kratší než 1,6 s. Pokud k tomu nedojde (např. vlivem zacyklení programu), je procesor resetován a tím uveden do definovaného stavu. Správné ošetření takové situace je již záležitostí programu. Ten musí zjistit, proč byl resetován, zda-li se jednalo o reset při zapnutí nebo reset od watchdogu. Resetování dohlížecího obvodu zprostředkovává signál $\overline{WDI}$ (WatchDog Input) připojený k pinu P0.14 procesoru.

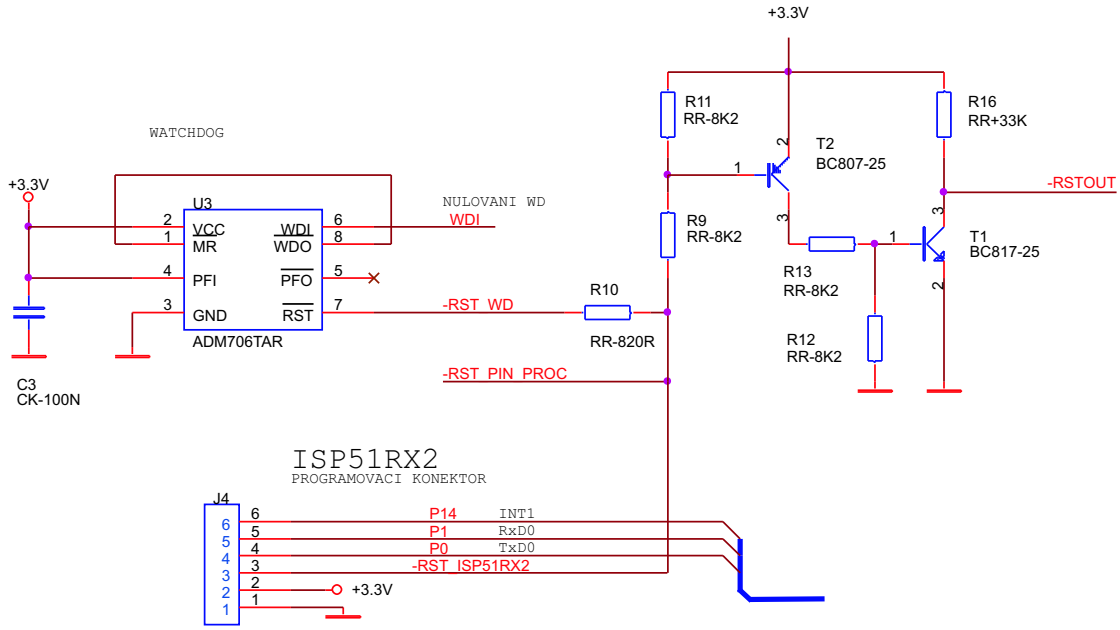
7.2.4 Paměť EEPROM M24256

Pro zálohování konfiguračních parametrů zařízení je modul osazen pamětí typu EEPROM[12]. Byla zvolena paměť s organizací 32k × 8bitů připojená k procesoru přes I^2C rozhraní. Může pracovat s přenosovou rychlostí až 400 kb/s a používá 7bitovou adresou *1010000b*. Adresování paměťových buněk je 16bitové. Paměť může být chráněna proti přepsání nastavením pinu $\overline{WD} = H$. Připojení paměti k procesoru je na obr. 7.5.

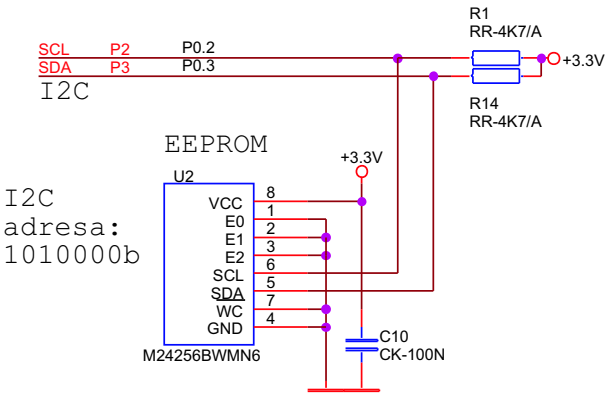
7.2.5 Obvod reálného času RTC RTC8564

Dalším požadavkem bylo použití obvodu reálného času. Byl vybrán obvod firmy EPSON připojený přes I^2C rozhraní. Obvod používá 7bitovou adresu *1010001b*. Připojení ke sběrnici I^2C je na obr. 7.6. Obvod napájen a zálohován baterií. Klíčovým parametrem bateriově zálohovaných obvodů je jejich klidový odběr, ten je pouze 275 nA při napájecím napětí 3 V.

[11] změna úrovně na vstupu WDI obvodu ADM706
[12] EEPROM (E^2PROM) Electrically-Erasable Programmable Read-Only Memory



Obrázek 7.4: Zapojení obvodu Watchdog a resetovacích obvodů procesoru



Obrázek 7.5: Připojení paměti EEPROM M24256 k I^{2}C a procesoru

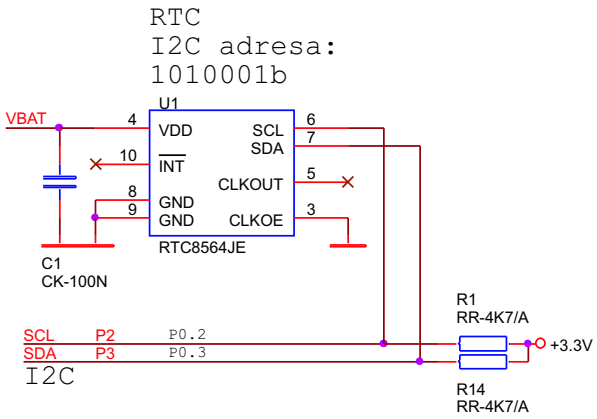
V obvodu RTC je umístěn krystal hodinového kmitočtu 32,768 kHz a obvod nabízí kromě základní funkce kalendáře i funkce alarmu, časovače a může poskytovat hodinové výstupy 32,768 kHz, 1024 Hz, 32 Hz a 1 Hz. Jeho připojení ke sběrnici I^{2}C je na obr. 7.6.

7.3 Realizace a popis modulu AMCLPC

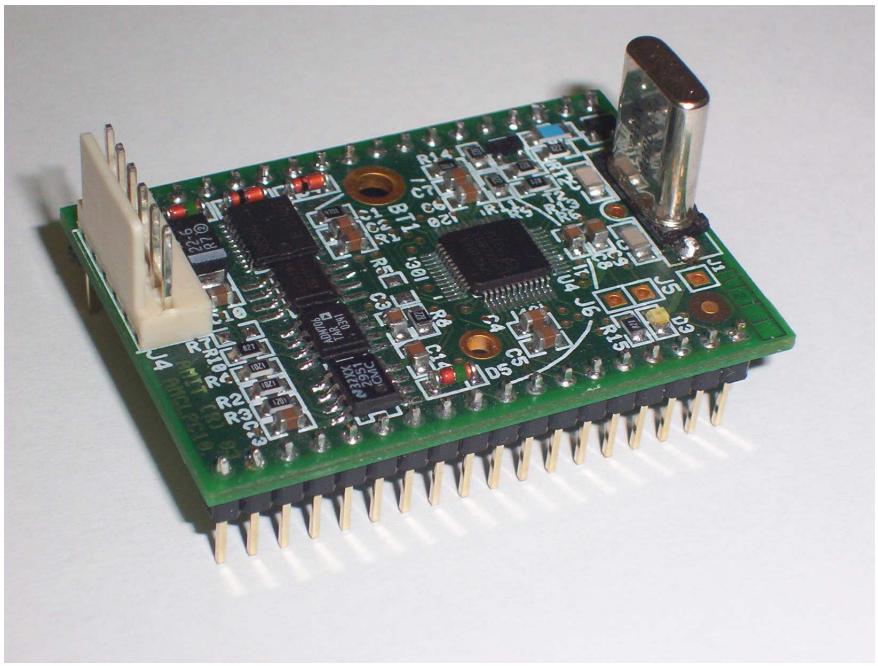
Schéma výsledného modulu[13] AMCLPC je na obr. 7.9. Deska plošných spojů je čtyřvrstvá kvůli lepší odolnosti proti elektromagnetického rušení a tím lepší stabilitě systému. Pro aplikace s taktovacím kmitočtem přesahujícím několik desítek MHz je vhodnější použít čtyřvrstvé desky namísto levnějších dvouvrstvých. Velikost desky je 34,3 × 49,5 mm a fotografie na obr. 7.7. Konektor ve spodní části desky (bílý konektor na fotografii vlevo) je určet pro servisní nahrání aplikace. Na něj jsou signály (zleva) GND, +3,3 V, $\overline{RST}$ procesoru, TxD0, RxD0 a BOOT. Rozložení pinů na bočních konektorech AMCLPC je na obr. 7.8. Je důležité zdůraznit, že modul AMCLPC **nemá tlačítko Reset**.

V obou vnitřních vrstvách desky je rozlitá měď, vrstvy jsou připojeny na +3,3 V a

[13]Návrh byl proveden v prostředí OrCAD 386+.



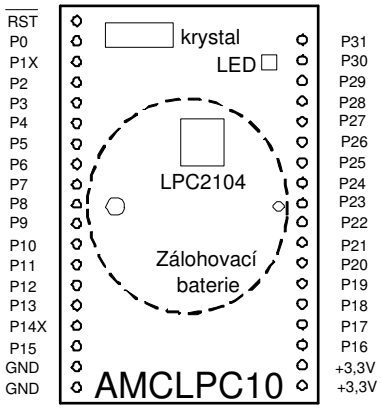
Obrázek 7.6: Připojení RTC8564 k I^{2}C sběrnici procesoru



Obrázek 7.7: Fotografie modulu CPU jádra AMCLPC bez osazené baterie

GND a tvoří tak rozprostřený blokovací kondenzátor s vysokým rezonančním kmitočtem. Vnitřní vrstvy minimalizují plochu napájecích smyček a nízkou parazitní indukčnost přívodů napájení. Deska je ve výrobě osazována automatem. Proto jsou na desce umístěny osazovací značky, pomocí kterých je deska přesně usazena pod osazovací hlavou. Boční konektory AMCLPC mají klíč, jsou nezaměnitelné, jednotku tak nelze zasunout do patice obráceně, na levé straně je 19 a na pravé straně 18 konektorů.

Ve spodní části modulu je napájení, dvě napájecí svorky na každé straně, GND vlevo a VCC (+3,3 V) vpravo, viz obr. 7.8. V levé spodní části je programovací konektor J4 (s interním rozhraním firmy AMiT, spol. s.r.o.,) ISP51RX2 se signály TxD a RxD portu P0, napájením +3,3 V a GND, signálem pro reset AMCLPC jednotky a signálem pro uvedení procesoru do režimu bootloader, který je spojen s pinem P0.14 procesoru. V pravém horním rohu desky jsou konektory J1 pro připojení země osciloskopické sondy a J5 a J6, jednoúčelové piny procesoru pro JTAG rozhraní, které není osazeno. V pravém horním rohu je umístěna



Obrázek 7.8: Rozložení konektorů CPU jádra AMCLPC

žlutá LED signalizující přítomnost napájecího napětí 3,3 V. Kompletní seznam součástek je v příloze D.1.

Součástí modulu je i baterie pro zálohování napájení obvodu reálného času RTC8564JE. Ta zatím není osazena, protože funkce, pro které je baterie třeba, nejsou zatím třeba implementovat. Obvod reálného času je spolu s EEPROM připojen k procesoru přes sběrnici I^2C. Rezistory R1 a R2 slouží jako pull-up pro I^2C sběrnici.

Aby resetovací signál watchdogu $\overline{WDO}$ resetoval procesor, musí být vývody ($\overline{WDO}$-WatchDog Output) a ($\overline{MR}$-Manual Reset) obvodu ADM706 propojeny. Celé zapojení je na obr. 7.4. Signál $-RST\_WD$ resetuje procesor a je vyveden jako výstup i na konektor AMCLPC (pin 1), na obr. 7.8 je označený jako $\overline{RST}$. Pro vstup signálu přerušení od WD byl vybrán pin P0.14. Volba byla provedena na základě této úvahy:

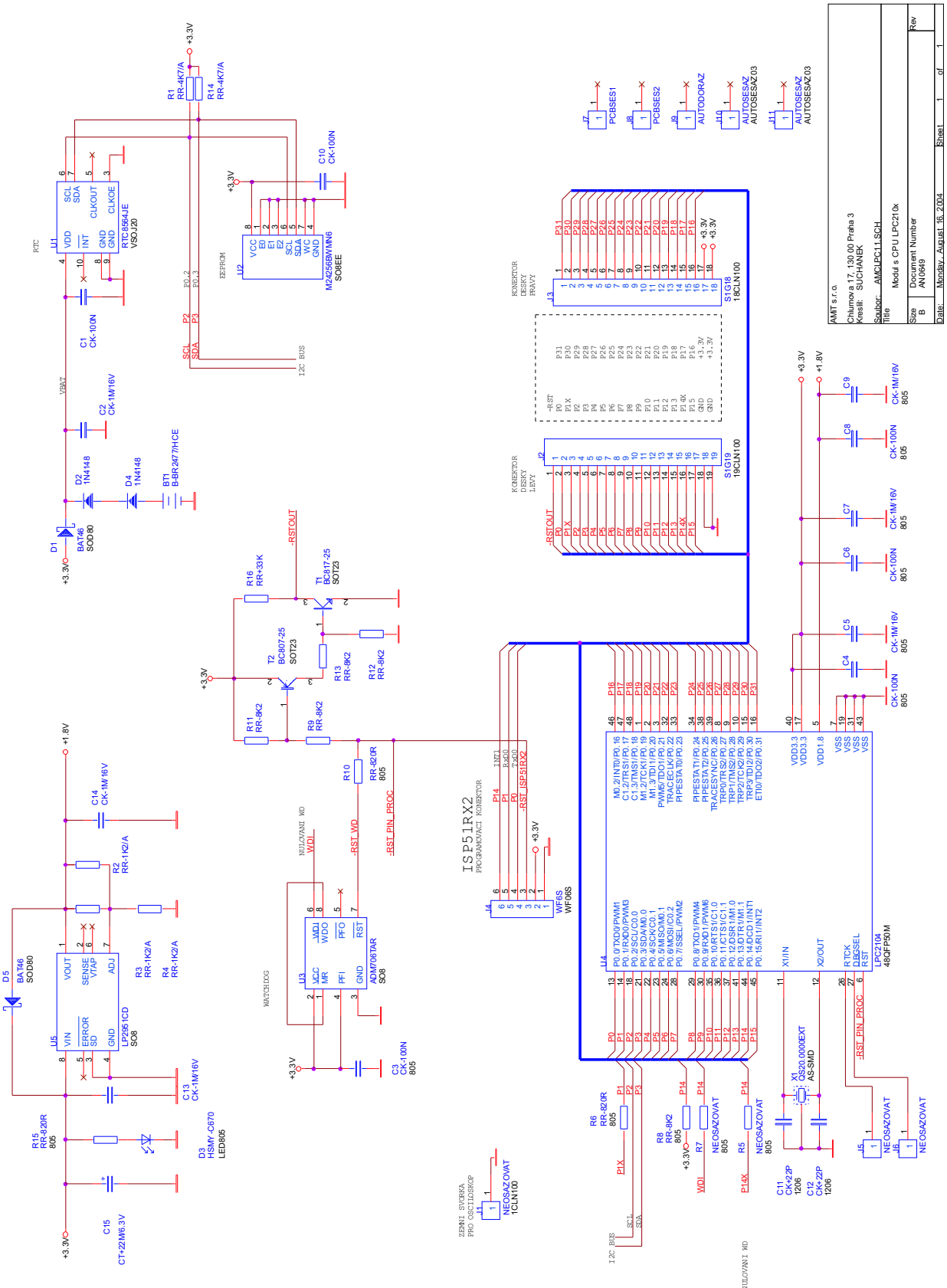
- Na pinu P0.14 se nachází vstup procesoru pro přerušení od externího zařízení (kromě základní funkce jako I/O pin),
- pin je používán pro exkluzívní činnost uvedení procesoru do boot režimu, ve kterém je aktivován interní bootloader, a automaticky dojde k nahrání programu do paměti procesoru.

