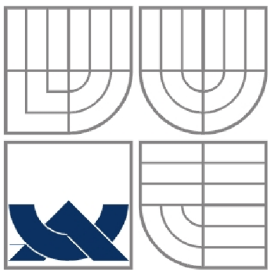
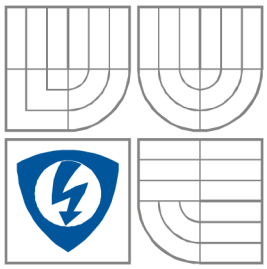


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TEORETICKÉ A EXPERIMENTÁLNÍ ELETROTECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF THEORETICAL AND EXPERIMENTAL ELECTRICAL ENGINEERING

ŘÍZENÍ MĚNIČE
CONVERTERS CONTROLLING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jiří Nevoral
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Pavel Fiala, Ph.D.
SUPERVISOR

BRNO, 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Jiří Nevoral
Bytem:	Stanislava Slavíka 1440, Moravské Budějovice, 676 02
Narozen/a (datum a místo):	1. července 1982 v Třebíči

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Dr. Ing. Zbyněk Raida, předseda rady oboru Elektronika a sdělovací technika
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

   ☐ disertační práce
   ☐ diplomová práce
   ☒ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

Název VŠKP:	Řízení měniče
Vedoucí/ školitel VŠKP:	doc. Ing. Pavel Fiala, Ph.D..
Ústav:	Ústav teoretické a experimentální elektrotechniky
Datum obhajoby VŠKP:	_____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.

3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

* hodící se zaškrtněte

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti

   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)

4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.

2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 6. června 2008

……...	……
Nabyvatel	Autor

Abstrakt: Bakalářská práce je studiem problematiky mikroprocesorů DSP vhodných k řízení frekvenčních měničů používaných k řízení asynchronních motorů a dále vhodné realizace vlastního řízení. Práce obsahuje informace o vlastnostech jednotlivých typů procesorů, výhody a nevýhody celého řešení. Cílem je realizace řízení frekvenčního měniče pomocí vývojové desky s DSP procesorem a výkonovým měničem.

Klíčová slova: měnič, řízení, elektromotor, třífázový, pcm, asynchronní, dsp, frekvenční, ccs, mikrokontrolér, mikroprocesor, programování, pulzně šířková modulace, pwm, otáčky, střídavý, Texas, Instruments, flash, DMC1500, F2808, generování

Abstract: Bachelor´s project is study of DSP microprocessor problems to converters controlling with asynchronous motors and execution of a project. Study includes information on quality single type processor, benefits and disadvantage whole solving. Of purpose is DSP converters control with DMC1500.

Keywords: dsp, converter, controlling, frequency, microprocessor, pulse width modulation, rpm (revolutions per minute), programming, code composer studio, pwm, AC (alternating-current), texas, Instruments, flash, DMS1500, F2808, pulse generation

NEVORAL, J. *Řízení měniče.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 42 s.
Vedoucí semestrální práce doc. Ing. Pavel Fiala, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Řízení měniče jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne 6. června 2008 .....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce doc. Ing. Pavlu Fialovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce

V Brně dne 6. června 2008 .....
podpis autora

Obsah

1 Úvod – DSP procesor 8
1.1. Z historie 9
1.2. Použití 11
1.3. Architektura obvodu 11
1.4. CPU jednotka 11
1.5. Paměť 14
1.6. Sériová komunikační zařízení 14
1.7. Generátor hodinových signálů 14
1.8. Jednotka čítače/časovače 16
1.9. A/D převodník 18
1.10. Shrnutí 18

2 Pulzně šířková modulace 18
2.1. Obecně o PWM 19
2.2. Generování PWM pomocí procesoru 8051 20

3 Vývojový kit eZdsp 21
3.1. Code Composer Studio 22

4 Frekvenční měnič kmitočtu 23
4.1. Podrobný popis produktu 23
4.2. Návrh zapojení 24
4.3. Možnosti řízení 25

5 AC/DC měnič DMS1500 28
5.1. Realizace obvodu 28

6 Aplikace vlastního řízení 29

7 Závěr 32

8 Seznam literatury 33

9 Použité zkratky a symboly 35

10 Příloha 36
10.1. Technické parametry frekvenčního měniče kmitočtu 36
10.2. Fotografie pracoviště 37
10.3. Testovací program pro generování PWM 38
10.4. Program pro řízení otáček 41
10.5. Plošná deska měniče DMC1500 42

1. Úvod – DSP procesor

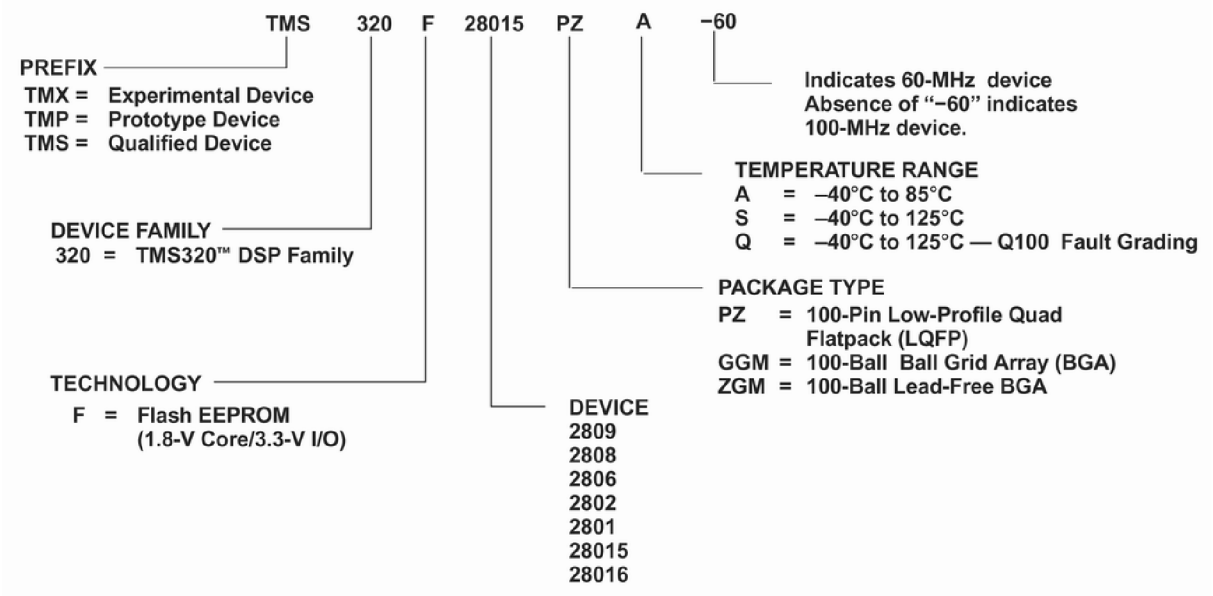
Firma Texas Instruments je již dlouhou dobu tradičním výrobcem špičkových signálových procesorů (zkráceně DSP, *obr. 1.,* [1] [15]). Poslední novinkou v této oblasti je i "malý" DSP řady F28xx. Výhodou proti klasickým DSP je FLASH paměť na čipu, 12bitový A/D převodník, sběrnice CAN, PWM výstupy, čímž se klasický DSP více přiblížil mikrokontrolérům. DSP procesory jsou vyráběny pro širokou škálu využití, od čipů pro zpracování obrazu a zvuku, tak i pro výkonovou elektroniku. Pro svoji vysokou rychlost a rozmanitost periferií jsou vhodné zejména pro řízení elektromotorů, což bylo klíčovým kritériem výběru.

Obr. 1. Čip DSP výrobce Texas

Digitální signálové procesory firmy Texas Instruments vždy patřily k těm nejznámějším a nejvyspělejším na trhu a jediný, kdo jim vždy byl schopen konkuroval, byly DSP firmy Motorola. DSP se však až do současnosti vždy vyznačovaly vysokým výkonem z pohledu zpracování dat (hardwarová násobička = rychlé násobení/dělení, víceúrovňový pipelining, duální přístup do SRAM paměti, více násobné oddělené datové a adresové sběrnice apod.), ale z pohledu počtu uživatelských periferií na čipu se vždy spíše blížili "klasickým počítačovým" CPU, tzn. A/D a D/A převodník, programová EEPROM apod., vždy musely být doplněny jako externí součástky. V současnosti však stejně jako MCU, které dostávají některé vlastnosti DSP (např. hardwarová násobička), DSP získávají vlastnosti MCU (např. A/D, PWM, FLASH, watchdog přímo na čipu DSP). Jejich použití se tak zjednodušuje. To však neznamená, že je rozdíl mezi MCU a DSP smazán. Stále DSP se vyznačují rychlejší a dokonalejším zpracováním velkých bloků dat např. z řeči, hudby nebo obrazu, zatímco MCU se stále vyznačují jednodušší konstrukcí, menšími rozměry a snadnější implementací.

FEATURE		F2809	F2808	F2806	F2802	F2801/ 9501	C2802	C2801
Instruction cycle (at 100 MHz)		10 ns	10 ns	10 ns	10 ns	10 ns	10 ns	10 ns
Single-access RAM (SARAM) (16-bit word)		18K (L0, L1, M0, M1, H0)	18K (L0, L1, M0, M1, H0)	10K (L0, L1, M0, M1)	6K (L0, M0, M1)	6K (L0, M0, M1)	6K (L0, M0, M1)	6K (L0, M0, M1)
3.3-V on-chip flash (16-bit word)		128K	64K	32K	32K	16K	–	–
On-chip ROM (16-bit word)		–	–	–	–	–	32K	16K
Code security for on-chip flash/SARAM/OTP blocks		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Boot ROM (4K X16)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
One-time programmable (OTP) ROM (16-bit word)		1K	1K	1K	1K	1K	–	–
PWM outputs		ePWM1/2/3/4/5/6	ePWM1/2/3/4/5/6	ePWM1/2/3/4/5/6	ePWM1/2/3	ePWM1/2/3	ePWM1/2/3	ePWM1/2/3
HRPWM channels		ePWM1A/2A/3A/ 4A/5A/6A	ePWM1A/2A/ 3A/4A	ePWM1A/2A/ 3A/4A	ePWM1A/2A/3A	ePWM1A/2A/3A	ePWM1A/2A/3A	ePWM1A/2A/3A
32-bit CAPTURE inputs or auxiliary PWM outputs		eCAP1/2/3/4	eCAP1/2/3/4	eCAP1/2/3/4	eCAP1/2	eCAP1/2	eCAP1/2	eCAP1/2
32-bit QEP channels (four inputs/channel)		eQEP1/2	eQEP1/2	eQEP1/2	eQEP1	eQEP1	eQEP1	eQEP1
Watchdog timer		Yes	Yes	Yes	Yes	Yes	Yes	Yes
12-Bit, 16-channel ADC conversion time		80 ns	160 ns	160 ns	160 ns	160 ns	160 ns	160 ns
32-Bit CPU timers		3	3	3	3	3	3	3
Serial Peripheral Interface (SPI)		SPI-A/B/C/D	SPI-A/B/C/D	SPI-A/B/C/D	SPI-A/B	SPI-A/B	SPI-A/B	SPI-A/B
Serial Communications Interface (SCI)		SCI-A/B	SCI-A/B	SCI-A/B	SCI-A	SCI-A	SCI-A	SCI-A
Enhanced Controller Area Network (eCAN)		eCAN-A/B	eCAN-A/B	eCAN-A	eCAN-A	eCAN-A	eCAN-A	eCAN-A
Inter-Integrated Circuit (I^2C)		I^2C-A	I^2C-A	I^2C-A	I^2C-A	I^2C-A	I^2C-A	I^2C-A
Digital I/O pins (shared)		35	35	35	35	35	35	35
External interrupts		3	3	3	3	3	3	3
Supply voltage	1.8-V Core, 3.3-V I/O	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Packaging	100-Pin PZ	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	100-Ball GGM, ZGM	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Temperature options(1)	A: -40°C to 85°C	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)
	S: -40°C to 125°C	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)	(PZ, GGM, ZGM)
	Q: -40°C to 125°C	(PZ)	(PZ)	(PZ)	(PZ)	(PZ)	(PZ)	(PZ)
Product status(2)		***TMX***	TMS	TMS	TMS	TMS	TMS	TMS

tab. 1. Tabulka vybavení DSP řady F28xx/C28xx v závislosti na typovém označení



tab. 2. Označení obvodu podle jeho vlastností

1.1. Z historie

V roce 1978 uvedla firma Intel na trh nový procesor. Jednalo se o analogový procesor i2920. Procesor byl složen z převodníků analogové reprezentace signálu na digitální a zpět a z 25-bitového digitálního procesoru. Komerčně úspěšné byly však až DSP firem NEC a AT&T, oba uvedené na trh v roce 1980. Oba DSP si našly uplatnění v telekomunikačním průmyslu. Firma, spojovaná v současné době s DSP asi nejvíc, Texas Instruments, do této oblasti vstoupila roku 1983 uvedením procesoru 320C10. Známějším následníkem tohoto

DSP je 32bitový float-pointový procesor 320C30, uvedený v roce 1988. Tento je významný myšlenkou nezávislé činnosti ALU, adresní a řídící logiky, která přinesla větší propustnost dat. Jeho design se promítnul mj. i do procesorů Silicon Graphics (MIPS R8000) a do architektury Power firmy IBM (v současné době PowerPC). 320C30 byl vybaven 4kB ROM, 2x1kB RAM a zpracovával instrukce ve čtyřfázové pipeline.

Klíčový přínos DSP osmdesátých let je Harvardská architektura. Typický obecný procesor té doby je Von-Neumanova typu s jednou sběrnicí, společnou pro paměť i data. DSP naproti tomu začaly v té době používat oddělené sběrnice pro program a data. Výhodou je zde efektivnější přístup do paměti. Typickou aplikací DSP je FIR filtr. Tento se na DSP skládá ze smyčky, obsahující převážně přístupy do paměti a instrukce MAC (Multiply-Accumulate, tj. suma součinů). Pokud dokážeme instrukci MAC provést v každém cyklu, poběží program rychleji. A právě Harvardská architektura toto zrychlení umožňuje.

Novějším prvkem zrychlení běhu programu je paralelizace. DSP se časem vyvinul do procesorů, složených z několika nezávislých jednotek, viděných programátorem jako nezávislé asymetrické procesory. Toto uspořádání struktury procesoru umožňuje v jednom cyklu provést přístup (i několik přístupů) do paměti a provést i několik ekvivalentů původních MAC instrukcí. Zajímavým jevem okolo DSP je predikovatelnost. Obecné procesory neřeší typicky úlohy, u kterých by bylo možné jednoduše předvídat např. vzory přístupu do paměti. U DSP, které realizuje např. výše zmíněný FIR, toto možné je. A lze to s výhodou uplatnit při optimalizaci kódu pro DSP. Typický signálový procesor dnešní doby je DSP s několika podpůrnými obvody ve svém pouzdře. Procesor má instrukční sadu upravenou pro zpracování signálu. Dále je optimalizován, aby dosáhnul kompromisu mezi třemi klíčovými požadavky: vysoký výpočetní výkon, nízká spotřeba energie a nízká cena. Z procesorů od TI jsou k těmto účelům vyvíjena DSP řad 320C3000, 320C5000 a 320C6000. Hned na počátku je třeba zdůraznit vhodnou volbu procesoru. Měli bychom si uvědomit, že se též jedná o rozhodnutí optimalizačního charakteru. Pokud např. pracujeme s 32bitovými daty a chceme je zpracovávat na 64bitovém procesoru, děláme špatnou volbu – v daných okamžicích totiž běží polovina zúčastněných součástek zcela zbytečně. DSP se z těchto důvodů vyrábějí s různými šířkami sběrnic a s různými schopnostmi podle zpracovávaného typu dat – např. fixed vs. float point. Procesor je tvořen z logických obvodů. Rozepíšeme-li si spotřebu výstupních obvodů jednoho členu, dostaneme $P = C_L U^2 f + I_L U$, kde dynamická složka je reprezentována $P_D = C_L U^2 f$ a statická $P_S = I_L U$. První, dynamická, složka je daná kapacitou zátěže C_L, kvadrátem napájecího napětí U a frekvencí f. Je přítomna vždy, když je daný obvod aktivní a pokouší se o nějakou činnost. Odstranit ji můžeme např. zastavením lokálních hodinových pulsů. Druhá, statická, je tvořena trvalým proudem v čipu (I_L, *leakage current*) a napájecím napětím U. Přítomna je vždy při připojení napájecího napětí. Lze ji eliminovat pouze odpojením napájení do části čipu. Z výše uvedeného vyplývají též hardwarové techniky optimalizace spotřeby.

