EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Použití mikroprocesorů pro řízení pohonů s BLDC motory

Učební texty ke kurzu

Autoři:

Ing. Jaroslav Lepka (Freescale Semiconductor, Rožnov p. R.)

Ing. Pavel Grasblum, PhD (Freescale Semiconductor, Rožnov p. R.)

Datum:

10. 11. – 11.11. 2011

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií CZ.1.07/2.3.00/09.0031

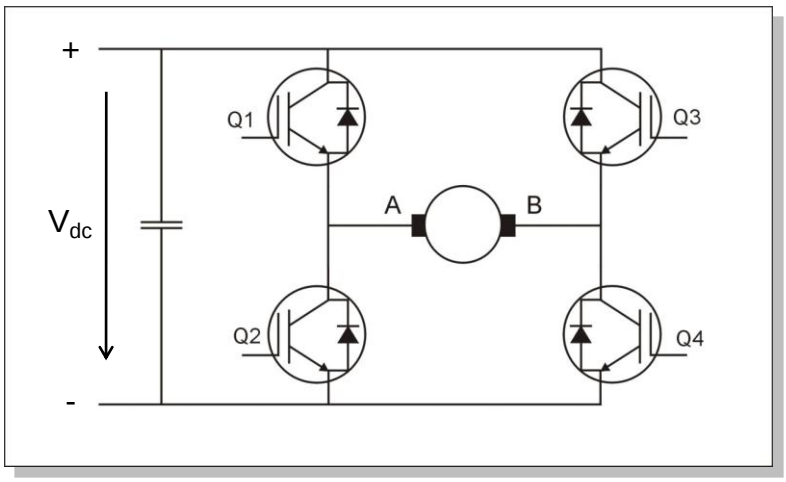
TENTO STUDIJNÍ MATERIÁL JE SPOLUFINANCOVÁN EVROPSKÝM SOCIÁLNÍM FONDEM A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

OBSAH

1. PRINCIPY SPÍNÁNÍ 3-FÁZOVÉHO STŘÍDAČE

1.1. Unipolární vs. bipolární spínání

Termíny "unipolární" a "bipolární" spínání souvisí s tím, jaká napětí vidí motor na svých svorkách. Pro jednoduchost předpokládejme, že budeme řídit stejnosměrný motor s cizím buzením (permanentní magnet) a tudíž máme k dispozici jedno vinutí se dvěma svorkami. K tomu, abychom byli schopni plně řídit tento motor, potřebujeme jednoduchou topologii střídače se čtyřmi tranzistory (dvě větve), kde máme možnost zcela nezávisle spínat kterýkoliv ze čtyř tranzistorů. Topologii znázorňuje Obrázek 1.1.