Pokud by pin byl použit jako I/O, mohlo by dojít k situaci, kdy by pin P0.14 byl aplikací držen v úrovni L a současně by došlo k resetu od watchdogu (připojeného na jiný pin). Úroveň L na pinu P0.14 by zavedla procesor do boot módu a očekávala by nahrávání programu, což by v tu chvíli nebylo požadované. Navozená situace by vyžadovala zásah obsluhy (odstranění úrovně L z pinu P0.14 a resetování), což je nepřípustné.

Proto je pin P0.14 vyveden na výstupní konektor AMCLPC přes sériový odpor R5 (standardně neosazovaný) a označený P14X. Zapojení je patrné z obr. 7.3 a 7.4. Podobně, pokud by nebyl dostatek I/O pinů, je možné použít piny sériové linky na P0.0 (TxD) a P0.1 (RxD) procesoru jako I/O. Aby připojené obvody nezasahovaly do případné probíhající komunikace, je pin P0.1 vyveden na AMCLPC konektory přes sériový odpor R6 s hodnotou 820 Ω. Osazený krystal je 20 MHz a lze bez problémů nastavit sériovou komunikaci na běžně požadované rychlosti až do 115 kBd.

7.4 AMCLPC - Zhodnocení návrhu

Modul AMCLPC byl bez problémů oživen. Při vývoji aplikačních knihoven pro modul jsem nenašel žádný závažný problém týkající se návrhu a funkčnosti desky, který by znemožnil použití modulu pro další aplikace.



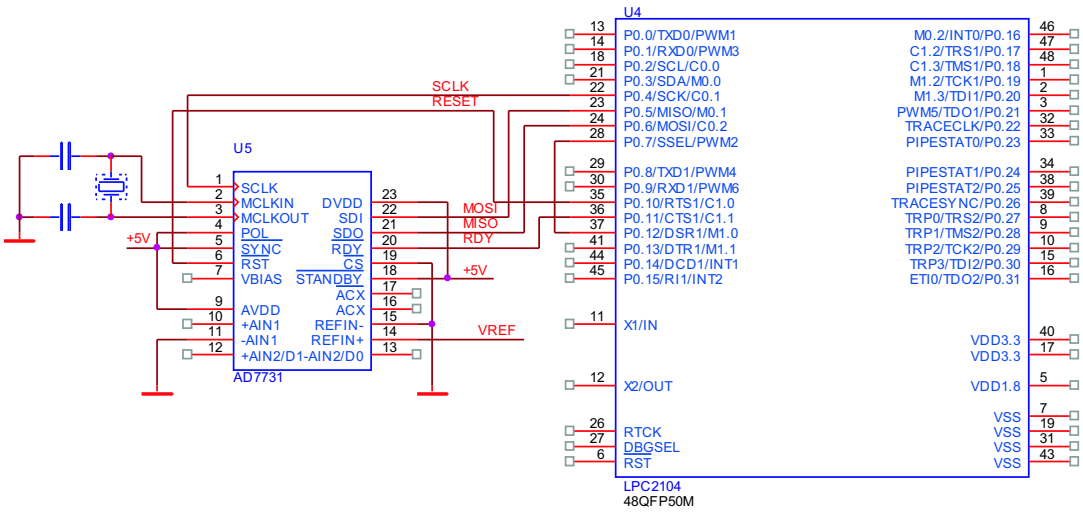
Obrázek 7.9: Schéma modulu CPU jádra AMCLPC

7.5 AMCLPC - Připojení externích převodníků

Protože procesor LPC2104 není vybaven interním A/D a D/A převodníkem, musí být připojen externí převodník. Z důvodu úspory zabraných I/O pinů byly testované převodníky připojeny přes sériové rozhraní SPI. V případě připojení paralelního převodníku by sice došlo k zabrání většího počtu vstupně/výstupních pinů, ale mnohonásobně by se zvýšila dosažitelná vzorkovací frekvence.

7.5.1 AMCLPC a externí A/D převodník AD7731

Analogově-číslicový převodník AD7731 od Analog Devices je 24bitový Sigma-Delta převodník s maximální vzorkovací frekvencí 800 Hz určený pro měření tenzometrických můstků. Převodník má sériové rozhraní SPI a obsahuje vlastní generátor hodin. Vnitřní registry převodníku jsou 8bitové, což je výhodné pro spolupráci s LPC210x také s 8bitovým SPI data registrem. Přestože převodník nevyhovuje pro aplikace požadované od jednotky pro sběr dat (je moc pomalý) byla s jeho pomocí otestována možnost připojit sériový převodník přes SPI. Ve schéma na obr. 7.10 je vyznačeno připojení převodníku k procesoru. Většina dalších obvodů (napájení, komunikace atd.) ve schématu vyznačeny nejsou. Program pro komunikaci s převodníkem je uveden na CD ROM. Převodník je v pouzdru SOIC-24. Proto musel být naletován na malou redukční desku spojů a osazen konektory pro zasunutí do nepájivého pole.

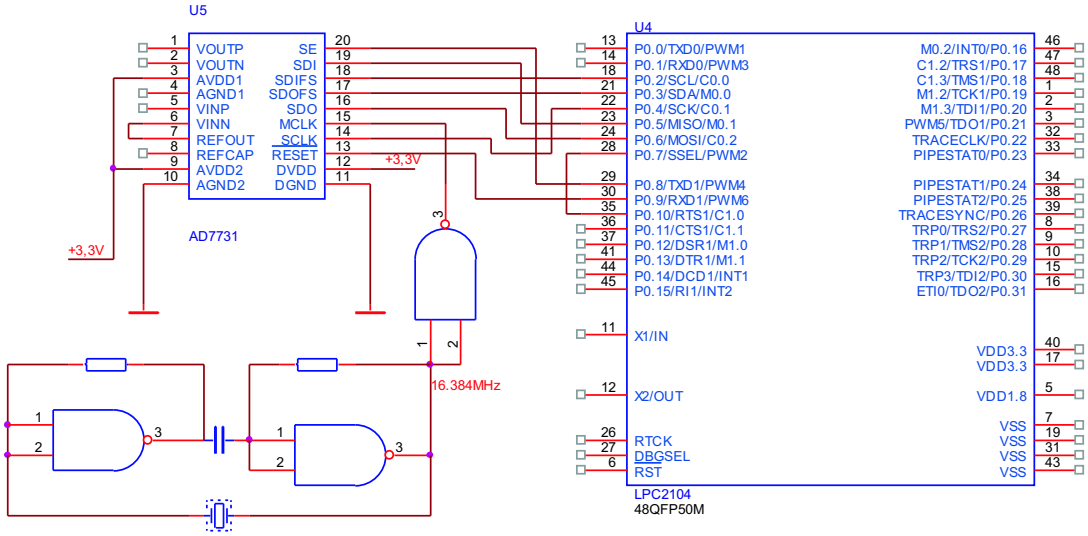


Obrázek 7.10: Blokové schéma propojení AMCLPC a AD7731

7.5.2 AMCLPC a externí převodník AD73311

Kvůli nedostačující rychlosti AD7731 byl navržen převodník, který v jednom pouzdru obsahuje A/D i D/A převodník. Obvod AD73311 od Analog Devices je 16bitový Sigma-Delta A/D a D/A převodník s maximální vzorkovací frekvencí 64 kSa/s a interní referencí. Je určen pro aplikace zpracování řeči a telefonní obvody. Sériové rozhraní SPORT je rozšířením standardního SPI a tento převodník se na SPORT chová jako MASTER. SPORT je rozhraní typické pro obvody spolupracující s DSP procesory. Ukázalo se, že pomocí procesoru LPC2104 nelze komunikovat s obvody s 16bitovým SPORT rozhraním. Procesor má 8bitový datový registr SPIDR společný pro vysílání i příjem dat po SPI. SPORT interface je však 16bitový. V případě, kdy si převodník vyžádá nastavení (nebo jiná data), generuje SLCK hodiny, a očekává přijem 16 bitů dat. Procesor ale nestihne bez přerušení vyslat 16 bitů dat, vždy pouze vyšle 8 bitů z registru SPIDR, pak musí následovat prodleva daná několika instrukcemi, které do

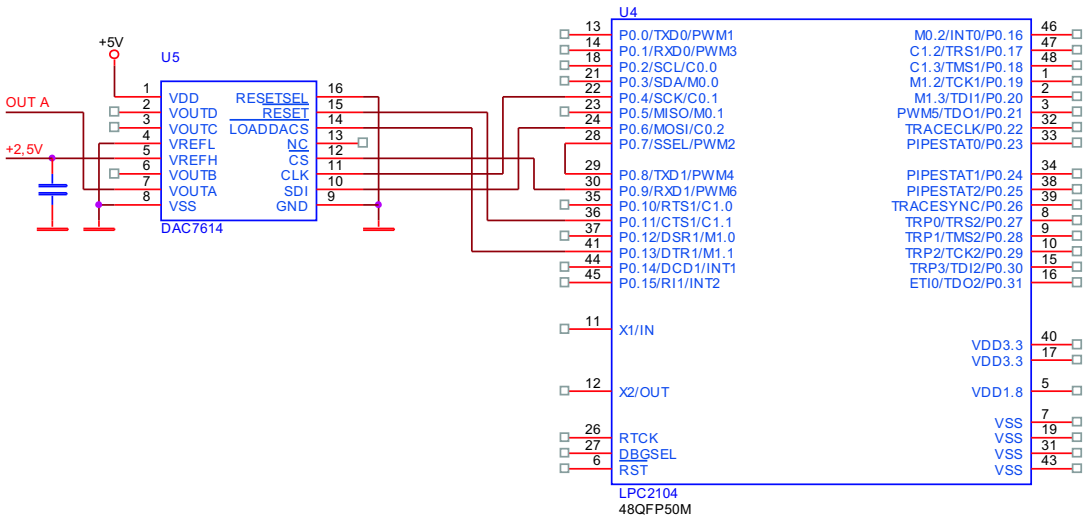
SPIDR nahrají další hodnotu a tu pak vyšle. Schéma připojení komunikačních vodičů je na obr. 7.11. Situaci by bylo možné řešit, pokud by procesor umožňoval přepínat mezi 8 a 16bitovým SPI rozhraním, nebo pomocí tzv. „buffered SPI“, u které je datový registr zálohován, takže do něj lze nahrát jednu hodnotu, která se začne vysílat, a mezitím než dojde k odelsání všech 8 bitů, lze nahrát hodnotu další a tím ji připravit k okamžitému odeslání po dokončení odeslání předchozí hodnoty. Převodník je v pouzdru SOIC-20. Proto musel být naletován na malou redukční desku spojů a osazen konektory pro zasunutí do nepájivého pole.



Obrázek 7.11: Blokové schéma propojení AMCLPC a AD73311

7.5.3 AMCLPC a externí D/A převodník DAC7614

Protože by jednotka sběru dat měla obsahovat i číslicově-analogový převodník pro generování průběhů, byla testována možnost připojit obvod firmy Burr-Brown DAC7614. Jedná se o 4kanálový 12bitový D/A převodník s žebříčkovou sítí R-2R a zesilovačem na výstupu, dobou ustálení 10 μs a sériovým rozhraním SPI. Požadované výstupní hodnoty jsou převedeny na výstup se sestupnou hranou signálu $\overline{LOADDACS}$. Data jsou pro každý kanál uvozeny v posloupnosti sériových dat dvěma adresními bity kanálu. Schéma zapojení je na obr. 7.12. Komunikace je 16bitová, první dva bity jsou adresa kanálu, pro který patří vzorek ve spodních 12 bitech. Vzorky jsou zapisovány do vnitřních záchytných registrů převodníku. Protože přepsání hodnot z registrů na vlastní D/A převodník se provádí najednou pro všechny 4 kanály, je možné nahráním hodnot vzorků do více záchytných registrů a následným aktivováním signálu $\overline{LOADDACS}$ generovat synchronně průběhy ve více kanálech.



Obrázek 7.12: Blokové schéma propojení AMCLPC a DAC7614

Pomocí D/A převodníku DAC7614 byla implementována metoda DDS pro generaci signálů. Popis programu a dosažené výsledky jsou v kap. 11.1 a na obrázku 11.1.

Kapitola 8

Vývojový kit MCB2100

Jednotka sběru dat musí být osazena A/D převodníkem. Pokud je převodník na čipu procesoru je to z hlediska jednoduššího zapojení výhodnější. V průběhu práce docházelo k postupnému uvolňování nových procesorů a vývojových kitů na trh. Jedním z nich je i kit[14] MCB2100 od firmy Keil. Vývojové kity[15] jsou osazeny těmito obvody:

- procesor LPC2129
  - s interním 10bitovým A/D převodníkem,
  - pamětí programu a dat,
  - SPI/I^{2}C interface
  - a 2 moduly CAN 2.0B rozhraní,
- JTAG (a volitelně ETM) konektor,
- řadič fyzické vrstvy CAN, obvody TJA1040,
- obvody pro uvedení procesoru do BOOT módu a jeho reset pomocí sériové linky,
- stabilizátory pro napájení,
- konektorem sériové linky,
- 8 LED diodami a
- dvěma tlačítky.

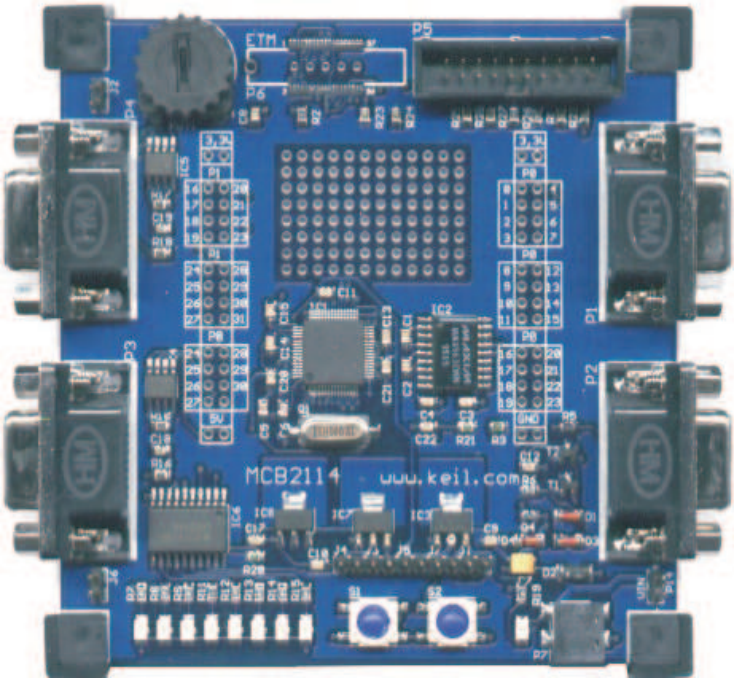
8.1 Popis vývojového kitu

Na desce je ochranná dioda proti přepólování a žlutá LED signalizující přítomnost napájecího napětí. Obvody pro generování signálů pro uvedení procesoru do BOOT módu a jeho reset pomocí sériové linky, jsou tvořeny dvěma tranzistory, protože převodník TTL/RS232, obvod MAX563, je již plně obsazen signály TxD0, RxD0, TxD1 a RxD1 portů označených COM0 a COM1. Reset a uvedení do boot módu je možné pouze přes sériovou linku COM0.

Vnější napájecí napětí je 9 V. LED diody jsou připojeny přes záchytný obvod 74LVC244. Rozhraní JTAG umožňuje připojení převodníku uLINK od Keilu. Tento převodník je k PC připojitelný přes USB. Pomocí něho lze ladit program přímo pomocí vývojového prostředí na PC. Při používání této funkce je třeba zkratovat jumper J9 umístněn v blízkosti JTAG konektoru. Komunikace při ladění prostřednictvím JTAG probíhá mnohem pomaleji, než je skutečný běh aplikace, proto nelze ladit real-time chování programů. K tomu je však možné použít tzv. ETM (Embedded Trace Modul) konektor, který sdílí některé signály JTAG a přidává některé další. Pro povolení této funkce je nutné zkratovat jumper J8 v blízkosti ETM konektoru. V tom případě nelze piny P1.16 – P1.31 použít jako I/O, protože právě tyto mají alternativní funkci pinů JTAG A ETM rozhraní.

[14] Kity byly zakoupeny katedrou měření z grantu Ing. Nováka na podzim 2004.

[15] Bylo zakoupeno 10 kitů od společnosti HiTEX, přičemž 5 z nich nefungovalo správně. Nebylo možné používat JTAG a ETM rozhraní, pravděpodobně kvůli špatně osazenému konektoru ETM.