Napěťové:

- snižování napájecího napětí
- vypínání neaktivních částí čipu.

a frekvenční

- snižování frekvence
- modulace frekvence dle aktivity na čipu

Pro mikroprocesory DSP je specifická technika snížení pracovní frekvence při současném zvýšení počtu prováděcích jednotek. Obecné procesory se tuto techniku pokoušejí

převzít. Softwarové techniky optimalizace spotřeby vyplývají z jednoduchého vztahu $W = P \cdot t$
tj. čím déle je procesor nebo jeho část aktivní t, tím větší je spotřeba energie W. Zkracuje se doba běhu programu a následně i klesá spotřebovaná energie. DSP mají pro řízení *power-managementu* ze strany software implementované instrukce SLEEP (někdy též IDLE), na které je navázaných několik hardwarových režimů šetření energií.

1.2. Použití

- Zpracování dat a řízení v reálném čase
- Zpracování zvuku = řeči, hudby včetně vícekanálové
- Odstranění šumu řeči, rozpoznávání hlasu, přepis řeči do textu, ekvalizéry
- Zpracování obrazu
- Zvýraznění obrysů, rozpoznání objektů, kontrola tvarů, stabilizace obrazu fotoaparátů
- Předzpracování videa

Okrajově i tyto oblasti využití:

- Komprese dat v reálném čase
- Složité řízení robotů a víceosých polohovacích zařízení
- atd.

1.3. Architektura obvodu

DSP procesory výrobce Texas Instruments se již klasicky vyznačují harvardskou strukturou, tzn. s odděleným mapováním paměti programu a dat, citace ze zdroje [11], i když fyzicky jsou samozřejmě sjednoceny. Řada F28xx/C28xx stojí na nejnižší pozici v sortimentu DSP TI, přičemž je samotným výrobcem označována jako digitální signálové kontroléry. Zvláště obvody označované jako F28xx se výbavou periferií na čipu blíží "velkým" mikrokontrolérům. Obvody C28xx jsou již více klasické DSP, kde musí být programová FLASH paměť připojena externě. Detailní blokovou strukturu vyjadřuje *obr. 2.*

Celou strukturu obvodu DSP řady F28xx lze rozdělit do následujících bloků:

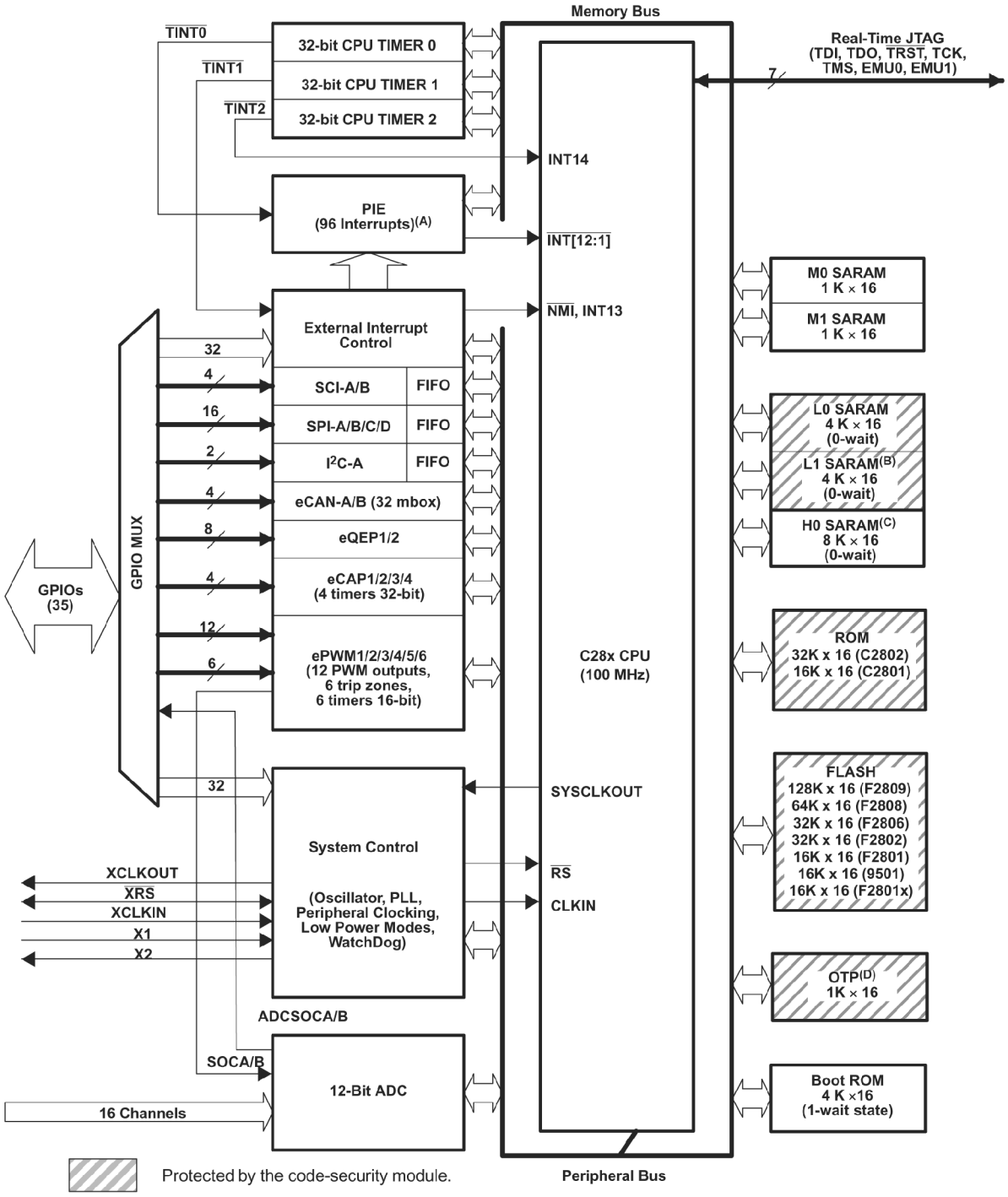
- CPU s ALU, hardwarovou násobičkou, registry
- FLASH, ROM, OTP, SRAM paměť (SARAM) - mapování dle potřeby
- Generátoru hodin - OSC a PLL
- Řízení přerušení (PIE, External Interrupt control)
- Periferie: víceúčelové vstupy/výstupy (GPIO), komunikační rozhraní (eCAN, I2C, JTAG, SPI, SCI), A/D převodník, PWM
- Čítače/časovače (timer), CAP, Watchdog, kvadraturní modulace (eQEP)

1.4. CPU jednotka

CPU jednotka (vlastní jádro) řady F28xx je shodná s obvody C28xx a je součástí DSP platformy TMS320C2000™ a skládá se z těchto prvků (*obr. 3.*), převzato z [13]:

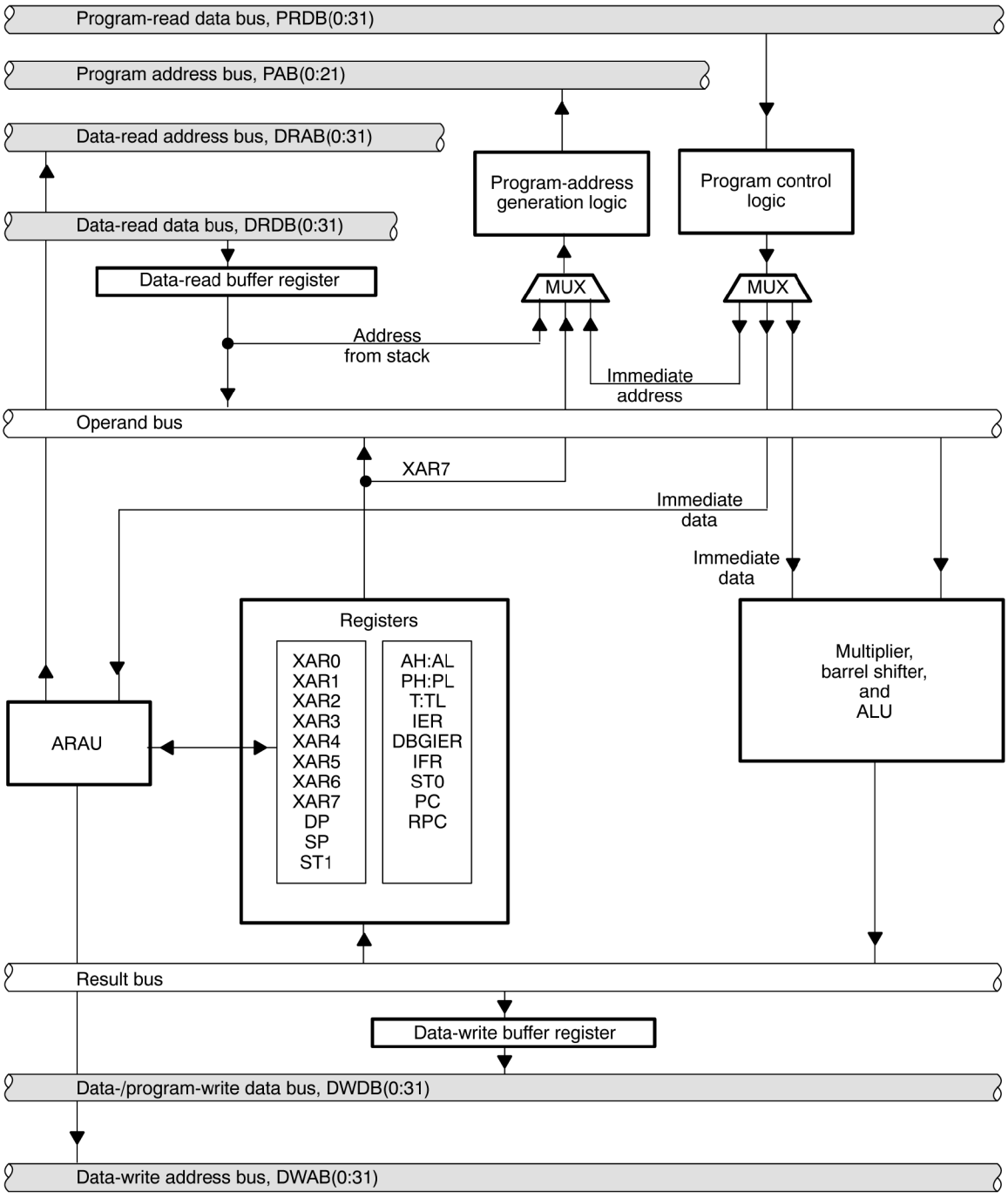
- Program and data control logic = programová a datová řídící logika
- Real-Time emulation and visibility = emulace a zobrazení dat v reálném čase
- Address register arithmetic unit (ARAU) = adresový registr aritmetické jednotky

- Atomic arithmetic logic unit (ALU) = aritmeticko-logická jednotka
- Prefetch queue and instruction decode = předzpracovávající fronta a dekódování instrukcí
- Address generators for program and data = generátor adres pro program a data
- Fixed-point MPY/ALU = Hardwarová násobička a ALU v pevné řádové čárce
- Interrupt processing = zpracování přerušení



Obr. 2. Bloková struktura DSP řady F28xx

V konečném výsledku jde o velmi výkonnou výpočetní a řídící jednotku pro programovací jazyk C/C++. Výpočetní jádro, pracující v pevné řádové čárce, tvoří 32 x 32bitová jednotka MAC se 64bitovými akumulátory. To umožňuje 64bitové zpracování dat, což eliminuje nutnost použít procesory pracující v plovoucí řádové čárce i pro data s velkým dynamickým rozsahem. 8 úrovňový pipelining a velký počet registrů (XAR0 až XAR7) umožňuje velmi rychlé zpracování dat i asynchronních událostí s minimálním zpožděním. Oddělené programové/datové adresové a datové sběrnice pro čtení a zápis dat (adresové - PAB, DRAB, DWAB, datové - PRDB, DRDB, DWDB) umožňují rychle přistupovat do pamětí a tím s minimálním prodlením připravovat potřebná data a příkazy ke zpracování.



Obr. 3. Blokové schéma CPU jednotky DSP řady C28xx/F28xx

Při časování hodinového cyklu o frekvenci 100 MHz je jeden instrukční cyklus pouze 10 ns. I když zpracování samostatně jednotlivých instrukcí v praxi může trvat více cyklů, jejich předzpracování (pipelining) ve struktuře více za sebou jdoucích příkazů, virtuálně zkrátí jejich provedení.

1.5. Paměť

Důležitým rozdílem řady F28xx proti C28xx je přítomnost FLASH paměti na čipu DSP. Ta může mít velikost až 128K x 16 bitů (F2800) rozdělená na 16 sektorů po 16 kB (viz (*obr. 4. Adresace paměti*). Navíc každý DSP má ještě 1K x 16 OTP paměti. Uživatel/programátor může individuálně každý sektor smazat či přepsat, aniž by se jakkoliv změnila data dalších sektorech. Není však možné provést mazání program právě běžícím v jiném sektoru. FLASH paměť může být mapována do programového i datového prostoru, tzn. že ji lze využít jako pro program, tak pro data.

ROM paměti může být až 32K x 16, přičemž v ní mohou být umístěny různé pevné tabulky konstant, například hodnoty funkce sinus apod. Mimo ni je zde ještě tzv. Boot ROM, kde je od výrobce nahrán bootovací program, který umožní rozběh procesoru pro připojení napájení nebo automatickému načtení programu a dat z externí paměti do vnitřní SRAM.

Interní SRAM, označená jako SARAM (Single Access RAM), je rozdělena do několika bloků lišící se svojí velikostí. Bloky M0 a M1mají velikost 1K x 16 a jsou mapovány do programového i datového paměťového prostoru. Po resetu se stack pointer (ukazatel zásobníku) nastaví do bloku M1. Bloky L0, L1, H0 poskytují až 16K x 16 bitů paměti. Každý blok může být nezávisle adresován CPU během pipeliningu.