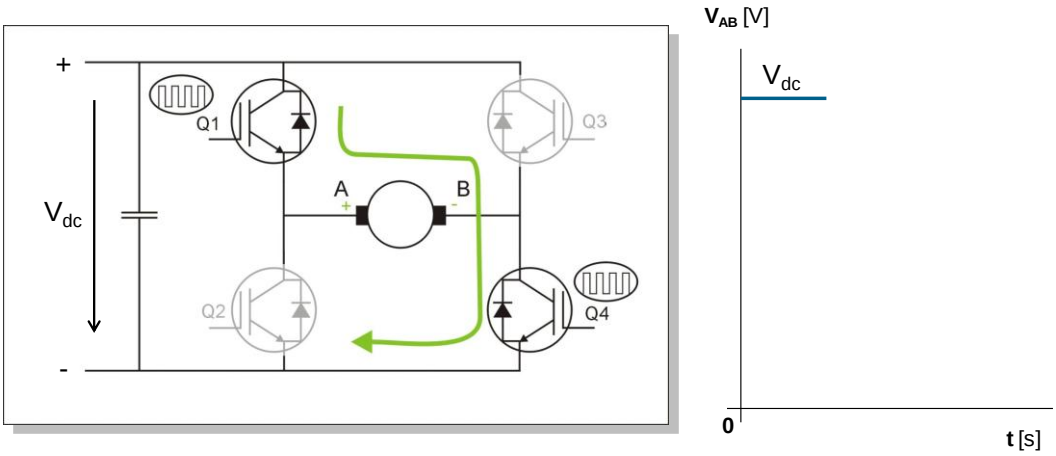


Obrázek 1.1 Topologie střídače ve tvaru H-můstku se 4 tranzistory

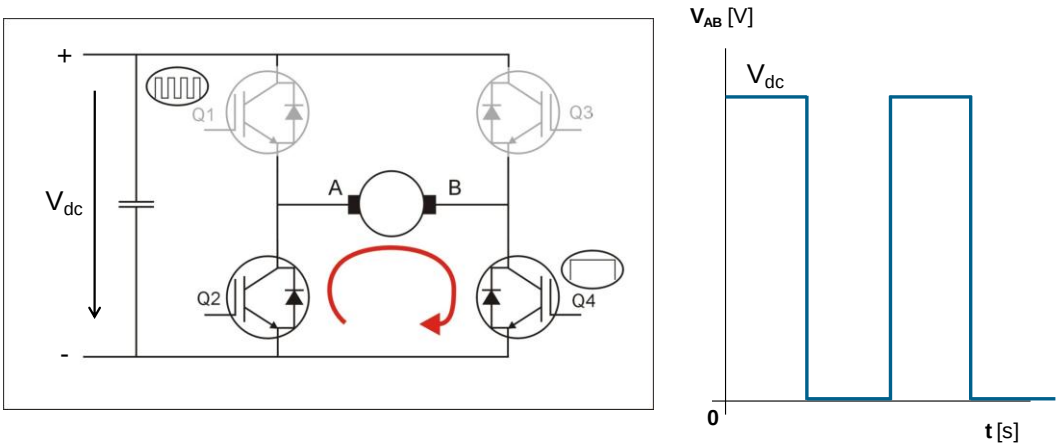
1.1.1. *Princip unipolárního spínání*

V případě unipolárního spínání pro jeden směr otáčení dochází k následnému postupnému spínání tranzistorů Q1-on a Q4-on (Q2-off, Q3-off) a poté Q2-on a Q4-on (Q1-off, Q3-off).

1. Q1-on a Q4-on (Q2-off, Q3-off) - v tomto případě na svorkách motoru A, B je přímo připojeno napětí V_{dc}. Tento stav ukazuje Obrázek 1.2.
2. Q2-on a Q4-on (Q1-off, Q3-off) - zde dochází ke zkratování svorek motoru A, B a de facto k připojení nulového napětí. Tento stav ukazuje Obrázek 1.3.



Obrázek 1.2 Unipolární spínání - případ 1.

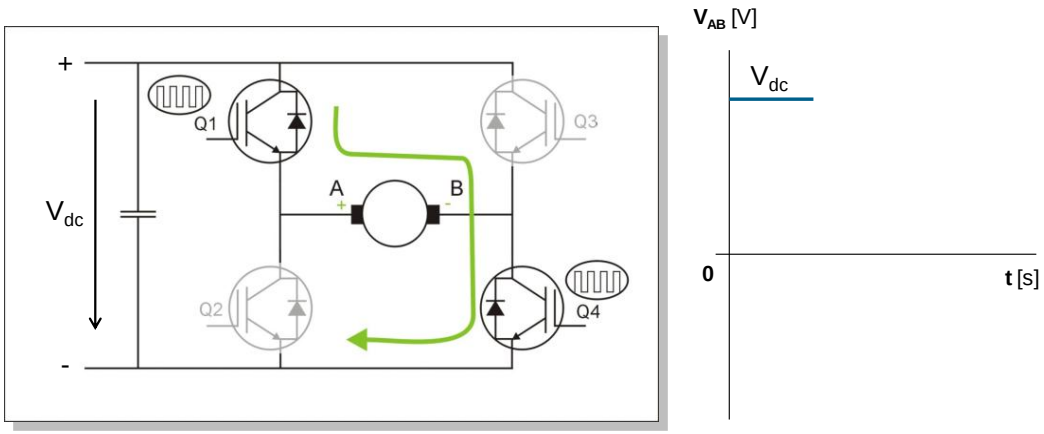


Obrázek 1.3 Unipolární spínání - případ 2.