Obrázek 8.1: Fotka vývojového kitu MCB2100

Pro otestování práce s A/D převodníkem je kit vybaven potenciometrem, který je přes jumper J2 připojen k analogovému vstupu AIN0. Pro sítě CAN je kit vybaven řadiči CAN fyzické vrstvy TJA1040. význam jednotlivých jumperů je v tab. 8.1.

Obrázek 8.2: Převodník USB/JTAG pro připojení ke kitu MCB2100

Jumper	Význam
J1	Povolí generovat signál BOOT signálem RTS
J2	připojí analogové napětí na AIN0
J3	Napájecí napětí 3,3 V pro CPU
J4	Napájecí napětí 1,8 V pro CPU
J5	Napájení analogové reference CPU
J6	Povolí LED
J7	Povolení Resetu procesoru tlačítkem S2
J8	Povolí modulu ETM
J9	Povolení funkce JTAG
J10	Povolí RESET signálem DTR sériové linky
J11	Enable pro CAN řadiče TJA1040
J12	Enable pro vstupní signál procesoru CAN_RD1 na pinu P0.25
J13	Enable pro vstupní signál procesoru CAN_RD2 na pinu P0.23

Tabulka 8.1: Význam jumperů na kitu MCB2100

Pro vývoj aplikací je možné použít prostředí μVision, které umožňuje propojení s kitem přes JTAG. Na obrázku 8.2 je fotografie převodníku USB/JTAG uLINK a vývojového kitu MCB2100.

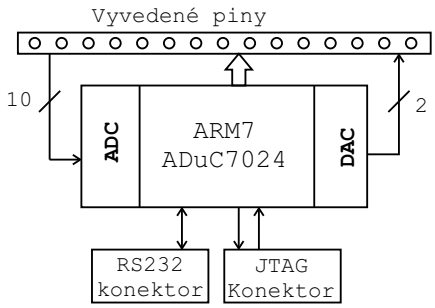
Kapitola 9

Vzorový modul s ADuC7024

Požadavky na jednotku pro sběr dat jsou nejen ze strany A/D převodníku, ale i ze strany D/A převodníku pro generování výstupních průběhů. Lze použít procesor s interním D/A převodníkem. Takové procesory byly ke konci práce, v prosinci 2004, poskytnuty Ing. Dohnalem, zástupcem firmy AMTEK. Jednalo se o procesory Analog Devices ADuC7024, vybavené dvoukanálovým D/A převodníkem na čipu procesoru. Koncept požadavku pro nový modul byl vytvořen na základě zkušeností s prototypovým modulem AMCLPC na průniku možností AMCLPC a MCB2100. Jedná se o modulek s procesorem, napájecím zdrojem, krystalem, konektorem JTAG pro ladění a programování procesoru a konektorem sériové linky. Prostřednictvím řadových kontekorů umožňuje zasunutí do dutinek na desce aplikace. Na konektory jsou vyvedeny všechny piny procesoru. Jeho blokové schéma je na obr. 9.1.

Požadavky návrhu jsou:

- procesor ADuC7024, varianta BST62 (pouzdro LQFP),
- obvody pro resetování a zavedení do boot módu prostřednictvím signálů sériové linky,
- konektor sériové linky,
- tlačítkem pro resetování,
- programovací a ladící konektor JTAG,
- stabilizátor napětí, externí napájení +5 V, ochranné prvky napájení,
- svorku pro připojení analogové země zdroje signálu,
- hodinkový krystal s možností připojení externího zdroje taktovacího kmitočtu,
- všechny ostatní piny procesoru vyvést na konektory, aby bylo možné modul zasunout do kontaktního pole.



Obrázek 9.1: Blokové schéma modulu s procesorem ADuC7024

9.1 Popis použitých součástek

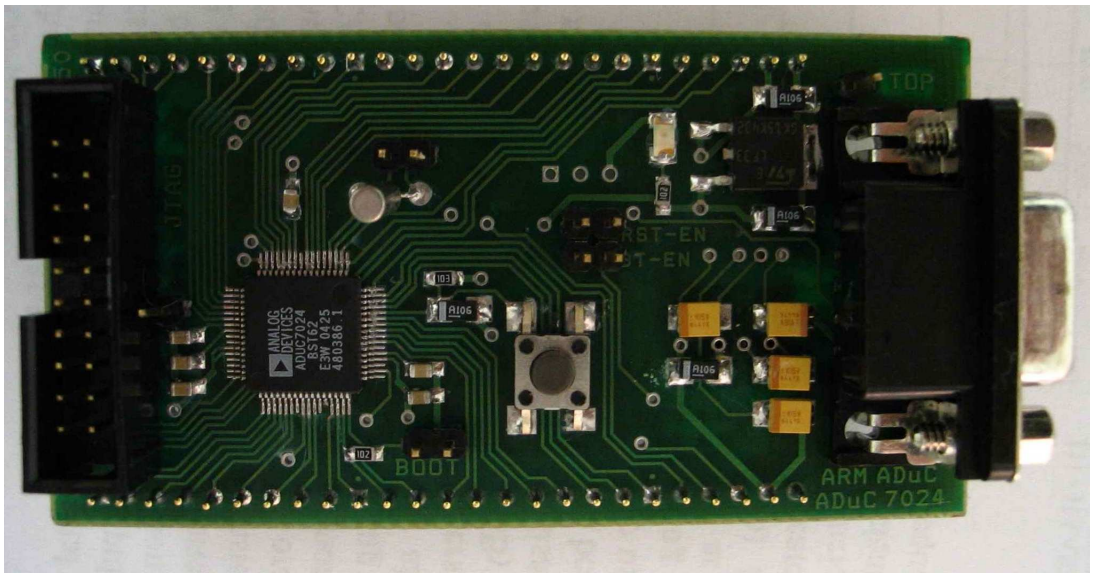
9.1.1 Procesor ADuC7024

Procesor má na čipu integrovány tyto funkční moduly:

- paměť programu a dat (62 kB/8 kB)
- 12bitový desetikanálový A/D převodník s referencí a teplotním čidlem na čipu,
- 12bitový dvoukanálový D/A převodník s žebříčkovou sítí,

- programovatelné pole PLA (Programable-Logic-Array) vhodné např. pro řízení CCD snímače,
- blok třífázové PWM.

Procesor jediný mezi ostatními současnými procesory nabízí také možnost zamknutí paměti programu „Memory protection“. Díky této vlastnosti se tento procesor stává vhodným pro komerčně vyvíjené aplikace.



Obrázek 9.2: Fotografie modulu s procesorem ADuC7024 a JTAG konektorem

Kompletní seznam součástek je v příloze D.2.

9.1.2 Napájecí zdroj

Procesor je napájen napětím 3,3 V a jádro napětím 2,5 V. Stabilizátor 2,5V části je integrován na čipu. Vyžaduje tedy pouze připojení napětí o velikosti 3,3 V. Pro stabilizaci napětí pro jádro je nutné pouze připojit kondenzátor proti zemi na svorku 21.

Pro stabilizaci napájecího napětí modulu byl zvolen pevně nastavený low-dropout stabilizátor LF33 v pouzdru DPAK, s maximálním úbytkem 0,45 V v propustném směru při proudu až 500 mA. Je vybaven interními ochranami proti tepelnému přetížení. Přesnost výstupního napětí je ±2 %, což je v povolených tolerancích napájení procesoru. Na vstupu napájení modulu je ochranná dioda D3 proti přepólování.

9.1.3 Zdroj hodin

Procesor může být taktován z interního oscilátoru 32,768 kHz s malou kmitočtovou stabilitou (±3 %). Proto je zapojení doplněno externím hodinkovým krystalem (32,768 kHz). Standardně je po resetu procesor taktován z interního oscilátoru. Pokud je třeba používat mnohem přesnější externí krystal, je nutné programově přepnout nastavení. Krystal je možné zkratovat a prostřednictvím jumperu J4 připojit vnější zdroj taktovacího kmitočtu. Procesor má na čipu násobičku a výsledný kmitočet jádra je 45 MHz.

9.2 Popis zapojení

Zapojení vychází z doporučení výrobce. Výsledné schéma je na obr. 9.3.

Již při návrhu desky se vědělo, že současný loader programu do procesoru ARMWSD neumí využívat signály sériové linky pro generaci signálu RESET a uvedení do boot módu. Přesto byla tato možnost zapojení ponechána a počítá se s tím, že se takto vylepšené verze

programu dočkáme buď od výrobce[16] (Analog Devices) nebo v rámci další studentské práce s tímto modulem. Tyto funkce je nutné povolit osazením jumperů J7 a J8 označenými na DPS „BT_EN“ a „RST_EN“.

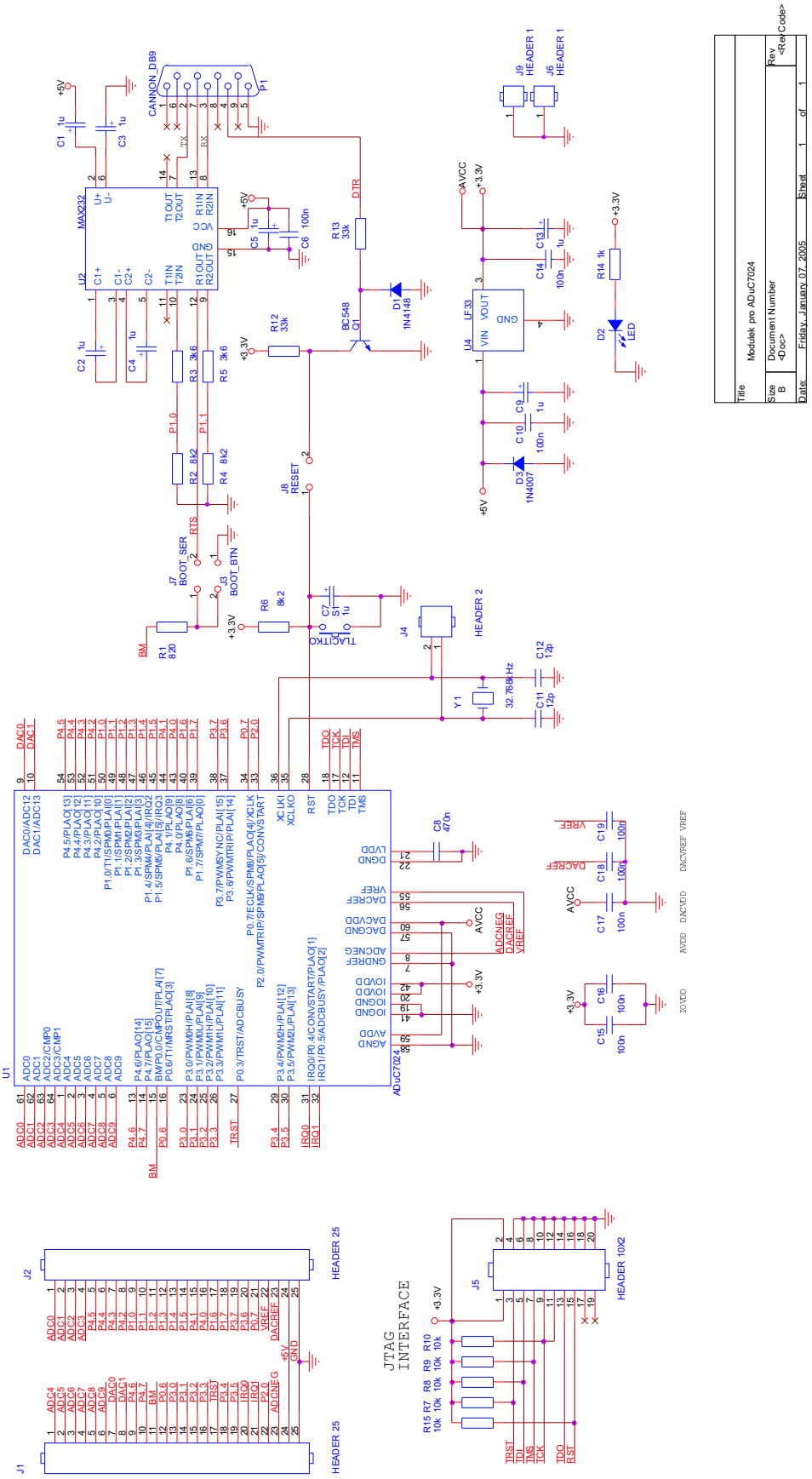
Deska byla vyrobena až během ledna 2005 a nebyl již dostatek času k ověření vlastností vyvinutého modulu. To může být předmětem další práce např. v rámci semestrální nebo diplomové práce.

9.3 Zjištené problémy

Jedinými doposud zjištěnými problémemy jsou:

- prohození signálů sériové linky TxD a RxD. K chybě došlo přehlédnutím v uživatelském manuálu procesoru. Náprava je možná překřížením vodičů mezi odpory R2, R3, R4 a R5.
- Nevhodné pouzdro pro hodinkový krystal (vhodnější by bylo pouzdro MTF32) a převodník sériové linky .

[16]Ing. Dohnal z firmy AMTEK přislíbil v tomto ohledu pomoc.



Obrázek 9.3: Schéma modulu s procesorem ADuC7024

Kapitola 10

Vývojové nástroje pro ARM

Pro vývoj programů pro procesory s jádrem ARM je možné volit z několika vývojových prostředí. Firma Keil nabízí grafické prostředí μVision, které v sobě obsahuje kompilátor assembleru a jazyka C, linker a debugger omezený velikostí testovaného programu. Tato možnost je vhodná pro vývoj malých programů, jejich odladění a je proto velmi vhodná pro začátečníky. Druhou možností je volba kompilátoru distribuovaného pod licencí GPL (General Public License). Obě prostředí jsou stručně popsána v následujím textu. Přestože byla většina práce tvořena pomocí volně šiřitelných nástrojů, začnu s popisem jednoduššího prostředí a tím je μVision. Dalším možným prostředím je např. kompilátor WinARM [28] nebo Cygnus.

10.1 Vývojové prostředí μVision

Prostředí je možné získat z webových stránek výrobce `www.keil.com`. Jeho výhodou je možnost spojení s testovací destičkou pomocí JTAG kabelu pomocí převodníku JTAG/USB označovaného *uLINK* (obr. 8.2) také od firmy Keil a jehož prostřednictvím je možné nahrát program do paměti procesoru a zpřístupnit jednotlivé periferní obvody na čipu. Prostředí je vybaveno simulátorem vhodným pro odladění jednoduchých programů. V instalačním balíčku je i velký počet příkladů. V prostředí je zpracovaná také kvalitní nápověda, které se doporučuji držet. Používaná verze je Keil μVision 3. Vzhledem k tomu, že prostředí je relativně nové a vzniká souběžně s vývojem mikrokontrolérů ARM, má stále několik nedostatků. Nejde ani tak o nedostatky z hlediska ovládání programu, které je velmi intuitivní, ale o ověření parametrů kompilace. Proto silně doporučuji provést kontrolu, zda je nastavena optimalizace překladu a jak se taková optimalizace chová. Proto je vhodné nejprve se seznámit s ARM assemblerem a poté začít pracovat v jazyce C. Prostředí je vybaveno konfiguračními předpisy pro většinu současně vyráběných procesorů.

10.2 Vývojové prostředí CYGWIN

Toto prostředí je vhodnější pro zkušenější programátory. Vývoj aplikací pro procesory ARM je přístupný prostřednictvím prostředí CYGWIN [30]. Jedná se o Unixové prostředí emulované pod Windows, ve kterém jsou nainstalované nástroje pro tvorbu aplikací pro procesory ARM.

10.2.1 Instalace prostředí CYGWIN do Windows

Vývojové nástroje byly instalovány do prostředí CYGWIN. To bylo zkomprimováno a uloženo na přiložené CD v archivu `cygwin.zip` (115MB). Instalace je rozdělena do dvou kroků:

- Archiv `cygwin.zip` je nutné rozbalit na požadované místo, např. do adresáře C:\Cygwin. Po rozbalení program zabere na disku cca 350 MB. V adresáři je obsaženo vlastní prostředí CYGWIN včetně zkompilovaných vývojových nástrojů jmenovaných dále.
- Před spuštěním je nutné ještě přidat soubor `cygwin98.reg` do registru Windows 98 nebo `cygwinXP.reg` do registru Windows XP.