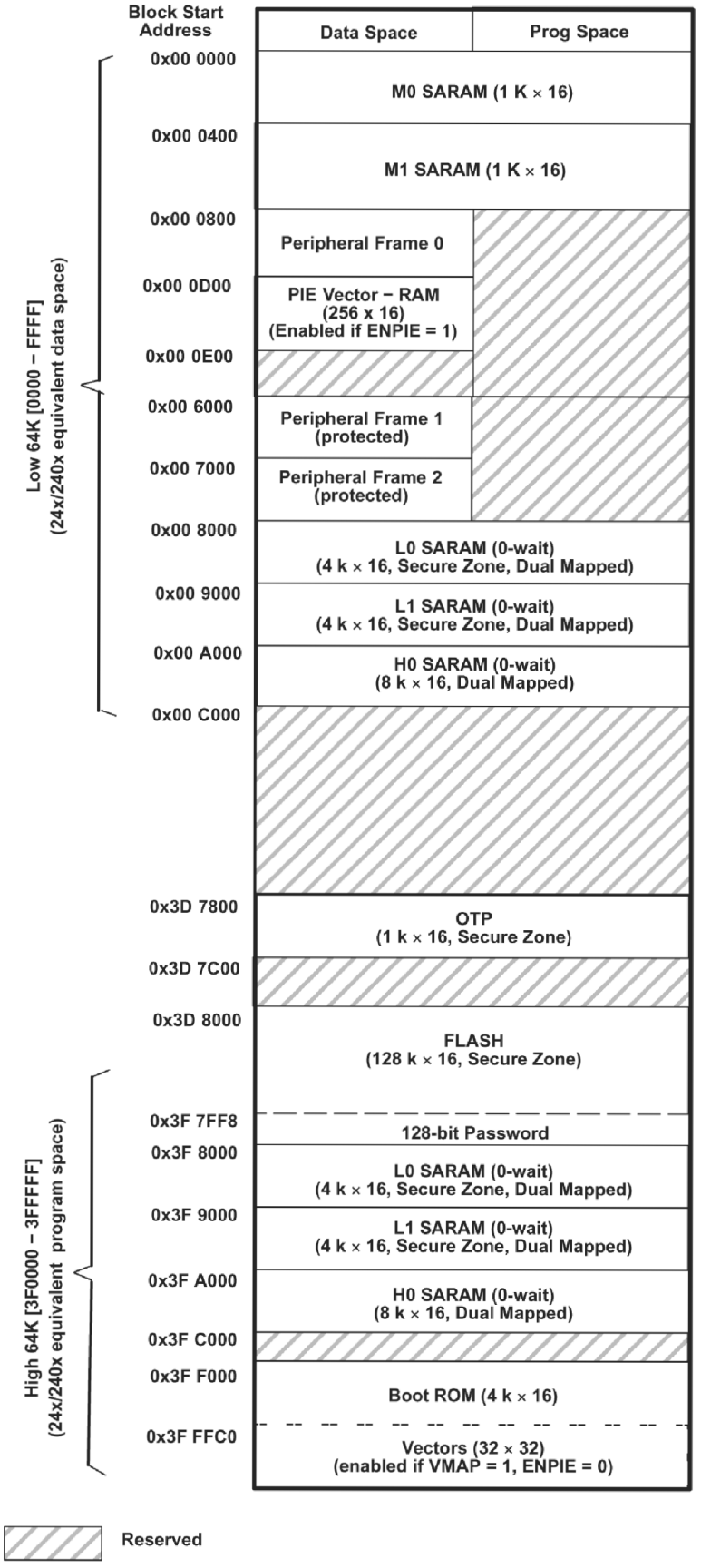
1.6. Sériová komunikační rozhraní

Na čipu jsou integrovány 4 různá sériová komunikační rozhraní:

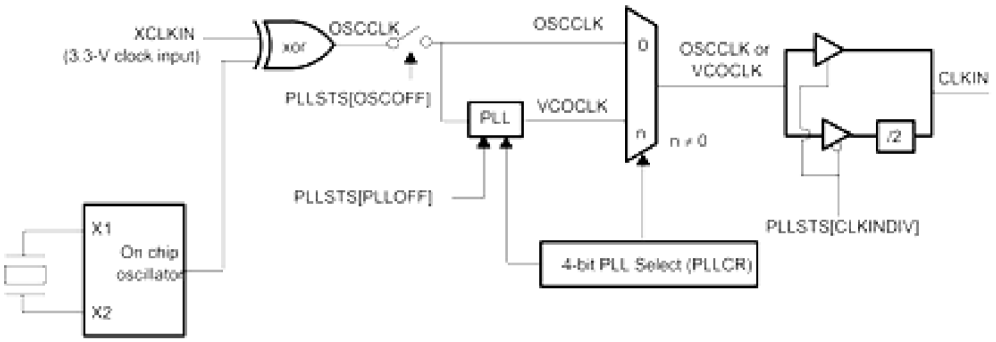
- **4x SPI rozhraní** synchronní sériové rozhraní/port, délka slova 1 až 16 bitů, režim Master/Slave,
- **2x tradiční SCI/UART rozhraní** (1 start bit, až 8 datových bitů, sudá/lichá/žádná parita, 1 nebo 2 stop bity) - max. přenos. rychlost až 6.25 Mb/s
- **I2C rozhraní** formát slova 1 až 8 bitů, 7/10 bitů adresovací mód, přenos. rychlost 10 kb/s až 400 kb/s, 16bitové FIFO vysílací a přijímací registry
- **2x CAN rozhraní** kompatibilní s CAN2.0B, 3.3 V CAN, přenos. rychlost až 1Mb/s (při taktování frekvencí 100 MHz je minimální přenos. rychlost 15.6 kb/s), Řízení zpráv - 32 schránek (mailboxes) = 512 B, datový formát 0 až 8 bitů

1.7. Generátor hodinových signálů

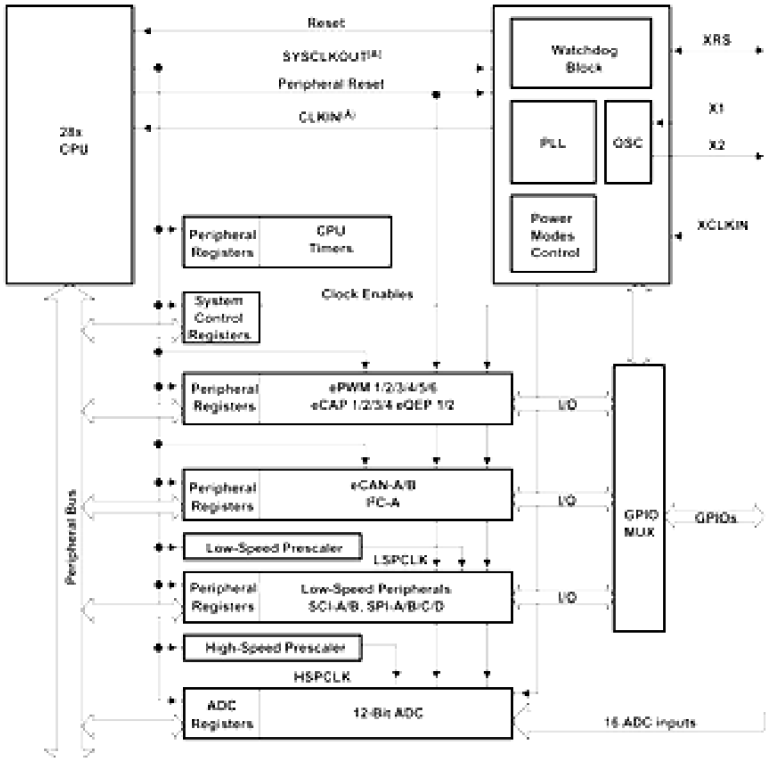
Součástí jádra je i interní generátor CLK hodinových signálů (*obr. 5a-b.*). Je řízen oscilátorem s frekvencí krystalu 20 MHz a upravován smyčkou fázového závěsu PLL. Generátor tak poskytuje až 10 poměrů - frekvencí hodin. signálů, daných poměrem PLL. Ten se nastavuje softwarově 4bitovým registrem. Nižší zvolenou frekvencí lze snížit spotřebu procesoru. Zároveň lze funkci PLL závěsu i úplně vyřadit. Maximální frekvence hodin je 100 MHz, což k tomu odpovídá doba jednoho instrukčního cyklu 10 ns.



Obr. 4. Adresace paměti



Obr. 5a. Blokové schéma generátoru hodinového signálu



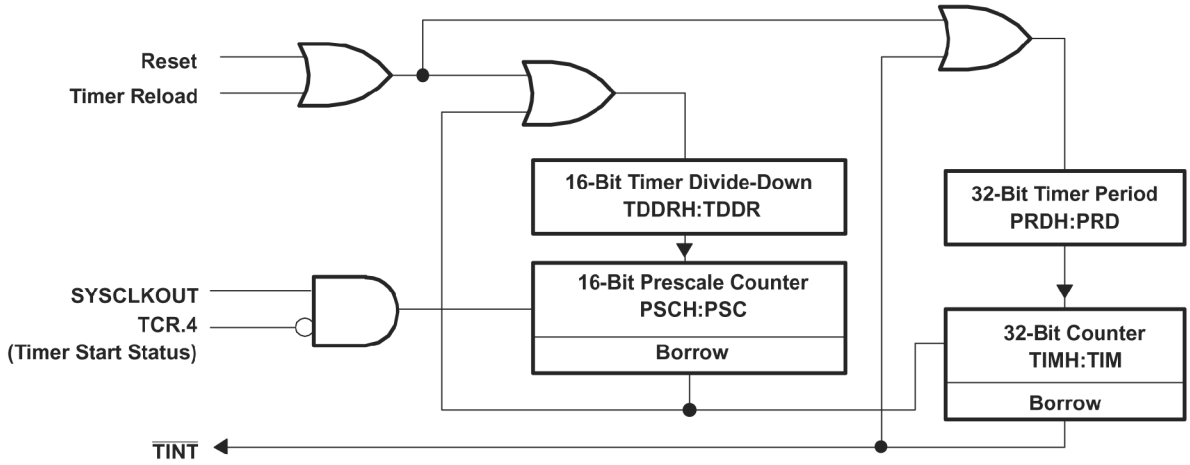
Obr. 5b. Blokové schéma použití hodin CLKIN a SYSCLKOUT

1.8. Jednotka čítače/časovače

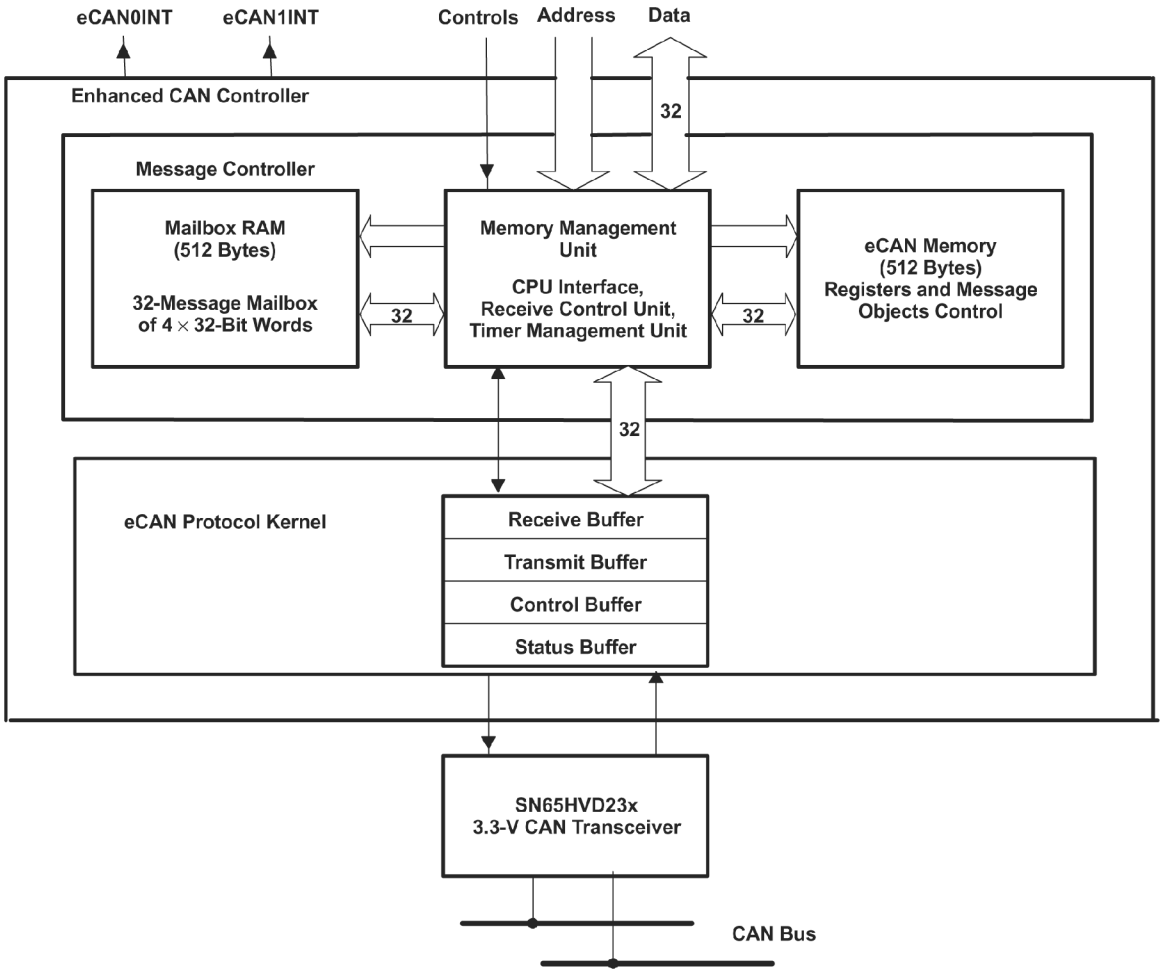
Na celém čipu mohou být až 3 jednotky 32bitových čítačů/časovačů (Counter/Timer). Ty pracují následovně: 32bitový čítací registr "TIMH:TIM" se naplní hodnotou uloženou v registru "PRDH:PRD". Čítač (*obr. 6.*) je dekrementován frekvencí hodin SYSCLKOUT. Když čítač dosáhne hodnoty 0, je vygenerován výstupní signál přerušení, generující puls přerušení (Interrupt Pulse) TINT. Frekvenci hod. signálu SYSCLKOUT lze měnit prostřednictvím 16bitové předděličky (registry TDDRH:TDDR a PSCH:PSC).

K časovému řízení softwaru slouží i 4úrovňový modul zachytávání eCAP (Enhanced CAPture Module). Ten umožňuje ze vstupu eCAP (*obr. 7.*) čítat signál a ten porovnávat se čtyřmi nezávislými 32bitovými registry. Použít lze kontinuální nebo jednorázový mód, který provádí porovnání opakovaně nebo jednorázově. Při zjištění shody nasčítané a nastavené hodnoty se generuje přerušení.

Naopak k časování řízené aplikace slouží ePWM blok (Enhanced Pulse Width Modulator) s až 7 PWM jednotkami. Ten tak poskytuje až 16 PWM výstupů podporující vysoké rozlišení, kdy je možné vytvořit PWM signály s přesností větší než 10 bitů pro frekvence vyšší než 200 kHz. To obvykle bývá již problém. Nezávisle lze řídit střídu i fázový posuv.



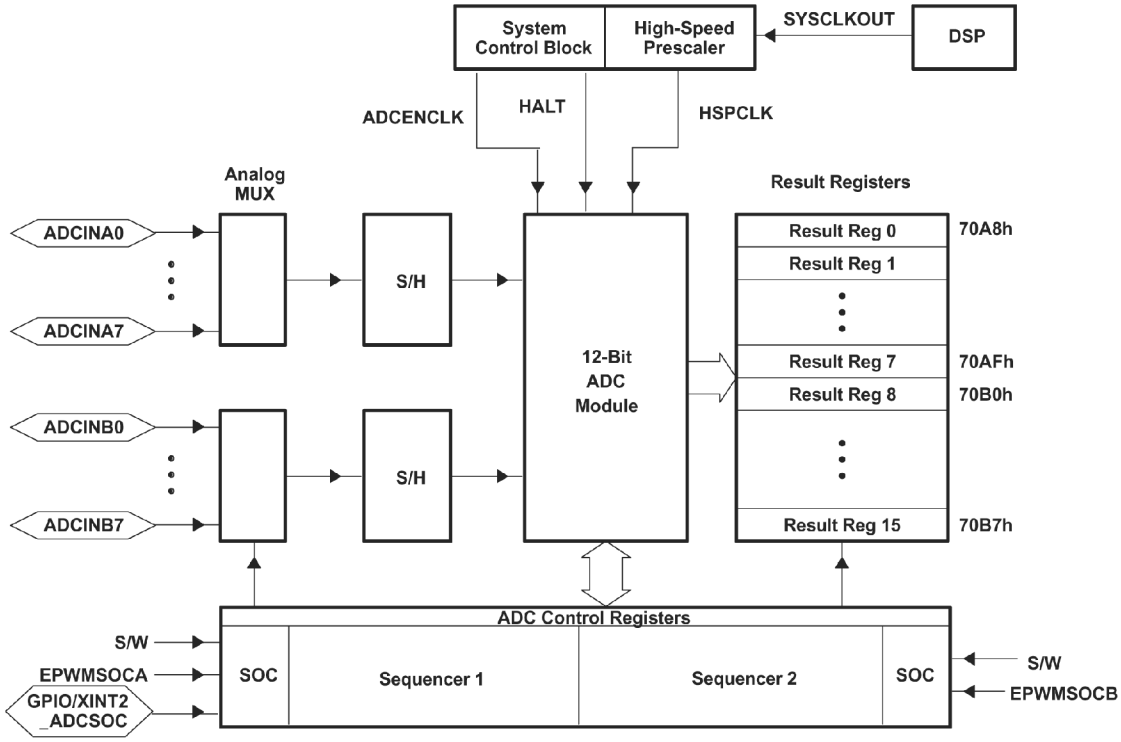
Obr. 6. Blokové schéma jednotky čítače/časovače



Obr. 7. Blokové schéma komunikačního rozhraní eCAN

1.9. A/D převodník

Na čipu všech DSP řady F28xx je integrovaný 12bitový rychlý A/D převodník s 2x S/H (Sample and Hold) obvody, což v praxi rozděluje 16 kanálů na dvě 8kanálové multiplexované větve s vlastním S/H. Vzájemnou synchronizaci obou větví řídí tzv. sekvencer (Sequencer). To umožňuje jednodušeji řešit převod například u stereofonního zvuku s jedním A/D převodníkem (*obr. 8.*). Každý kanál má vlastní registr pro uložení výsledku převodu. Na vstup je možné přivést napětí v rozsahu 0 až 3 V a výkon převodníku je 6.25 MS/s => převodní čas 160 ns při frekvenci A/D hodin 12.5 MHz.