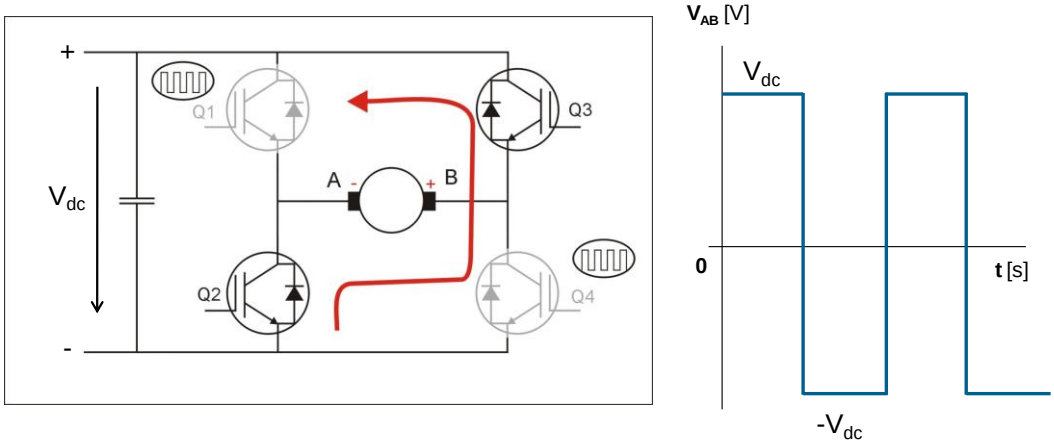
1.1.2. *Princip bipolárního spínání*

V případě bipolárního spínání dochází k následnému postupnému spínání tranzistorů Q1-on a Q4-on (Q2-off, Q3-off) a poté Q2-on a Q3-on (Q1-off, Q4-off).

1. Q1-on a Q4-on (Q2-off, Q3-off) - v tomto případě na svorkách motoru A, B je přímo připojeno napětí V_{dc}. Tento stav ukazuje Obrázek 1.4.
2. Q2-on a Q3-on (Q1-off, Q4-off) - v tomto případě na svorkách motoru A, B je připojeno napětí V_{dc} s opačnou polaritou, tedy $-V_{dc}$. Tento stav ukazuje Obrázek 1.5.



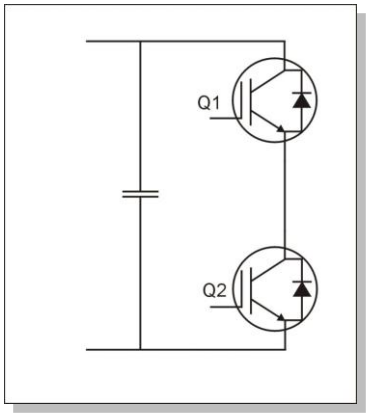
Obrázek 1.4 Bipolární spínání - případ 1.



Obrázek 1.5 Bipolární spínání - případ 2.

1.2. Nezávislé vs. komplementární spínání

Termíny "nezávislé" a "komplementární" spínání souvisí s tím, jaký je stav jednotlivých tranzistorů v jedné větvi střídače během jedné periody sepnutí. Pro jednoduchost předpokládejme, že budeme hovořit o jedné větvi střídače, tedy o tranzistorech Q1 a Q2 viz Obrázek 1.6.

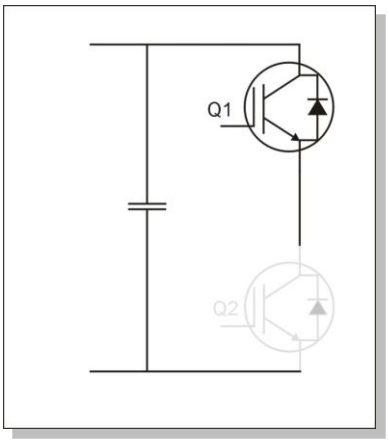


Obrázek 1.6 Topologie jedné větve střídače.