Jednotlivé vývojové nástroje jsou nyní připraveny k použití. Jedná se hlavně o překladač asembleru `arm-elf-as`, kompilátor jazyka C `arm-elf-gcc` a linker `arm-elf-ld`.

10.2.2 Zpracování projektu prostředím CYGWIN

Zpracováním je v tomto případě myšleno vytvoření výsledného souboru `*.hex` nutného pro nahrání aplikace do mikrokontroléru. K tomuto cíli vedou dvě cesty.

- Spusťte dávku C:\cygwin\cygwin.bat[17] a podle potřeby použijte jednotlivé nástroje k sestavení výsledného objektu. Tato možnost je velmi zdlouhavá a pracná, neboť při každé byť sebemenší opravě v programu musíme celou proceduru znovu opakovat.
- V každém adresáři projektu vytvořte instalační předpis `Makefile` podle příkladu v příloze E a skript, který ho volá, podle následujícího příkladu:[18]

  ```
  cd /cygdrive/f/siplomka/Amclpc.649/sw/examples/dds
  make clean
  make all
  ```

  Tyto soubory nejsou využívány přímo, ale přes dávkový soubor make.bat, který obsahuje

  ```
  copy script c:\Cygwin\home\petr\script
  c:\Cygwin\bin\bash --login -c c:/cygwin/home/petr/script
  ```

  Tím se prostředí CYGWIN nespustí, ale pouze se využívají potřebné nástroje automaticky podle „předpisu“ vedeném v souboru makefile).

10.2.3 Adresářová struktura projektů

Adresářová struktura běžného projektu a knihovních modulů je podobná štábní kultuře při programování pod Unixem, a může vypadat např. takto:

```
\ SW \ examples \ blick
     \ libs.100 \ include
                \ ldi
                \ lib
                \ lst
                \ obj
                \ src \ lpc210x
                      \ startup
                      \ stdhal
```

10.2.4 Popis jednotlivých adresářů projektu

Adresář *src* obsahuje zdrojové kódy funkcí programových knihoven. Přeložené objekty jsou překladačem ukládány do adresáře *obj*.

Podadresář *src**lpc210x* obsahuje zdrojove kódy obsluhy jednotlivých periferií daného procesoru včetně obslužných rutin přerušení (*interrupt handlers*), např. soubory uarts.c, i2c.c nebo spi.c. Obsluhy přerušení (výjimek) jsou psané samozřejmě v assembleru (soubory vsr.s a isr.c).

Podadresář *src**startup* je nejdůležitější. Obsahuje zaváděcí soubor startup_ram16.s psaný v assembleru. V něm je provedeno základní nastavení proceosru před vlastním zavoláním funkce `main` instrukcí `bl main`. V této části dochází k inicializaci velikostí zásobníku pro různé pracovní módy procesoru, zapnutí násobičky kmitočtu PLL pro obvod hodin a nastavení rychlosti sběrnice VPB. Soubor *startup.s* odkazuje na linkovací skript (soubor ram16.ldi),

[17] pokud byla instalace provedena na disk C:

[18] kompilovaný projekt je v adresáři F:\diplomka.

ve kterém je mapa jednotlivých paměťových oblasti procesoru. Funkci `main()` lze předávat parametry prostřednictvím registru `r0`, určuje počet parametrů, a registr `r1` s ukazatelem na pole předávaných parametrů ukončené nulovým znakem.

Podadresář *src**stdhal* obsahuje základní funkce pro práci (zápis, čtení) se sériovou linkou.

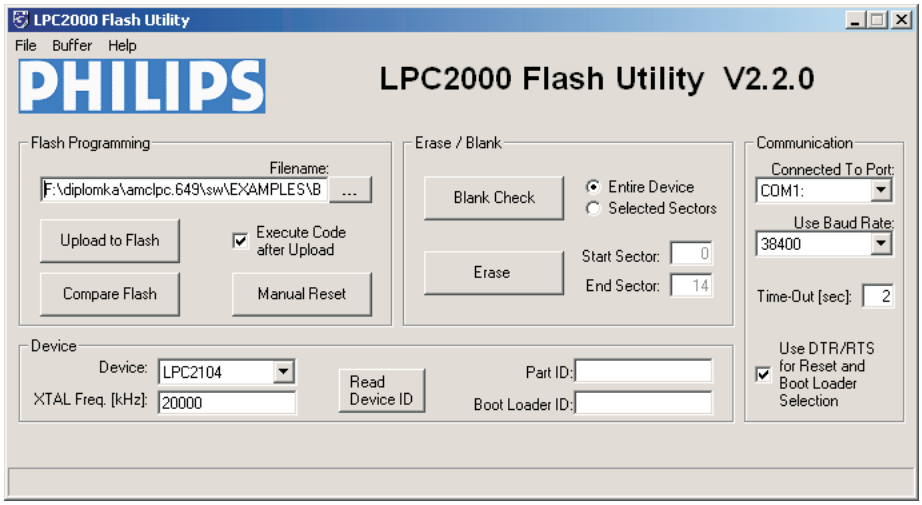
Adresář *include* obsahuje hlavičkové soubory. Pro správný překlad je důležité, aby absolutní adresa registru byla definována jako proměnná typu *volatile*. To překladači říká, že takto definovaná proměnná může být měněna nezávisle na chodu programu a při optimalizaci nečiní překladač žádné závěry o jejím obsahu (nesanží se optimalizovat operace přístupu na tuto adresu) [10]. Například definování registru `PINSEL0`[19] je provedeno příkazem:

```
#define PINSEL0 (*((volatile unsigned long *) 0xE002C000))
```

10.3 Programování procesorů Philips

Nahrání (*download*) výsledného *.hex souboru do procesoru Philips LPC2xxx je zajištěno programem **LPC210x FLASH ISP Utility**, který lze volně stahnout přímo z internetových stránek výrobce procesoru Philips [32]. Procesor má v sobě bootloader, proto je situace jednoduchá.

Tato verze umožňuje ovládání signálů reset a boot na modulu AMCLPC přes sériovou linku. To je provedeno zaškrtnutím možnosti *„Use DTR/RTS for Reset and Boot Loader Selecion“* v sekci „Communication“. Není třeba zadávat správný kmitočet procesoru v sekci *„Device“*, neboť v procesoru je po vstupu do boot módu automaticky rychlost komunikace detekována. Pokud je zaškrtnuta i možnost *„Execute code after Upload“*, dojde po nahrání programu do procesoru automaticky k resetu zařízení a aplikace, která byla nahrána se automaticky spustí. Při nahrávání větších aplikací je vhodné nastavit co nejvyšší rychlost nahrávání v menu *„Use Baud Rate:“*. Program, který má být nahrán do procesoru se nalistuje v menu *„Filename“*.



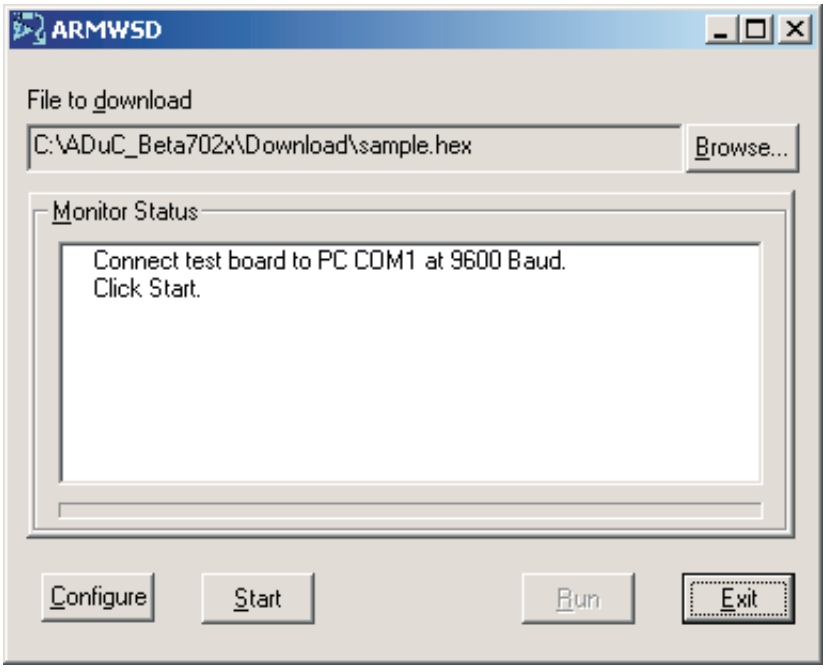
Obrázek 10.1: Programovací utilita pro ARM procesory Philips LPC2xxx

10.4 Programování procesorů Analog Devices

Nahrání (*download*) výsledného *.hex souboru do procesoru ADcC702x je zajištěno programem **ARMWSD**, který je volně dostupný ke stažení na FTP serveru výrobce Analog Devices `ftp://ftp.analog.com` [33]. Procesor má v sobě integrovaný bootloader. Po stisku tlačítka *„Start“* se čeká na reset procesoru do boot módu. To je provedeno stažením pinu P0.0 na

[19] Registr `PINSEL0` nastavuje alternativních funkce vstupně/výstupních pinů

zem. Tím dojde k vyslání identifikačního řetežce mikroprocesoru do programu na PC. Pak již probíhá download. Po nahrání je nutné odstranit jumperboot módu a provést reset procesoru. Tím dojde ke spuštění běhu aplikace.



Obrázek 10.2: Programovací utilita pro ARM procesory Analog Devices ADuC702x

Kapitola 11

Aplikační programy

11.1 Software — Aplikační knihovny pro AMCLPC

Jedním z požadavků na vývoj jednotky AMCLPC je vytvoření knihovny základních funkcí. Ty by měly převážně sloužit k rychlému otestování funkčnosti modulu jako celku a dále pak ukázkové aplikace s použitými převodníky. Testovány by měly být hlavně RTC a EEPROM.

Vznikla tak struktura projektu popsaném v kap. 10.2.3. V adresáři *sw\libs.100\src* jsou zdrojové soubory knihovních funkcí (verze 1.00), mezi které patří:

- obsluha I^2C a SPI rozhraní,
- inicializace PLL, MAM a sériové komunikace,
- rutiny přístupu k EEPROM a RTC.

Protože v době vývoje této části zadání jsem ještě nepracoval s vývojovým prostředím μVision, které umožňuje provést simulaci programu, bylo možné kontrolovat správnou činnost jednotlivých programů pouze blikáním LED diodami připojenými k výstupním pinům. Tato část práce byla velmi náročná a zdlouhavá.

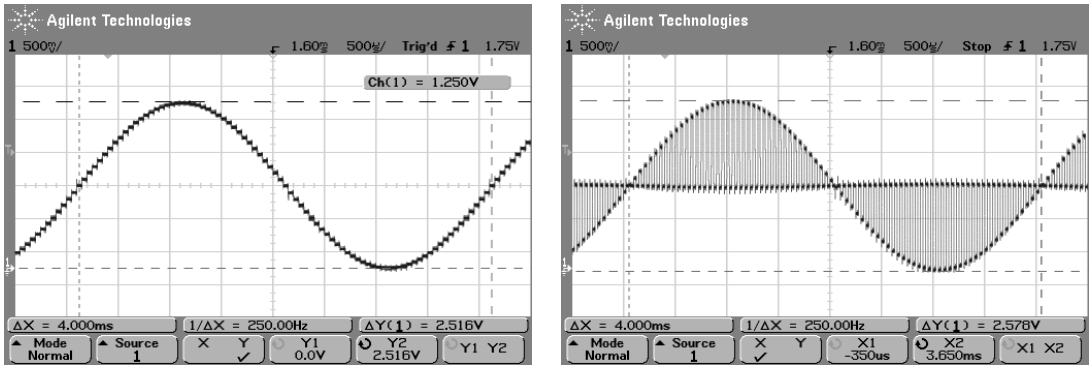
V adresáři *sw\examples* jsou příklady programů vytvořených v assembleru i v jazyce C. Tyto jsou určeny k inspiraci pro případné zájemce o aplikace procesorů ARM. V jednotlivých adresářích jsou nejen zdrojové kódy příkladů, ale i funkční přeložené *.hex soubory vhodné pro okamžitou otestování, někdy i ve více variantách. vhodné je začít programy s blikáním LED.

Projekty, které vznikly pro testování RTC a EEPROM jsou v adresářích *examples\i2c-rtc* a *examples\i2c-eeprom*. S jednotkou AMCLPC jsem testoval možné způsoby připojení A/D i D/A převodníků. Tyto projekty jsou v adresářích *examples\Spi_7731*, *examples\Spi_73311*, *examples\spi-tlc2578* a *examples\Spi_dac7614*. Většinou se jedná o jednoduchou ukázku komunikace s konkrétním převodníkem.

S převodníkem DAC7614 byla implementována generace průběhů metodou DDS. Projekt je v adresáři *examples\dds*. Program umožňuje nastavit výstupní frekvenci i vzorkovací frekvenci. Ta je v případě omezena na 50 kHz. Je možné vybrat prokládanou a neprokládanou variantu DDS, kdy mezi dva následující vzorky je vložen vzorek s nulovou amplitudou. Program je ovládán z terminálu prostřednictvím sériové linky. Výstupní průběhy pro prokládanou i neprokládanou metodu DDS generátoru výstupního kmitočtu $f = 250$ Hz jsou na obr. 11.1.

Pro AMCLPC vznikla první varianta programu pro měření parametrů periodických průběhů. Projekt je uložen v adresáři *examples\parametry* a není využíváno datových struktur. Proto není její použití vhodné. Později byl vyvinut více komplexní program pro kit MCB2100.

Při metodách generování průběhů pomocí AMCLPC a externího D/A převodníku DAC7614 byl odhalen poměrně velký jitter hrany signálu $\overline{LOADDACS}$, která přepisuje hodnoty ze záchytných registrů převodníku na jeho výstup. Protože v dostupné literatuře k procesoru [22] není toto chování popsáno, bylo nutné zjistit jeho příčinu a pokusit se ho vysvětlit. Popis měření a vysvětlení je v kap. 11.5.



(a) Neprokládaná DDS (b) Prokládaná DDS

Obrázek 11.1: Generování průběhu metodou DDS

11.2 Software — Měření parametrů signálů s MCB2100, implementace algoritmů

S tímto kitem byl vytvořen program, který využívá integrovaného A/D převodníku a měří parametry periodických signálů. Program komunikuje s uživatelem prostřednictvím terminálu připojeného pomocí sériové linky. V případě nasazení vytvořeného programu do modulu, který komunikuje s nadřazeným systémem pomocí jiné sběrnice než RS–232 je nutné pouze přepsat obsluhu vyslání a příjmu znaků.

Vstupní průběh je vzorkován s volitelnou vzorkovací frekvencí tak, aby byl odebrán dostatečný počet vzorků. Teoreticky stačí navzorkovat více než jednu periodu vstupního signálu. V tomto případě, kvůli použitému algoritmu vyhledání periody ve vstupním signálu, je nutné zvolit vzorkovací frekvence tak, aby došlo k navzorkování zhruba více než 1,5 periody vstupního průběhu. Vhodnou alternativou této metody by bylo vzorkování maximální dosažitelnou frekvencí s následnou decimací vzorků. Tím by se díky převzorkování snížila úroveň šumu v měřeném pásmu.

Program je tvořen formou stavového automatu, jak je naznačeno ve vývojovém diagramu na obr. 11.2. Hlavním úkolem práce bylo demonstrovat výpočetní možnosti procesorů ARM, proto jsem se přiklonil k metodě výpočtů off-line popsané v rozboru práce.

Jednotlivé algoritmy pro měření parametrů jsou implementovány v programovacím jazyce C podle těchto vztahů:

- perioda podle vztahu (4.3) zpřesněného lineární interpolací vztahem (4.1),
- fázový rozdíl podle vztahu (4.5),
- efektivní hodnota podle (4.11),
- výkon podle (4.17).

Dosažené výsledky jsou popsány v kapitole 13.

11.3 SW – ADuC7024

Jak bylo uvedeno, vyrobená deska plošných spojů (DPS) byla dodána až v lednu 2005. Proto jsem stihl modul pouze oživit s použitím ukázkových příkladů pro tento typ procesorů pro prostředí μVision. Tyto programy jsou ve formě projektů volně ke stažení z webových stránek `http://www.keil.com`.