Obr. 8. Blokové schéma A/D převodníku

1.10. Shrnutí

TMS320F28xx patří mezi nejmenší DSP v nabídce firmy Texas Instruments. Svým výkonem je však lze použít pro široké spektrum aplikací. Rozhodně bych po něm sáhl v případě, že již pro aplikaci výkonnostně nestačí žádný z běžných MCU. Myslím si, že právě TMS320F28xx je takový šikovný mezistupeň právě mezi MCU a "klasickými" velkými DSP.

2. Pulzně šířková modulace

Pulzně šířková modulace, neboli PWM (Pulse Width Modulation) je proces, při kterém dochází ke změně šířky pulsu nějakého nosného signálu (signálu, který má konstantní frekvenci, např. sinusový signál o frekvenci 1 MHz). Tohoto typu modulace se využívá např. při řízení elektromotorů, kde rychlost otáčení je přímo úměrná šířce pulsu, což je energeticky úspornější, než řízení pomocí reostatu.

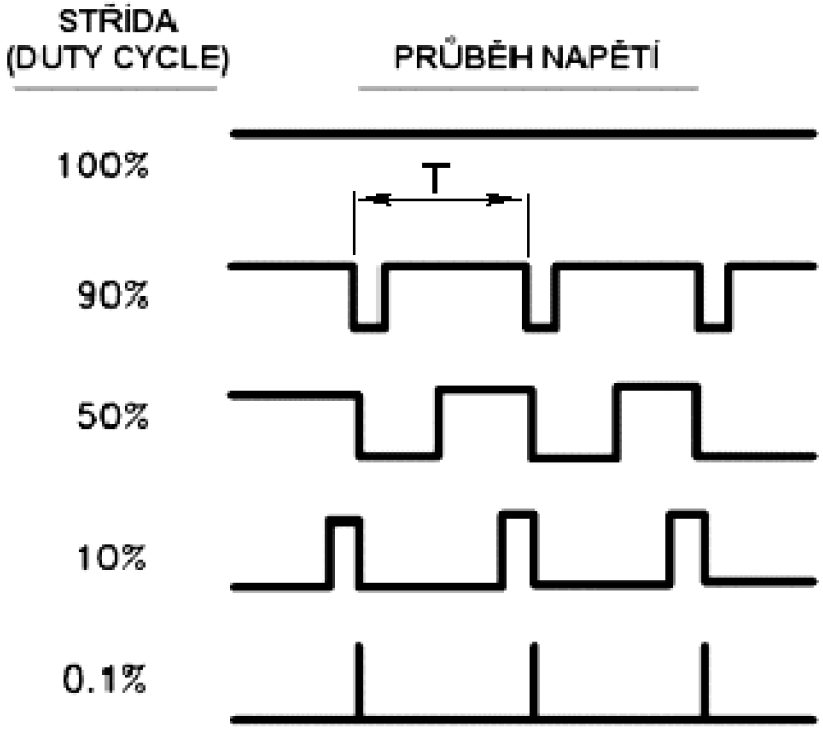
Zařízení, které PWM modulaci provádí, se nazývá pulzní šířkový modulátor.

2.1. Princip

Pulzně šířková modulace (dále jen PWM, angl. Pulse Width Modulation, zdroj [2]) je způsob kódování dat, která se přenášejí z vysílajícího zařízení k přijímacímu zařízení zvolenou přenosovou cestou. Přenosovou cestou může být pevné drátové spojení nebo bezdrátové spojení, kdy se data přenášejí vzduchem např. při IR komunikaci. Nás bude zajímat samotná pulzně šířková modulace, která reprezentuje sled bitů přenášeného rámce dat - jak ji generovat na straně vysílacího zařízení, příp. zpracovávat na straně přijímacího zařízení.

A jak takový signál kódovaný pulzně šířkovou modulací vypadá?

Jde tedy o signál s konstantní periodou T (vysvětlení na *obr. 9.,* případně [9]), kde se mění střída napětí (tj. poměr délky impulzu ku délce mezery uvažovaný v jedné periodě). Střída se uvádí někdy jako poměr (1:1,2:1,1:5 atd.), kdy je nutné uvést které číslo představuje impulz a které mezeru. Někdy se střída vyjadřuje procentuálně (100%,50%,0.1% atd.), kde 100% představuje ideální poměr 1:0, 50% poměr 1:1 atd. Poměr délky impulzu ku délce mezery bývá v zahraniční literatuře nazýván Duty Cycle.



Obr. 9. Diagram průběhů napětí

Využití pulzně šířkové modulace ve spojení s procesorem řady x51 je možné použít (mimo jiné) v těchto aplikacích:

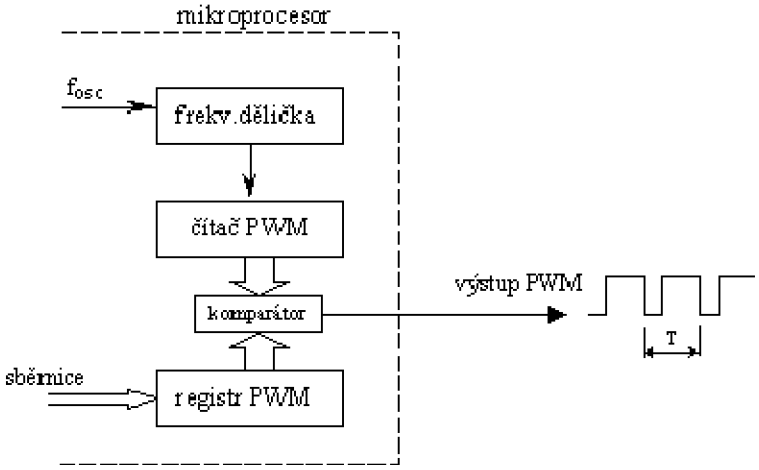
- řízení stejnosměrného motorku
  Generování PWM signálu na straně 8051, přijímací část představuje budič stejnosměrného motorku.
- zpracování dat z inteligentního snímače teploty
  Obecně zpracování PWM signálu na vstupu 8051, vysílajícím zařízením může být snímač, jehož výstupem je PWM signál. Příkladem může být snímač teploty SMARTEC SMT160, který funguje právě jako převodník teplota/střída.

- IR datový přenos
  Jeden z používaných druhů kódování při IR přenosu dat, 8051 se používá obvykle jako přijímající část, vysílající částí může být vysílací modul dálkového ovládání.

Pulzně šířková modulace se využívá i v konvenčním řízení otáček stejnosměrných motorů (nemusí zde být k řízení použit mikroprocesor), stejně tak i ve frekvenčních měničích pro řízení střídavých asynchronních motorů. Energetická úspora oproti řízení otáček změnou velikosti nap.napětí může být u velkých motorů značná. Také se tato modulace používá ve spínaných napájecích zdrojích.

2.2. Generování PWM signálu na výstupu 8051

Pokud budeme ve vyvíjeném zařízení potřebovat generovat PWM signál (uvažujme příklad pro ovládání stejnosměrného motorku, viz [2]), je vhodné se hned na začátku rozhodnout, jestli použijeme standardní typ mikroprocesoru a PWM signál budeme vytvářet softwarově, nebo použijeme mikroprocesor, který má už v sobě integrován obvod PWM. Přehled mikroprocesorů obsahujících hardwarově obvod PWM najdete na stránce výrobců. Použitím takovéhoto procesoru si můžete práci usnadnit, ale např. nyní hodně rozšířené mikroprocesory ATMEL AT89Cxxxx obvod PWM nemají a pak nezbývá než generování PWM signálu zajistit čistě softwarově. Obvod PWM integrovaný v mikroprocesoru funguje podle *obr. 10*.

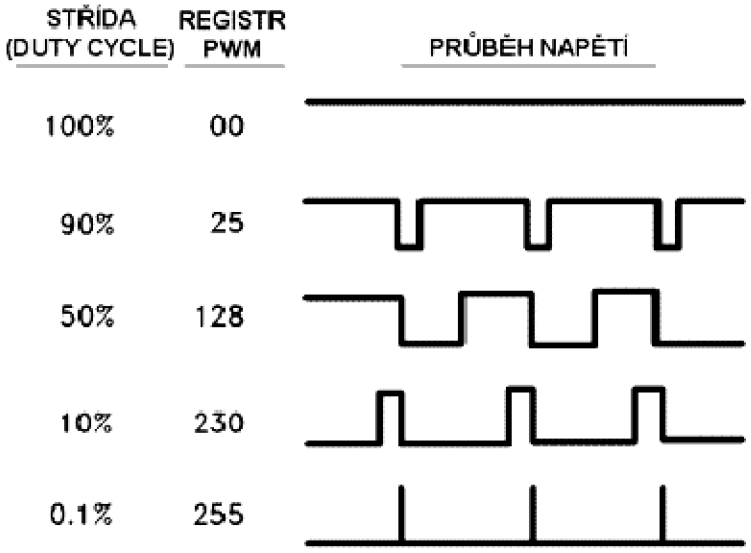


Obr. 10. Obecná struktura obvodu PWM

Do frekvenční děličky se přivádí hodinový signál f_{osc}, který se může ponechat nezměněn nebo se vydělí nějakou dělící konstantou. Dělící konstantu lze obvykle zvolit nastavením příslušného řídícího registru. Místo vnitřního hod.signálu lze u některých mikroprocesorů použít i externí zdroj hod. signálu. Čítač PWM je volně běžící čítač, který čítá hodinový signál z frekvenční děličky. Obsah tohoto čítače je komparátorem porovnáván s obsahem registru PWM, do kterého programově nastavíme požadovanou hodnotu. Na výstupu komparátoru dostáváme PWM signál - jsou-li obsahy čítače a registru stejné, je na výstupu log.0, jsou-li různé, je na výstupu log.1. Střída výstupního signálu je tedy úměrná hodnotě zapsané v registru PWM (přibližně podle *obr. 9.*). Perioda výstupního signálu je pevná a je dána čítaným signálem a modem čítače PWM.

Popíšeme si ještě obvod PWM u mikroprocesoru Dallas DC87C550, kde je tento obvod řešen trošku odlišně:

Mikroprocesor obsahuje 4 výstupy s 8-bitovou pulzně šířkovou modulací. Obvod PWM sestává se 3 částí: frekvenční děličky, hodinového generátoru a pulzního generátoru. Do frekvenční děličky přivádíme hodinový signál (můžeme použít i signál z externího zdroje). Nastavením bitů PWxS2, PWxS1, PWxS0 (v tabulce symbolicky PWxS2:0), kde x představuje číslo jednoho ze čtyř možných výstupů (0-3), se zvolí dělící konstanta dle tabulky níže.



Obr. 11. Diagram průběhů napětí

Výstup z děličky se přivádí na všechny čtyři přítomné hodinové generátory. Hodinovým generátorem je zde 8-bitový čítač s automatickým přednastavováním, který udává periodu výstupního signálu. Pulzní generátor představuje 8-bitový časovač časovaný signálem z 8-bitového čítače. Procesor umožňuje kaskádové spojení dvou PWM obvodů, čímž dostaneme 16-bitový PWM signál. To nám umožní dosáhnout většího rozlišení (např. při řízení otáček ss motorku jemnější nastavení). Jinými slovy - když budeme mít k dispozici 8-bitový pulzně šířkový modulátor, můžeme měnit rychlost otáčení motorku maximálně v 255 krocích (rozlišení 0,4%).

3. Vývojový kit eZdsp

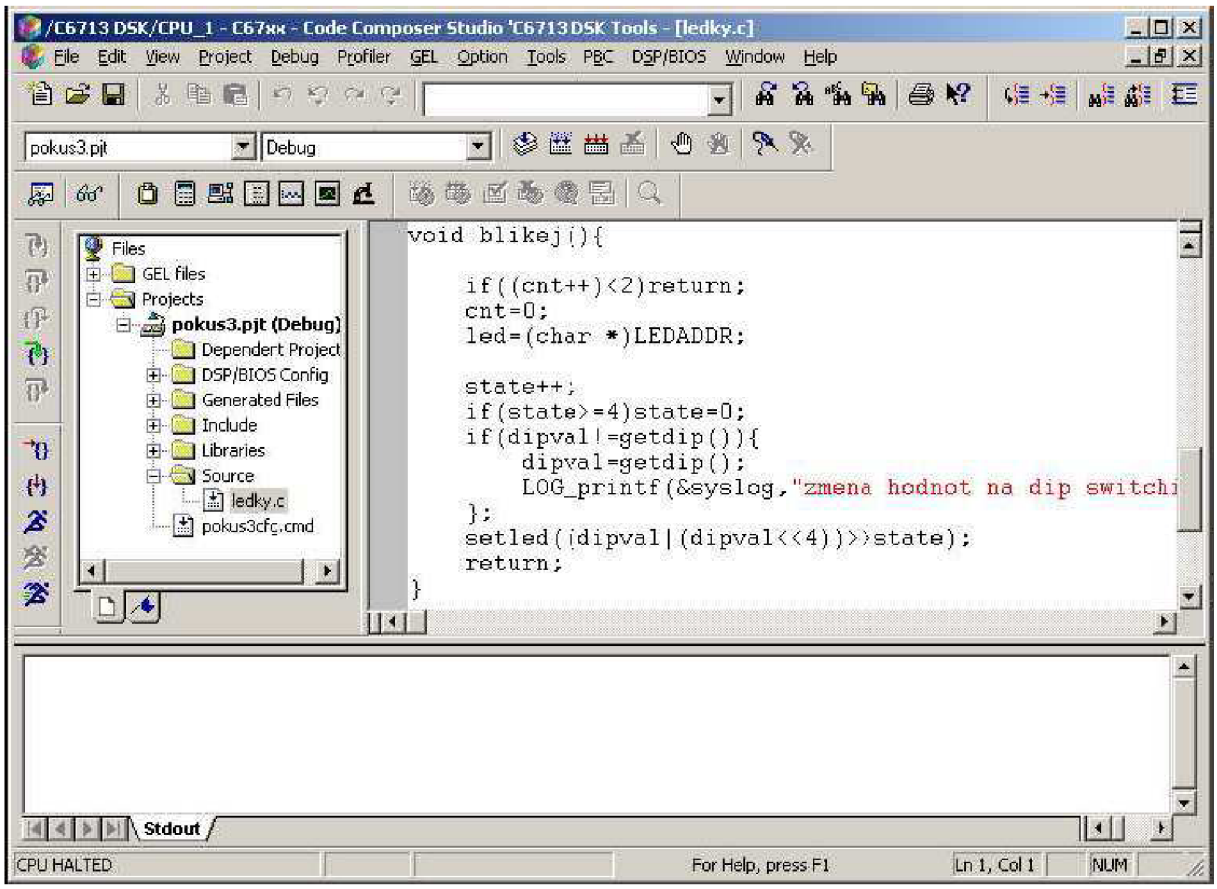
Z řady výrobků firmy TI jsem pro svou práci vybral procesor typ TMS320F2808 (viz *kap. 1.*). Abych předešel problémům s osazováním součástek, kompilací programů, driverů, objednali jsme od výrobce tzv. vývojový kit. Jedná se o hotovou desku, přímo určenou k programování čipu pomocí osobného počítače. Konkrétně produkt TMS320F2808 eZdsp Starter Kit (DSK), [1], [15].