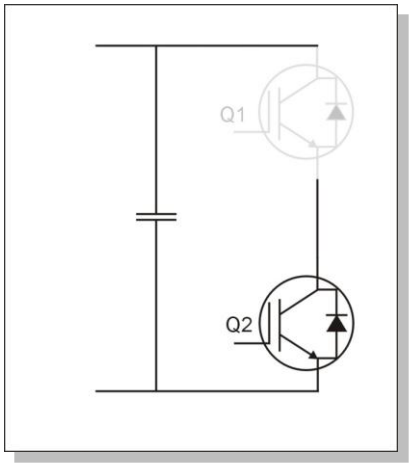
1.2.1. *Princip nezávislého spínání*

V případě nezávislého spínání je vždy jeden z tranzistorů sepnut po celou periodu spínací periodu a druhý je řízen PWM signálem tak, aby bylo dosaženo potřebné střední hodnoty napětí na svorkách motoru.

1. Q2-on po celou dobu periody, Q1-on/off v závislosti na střední hodnotě napětí na motoru, viz. Obrázek 1.7.
2. Q1-on po celou dobu periody, Q2-on/off v závislosti na střední hodnotě napětí na motoru, viz. Obrázek 1.8.



Obrázek 1.7 Nezávislé spínání - případ 1.



Obrázek 1.8 Nezávislé spínání - případ 2.

1.2.2. *Princip komplementárního spínání*

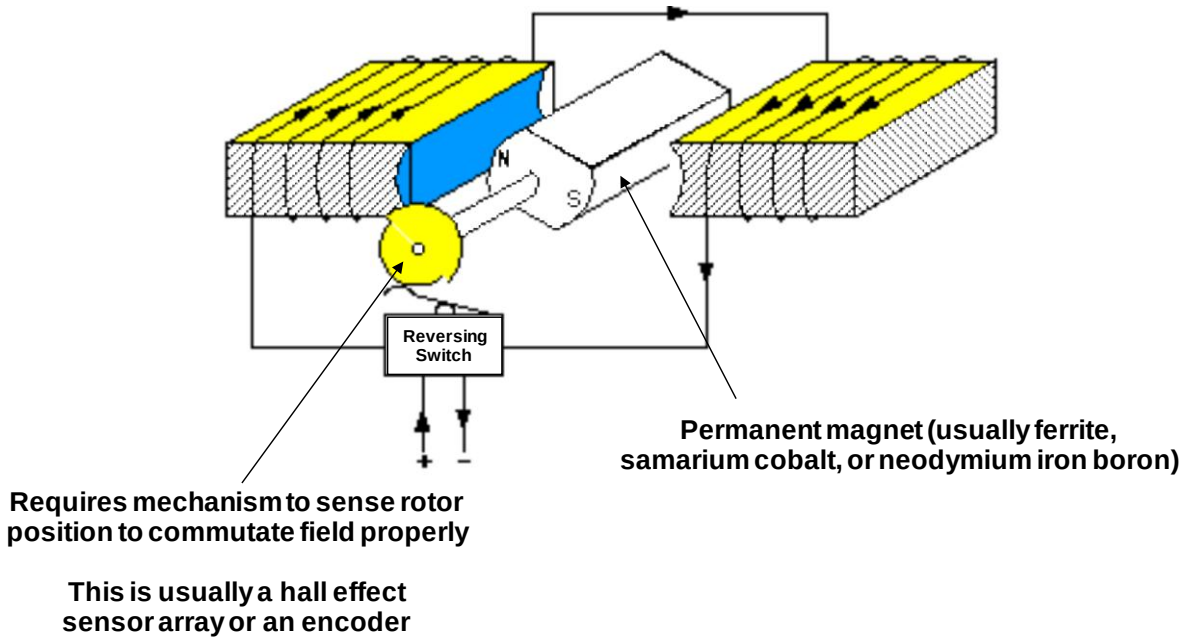
V případě komplementárního režimu spínání jsou vždy ovládány oba tranzistory v jedné větvi invertoru tzv. komplementárním způsobem, což znamená, že pokud je jeden tranzistor sepnutý (např. Q1), druhý (Q2) je vypnutý a naopak. Specifický případ nastává v okamžiku přepnutí, kdy dochází k vypínání tranzistoru Q1 a spínání tranzistoru Q2. V tomto případě, abychom zabránili zkratu střídače, jsou po určitou dobu vypnuty oba tranzistory. Době, po kterou jsou oba tranzistory vypnuty, se říká "deadtime".