11.4 Výkonové možnosti LPC2129

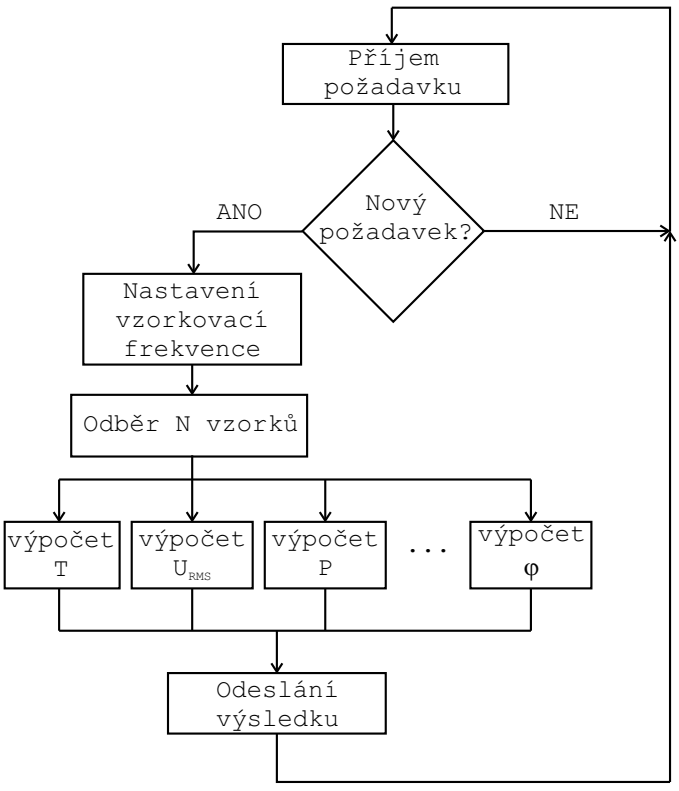
Chování procesoru a naměřené parametry (uvedeno dále v textu) bude pravděpodobně stejné pro celou rodinu procesorů Philips LPC2xxx. Měření bylo provedeno na procesorech LPC2129

osazených na vývojovém kitu MCB2100. U procesorů jiných výrobců bude metoda urychlení vyčítání z paměti realizována jinak. Pro jednotlivé testy byly napsány testovací programy v asembleru.

Nejprve je nutné představit architekturu jednotky MAM (Memory acceleration Module). Paměť flash je rozdělena na dvě nezávisle přístupné banky [22], každá se dvěma vyrovnávacími registry označenými jako Prefetch Buffer (PB) a Branch Trail Buffer (BTB) o celkové velikosti 128 kByte (pro LPC210x), což jsou 4 kByte kódu (jedna instrukce má 4 Byte). Obě banky jsou propojeny metodou interleaving a během vyčítání se mezi nimi přepíná. Obě banky jsou vzájemně transparentní a tvoří souvislý blok paměti. Režim nastavení jednotky MAM je dán kompromisem mezi rychlostí běhu programu a predikovatelností časování jednotlivých událostí.

Jednotka MAM má na starosti, aby nedošlo k zastavení pipeliningu a aby byla další vykonávaná instrukce připravena v záchytných registrech. Instrukce se z paměti programu načítají po 128 bitech, tedy čtyři 32bitové ARM instrukce nebo osm Thumb instrukcí najednou. Pokud dojde k situaci nepodmíněného skoku (instrukce `B`) a na místo skoku byl již program nucen alespoň jednou skočit, je pravděpodobné, že první instrukce z cílové adresy se již nalézá v BTB. Není-li tomu tak, je nutný jeden přístup do paměti kódu FLASH a načtení souvislého bloku čtyř instrukcí. Proces plnění pipeliningu tak znovu začne. Celkem je tak v obou bufferech uchováno 8 ARM instrukcí. Toto neplatí pro data.

Jednotka MAM může pracovat ve třech režimech, kdy je zrychlené vyčítání úplně vypnuto, částečně povoleno a úplně povoleno. V prvním případě nejsou záchytné registry použity a každé čtení instrukce vyvolá přístup do Flash paměti. Ve druhém případě je funkce zrychleného vyčítání částečně povolena a při sekvenčním přístupu do paměti je povoleno prefetch instrukce. Při nesekvenční přístupu skoku (branch) se vyžaduje přístup do paměti. V posledním případě je funkce funkce MAM plně povolena a data i instrukce jsou čteny ze zá-



Obrázek 11.2: Vývojový diagram programu pro měření parametrů signálů

chytných registrů. Tím se minimalizuje počet přístupů do paměti FLASH, která je z podstaty pomalejší než RAM.

Jednotka MAM je připojena k lokální sběrnici procesoru AHB. Po resetu procesoru je MAM akcelerace vypnuta a v případě jejího zapnutí je default hodnota MAMTIM = 7, takže pro přístup do paměti programu a načtení instrukce (*fetch*) je třeba sedmi hodinových cyklů.

11.4.1 Vliv registru MAMTIM

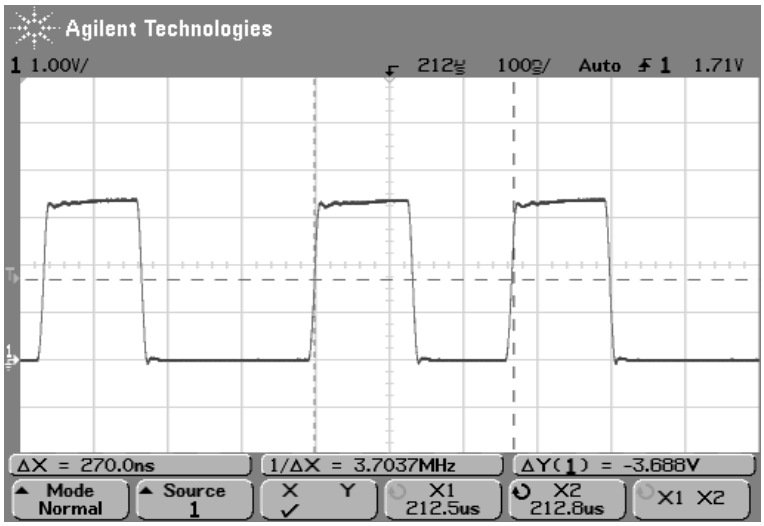
Nejprve byl testován vliv počtu wait stavů na rychlost přístupů do paměti programu. Počet wait stavů je dán hodnotou v registru MAMTIM. K testování byl použit následující program, kde v `r5` je adresa registru IOSET, v `r6` adresa IOCLR a v `r7` hodnota zapisovaná na výstupní piny, který střídavě nastavuje a shazuje výstupní pin. Procesor má nastavení: f=10 MHz, PLL vypnuta (hodinový cyklus sběrnice 100 ns) a MAMCR = 0, což znamená, že rychlost periférií (VPB bus) je stejná jako rychlost jádra a jednotka MAM je vypnuta. Testování spočívalo v měření doby trvání HI a LO úrovně výstupního pinu na nastavení MAMTIM a VPBDIV.

```
cyklus:
    str r7, [r5]    /* nastav vystupni pin do urovne HI*/
    str r7, [r6]    /* nuluj vystupni pin (do urovne LO) */
    b   cyklus      /* skok zpatky na zacatek programu */
```

Změřené doby tvrání jsou naznačeny v tab. 11.1. Údaje jsou počty cyklů (1 cykl = 100 ns) a možný výstupní průběh je zobrazen na obr. 11.3.

VPBDIV	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4	4
MAMTIM	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	6
t_{HI}	7	8	9	10	11	10	10	12	12	14	16	16	16	20	20	20
t_{LO}	10	12	14	16	18	12	14	16	18	20	20	20	24	24	28	28

Tabulka 11.1: Doba trvání (v počtu cyklů) HI a LO úrovně



Obrázek 11.3: Výstupní průběh blikání pinem

Pro nastavení VPBDIV = 1 je časování chování při nastavování pinu částečně predikovatelné a je zřetelně vidět vliv „wait stavů“ (reprezentováno MAMTIM registrem) při přístupu do paměti programu. S rostoucím počtem wait stavů roste doba na jednu instrukci právě

o dobu jednoho cyklu. Např. pro MAMTIM = 4 (platí, že rychlost APB je stejná jako AHB) je počet hodinových cyklů v úrovni HI o jedničku větší a počet cyklů v LO o dvě větší než pro MAMTIM = 3, neboť v programu dochází v úrovni LO k načtení instrukce skoku `B` navíc. Pro jiná nastavení registru VPBDIV nelze zřetelně vysledovat vliv počtu wait stavů.

Pokud je MAM funkce povolena úplně (tedy MAMCR = 2), nemá nastavení registru MAMTIM vliv a program běží efektivní rychlostí 0 wait stavů (zero wait states). Takto ideální chování je pouze v tomto příkladu, kde se neustále skáče na stejnou adresu a instrukce za ní následující jsou již načteny v BTB registru. V případě reálného programu, který skáče na mnoho míst, bude i v případě plně zapnuté MAM jednotky docházet k zastavení pipeliningu. Z toho plyne poznatek, že pro praktické použití musí být MAM jednotka plně zapnuta, aby se možnost přerušení pipeliningu pokud možno minimalizovala. Režimy 0 a 1 nejsou vhodné pro aplikace, kde je časování ke studiu vlastností pipeliningu.

Při běhu programu z RAM samozřejmě nezávisí na nastavení MAMTIM registru, neboť ten určuje počet wait cyklů pro přístup do FLASH. To by se projevilo pouze při přístupu k proměnným uložených ve FLASH. Na rychlost běhu programu také nemá vliv použití instrukcí pro 8 nebo 16bitový přístup (STRB a STRH) namísto STR. To je logické, sběrnice jsou 32bitové a proto nemá význam, aby komunikace byla po menších kvantech.

11.4.2 Měření přístupových rychlostí uvnitř procesoru

Další parametry, který nsjsou v literatuře uvedeny a jsou velmi důležitý pro použití procesoru v rychlých aplikacích, jsou maximální přenosové rychlosti s pamětí a periférierni, kterých lze s jádrem ARM dosáhnout. V tabulce 11.2 jsou uvedeny dosažené přístupové rychlosti měřené na procesoru LPC2129. Díky stejné vnitřní architektuře lze předpokládat stejné parametry u všech procesorů řady Philips LPC2xxx.

Měření bylo provedeno pro velký počet zápisů/čtení do/z požadovaného místa a následné vydělení počtem zápisů/čtení. Začátek a konec přístupů byl signalizován bliknutím výstupního pinu. Použitím velkého počtu zápisu se eliminuje vliv zpoždění vznikajícího při vlastní signalizaci (nastavování pinů, které je velmi omezující). Procesor byl nastaven na maximální výkon, kmitočet jádra 60 MHz, MAMTIM = 2, VPBDIV = 1.

	RAM				FLASH				Periferie			
	STR		LDR		STR[20]		LDR		STR		LDR	
VPBDIV	μs	MHz	μs	MHz	μs	MHz	μs	MHz	μs	MHz	μs	MHz
1	3,45	**29**	5,11	**19,6**	-	-	5,14	19,5	11,8	**8,5**	13,5	**7,4**
2	3,5	28,5	5,17	19,3	-	-	5,14	19,5	16,8	5,9	16,8	5,9
4	3,59	27,9	5,26	19	-	-	8,56	11,7[21]	26,9	3,7	26,9	3,7

Tabulka 11.2: Rychlost přístupu do paměti RAM, FLASH a periférií umístěných na čipu procesoru, procesor na max. výkon

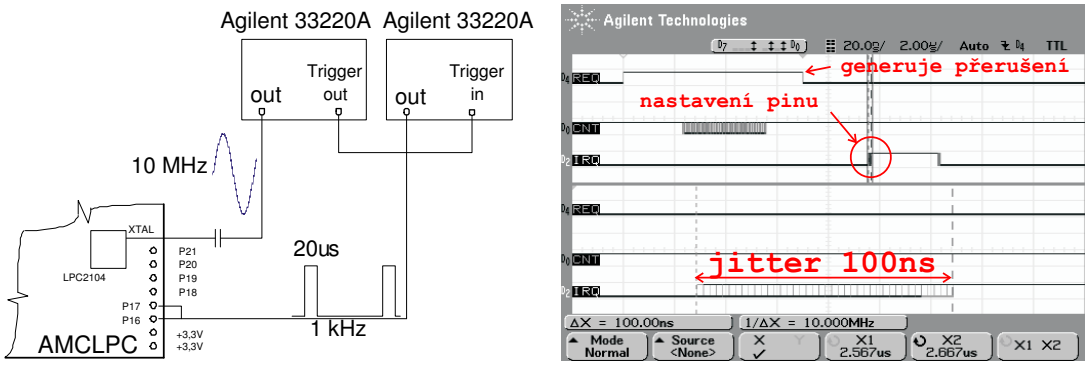
Z tabulky 11.2 je patrné, že maximální dosažitelné rychlosti jsou 7,9 MHz z periférií (např. z pinů), a 19 MHz z paměti. Do paměti FLASH nelze zapisovat (instrukcí), to lze pouze pomocí ISP příkazů. Je zajímavé, že nastavení VPBDIV má vliv i na rychlost paměťové sběrnice. Měřeno na procesoru LPC2129. Při těchto rychlostech však procesor nedokáže dělat nic jiného.

11.5 Zjištěný jitter při generování průběhů

Jitter byl pozorován v aplikaci generování průběhů s externím převodníkem, kap. 7.5.3.

11.5.1 Metoda měření jitteru

Pro měření byly použity dva generátory Agilent 33220A zapojené podle obr. 11.2 a). Jeden generuje hodiny, sinus, a taktuje procesor na 10 MHz. Druhý generuje pulzy přerušení a je nastaven na cca 1 kHz, 0 – 3V, šířka HI úrovně cca 20 μs (musí se pokaždé nastavit podle testovaného programu). Generátor přerušení je synchronizován s generátorem hodin (mód Gated burst). Tím je zajištěno, že přerušení je generováno synchronně s hodinovým cyklem. Synchronizace generátorů je provedena propojením generátoru hodin BNC konektorem vzadu se vstupem na generátor impulsů umístěným vpředu. Chování bylo měřeno osciloskopem pomocí datových vstupů.



(a) Zapojení při měření (b) Zjištěný jitter

Obrázek 11.4: Zapojení pro měření jitteru

Signál přerušení je kromě pinu P0.16 přiveden i na P0.17 pro testování stavu signálu, protože se nepodařilo testovat signál přímo na P0.16, na kterém byla nastavena alternativní funkce EXTINT0.

V oscilogramu je na prvním kanálu signál generátoru, který vyvolává přerušení. Druhý kanál slouží k signalizaci (pulz o úrovni L) začátku bloku instrukcí NOP (v signálu je jitter, proto je tolik rozmazán). Třetí kanál v detailu ve spodní části oscilogramu ukazuje jitter vystavení pinu v rutině přerušení. Nastavení výstupního průběhu generátoru 3 kHz, doba pulzu 8 μs, plnění (duty) 0.8 %. Nastavení procesoru MAMTIM=3, CPU=20MHz, VPBDIV=1.

11.5.2 Testovací program, SW

Program pracuje jako jednoduchý stavový automat, který čeká na pinu P0.17 na náběžnou hranu. Následující sestupná hrana vyvolá přerušení, které příjde v okamžiku bloku instrukcí NOP. Procesor začne vykonávat obsluhu přerušení od adresy 0x18. Na začátku rutiny jsou instrukce NOP následované nastavením pinu do úrovně HI. Tím je simulována situace, kdy je např. generován spouštěcí signál pro vzorkování externím převodníkem. Pozorovaný jitter tohoto signálu je na obr. 11.2 b). Testovací program je uveden v příloze C.

Následuje vysvětlení vlivu nastavení jednotlivých zůčastněných jednotek a pravděpodobné příčiny chování.

Dospěl jsem ke zjištění, že pokud je na začátku rutiny přerušení provedeno několik instrukcí NOP, dochází k výskytu jitteru při následném nastavování pinu s mnohem menší pravděpodobností. Důvod je ten, že při generování externího přerušení se tento signál musí do procesoru dostat přes VPB sběrnici, která je tím na několik strojových cyklů zablokovaná. Pokud je požadováno následné okamžité nastavení pinu (nebo jakákoliv operace s registry v periféríích), nelze ho provést vždy se stejným časováním, a tím dochází k jitteru. Při běhu

programu z FLASH stačí vložit relativně málo NOPů, ale při běhu z RAM je jich mnohem více, neboť přístupová doba do RAM je mnohem kratší.