Tato deska obsahuje samotný procesor, pro uložení dat se používají SDRAM a FlashEEPROM paměti, pro debugging 4 přepínače DIP a LED dioda. Pro analogový výstup je k dispozici stereofonní zvukový kodek, v našem případě nevyužitý. Pro komunikaci s nadřazeným počítačem je použité USB s emulací JTAG. Tento konektor je vyveden zvlášť. Konfigurace desky se provádí konfiguračními přepínači. Rozšiřování se provádí pomocí 3

externích konektorů – HPI, EMIF a Peripherial Extension konektorů. Karta je napájena pomocí adaptéru DC napětím.

3.1. Code Composer Studio (CCS)

Code Composer Studio je software pro vývoj v jazyce C++ a assembleru. V podstatě jazyk ANSI C se specifickými rozšířeními pro ovládání I/O. Program je určen pro vývoj aplikací formou projektů. Tyto lze navíc mezi sebou kombinovat. Prostředí (kopie okna *obr. 12.*) má integrovaný debugger se standardními možnostmi. Debugger navíc dokáže data zobrazit graficky, což může usnadnit ladění algoritmů, včetně grafických závislostí výstupních veličin.



Obr. 12. Ukázka aplikace CCS (kopie okna)

Code Composer Studio lze propojit s libovolným programem přes COM knihovny. Obdobně je také řešeno propojení programu Matlab s CCS, MS Excel apod.

CCS má u sebe řadu komponent pro efektivnější tvorbu software.

- knihovny pro jednotlivé DSP procesory
- knihovny pro čipy, integrované do pouzdra s DSP
- knihovny pro standardní externí hw
- knihovnu DSP/BIOS (dodávaný RTOS) + konfigurační nástroje
- výstup log do okna debuggeru
- měření zátěže CPU debuggerem
- XML rozhraní pro programování DSP
- zobrazování prováděcího grafu debuggerem

4. Frekvenční měnič kmitočtu

Frekvenční měnič kmitočtu (nebo také měnič frekvence, často nesprávně nazývaný frekvenční měnič) je zařízení, které slouží k přeměně elektrického proudu s určitým kmitočtem na elektrický proud s jiným kmitočtem. Dříve byl realizován jako rotační měnič, dnes se používají spíše elektronické obvody a moderní výkonové polovodičové součástky. Z nich je vytvořen usměrňovač a střídač, případně další řídící a stabilizační elektronika.

Pro svou práci jsem jako nejvhodnější vybral měnič výrobce Siemens typu Sinamics G110 (*obr. 13*), literatura [20]. Díky optimálnímu výkonu (viz *tab. 4.* v Příloze) se jevil jako nejvhodnější pro náš třífázový motor, zároveň disponuje širokou škálou připojitelnosti a ovládání, jak přes klávesnici, tak přes PC, dokonce i přes analogové vstupy. Frekvenční měnič se postupně stává přirozenou (a ekonomicky výhodnou) součástí pohonu s ASM (někdy bývá integrován v samotném motoru).

4.1. Podrobný popis produktu

Frekvenční měniče Sinamics G110 jsou novinkou v řadách měničů Siemens. Byly uvedeny jako reakce na požadavky jednoduchých měničů s nízkými pořizovacími náklady. Nízká cena ovšem zdaleka není jedinou výhodou - vysoký spínací kmitočet a pasivní chlazení snižují hlučnost pohonu, jednofázové vstupní napětí a možnost osazení filtrem třídy B umožňují nasazení měniče ve všech prostředích včetně domácího, výběr z varianty s analogovým nebo USS šetří Vaše peníze a přitom neomezují možnosti použití. Čerpáno z literatury [20].

Mezi další důležité vlastnosti G110 patří např.: jednoduchá instalace, nastavení a montáž, široké možnosti nastavení provozních parametrů jako je nastavení rozběhové a doběhové rampy, synchronizace na otáčející se motor, automatický restart v případě poruchy nebo přerušení dodávky proudu, rychlé proudové omezení FCL reagující na náhlé změny zátěže bez vyhlašování poruch, možnost stejnosměrného brzdění, možnost osazení panelem BOP pro nastavení a zobrazení hodnot, možnost klonování nastavených parametrů na více měničů pomocí panelu BOP, možnost provozu na vstupním kmitočtu 50Hz nebo 60Hz - volba přepínačem a další.

Druhy ochrany: proti přepětí a podpětí, proti zkratu zemnímu i mezi fázemi, chodu bez zátěže, proti přehřátí motoru i měniče.

Metody řízení: lineární, kvadratická a vícebodová charakteristika

Druhy řízení: externí, interní

Teoreticky lze funkci samotného měniče popsat následovně: mikroprocesor, který vše ovládá, využívá supermoderní bipolární tranzistory s izolovaným hradlem (IGBT), díky čemuž jsou spolehlivé a univerzální, blokové schéma je patrné z obrázku (*obr. 14*). Za pomocí metody pulzně šířkové modulace (PWM) s přepínatelným spínacím kmitočtem je dosaženo tichého a rovnoměrného chodu motoru. Ochranné funkce zajišťují dokonalou ochranu motoru a měniče.

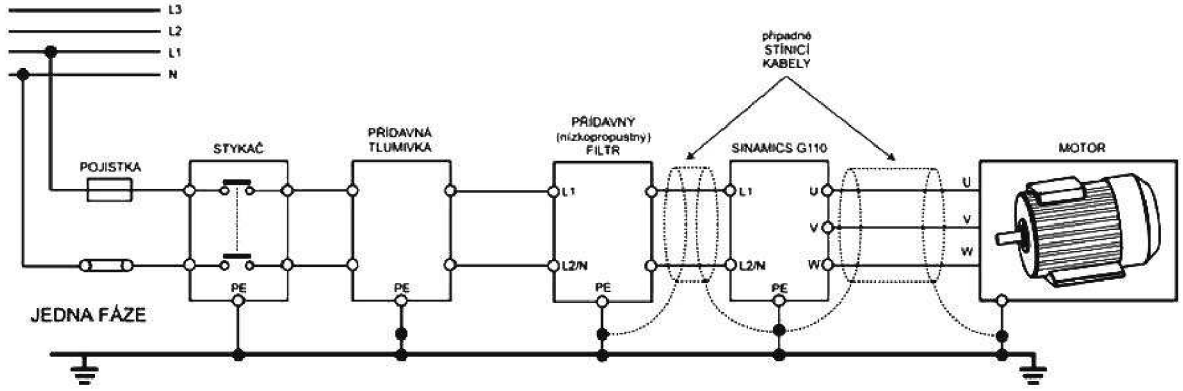
Měnič pracující ve standardním továrním nastavení je ideální volbou pro celou řadu jednoduchých aplikací řízení pohonu s jednoduchou U/f charakteristikou. Pomocí širokého spektra programovatelných parametrů, které jsou součástí měniče, lze přístroj adaptovat pro

široký okruh aplikací. Parametry lze měnit buďto pomocí univerzálního sériového rozhraní (USS) nebo pomocí ovládacího panelu (OP).

Obr. 13. Frekvenční měnič kmitočtu

4.2. Návrh zapojení

Po prostudování teoretických poznatků, výhod a nevýhod problému jsem přistoupil k částečné realizaci projektu. Základem byla deska rozvaděče (viz *obr. 26* v Příloze *kap. 10.2.*), na kterou jsem přišrouboval frekvenční měnič kmitočtu, třífázový motor a vývojový kit procesoru DSP. Všechny potřebné vodiče byly z bezpečnostních důvodů vedeny vespod desky. Zapojení bylo rozděleno na výkonovou a řídící větev. Výkonová větev spočívá v připojení napájecího napětí k měniči a následně motoru, s připojením adaptéru DC napětí k vývojové desce DSP. Řídící větev je složena z kabelu spojující DSP s počítačem přes sběrnici USB a druhého kabelu, spojujícího DSP s měničem. Celé pracoviště má tuto podobu:



Obr. 14. Blokové schéma frekvenčního měniče kmitočtu

4.3. Možnosti řízení

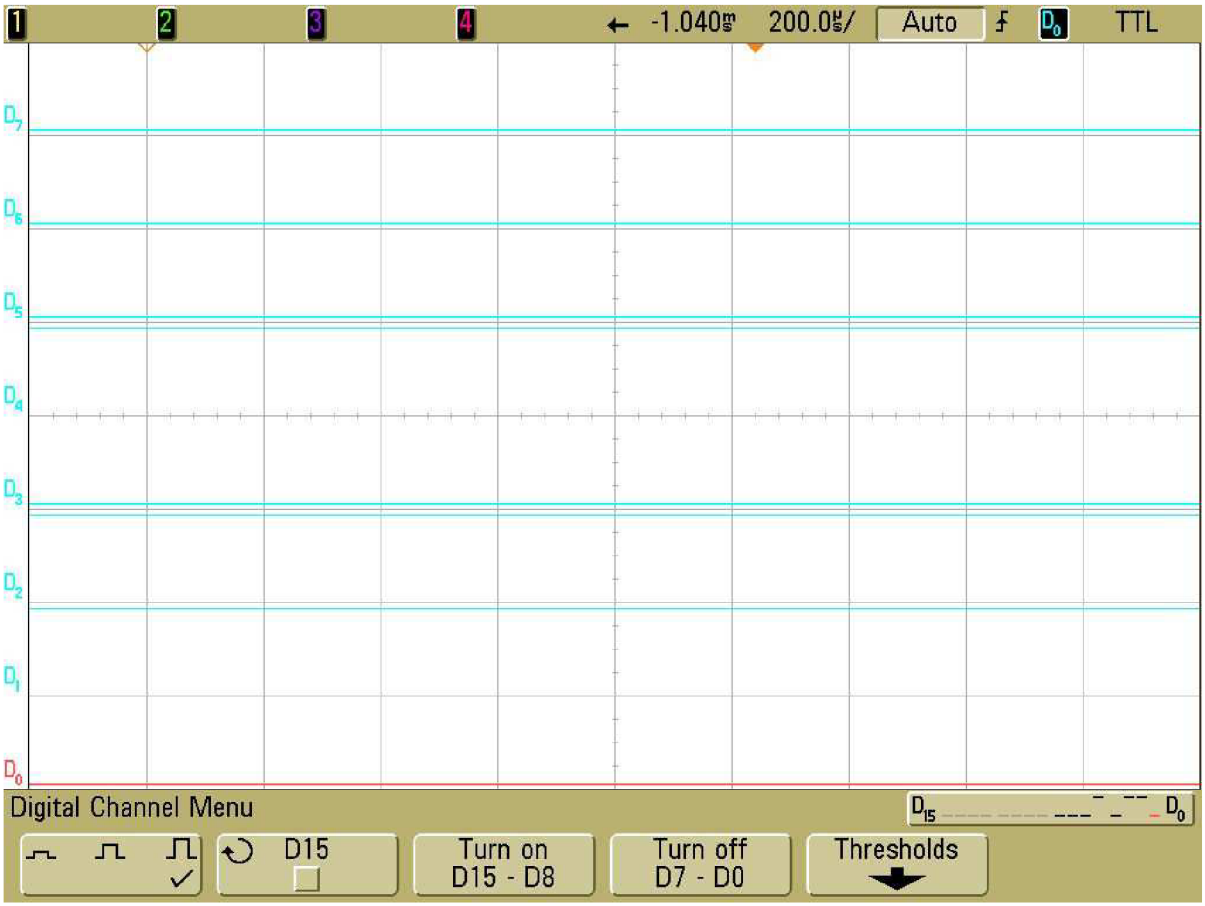
Po sestavení pracoviště jsem se pustil do experimentálního ověření funkce všech zařízení, vše pod odborným dohledem vyučujícího. V osobním počítači jsem spustil program CCS (více *kap. 3.1*), za pomocí webu výrobce [20] a uvedených knihoven [1] sestavil jednoduchý program pro obsluhu přerušení v kombinaci s generováním PWM signálu (samotný program viz *kap. 10.3* v Příloze). Po úspěšné kompilaci a nahrání do Flash paměti (vlastní programování čipu DSP) jsem za pomoci speciálního osciloskopu vybaveného digitálním vstupem ověřil, že se na určených výstupních I/O pinech desky objevuje obdélníkový signál, resp. průběh symbolizující log.1 či log.0 (dle *obr. 9,* střída 100%). Tím jsme ověřili funkčnost řídící části pracoviště a přistoupili ke komunikaci mezi dílčími zařízeními. Kopii okna osciloskopu vidíme na *obr. 15.*

Jak je patrné z uvedeného blokového schématu (*obr. 17*), měnič disponuje dvěma druhy řízení. Jedná se o digitální a analogové řízení. Mým cílem je za využití právě digitálního řízení nastavovat, resp. programovat tento měnič.

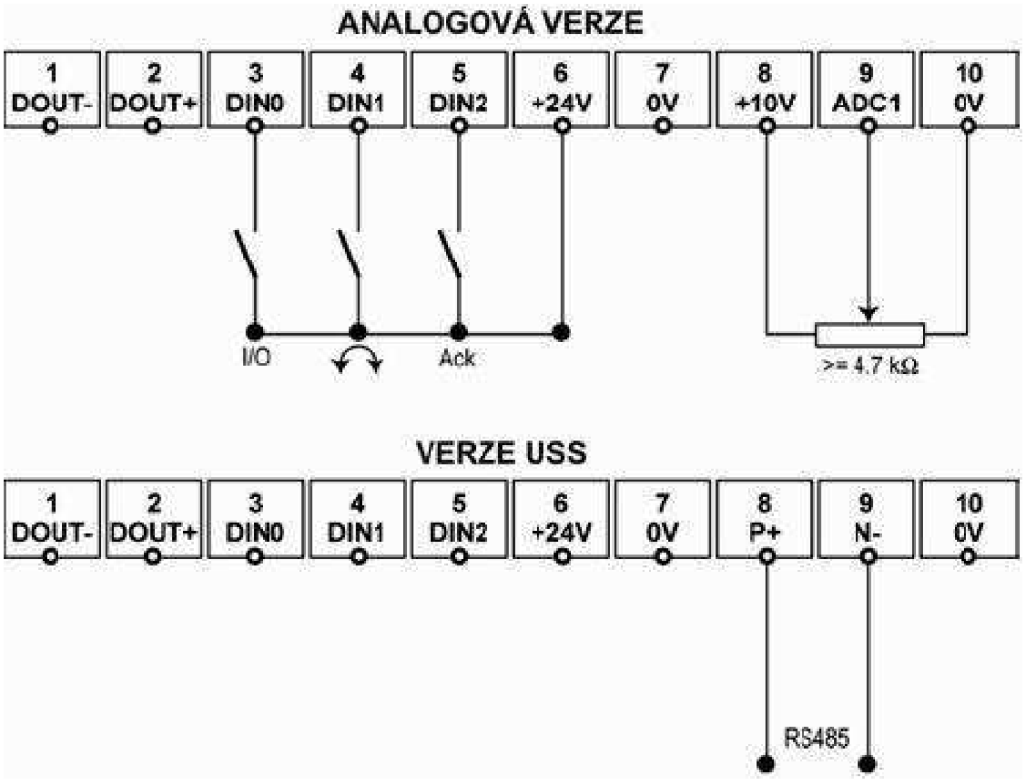
Bohužel jsem po prostudování manuálu dospěl k závěru, že tento měnič lze sice digitálně řídit, avšak pouze za pomocí speciální sběrnice (USS), která že nejen není kompatibilní s běžnými datovými standardy (obousměrné toky dat Tx, Rx, …), ale navíc vychází z příkazů a parametrů daných výrobcem pro tento účel. Závěrem lze říci, že pro digitální řízení měniče je zapotřebí komunikační kabel, disponující sběrnicí USS (na měniči) a USB (pro osobní počítač), v kombinaci s obslužným programem, rovněž dodávaným výrobcem. Pro zachování výrobního tajemství je nemožné výše zmíněné produkty poupravit pro naše potřeby.