2. BLDC MOTOR

BLDC motor (Brushless DC Motor) je bezkartáčový elektronicky komutovaný DC motor, který je hojně používaný v jednoduchých aplikacích, kde nepožadujeme rovnoměrnost momentu. Pro svoji konstrukční jednoduchost, relativně snadné řízení, cenovou dostupnost a spolehlivost téměř zcela vytlačil klasické stejnosměrné motory.

2.1. BLDC motor - princip

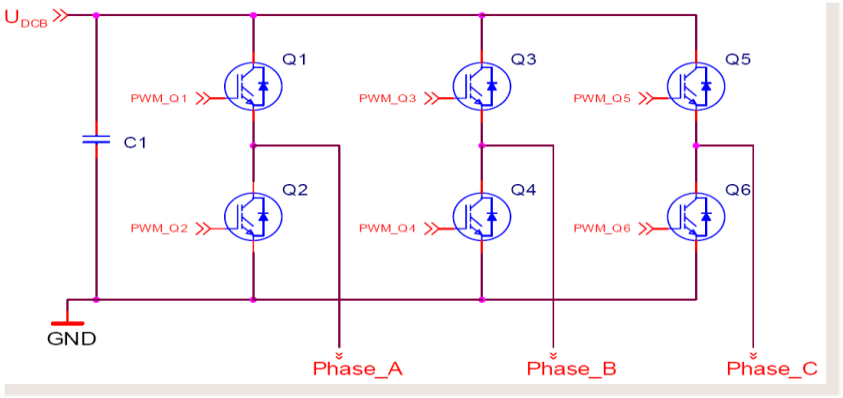
Konstrukčně je BLDC motor tvořen statorem, na kterém je navinuto statorové vinutí, obvykle 3-fázové. Rotor je tvořen permanentním magnetem, který vytváří konstantní magnetické pole ve vzduchové mezeře. BLDC motory se vyrábějí jako 2-pólové nebo n-pólové. BLDC motor na rozdíl od klasického DC motoru nemá mechanický komutátor. Komutaci je tedy zapotřebí provádět elektronicky. Abychom byli schopni provádět komutaci ve správném okamžiku, je zapotřebí znát informaci o poloze rotu. Toto se obvykle řeší pomocí Hallových sensorů v případě aplikací, kde záleží na ceně nebo pomocí obvykle inkrementálního snímače v případě vyšších nároků na pohon. Existují i řešení bez použití snímače a takováto řešení se pak nazývají obecně "sensorless řízení". Základní princip BLDC motoru ukazuje Obrázek 2.1.



Obrázek 2.1 Princip BLDC motoru.

2.1. Invertor pro řízení BLDC motoru

Základní řídicí algoritmus je tzv. "six step komutace". K tomu, abychom byli schopni elektronicky komutovat jednotlivé fáze statorového vinutí a tím docílit rotačního magnetického pole ve vzduchové mezeře a následně elektromagnetického momentu, který způsobí otáčení rotoru BLDC motoru, je zapotřebí mít možnost ovládat všechny tři fáze motoru (nejběžnější topologie motoru). K tomuto účelu se používá 3-fázový střídač (Obrázek 2.2).