Měřením detailního chování procesoru jsem dospěl k závěru, že bez potřebných konkrétních informací o vnitřní struktuře implementovaných částí procesoru je velmi těžké, ne-li nemožné, vysledovat a kvalitně popsat chování vnitřních sběrnic. I pro nejjednodušší aplikace typu nastavení pinu je to velmi těžké. Informace, které by bylo potřeba znát, a které výrobce neposkytuje, jsou především jaký typ AHB sběrnice je použit a zda-li je architektura jednovrstvá (je to velmi pravděpodobné, jsou tu pouze dva mastery, CPU a ETM) nebo vícevrstvá.

Kapitola 12

Měření interního A/D převodníku v procesoru LPC2129

Kit MCB2100 s procesorem LPC2129 je osazen 10bitovým převodníkem typu SAR (succcessive approximation register — převodník s postupnou aproximací). Vstupní rozsah je 0 – 3,3 V (v manuálu špatně uvedena hodnota 0 – 3 V), doba převodu při maximálním rozlišení 2,44 μs a má 4 multiplexované kanály. Velikost vzorkovací kapacity je 1 pF. Práce s ním však není jednoduchá, neboť je nutné podrobit ovládací SW 6 errata opravám. V začátku práce na diplomové práci nebyla k dispozici kvalitní dokumentace interního AD převodníku[22]. Z tohoto důvodu bylo třeba odměřit některé základní parametry interního AD převodníku.

Převodník je řízen dvěma registry, ADCR (A/D Control Register) a ADDR (A/D Data Register). Převod může být startován „hardwarově“ externím přerušením EINT0, sestupnou nebo vzestupnou hranou CAP0.0 nebo některými MAT výstupy obvodu CAP/COM, nebo přímo programově. Je možný i režim *BURST*, který automaticky přepíná nastavené kanály a realizuje tak funkci *autosweep*. Je také možné nastavit nižší rozlišení převodníku, a tak zvýšit maximální vzorkovací frekvenci. Pro maximální přesnost 10 bitů je třeba 11 hodinových cyklů, např. pro 8 bitů pouze 9 cyklů, vždy tedy o jeden více (nutné pro vlastní odběr vzorku). Nejnižší dosažitelné rozlišení je 3 bity, při kterém dosahuje vzorkovací frekvence 1,125 MHz. Maximální hodinová frekvence převodníku je 4,5 MHz, lze ji nastavit pomocí děličky VPB sběrnice nebo vlastní děličky v A/D převodníku. Pro aplikace s požadovanou minimální spotřebou lze napájení převodníku vypnout v registru ADCR.

V registru ADDR je výsledek převodu, platný po nastavení MSB bitu DONE. V dalších bitech je zakódováno číslo kanálu, ze kterého byl vzorek odebrán. Je dobré provést kontrolu, že data jsou z požadovaného kanálu, neboť pro samotný převodník již bylo vydáno 6 errata oprav! Je velmí vhodné takovou kontrolu provést, buď v debug módu při ladění programu aplikace, nebo přímo v SW aplikace.

12.1 Vstupní proud převodníku

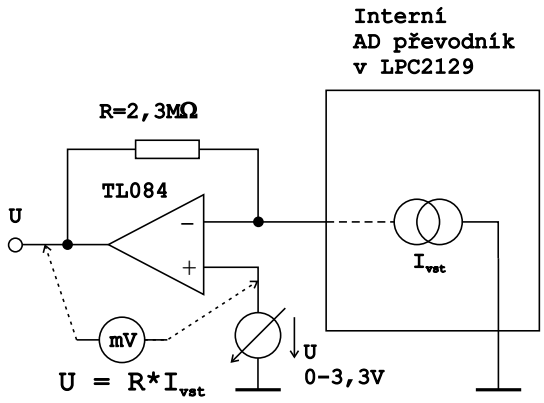
Vstupní vzorkovací obvod v převodníku se chová jako zdroj proudu. Jeho velikost je 1 pF podle dokumentace k procesorům řady LPC213x, ve kterých je osazen stejný interní převodník. Tato hodnota není uvedena v dokumentaci k procesoru řady LPC2129.

Pro měření byl použit operační zesilovač TL084 jako převodník proudu na napětí. Velikost výstupního napětí ovlivňují i vstupní proudy OZ (cca 40 pA) a vstupní napěťová nesymetrie. Po odpojení vstupu A/D převodníku a změření výstupního napětí a jeho odečtením od měřených hodnot lze vliv těchto parazitních jevů eliminovat. Zpětnovazební odpor byl $2,3$ MΩ.

Jinou možností měření vstupního proudu je metoda, při které je kondenzátor s mnohem větší kapacitou než předpokládaná velikost vzorkovacího kondenzátoru, např. 1000× větší než C_{sample} nabíjený vytékajícím proudem ze vstupu A/D převodníku. Pak je měřena rychlost nárůstu výstupního napětí sledovače, tvořeného OZ. Velikost vstupního proudu I_{vst} by se pak vypočetla jako $I_{vst} = C_x\,\Delta U/\Delta t$, kde ΔU je přírůstek výstupního napětí za dobu Δt.

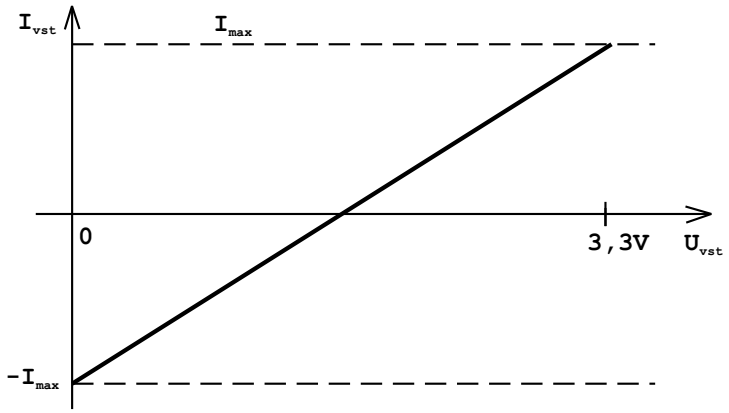
Průběh velikosti vstupního proudu je překvapující, neboť podle očekávání by měl vypadat

[22]Tento nedostatek není vyřešen dodnes, kvalitní dokumentace jsme se od firmy Philips nedočkali.



Obrázek 12.1: Zapojení obvodu při měření vstupního proudu

jako na obr. 12.2. Skutečná, naměřená velikost vstupního proudu je však na obr. 12.3. Příčina takového chování by mohla být ve vnitřní struktuře převodníku při přepínání jednotlivých proudových zdrojů podle jednotlivých nastavených výstupních bitů D/A převodníku v SAR struktuře převodníku. Měření je opakovatelné. Schéma zapojení pro měření vstupního proudu převodníku je naznačeno na obr. 12.1.

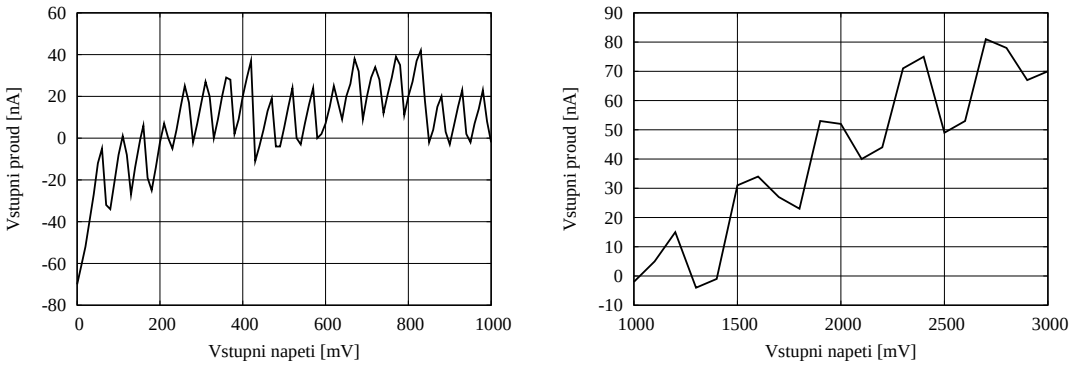


Obrázek 12.2: Teoretický průběh vstupního proudu u SAR A/D převodníku

Průběh naměřeného vstupního proudu je rozdělen na dvě části, podle velikosti vstupního napětí, a je uveden na obr. 12.3. Skutečně změřený průběh vstupního proudu je na obr. 12.3. Pro vstupní napětí 0 – 1 V byl měřen s krokem 10 mV a v rozsahu 1 – 3 V s krokem 100 mV. Průběh na obr. 12.1 b) **je pouze informativní**, protože se jednalo o diskrétní měření, přestože bylo provedené na plynule regulovatelném zdroji a současně měřeno voltmetrem HP34401A. Může se zdát, že se jedná o jistou formu obrazového aliasingu[23], ale je nutno poznamenat, že výstupní napětí měřicího řetězce (dle obr. 12.1) kolísalo podle naznačeného průběhu. **Oba grafy nejsou** měřeny se stejným krokem vstupního napětí.

Technická dokumentace o tomto chování toto chování nezmiňuje . S vysvětlením nejen tohoto, ale i dalších nedostatků v dokumentaci, jsem se několikrát obrátil na technickou podporu Philips, bohužel s neuspokojivým výsledkem.

[23]Nejedná se o ustálené slovní spojení. Krok (vzorkovací interval) vstupního napětí 100 mV nepostihuje nejrychlejší změnu průběhu a graf pak zobrazuje průměrnou hodnotu



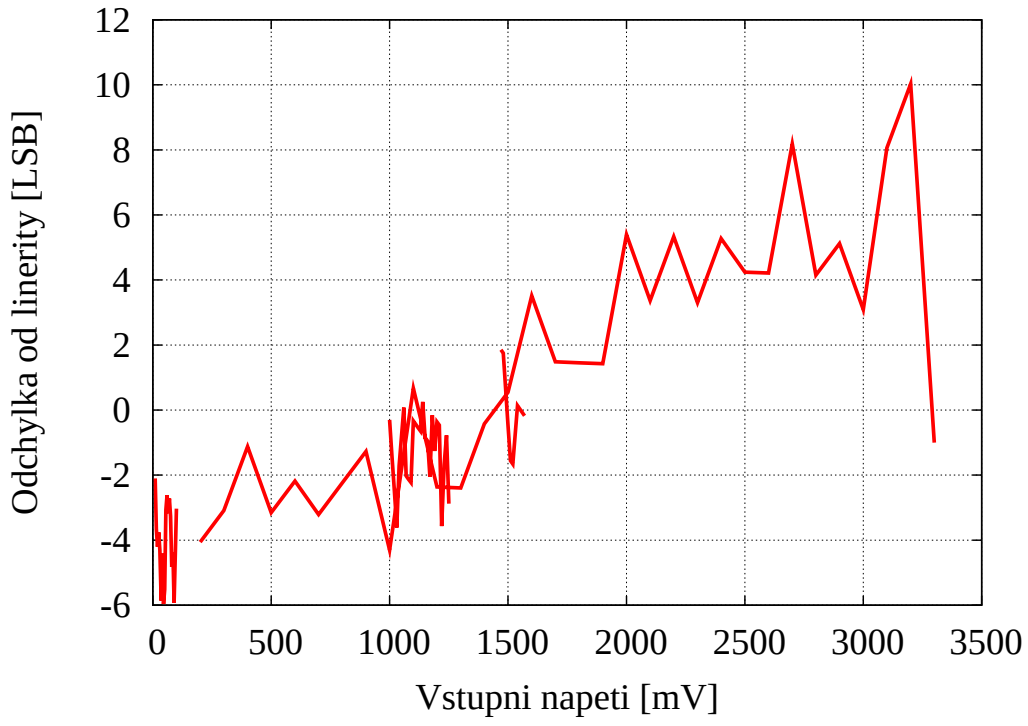
(a) Vstupní napětí 0 – 1 V, **krok 10 mV** (b) Vstupní napětí 1 – 3,3 V, **krok 100 mV**

Obrázek 12.3: Naměřený průběh vstupního proudu interního SAR A/D převodníku v LPC2129 pro dva rozsahy vstupního napětí

12.2 Statická převodní charakteristika

Statická převodní charakteristika je důležitá charakteristika A/D převodníku. Z ní je možné určit např. nelinearitu obvodu.

Zdroj stejnosměrného napětí byl připojen ke vstupu přes ochranný odpor 470 Ω. Přestože propojení bylo provedeno koaxiálním kabelem, bylo měření zatíženo šumem. Proto byly odměry průměrovány ze 4 vzorků. Možnou příčinou by mohl být nevhodný návrh plošného spoje a propojení analogových zemí. V dokumentaci ke kitu MCB2100, na kterém bylo měření provedeno, bohužel není naznačen layout PCB a jedinou dostupnou dokumentací je schéma kitu.



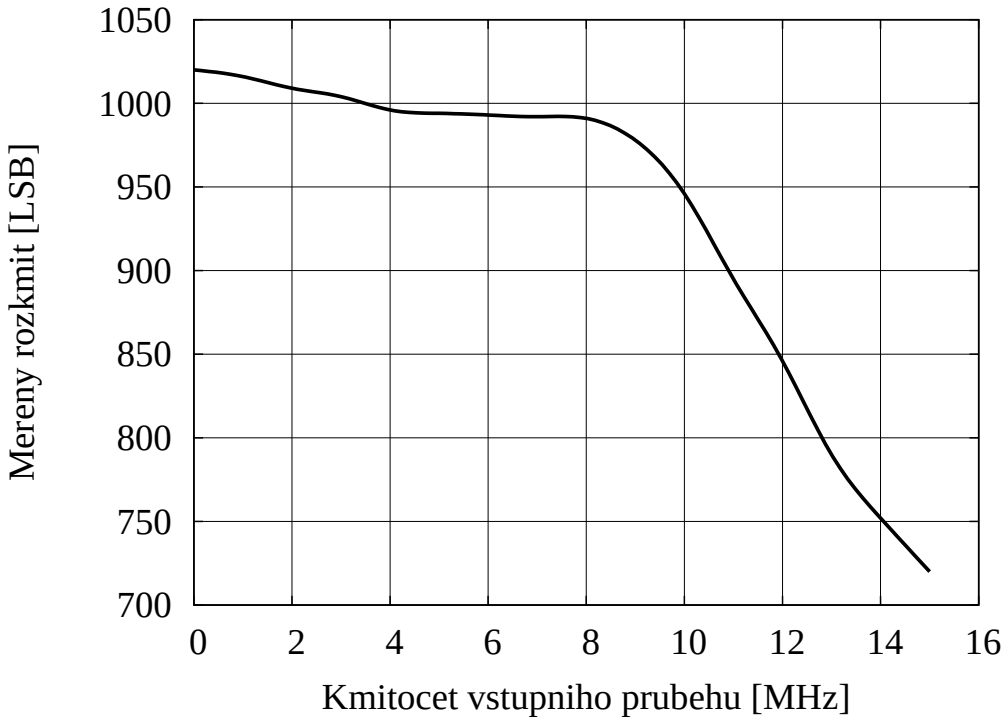
Obrázek 12.4: Statická převodní charakteristika

12.3 Šířka pásma

Šířka pásma je důležitý převodníku, který udává jeho dynamický rozsah. Je to frekvence vstupního signálu, při které poklesne amplituda rekonstuovaného signálu o -3 dB oproti amplitudě nízkofrekvenčního signálu.

Na vstup byl přiveden signál o konstantním rozkmitu $3,3$ V a proměnné frekvenci 10 kHz – 15 MHz. Změřenou amplitudu byla poslána po sériové lince do PC, který ji zobrazoval prostřednictvím jednoduchého programu pomocí protokolu NRC, rozšířeného pro 16bitová data.