Napadla nás též myšlenka vyřadit zmíněný řídící obvod a komunikovat přímo s tranzistory IGBT (viz *kap. 4.1*), kde bychom logickými signály z PWM modulace generované procesorem DSP spínali hradla v těchto tranzistorech namísto řídící logiky (motivace viz [8]). Avšak tento zákrok byl technicky nemožný vzhledem k obrovské složitosti tištěného spoje

Abychom vůbec ověřili funkčnost frekvenčního měniče kmitočtu a správnost zapojení střídavého motoru, využil jsem možnost analogového řízení. Princip spočívá v pouhém mechanickém spínaní, tedy v přívodu externího napětí 24 V na jednotlivé svorky. Více je patrné z obrázku (*obr. 16*)



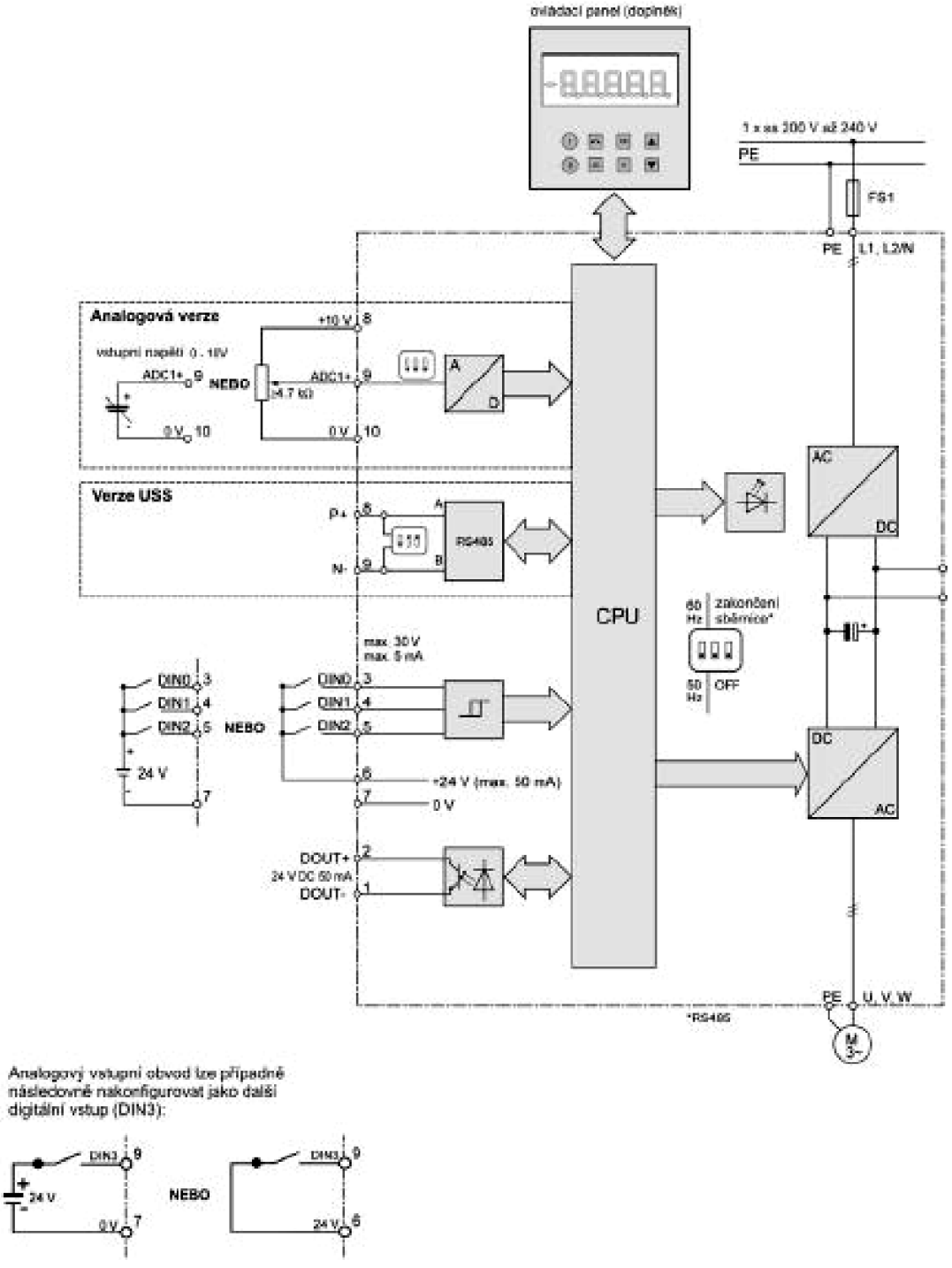
Obr. 15. Obrazovka osciloskopu



Obr. 16. Analogové řízení

Toto řízení však disponuje omezenými možnostmi. Umožňuje pouze rozběh/zastavení motoru, regulaci otáček potenciometrem, reverzaci, případně resetováním vnitřního nastavení v případě poruchy.

Využil jsem i Hallova snímače umístěného v blízkosti volného konce hřídele motoru k měření otáček a zobrazení časové závislosti otáček motoru na obrazovce osciloskopu.

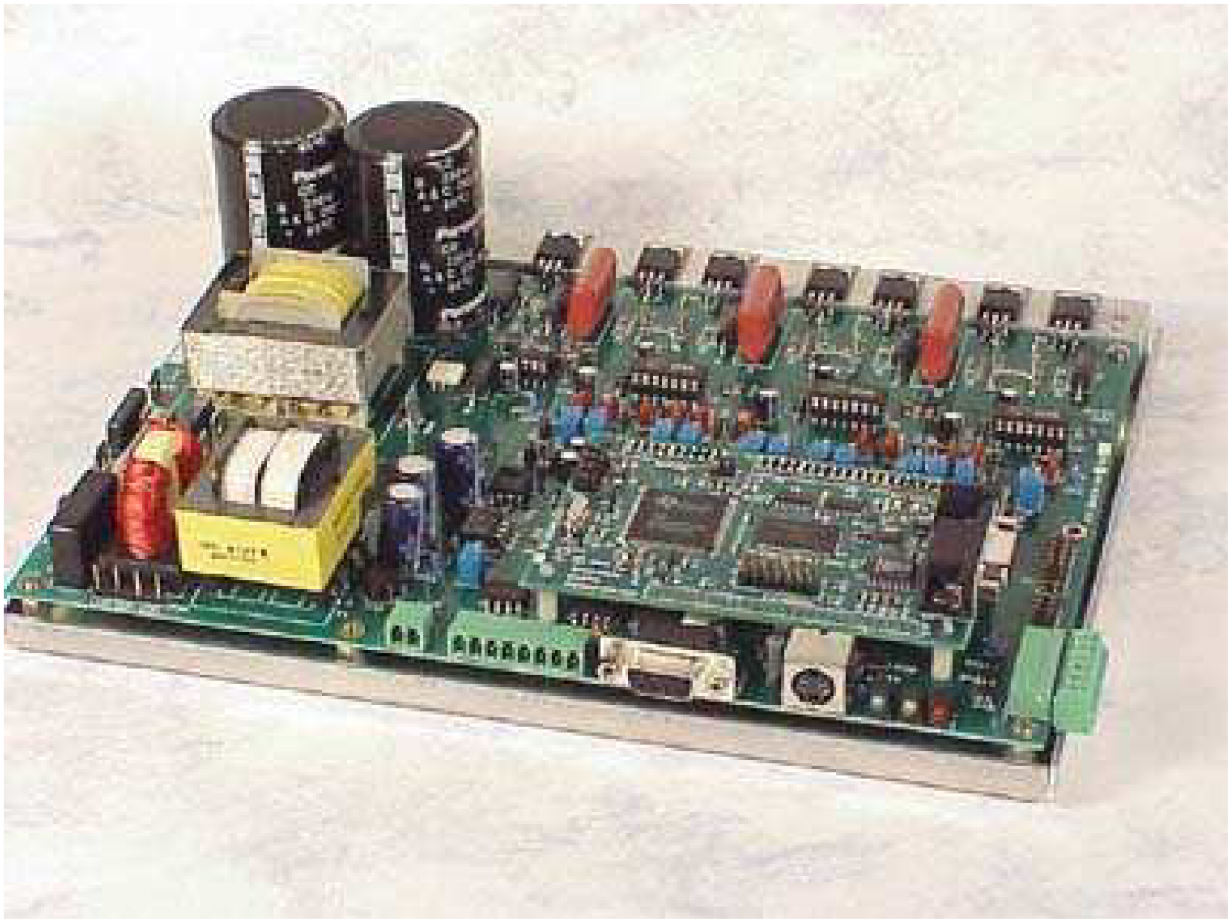


Obr. 17. Blokové schéma frekvenčního měniče kmitočtu

5. AC/DC měnič DMC1500

Po předchozích nezdarech s frekvenčním měničem kmitočtu Siemens, který je brán spíše jako koncové zařízení pro zákazníka a nelze řídit pomocí DSP procesoru, jsem využil měniče kmitočtu DMC1500 výrobce Spektrum Digital (ukázka zařízení na *obr. 18.*). Tento měnič je nejen určen pro experimentální využití a měřící účely, ale i plně kompatibilní a spolupracující s naším vývojovým kitem eZdsp. Osazení plošného spoje jednotlivým součástkami ukazuje *obr. 27.* (v Příloze *kap. 10.5.*).

Pro tento měnič výrobce (viz [1], [10], [15]) dodal nespočet volně stažitelných ukázkových knihoven programů, pro všemožné využit. Od krokování motoru, přes měření napětí, proudu, otáček a vibrací, k simulaci různých poruch motoru (stator, rotor, mechanické poruchy, …), s možností měnit podmínky provozu (zatížení, rozvážení napětí v síti, snížené/zvýšené napětí, směr otáčení, aj.).

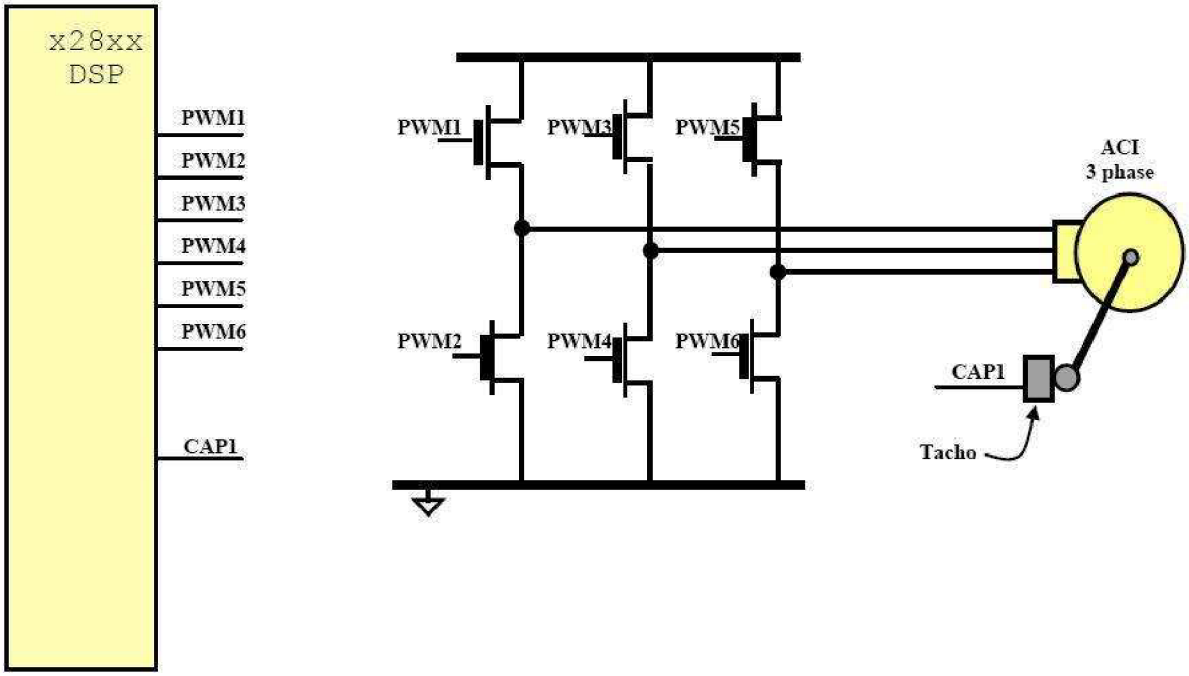


Obr. 18. Frekvenční měnič kmitočtu DMC1500

5.1. Realizace obvodu

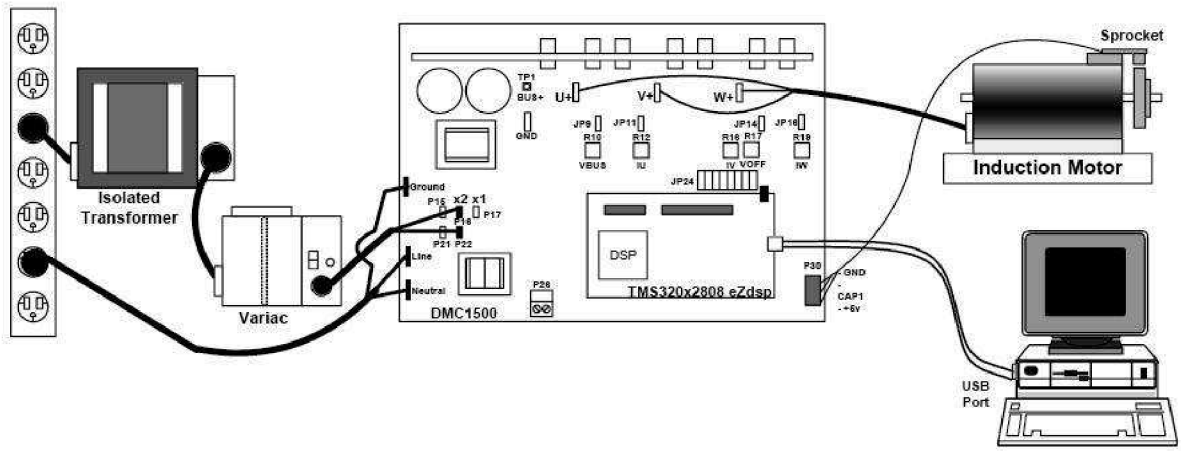
Princip činnosti vychází opět z PWM modulace, kde výstupní signál bude řídit spínaní tranzistorů. Probíhající děj je patrný z obrázku (*obr. 19*).

O zapojení vypovídá obrázek (*obr. 20*). Frekvenční měnič kmitočtu DMC1500 disponuje konektory pro připojení vývojového kitu eZdsp a oba obvody pak tvoří jeden celek.



Obr. 19. Funkce obvodu

Výstupem jsou vodiče napájející motor, výkonovým vstupem je napájecí napětí, datovým USB kabel pro komunikaci s počítačem



Obr. 20. Výsledné zapojení

6. Aplikace vlastního řízení

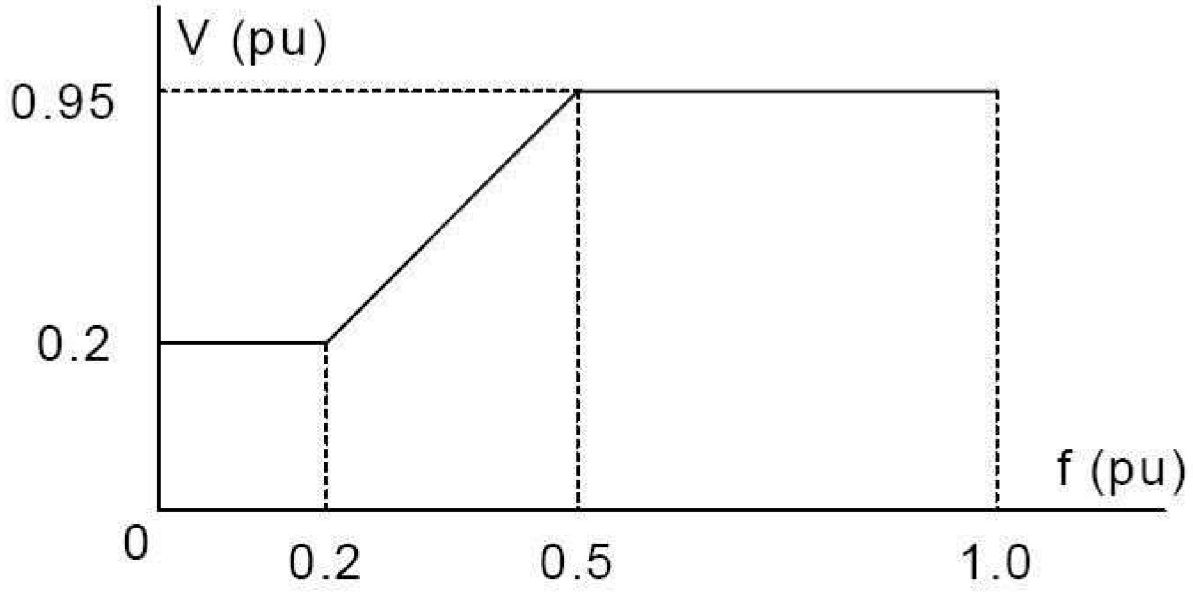
Při běhu programu je krok z krokem vyhodnocován zdrojový kód programu a dle předem stanovených pravidel přiřazuje dílčím hodnotám HEX kódu hodnoty výstupního napětí, jak je patrné z *tab. 3*. podle literatury [10].