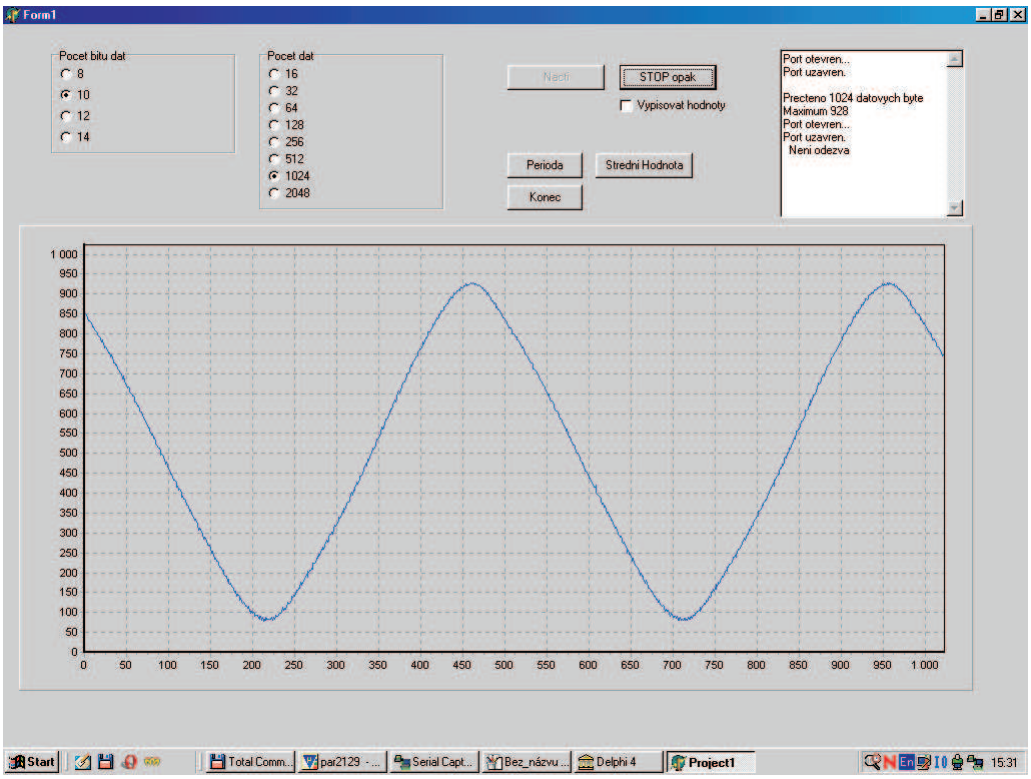
Maximální vzorkovací kmitočet je řádově 400 kHz. Proto docházelo k aliasingu (stroboskopické vzorkování), což ovšem nebylo při tomto měření na škodu. Zjištěná šířka pásma je cca 15MHz, s prvním zlomovým kmitočtem již na 9 MHz (může jít o vliv vstupní kapacity pinu procesoru), jak je vidět na obr. 12.5.



Obrázek 12.5: Šířka pásma vstupu A/D převodníku

Zdroj signálu byl původně použit generátor Tektronix AFG320, ale poté, co jsem u něj zjistil nevyhovující výrazný pokles aplitudy při vyšších frekvencích než 7 MHz, použil jsem generátor Hewlett-Packard a V-metr Hewlett-Packard HP34401A.

Ukázkou dobré šířky pásma je obrázek programu, který jsem pro tyto účely vytvořil v Delphi, a který demonstruje amplitudu stroboskopicky vzorkovaného signálu o kmitočtu 12,467 MHz s rozkmitem 3,3 V, tedy 1023 LSB. Na ose x jsou čísla jednotlivých odebraných vzorků a na ose y jejich hodnoty. Program je rozšířením zobrazovacího programu NRC, který pracuje pouze s 8bitovými daty. Tento umožňuje práci s 10-, 12- a 16bitovými vzorky dat přenášené sériovou linkou.



Obrázek 12.6: Šířka pásma vstupu A/D převodníku, vzniká aliasing

Kapitola 13

Naměřené parametry

Měření parametrů signálů bylo realizované vývojovým kitem MCB2100 firmy Keil. Pro něj byl napsán program typu stavového automatu popsaného v následující kapitole. V dalších jsou pak uvedeny dosažené naměřené výsledky.

13.1 Implementace algoritmů pro měření parametrů signálu

V jednotce měření a sběru dat bylo použito asynchronní vzorkování. V tabulce 13.1 jsou uvedeny časy potřebné pro provedení výpočtu jednotlivých parametrů. Pro výpočty je vždy vzorkováno cca 1024 vzorků s takovou vzorkovací frekvencí, že po nalezení periody zbývá na periodu cca 500 vzorků. Doby zde uvedené jsou pro měření bez průměrování.

	Perioda	**Střední hodn.**	**Efektivní hodn.**	**Výkon**
Čas [μs]	680 μs	700 μs	820 μs	1,2 ms

Tabulka 13.1: Doba trvání výpočtu měřených parametrů

Doby trvání výpočtu parametrů signálu jsou kratší než 1 ms, kromě výpočtu výkonu. Důvod je ten, že při výpočtu výkonu je nutné načíst data ze dvou datových struktur (vzorky napětí a proudu), a tím se doba výpočtu prodlouží.

Při implementaci jednotlivých algoritmů výpočtu je použita metoda off-line, neboť z uvedených dob trvání je zřejmé, že doba výpočtu je dostačující. Většina v praxi měřených signálů bude pravděpodobně o síťovém kmitočtu s dobou periody 20 ms. Vzhledek k době periody je doba výpočtu zanedbatelná.

Vývojový diagram jednotky měření a sběru dat je uveden na obr. 11.2.

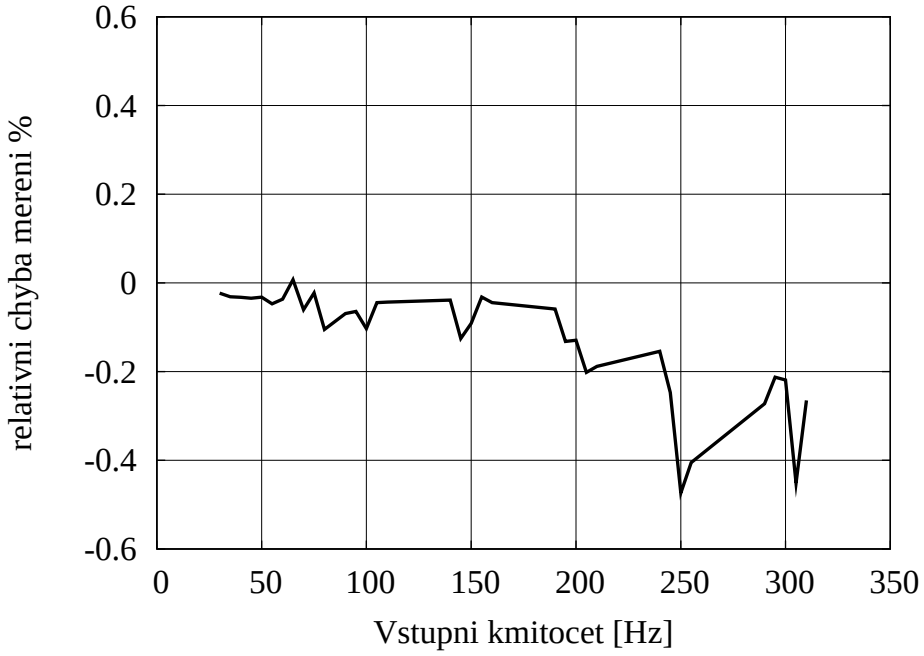
13.2 Výsledky měření periody

Měření periody bylo provedeno podle vzorce (4.1), který počítá hodnotu periody signálu a pro zpřesnění výpočtu provádí lineární interpolaci mezi odebranými vzorky. Funkce, která počítá periodu současně ukládá do datové struktury navzorkovaného kanalu jednotlivé průchody referenční úrovní, které jsou dále použity pro výpočet dalších parametrů signálu, integrujících přes délku periody. Průměrná dosažená velikost chyby je $-0,2$ %.

Vstupní průběh byl generován přístrojem AFG320. Z průběhu chyby vztažené k rozsahu měření na obr. 13.2 je patrné nárůst velikosti chyby při vyšší frekvenci. To je způsobeno tím, že měření bylo prováděno s pevně nastavenou vzorkovací frekvencí a proto pro vyšší vstupní kmitočty vychází menší počet vzorků na periodu a tím dochází k nárůstu chyby při výpočtu.

13.3 Výsledky měření střední hodnoty

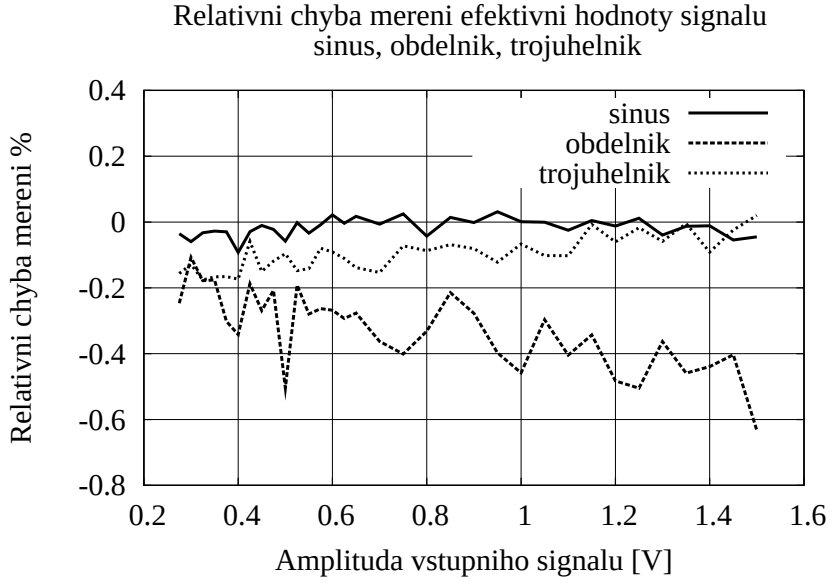
Střední hodnota je počítána podle vzorce (4.9). Průchody referenční úrovní jsou nalezeny funkcí pro výpočet periody. Neuvádím graf, neboť chyba byla téměř konstantní a maximálně dosahovala v absolutní velikosti 2 LSB, což je pro 10bitový A/D převodník cca 5 mV.



Obrázek 13.1: Měření periody

13.4 Výsledky měření efektivní hodnoty

Efektivní hodnota je počítána schodovitou aproximací vztahu (4.11). Průchody referenční úrovní jsou nalezeny funkcí pro výpočet periody. Měření jsem prováděl pro kmitočet vstupního signálu 250 Hz, což odpovídá 5. harmonické síťového kmitočtu. Střídavá složka o maximální velikosti 1,5 V je z důvodu unipolárního rozsahu převodníku modulována na stejnosměrnou úroveň 1,5 V, což odpovídá polovině vstupního rozsahu převodníku. Tím je dosaženo maximálního rozkmitu vstupního signálu.



Obrázek 13.2: Chyba měření efektivní hodnoty

Měřenou hodnotu jsem porovnával s 6,5místným přístrojem HP34401A, který má pro střídavá měření na vstupu oddělovací kondenzátor. Proto měří pouze efektivní hodnotu (TrueRMS) střídavé složky. Pro kontrolu správnosti měřených hodnot je proto nutné přepočítat správnou hodnotu měřenou přístojem z údajů střední DC a efektivní hodnoty (definované amplitudou AC) pro jednotlivé typy průběhů podle vztahů

$$U_{RMS}^{sin} = \sqrt{DC^2 + \frac{AC^2}{2}}, \qquad U_{RMS}^{obd} = \sqrt{DC^2 + AC^2}, \qquad U_{RMS}^{troj} = \sqrt{DC^2 + \frac{AC^2}{3}} \quad (13.1)$$

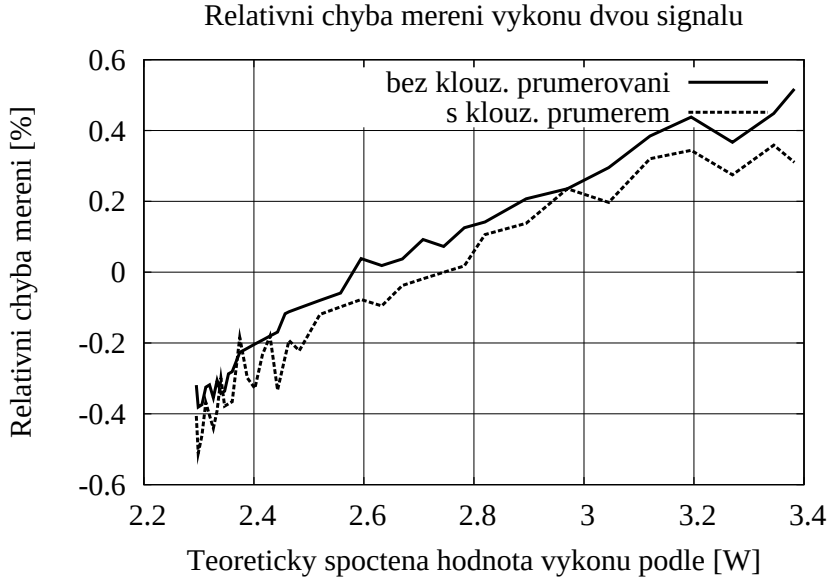
kde DC je stejnosměrná úroveň, na kterou je namodulována střídavá složka s amplitudou AC.

Pomocí tohoto vztahu je nutné výslednou hodnotu přepočítat podle konkrétní aplikace v případě, že vstupní signál bude upraven např. operačním zesilovačem a stejnosměrně posunut na polovinu vstupního rozsahu A/D převodníku 1,5 V.

13.5 Výsledky měření výkonu

Při měření výkonu jsem použil dva průběhy superponované na 1,5 V a tyto signály přivedl na vstup A/D převodníku. Pokud jsou vstupní signály obou kanálů popsány jako $u(t) = C + a\sin(\omega t)$ a $i(t) = D + B\sin(\omega t)$, platí pro výkon vztah

$$P = CD + \frac{AB}{2} \quad (13.2)$$



Obrázek 13.3: Chyba měření vykonu

Kapitola 14

Závěr

Úkolem diplomové práce bylo provedení rozboru možností použití mikrokontrolérů řady ARM v jednotkách sběru dat s vícekanálovým vstupem. Pro potvrzení těchto možností bylo dále úkolem vytvoření modulů osazených procesory ARM7TDMI. Práci lze rozdělit do několika částí.

V první části části jsem se zaměřil na teoretický popis vzorkovacích metod použitelných v jednotkách sběru dat a matematických metod číslicového zpracování dat při výpočtu parametrů signálů a metodě generování průběhů metodou DDS. Zde jsem čerpal převážně z odborných článků a příspěvků z odborných konferencí.

V další části jsem se poměrně obsáhle věnoval samotné ARM architektuře, protože tato práce je jedna z prvních, která se na katedře měření touto architekturou zabývá. Tato část může použita pro pedagogické účely při výuce mikroprocesorové techniky. V této části jsem provedl průřez jednotlivými rodinami procesorových jader ARM, naznačil metody jejich programování a limitace jejich použití. Zde jsem čerpal z internetových manuálů jednotlivých výrobců a dokumentace výrobce ARM Ltd. Protože se jedná o poměrně novou kategorii mikrokontrolérů, po celou dobu práce jsem se potýkal s nedostatkem dokumentace k používaným mikrokontrolérům, převážně výrobků firmy Philips.

V praktické částí práce jsme navrhl modul CPU jádra s procesorem LPC2104 označeným AMCLPC. Pro tento modul jsem vyvinul knihovnu obslužných funkcí pro přístup k jednotlivým perifériím v prostředí CYGWIN. Modul jsem vytvořil ve spolupráci s firmou AMiT, spol. s.r.o.

Algoritmy uvedené v teoretické části jsem implementoval do systému MCB2100, vývojového kitu od firmy Keil, se kterým jsem odměřil většinu parametrů. Programové vybavení jsem vytvořil v prostředí μVision od firmy Keil, které je velmi vhodné i pro účely výuky. Prostředí je omezené limitované na 16kB kódu pro ladění. Během vlastní práce jsem objevil některé nedostatky prostředí. Vlivem časově náročné práce při studiu vlastní architektury se mi nepodařilo implementovat algoritmus výpočtu spektra použít generované PWM.

Během posledních měsíců práce byl navržen a oživen modul s procesorem ADuC7024, který je vybaven interním A/D a D/A převodníkem. Mohl by proto být vhodnějším nástrojem i pro generování průběhů. Toto může být předmětem další práce.

Procesory ARM splnily počáteční představy pouze částečně. Přestože jádro procesoru je velmi výkonné, přístupy na výstupní piny takových výkonů nedosahují a maximálně dosažitelná frekvence programově ovládaného výstupu je 8,5 MHz a vstupu pouze 7,4 MHz při vnitřní frekvenci jádra 60 MHz. Rychlost přístupů do paměti dat je při zápisu 29 MHz a při čtení 19,5 MHz. Příčinu těchto omezení jsem vysvětlil v kapitole věnované architektuře ARM. Uváděný výpočetní výkon je dosažitelný pouze při práci s registry. Pokud nejsou tato omezení v aplikace limitující, jsou procesory ARM vhodnými procesory pro jednotky sběru a následného zpracování dat.