Freq		VoltOut	
Actual value (pu)	Q24* (Hex)	Actual value (pu)	Q24* (Hex)
0.1	0199999h	0.2	0333333h
0.3	04CCCCDh	0.45	0733333h
0.4	0666666h	0.70	0B33332h
0.8	0CCCCCDh	0.95	0F33333h

* Assuming GLOBAL_Q be Q24.

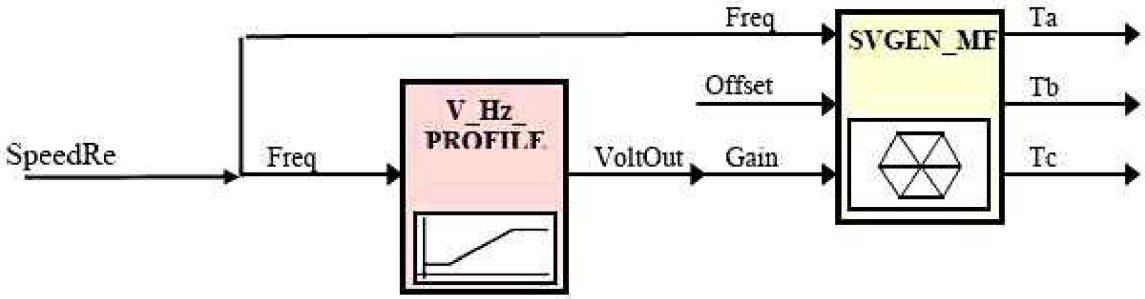
tab.3. Vyhodnocení kódu

Lze tedy pomocí jednoduché změny číselné hodnoty výstupního napětí přímo ovlivnit frekvenci, resp. otáčky motoru. Frekvenční závislost průběhu výstupního napětí ukazuje *obr.21.*



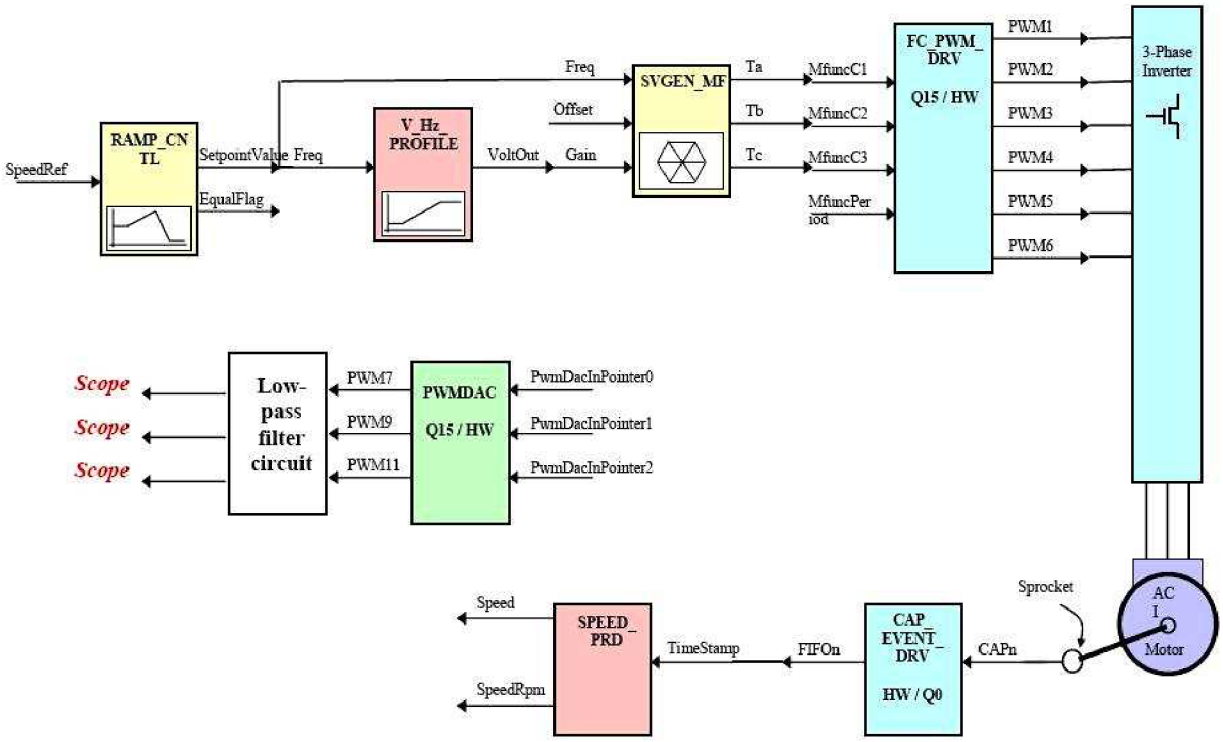
Obr. 21. Závislost U/f

Principielně děj vyjadřuje blokové schéma (*obr. 22)*



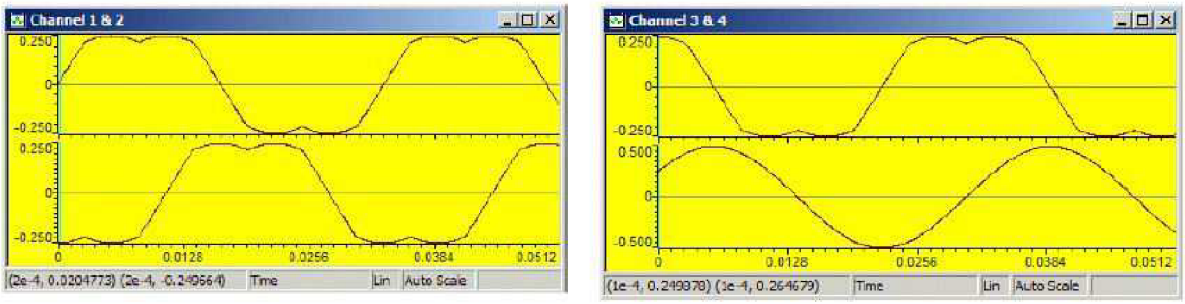
Obr. 22. Řízení otáček

Výstupy Ta, Tb a Tc jsou dále brány jako vstupní impulzy pro řízení čipu ovládající PWM modulaci, která je klíčová pro samotný běh motorku. Uvedené blokové schéma na *obr. 23.* vysvětluje celou problematiku.



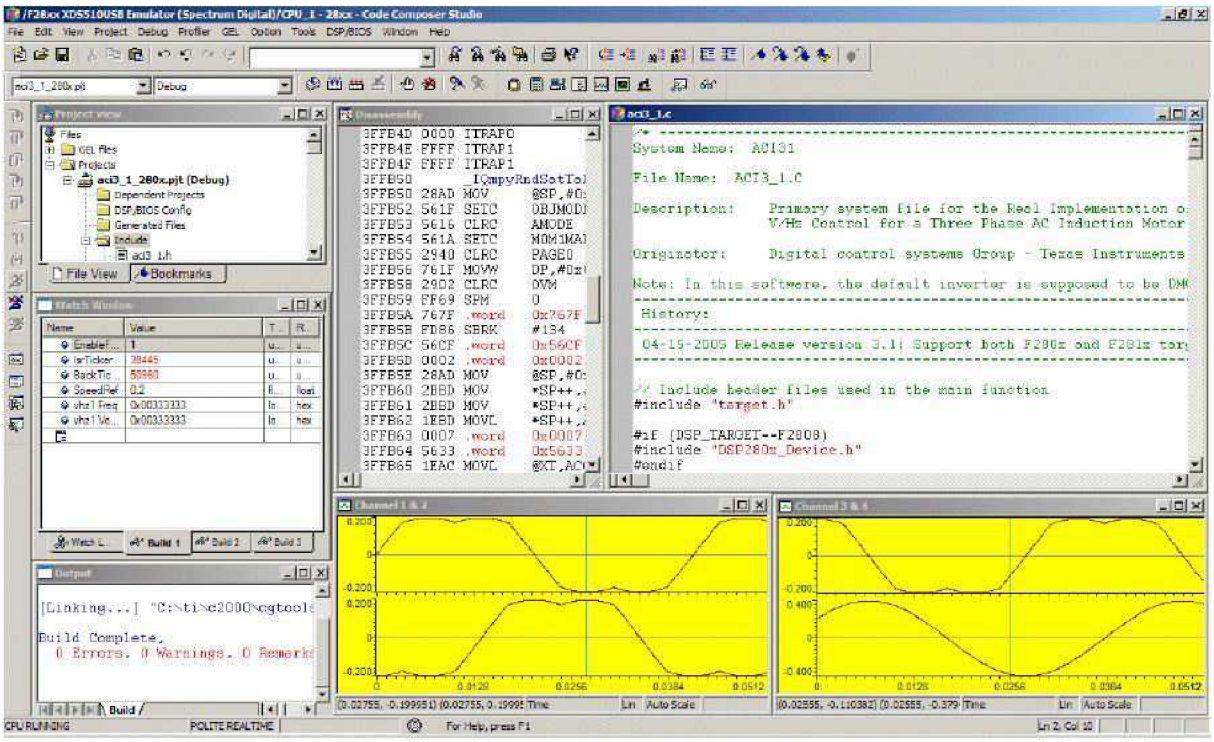
Obr. 23. Blokové schéma řízení

Samotný program CCS umožňuje veškeré časové průběhy zaznamenat a z výsledků vyhodnocovat grafické závislosti. V Příloze (*kap. 10.4.*) najdeme jednoduchý program pro řízení otáček.



Obr. 24. Průběhy jednotlivých výstupů

Vlastní program má podobu (viz *obr. 25*). Lze libovolně měnit jednotlivé hodnoty, jejichž variací dosáhneme změn v otáčkovém spektru, směru otáčení, krokování chodu a mnoho dalších. Měnič DMC1500 pro experimentální využití umožňuje i komunikaci s mnoha vedlejšími prvky, jako jsou snímače otáček, vibrací, teploty a tím je možné vybudovat rozmanité testovací pracoviště.



Obr. 25. Okno obrazovky

7. Závěr

Cílem této úlohy bylo podrobně prostudovat problematiku řízení měniče s použitím moderního procesoru DSP typové řady TMS320F2808 výrobce Texas Instruments Inc. Problém byl koncipován na řízení malého asynchronního třífázového motorku zapojeného přes frekvenční měnič kmitočtu. Protože testovaný frekvenční měnič kmitočtu nedisponoval možností externího řízení a programování, bylo využito měniče pro experimentální využití. Pomocí vývojového kitu a CCS studia pro procesory DSP byl sestaven jednoduchý program pro generování PWM signálu, který má za pomoci logických členů řídit běh motoru. Vhodné bylo ověření funkce pomocí snímače otáček, zejména pro porovnání hodnot se zobrazenými v programu. Avšak tento výkonový člen DMC1500 nesplňoval naše požadavky a nepracoval korektně, museli jsme se obrátit na jiný model tohoto koncového prvku. Náhradní modul se ale bohužel nepodařilo včas zajistit.

Závěrem lze říci, že veškeré třífázové střídavé elektromotory je možné externě řídit. První variantou je frekvenční měnič kmitočtu s vlastním řízením, zejména vhodný pro koncové zákazníky. Druhou možností je řízení pomocí rychlých procesorů DSP v kombinaci s výkonovým a programovatelným měničem. Popis, porovnání a ověření funkcí obou variant je obsahem této práce.

8 Seznam literatury

[1] http://www.ti.com/ - internetové stránky výrobce Texas Intruments

[2] http://cs.wikipedia.org/wiki/Hlavn%C3%AD_strana – virtuální encyklopedie

[3] GESCHEIDTOVÁ, E., STEINBAUER, M. *Klasifikace a vlastnosti automatizovaných měřících systémů.* Elektrotechnický magazín, 1999, č. 10-11, s. 24-27. ISSN 1210-5422 .

[4] BARTUŠEK, K., HEGT, GESCHEIDTOVÁ, E., STEINBAUER, M. *Měření v elektrotechnice – návody k laboratorním cvičením.* Skriptum. Brno: FEKT VUT v Brně, 2006. ISBN 80-214-3265-9.

[5] GESCHEIDTOVÁ, E., REZ, J., STEINBAUER, M. *Měření v elektrotechnice.* 1. vydání. Brno: Nakladatelství VUTIUM, 2002. ISBN 80-214-1990-3.

[6] DĚDKOVÁ, J. *Modelování elektromagnetických polí.* Skriptum. Brno: FEKT VUT v Brně, 2005.

[7] DĚDEK, L., DĚDKOVÁ, J. Elektromagnetismus. Skriptum. Brno: Nakladatelství VUTIUM, 2000.

[8] ŠEBESTA, V., SMÉKAL, Z. *Signály a soustavy.* Elektronické skriptum. Brno: FEKT VUT v Brně.

[9] JURA, P. *Signály a soustavy.* Elektronické skriptum. Brno: FEKT VUT v Brně.

[10] Spectrum Digital Inc. *DSK6713 Support Home* [online] [cit. 2004-10-11] http://c6000.spectrumdigital.com/dsk6713/

[11] Berkeley Design Technology Inc. *Inside DSP* [online] 2003-2005. [cit. 2004-10-20] http://insidedsp.com

[12] *TMS320C6713, TMS320C6713B Floating-Point Digital Signal Processors*. Texas Instruments, 2004. [cit. 2005-1-1]

[13] *TMS320C6000 CPU and instruction set reference guide*. Texas Instruments, 2000. index SPRU189F, [cit. 2005-1-1].