Domnívám se, že cíle vytyčené v zadání byly splněny.

Literatura

[1] Haasz V. *Měření parametrů periodických průběhů systémy bázi zásuvných měřicích desek do PC.* Automatizace ročník 39, č. 5 – 8, Praha, 1996

[2] Haasz V. *Synchronous or asynchronous sampling by measurement of parametres of periodical signals.* 8th International IMEKO TC-4 Symposium, Budapest 1996, str. 115 – 117

[3] Haasz V., Sedláček M. *Modelování vlivu vzorkování a kvantování na přesnost určení efektivní hodnoty napětí a proudu a přesnost určení výkonu.* EMISKON 89, Praha 1989, str. 64 – 68

[4] Haasz V. *Use of sampling methods for multichannel measuring of quality of active compensation in three-phase systems.* 6th Int. conference on industrial metrology CIMI'95. Zaragoza 1995, str. 229 – 308

[5] Sedláček M., Matějovský J. *An alternative method for leakage reduction in FFT spectrum analysis — experimental results.* 8th IMEKO TC4 Symposium, Budapest 1996, str. 198 – 201

[6] Sedláček M., Titěra m. *Finding RMS values of sampled signal using time domain or frequency domain signal processing.* Proc. of 3rd International Conference on Digital Signal Processing, Herl'any 1997, str. 140 – 143

[7] Ďaďo S., Vedral J. *Číslicové měření, přístroje a metody.* Vydavatelství ČVUT, Praha 2002

[8] Nonovičová J. *Pravděpodobnost a matematická statistika.* Vydavatelství ČVUT, Praha, 1999

[9] Smeljakov B.B. *Čifrovaja izmeritelnaja apparatura infranizkich častot*, Energija, Moskva, 1975

[10] Herout P. *Učebnice jazyka C.* České Budějovice: Kopp, 2003

[11] Buchanan D. *Choosing DAC for Direct Digital Synthesis.* Application note AN-237, Analog Devices

[12] Ganapathiraju A. `http://www.isip.msstate.edu/publications/reports/hpclpdsp/1997/status/reportv1.pdf`, Missippi State University

Literatura o ARM procesorech

[13] ARM Ltd. *ARM7TDMI — Technical reference manual, Rev. 4*, ARM Limited, 2001. [cit. 2004-09-13]. `http://www.arm.com/pdfs/DDI0234A\_7TDMIS\_R4.pdf`

[14] IHI 0011A

[15] Technical Reference Manual, ARM DDI 0169B

[16] `http://www.arm.com/aboutarm/milestones.html`

[17] ARM Architecture reference manual, DDI0100E_ARM_ARM

[18] `http://home.in.tum.de/~atterer/acorn.html`

[19] `http://en.wikipedia.org/wiki/Acorn\_RISC\_Machine`

[20] ARM Instruction Set, Quick Reference Card

[21] http://www.pinknoise.demon.co.uk/ARMinstrs/ARMinstrs.html

[22] LPC2106/2105/2104 User Manual, Preliminary Release, October 02, 2003, Philips Semiconductors

[23] http://www.esemagazine.co.uk/common/viewer/archive/2004/Feb/1/feature2.phtm

[24] http://www.directinsight.co.uk/microcontroller/arm-devices.html

[25] http://atterer.net/acorn.html

[26] Performance of the ARM9TDMI^{TM} and ARM9E-S^{TM} cores compared to the ARM7TDMI^{TM} core

[27] An Introduction to ThumbTM, ARM DVI-0001A

[28] Webová stránka projektu WinARM, http://www.siwawi.arubi.uni-kl.de/avr_projects/arm_projects/\#winarm

[29] Stránky vývojového prostředí μVision firmy Keil http://www.keil.com/uvision2/

[30] Stránky vývojového prostředí CygWin, http://www.cygwin.com/

[31] Katalogové listy použitých součástek

Webové stránky výrobců mikroprocesorů ARM:

[32] http://www.semiconductors.philips.com/

[33] http://www.analog.com

[34] http://focus.ti.com

[35] http://www.altera.com/products/devices/arm/arm-index.html

[36] http://www.atmel.com/products/AT91/

[37] http://www.sharpsma.com/sma/products/MCUSoC.htm

[38] http://developer.intel.com/design/pca/applicationsprocessors/index.htm

[39] http://www.cirrus.com/en/products/pro/areas/embedded.html

[40] http://www.st.com/stonline/products/selector/696.htm

[41] http://www.ni.com/dataacquisition/

Kapitola A

Seznam použitých zkratek a symbolů

- **AMBA** – *Advanced Microcontroller Bus Architecture*, ARM^{TM}, definuje standard komunikace mezi jednotlivými bloky na čipu mikrokontroléru, definuje sběrnice AHB, ASB a APB
- **AHB** – *Advanced High-performance Bus*, specifikace sběrnice s velkou datovou propustností pro procesory ARM
- **APB** – *Advanced Peripheral Bus*, definice sběrnice pro periferní nízkopříkonové moduly procesoru, někdy označovaná jako VPB
- **ARM** – *Advanced RISC Machines*, jméno společnosti ARM Ltd., označení jádra, označení instrukční sady
- **ARM7** – Rodina procesorů ARM, další jsou např. ARM9, ARM10
- **ARM7TDMI** – Jádro procesoru z rodiny ARM7.
- **ARMv4T** – Architektura a instrukční sada
- **ARMv6** – Architektura a instrukční sada
- **ASB** – *Advanced System Bus*, alternativa ke složité sběrnici AHB, definováno standardem AMBA
- **DMA** – *Direct Memory Access*, přímý přístup do paměti, datový přenos, nezatěžuje procesor, vlastní řadič
- **DPS** – *Deska plošných spojů*
- **DSP** – *Digital Signal Processor*, procesory určené pro matematické zpracování dat, *Digital Signal Processing* číslicové zpracování dat, algoritmy
- **ETM** – *Embedded Trace Module*, modul na čipu procoserů ARM, umožňuje trasování běhu programu přes speciální konektor
- **MAM** – *Memmory Acceleration Module*, zrychluje vyčítání instrukcí a dat z paměti
- **MMU** – *Memory Management Unit*, provádí překlad logických adres na fyzické adresy
- **Mikrokontrolér** – Čip obsahující procesor, paměť a další periferie v jednom pouzdře
- **Pipelining** – Zřetězené zpracovávání instrukcí. V jednom okamžiku dochází k dekódování instrukce (*decode*), načítání další instrukce z paměti (*fetch*) a vykonávání již dekódované instrukce (*execute*).
- **IP** – *Intelectual Properties*
- **Jazelle** – Technologie ARM, která umožňuje vykonávání instukcí v Java kódu.
- **JTAG** – *Joint Test Action Group*, sdružení výrobců součástek. Definovali rozhraní IEEE 1149.1, používané pro testování, programování a ladění mikroprocesorů. Jsou přístupné všechny registry v programátorském modelu. Základem je pětivodičové připojení včetně zemního vodiče.
- **jitter** – časová nejistota, např. jitter při odběru vzorku znamená, že vzorek může být odebrán v přesně neznámém okamžiku
- **SFDR** – *Spurious Frequency Dynamic Range*, udává dynamický rozsah D/A převodníků, vyjadřuje se v dB
- **SNR** – *Signal Noise Ratio*, odstup signálu a šumu, vyjadřuje se v dB
- **SoC** – *System-On-Chip*, sběrnicová struktura propojení jednotlivých bloků na čipu procesoru
- **Thumb** – komprimovaná instrukční sada, délka instrukcí 16 bitů, jsou dekódovány na 32bitové ARM instrukce
- **UART** – řadič sériové linky
- **Watchdog** – dohlížecí obvod, má za úkol resetovat procesor, pokud dojde k zacyklení programu, např. vlivem rušení
- **Xrms** – efektivní hodnota rušivého napětí

Kapitola B

Použité měřicí přístroje - parametry

Při měření jsem používal přístroje vyjmenované v této kapitole.

- Multimetr HP-34401A
- Generátor Tektronics AFG320
- Osciloskop Agilent Megazoom HP54622D
- Osciloskop Tektronics TDS3052B

Kapitola C

Měření parametrů jádra - testovací SW

Následuje výpis zdrojového kódu pro testování jitteru v přerušení.

```
    ldr r3, =MAMCR
    ldr r4, =MAMTIM
    ldr r5, =VPBDIV

    ldr r2, =0
    str r2, [r3]

    ldr r2, =1
    str r2, [r4]    // nastavi prislusny počet wait stavu

    ldr r2, =1
    str r2, [r3]    // zapne MAM v˜rezimu 1

    ldr r2, =1
    str r2, [r5]    // nastavi VPBDIV=1

    @-------------------------------------------------------------
    @ predem potreba nastavit směr výstupních pinu v˜registru IODIR,
    @ a alternativni funkci prerusovaciho pinu P0.16
    @ -------------------------------------------------------------
    cykl:
        @ preddefinovani registru
        ldr r1, =IOPIN      // nacte port
        ldr r3, =IOSET      // nastav port do 1
        ldr r4, =IOCLR      // nastav port do 0
        ldr r5, =0x80000000 // ovlada pin P0.31
        ldr r6, =0x40000000 // ovlada pin P0.30


    opakuj:
    cekam_nulu:
        ldr r2, [r1]
        tst r2, #0x20000    // otestuje bit P0.17
        beq je_nula         // pokud 0 pokracuj cekanim na 1
        b   cekam_nulu      // cekej dal na 1
    je_nula:

    cekam_jednicku:
        ldr r2, [r1]
```

```
        tst r2, #0x20000
        bne je_jednicka
        b cekam_jednicku
    je_jednicka:                // nasledujici hrana bude sestupna a vyvola preruseni

    strimm  EXTINT, 1               // pro jistotu
    strimm  VICIntEnable, 0x4000    // povoli preruseni

    cekej_na_preruseni:
    nop
    str r6,[r3]             // set pin, kontrola oblasti nopu
    str r6,[r4]             // set pin, kontrola oblasti nopu
    str r6,[r3]             // set pin, kontrola oblasti nopu
    str r6,[r4]             // set pin, kontrola oblasti nopu
    nop
    nop
    ...
    nop
    nop
    nop
    nop
    nop
    str r6,[r3]             // set pin, kontrola oblasti nopu
    str r6,[r4]             // set pin, kontrola oblasti nopu
    nop
    nop
    nop
    b   opakuj                      @ vrat se zpatky na zacatek programu
    @-------------------------------------------------------------

    @-------------rutina preruseni-------------------------------
    SystemIRQ_Handler:
        stmfd   sp!, {r0-r12, lr}   @ uloz registry na stack

        str r5, [r3]                @ tuto hranu kontrolujeme
        str r5, [r4]

        str EXTINT, 1               @ nutne pro zruseni flagu preruseni

        ldmfd   sp!, {r0-r12, lr}   @ obnov registry ze stacku
        subs    pc, lr, #4          @ navrat z~preruseni (jako RETI)
@-------------------------------------------------------------
```

Kapitola D

Seznam součástek

D.1 AMCLPC - Seznam součástek

Počet	Označení	Popis	Hodnota
6	C2,C5,C7,C9,C13,C14	Kondenzátor 0805	1M/16V
6	C1,C3,C4,C6,C8,C10	Kondenzátor 0805	100N/63V
1	C15	Kondenzátor 0805	22M/6.3V
2	C11,C12	Kondenzátor 0805	22P/63V
1	R16	Odpor 1206	33K
5	R8,R9,R11,R12,R13	Odpor 0805	8K2
3	R6,R10,R15	Odpor 0805	820R
3	R2,R3,R4	Odpor 0805	1% 1K2/A
2	R1,R14	Odpor 0805	1% 4K7/A
	R5,R7	Odpor 0805	NEOSAZOVAT
1	X1	Krystal	20.0000 MHz
1	D3	LED SMD 0805	HSMY-C670
2	D2,D4	Dioda SMD	1N4148
2	D1,D5	Dioda SMD	BAT46
1	T1	Tranzistor SMD	BC817-25
1	T2	Tranzistor SMD	BC807-25
1	U5	Stabilizátor	LP2951CD
1	U4	Procesor	LPC2104BBD48
1	U3	Watchdog	ADM706TAR
1	U2	EEPROM	M24256BWMN6
1	U1	RTC	RTC8564JE
1	BT1	Baterie	BR2477/CHCE

Tabulka D.1: Seznam součástek pro prototypový modul AMCLPC

D.2 ADuC - Seznam součástek

Počet	Označení	Hodnota
8	C1,C2,C3,C4,C5,C3 C7,C9,C1	1u
9	C6,C10,C14,C15, C16,C17,C18,C19,C8	100n
2	C12,C11	12p
1	D1	1N4148
1	D2	LED
1	D3	1N4007
2	J1,J2	HEADER 25
4	J3,J4,J7,J8	JUMPER
1	J5	HEADER 10X2
2	J9,J6	HEADER 1
1	P1	CANNON DB9
1	Q1	BC548
4	R1,R3,R5,R14	1k
3	R2,R4,R6,R7,R8,R9 R10,R13,R12,R15	10k
1	S1	TLACITKO
1	U1	ADuC7024
1	U2	MAX232
1	U4	LF33
1	Y1	32.768kHz

Tabulka D.2: Seznam součástek pro vzorový modul ADuC

Kapitola E

Cygwin - Kompilační předpis Makefile

```
#LPC210x common library Makefile
PACKNAME=lpc210x

#directories
LIBSDIR=../..
INCDIR=$(LIBSDIR)/include
OBJDIR=$(LIBSDIR)/obj
LIBDIR=$(LIBSDIR)/lib
LSTDIR=$(LIBSDIR)/lst

#filenames
PACKLSTFILE=$(LSTDIR)/lib$(PACKNAME).lst
PACKLIBFILE=$(LIBDIR)/lib$(PACKNAME).a
INTLSTFILE=$(LSTDIR)/libinterrupts.lst
INTLIBFILE=$(LIBDIR)/libinterrupts.a

#defines
CC=arm-elf-gcc
AS=arm-elf-as
AR=arm-elf-ar
CFLAGS=-O1 -ggdb -Wall
#-o1 level of optimalization, -Wall print all warnings

all: build $(PACKLIBFILE) $(INTLIBFILE)
build:
@echo "Building LPC210x common library"
@mkdir -p $(OBJDIR)
clean:
@echo "Cleaning LPC210x common library"
@rm -f $(PACKLSTFILE) $(PACKLIBFILE) $(INTLSTFILE) $(INTLIBFILE)
@rm -f $(OBJDIR)/isr.o
@rm -f $(OBJDIR)/uarts.o
@rm -f $(OBJDIR)/vsr.o
@rm -f $(OBJDIR)/pllvpb.o

#****************** PACKLIBFILE *********************
# uart and timing PLL and VPB library target
$(PACKLIBFILE): $(OBJDIR)/uarts.o $(OBJDIR)/pllvpb.o
$(AR) -ru $@ $(OBJDIR)/uarts.o
$(AR) -ru $@ $(OBJDIR)/pllvpb.o
```

```
$(AR) -t  $@ > $(PACKLSTFILE)

#****************** INTLIBFILE *********************
# interrupts
$(INTLIBFILE): $(OBJDIR)/isr.o $(OBJDIR)/vsr.o
$(AR) -ru $@ $(OBJDIR)/vsr.o
$(AR) -ru $@ $(OBJDIR)/isr.o
$(AR) -t  $@ > $(INTLSTFILE)

#*************** creating target objects ***********
$(OBJDIR)/uarts.o: uarts.c $(INCDIR)/uarts.h
$(CC) -I$(INCDIR) -c $(CFLAGS) uarts.c -o $@

$(OBJDIR)/pllvpb.o: pllvpb.c $(INCDIR)/pllvpb.h
$(CC) -I$(INCDIR) -c $(CFLAGS) pllvpb.c -o $@

# interrupts obj targets
$(OBJDIR)/isr.o: isr.c $(INCDIR)/ints.h
$(CC) -I$(INCDIR) -c $(CFLAGS) isr.c -o $@

$(OBJDIR)/vsr.o: vsr.s
$(AS) -I$(INCDIR) vsr.s -o $@
```