[14] *TMS320C6000 Peripherial overview reference guide*. Texas Instruments, 2004. Index SPRU190G, [cit. 2005-1-1]

[15] *TMS320 Programmer's guide*, Texas Instruments, 2002. Index SPRU198G, [cit. 2005-1-1]

[16] T*MS320C6013 Errata*, Texas Instruments, 2004. Index SPRZ191G, [cit. 2005-1-1]

[17] *TMS320 Cross platform daughterboard specification*. Texas Instruments, 2000. Index SPRA711, [cit. 2005-1-1]

[18] *TMS320C6713 Hardware designer's reference guide*. Texas Instruments, 2004. Index SPRA33, [cit. 2005-1-1]

[19] Varella, L.R., Aparício, J.N., Silva, S.C.*A schedulling web service based on XMLRPC*, 2001

[20] http://www.kvelb.cz/ - internetové stránky prodejce výkonové elektroniky

9 Použité zkratky a symboly

A/D	analogicko/digitální převodník
BIOS (Basic Input-Output Systém)	základní programové vybavení osobního počítače
CAN (Controller Area Network)	systémová sběrnice
CCS (Code Composer Studio)	prostředí pro programování mirokontrolérů
CLK	interní generátor hodinových signálů
CMOS	integrovaný obvod
CPU	centrální procesová jednotka uvnitř procesoru
DC	stejnosměrné napětí
DIP	druh tlačítkového přepínače
DSP	dynamický signálový procesor
EEPROM (Erasable Programmable Read-Only Memory)	přepisovatelný typ paměti
eCAP	4-úrovňový modul zachytávání
eZdsp	vývojový kit obsahující DSP procesor
eQEP	kvadraturní modulace
FCL	rychlé proudové omezení
FET	typ tranzistoru
FIR	číslicové filtry pro zpracování digitálních signálů
FLASH	Flash paměť typu RAM
GPIO (General Purpose Input/Output)	piny, které mohou být jako vstupy i jako výstupy
I/O (Input/Output)	vstup/výstup
IGBT	bipolární tranzistor s izolovaným hradlem
IR (InfraRed)	infračervené záření
JTAG	komunikační standard IEEE 1149.1
MAC (Multiply-Accumulate)	suma součinů
PC	osobní počítač
PLL (Phase Locked Loop)	smyčka fázového závěsu
PWM (Pulse Width Modulation)	pulzně šířková modulace
RAM (Random-Access Memory)	paměť s libovolným (náhodným) přístupem
ROM (Read-Only Memory)	paměti, jejíž obsah nelze přepsat běžným způsobem.
TI (Texas Instruments)	výrobce mikropočítačů
USB (Universal Serial Bus)	univerzální sériová sběrnice
USS	sériové rozhraní pro komunikaci

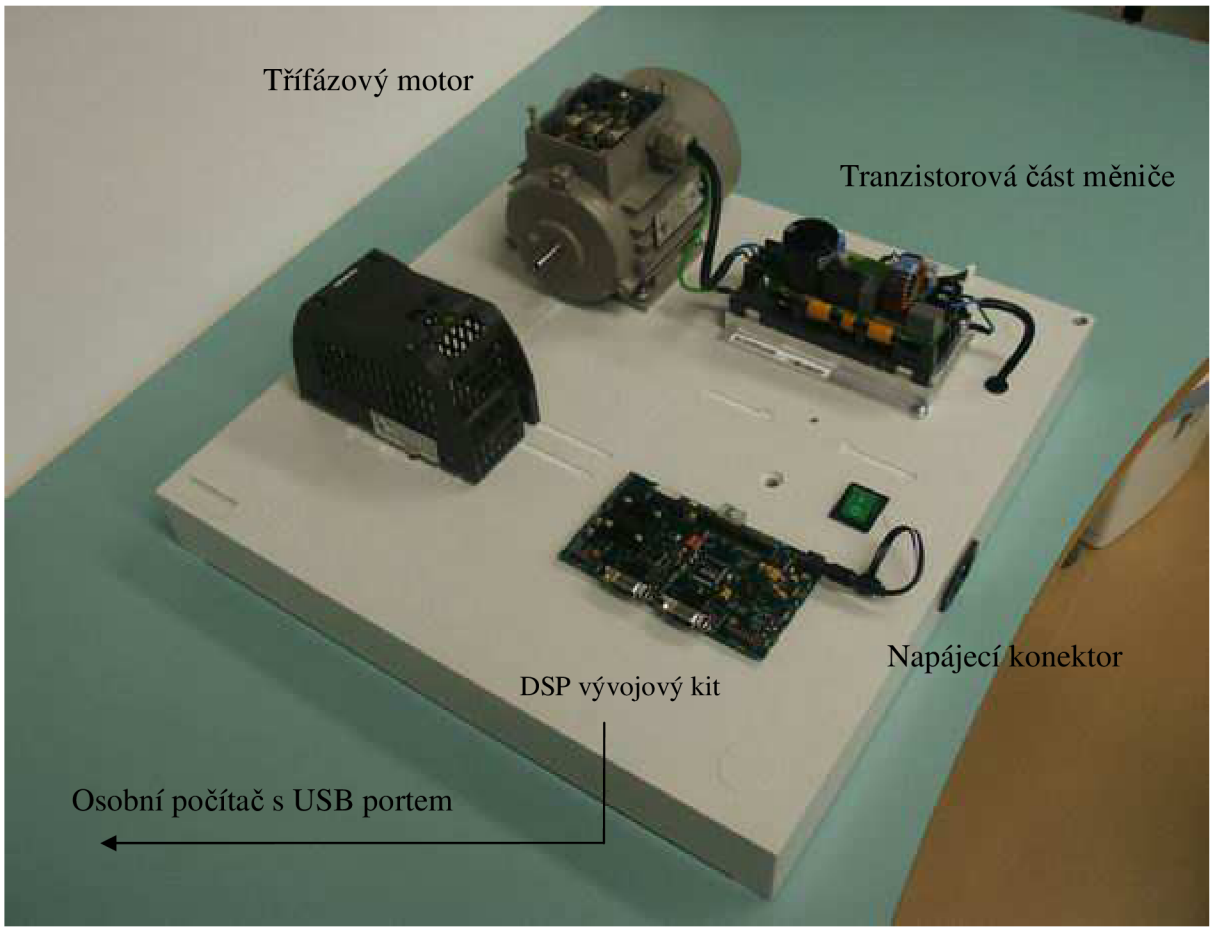
10 Příloha

10.1. Technické parametry frekvenčního měniče kmitočtu

Výkon (kW) pro 4-pól. Motor	0,12
Vstupní napětí (V)	1x200 – 240 +/-10%
Max. výstupní napětí Umax (V)	3x240
Max. výstupní proud (A)	0,9
Kmitočet min-max (Hz)	0 – 650
Vstupy digital/analog (počet)	3 – 4 / 1 verze analog, 3 / 0 verze USS
Výstupy digital/analog (počet)	1 / 0
Přetížitelnost (%/čas)	150% / 60s
Modulační kmitočet (kHz)	2 – 16 (přednastaveno 8)
Brzdné režimy	stejnosměrný
Sady parametrů (počet)	1
Řízení výstupu – lineární	ANO
Řízení výstupu – FCC	NE
Řízení výstupu – vektorové	NE
technologický regulátor	-
zabudovaný odrušovací filtr	dle verze bez filtru nebo třída B
výbava navíc v USS verzi	sériový rozhraní RS485

tab.4. Technické parametry

10.2. Fotografie pracoviště



Obr. 26. Pracoviště

10.3. Testovací program pro generování PWM signálu

```
void HRPWM1_Config(int);
void HRPWM2_Config(int);
void HRPWM3_Config(int);
void HRPWM4_Config(int);

Uint16 i,j, duty, DutyFine, n,update;

Uint32 temp;
void main(void)
{
   HRPWM1_Config(10);    // ePWM1 target, 10 MHz PWM
   HRPWM2_Config(20);    // ePWM2 target,  5 MHz PWM
   HRPWM3_Config(10);    // ePWM3 target, 10 MHz PWM
   HRPWM4_Config(20);    // ePWM4 target,  5 MHz PWM

   EALLOW;
   SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;
   EDIS;
   while (update ==1)
        {
             for(DutyFine =1; DutyFine <256 ;DutyFine ++)
         {


         EPwm1Regs.CMPA.half.CMPAHR = DutyFine << 8;
         EPwm2Regs.CMPA.half.CMPAHR = DutyFine << 8;

         EPwm3Regs.CMPA.all = ((Uint32)EPwm3Regs.CMPA.half.CMPA << 16) +
(DutyFine << 8);
         EPwm4Regs.CMPA.all = ((Uint32)EPwm4Regs.CMPA.half.CMPA << 16) +
(DutyFine << 8);
        for (i=0;i<10000;i++){}
        }
        }
        }
void HRPWM1_Config(period)
{
        EPwm1Regs.TBCTL.bit.PRDLD = TB_IMMEDIATE;
        EPwm1Regs.TBPRD = period;
        EPwm1Regs.CMPA.half.CMPA = period / 2;
        EPwm1Regs.CMPA.half.CMPAHR = (1 << 8);
        EPwm1Regs.CMPB = period / 2;
        EPwm1Regs.TBPHS.all = 0;
        EPwm1Regs.TBCTR = 0;
        EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
        EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;
        EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
        EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
        EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;

        EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
        EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;
        EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
        EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;

        EPwm1Regs.AQCTLA.bit.ZRO = AQ_CLEAR;
        EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
        EPwm1Regs.AQCTLB.bit.ZRO = AQ_CLEAR;
```

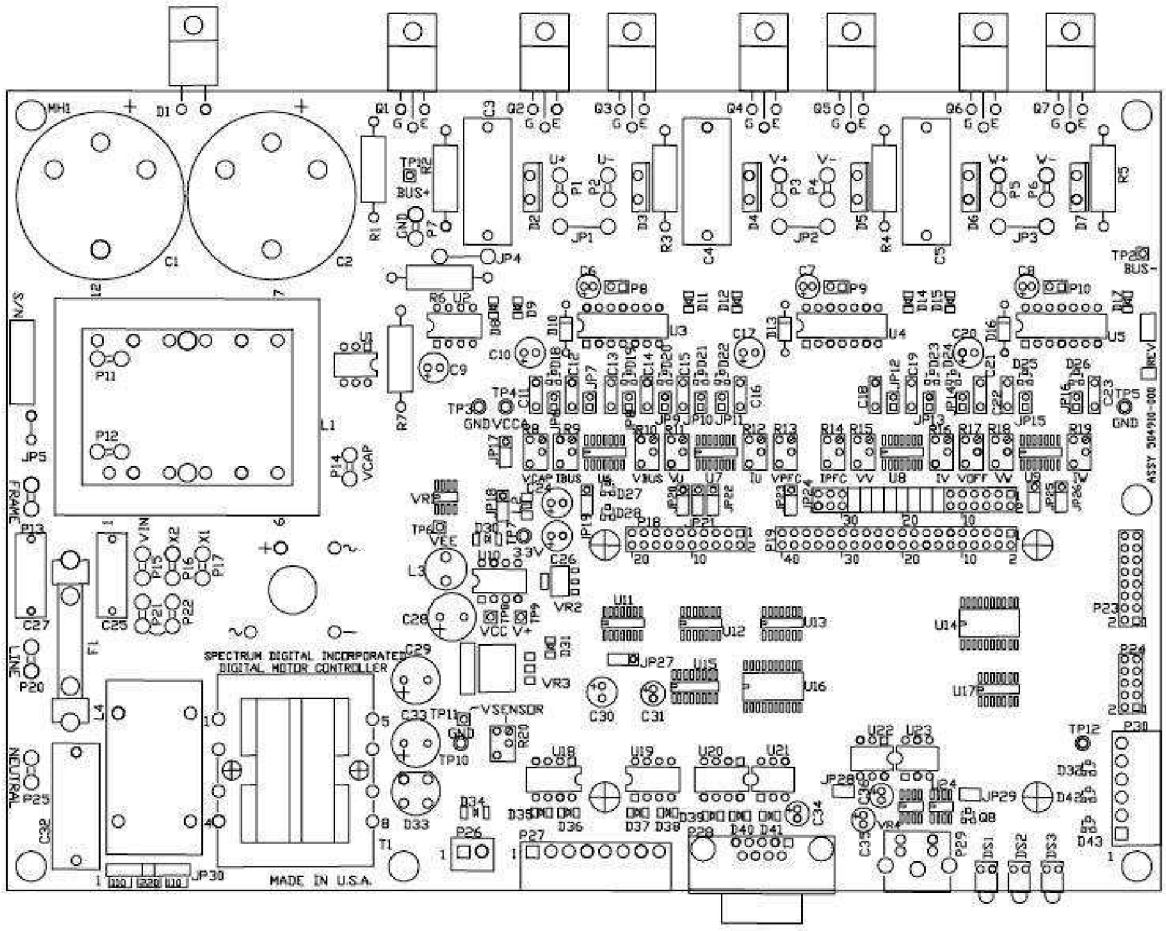
```
        EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;

        EALLOW;
        EPwm1Regs.HRCNFG.all = 0x0;
        EPwm1Regs.HRCNFG.bit.EDGMODE = HR_REP;
        EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;
        EPwm1Regs.HRCNFG.bit.HRLOAD  = HR_CTR_ZERO;
        EDIS;
}
void HRPWM2_Config(period)
{
        EPwm2Regs.TBCTL.bit.PRDLD = TB_IMMEDIATE;
        EPwm2Regs.TBPRD = period;
        EPwm2Regs.CMPA.half.CMPA = period / 2;
        EPwm1Regs.CMPA.half.CMPAHR = (1 << 8);
        EPwm2Regs.CMPB = period / 2;
        EPwm2Regs.TBPHS.all = 0;
        EPwm2Regs.TBCTR = 0;

        EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
        EPwm2Regs.TBCTL.bit.PHSEN = TB_DISABLE;
        EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
        EPwm2Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
        EPwm2Regs.TBCTL.bit.CLKDIV = TB_DIV1;

        EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
        EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;
        EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
        EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;

        EPwm2Regs.AQCTLA.bit.ZRO = AQ_CLEAR;
        EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;
        EPwm2Regs.AQCTLB.bit.ZRO = AQ_CLEAR;
        EPwm2Regs.AQCTLB.bit.CBU = AQ_SET;

        EALLOW;
        EPwm2Regs.HRCNFG.all = 0x0;
        EPwm2Regs.HRCNFG.bit.EDGMODE = HR_REP;
        EPwm2Regs.HRCNFG.bit.CTLMODE = HR_CMP;
        EPwm2Regs.HRCNFG.bit.HRLOAD  = HR_CTR_ZERO;

        EDIS;
}
void HRPWM3_Config(period)
{
        EPwm3Regs.TBCTL.bit.PRDLD = TB_IMMEDIATE;
        EPwm3Regs.TBPRD = period;
        EPwm3Regs.CMPA.half.CMPA = period / 2;
        EPwm3Regs.CMPA.half.CMPAHR = (1 << 8);
        EPwm3Regs.TBPHS.all = 0;
        EPwm3Regs.TBCTR = 0;

        EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
        EPwm3Regs.TBCTL.bit.PHSEN = TB_DISABLE;
        EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
        EPwm3Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
        EPwm3Regs.TBCTL.bit.CLKDIV = TB_DIV1;
        EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
        EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;
        EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
        EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
```

```
        EPwm3Regs.AQCTLA.bit.ZRO = AQ_SET;
        EPwm3Regs.AQCTLA.bit.CAU = AQ_CLEAR;
        EPwm3Regs.AQCTLB.bit.ZRO = AQ_SET;
        EPwm3Regs.AQCTLB.bit.CBU = AQ_CLEAR;

        EALLOW;
        EPwm3Regs.HRCNFG.all = 0x0;
        EPwm3Regs.HRCNFG.bit.EDGMODE = HR_FEP;
        EPwm3Regs.HRCNFG.bit.CTLMODE = HR_CMP;
        EPwm3Regs.HRCNFG.bit.HRLOAD  = HR_CTR_ZERO;
        EDIS;
}
void HRPWM4_Config(period)
{
        EPwm4Regs.TBCTL.bit.PRDLD = TB_IMMEDIATE;
        EPwm4Regs.CMPA.half.CMPA = period / 2;
        EPwm4Regs.CMPA.half.CMPAHR = (1 << 8);
        EPwm4Regs.CMPB = period / 2;
        EPwm4Regs.TBPHS.all = 0;
        EPwm4Regs.TBCTR = 0;

        EPwm4Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
        EPwm4Regs.TBCTL.bit.PHSEN = TB_DISABLE;
        EPwm4Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
        EPwm4Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
        EPwm4Regs.TBCTL.bit.CLKDIV = TB_DIV1;

        EPwm4Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
        EPwm4Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;
        EPwm4Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
        EPwm4Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;

        EPwm4Regs.AQCTLA.bit.ZRO = AQ_SET;
        EPwm4Regs.AQCTLA.bit.CAU = AQ_CLEAR;
        EPwm4Regs.AQCTLB.bit.ZRO = AQ_SET;
        EPwm4Regs.AQCTLB.bit.CBU = AQ_CLEAR;
        EALLOW;
        EPwm4Regs.HRCNFG.all = 0x0;
        EPwm4Regs.HRCNFG.bit.EDGMODE = HR_FEP;
        EPwm4Regs.HRCNFG.bit.CTLMODE = HR_CMP;
        EPwm4Regs.HRCNFG.bit.HRLOAD  = HR_CTR_ZERO;
        EDIS;
}
```

10.4. Program pro řízení otáček

```
#ifndef __VHZ_PROF_H__
#define __VHZ_PROF_H__

typedef struct    { _iq  Freq;
                    _iq  VoltOut;
                    _iq  LowFreq;
                    _iq  HighFreq;
                    _iq  FreqMax;
                    _iq  VoltMax;
                   _iq  VoltMin;
                    void  (*calc)();
calculation function
                   } VHZPROF;

typedef VHZPROF *VHZPROF_handle;
#define VHZPROF_DEFAULTS { 0,0, \
                           0,0,0,0,0, \
                                 (void (*)(Uint32))vhz_prof_calc }

void vhz_prof_calc(VHZPROF_handle);

#endif // __VHZ_PROF_H__
```

10.5. Plošná deska měniče DMC1500



Obr. 27. Plošná deska DMC1500