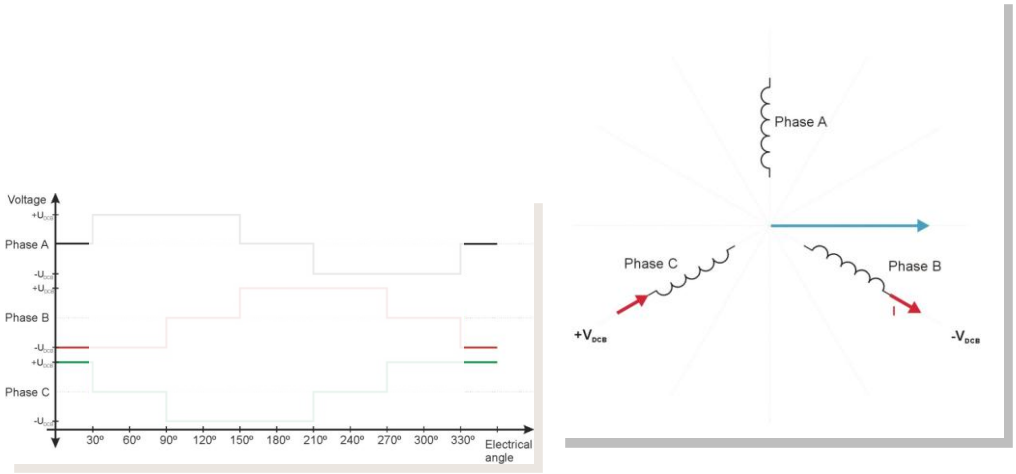


Obrázek 2.2 3-fázový střídač.

Z obrázku je patrno, že se střídač skládá ze tří větví tranzistorů, po dvou tranzistorech v každé větvi, což dohromady činí 6 tranzistorů. Středy jednotlivých větví jsou pak připojeny na jednotlivé fáze statorového vinutí BLDC motoru. Jednotlivé tranzistory jsou řízeny mikropočítačem, kde je implementovaný řídicí algoritmus.

2.1. Six step komutace

Tento základní řídicí algoritmus vyžaduje znalost o poloze rotoru. V případě "six step" komutace jsou vždy napájeny pouze dvě fáze motoru. Odpovídající fáze jsou napájeny na základě znalosti aktuální pozice rotoru a požadovaného směru otáčení. Jelikož na jednu elektrickou otáčku generujeme pouze 6 spínacích vzorů, mění se vektor magnetického pole po skocích 60° elektrických, což způsobuje zvlnění momentu, které je pro některé aplikace nepřípustné. Proto je zapotřebí zvážit, jestli je algoritmus "six step" komutace vhodný pro daný pohon. Příklad jednoho spínacího vzoru ukazuje Obrázek 2.3.

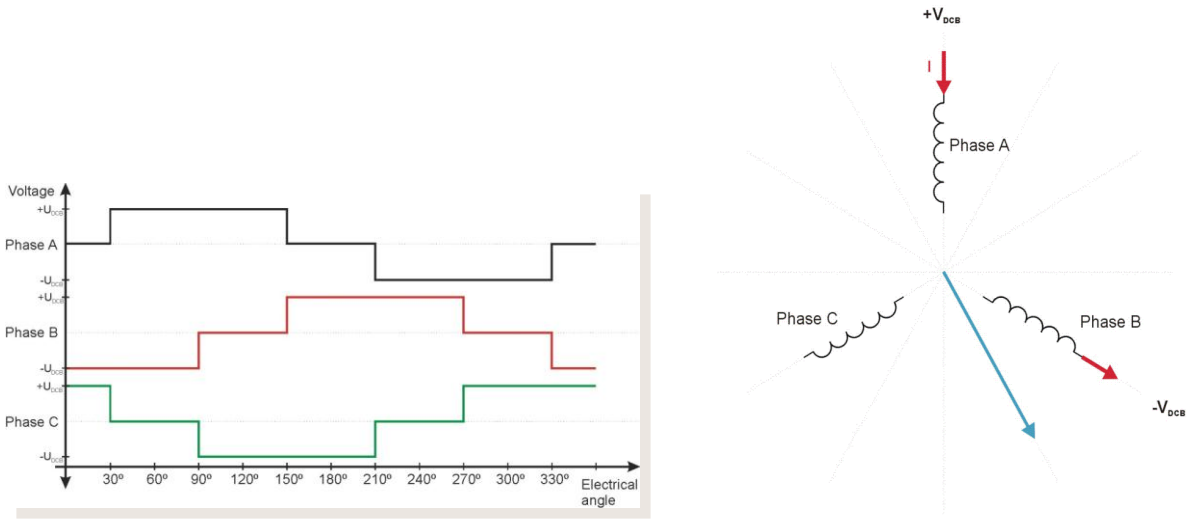


Phases voltage

Obrázek 2.3 Příklad jednoho komutačního vzoru.

Z obrázku je zřejmé, že se napájí pouze fáze B a C a fáze A je nezapojena. Vektor magnetického pole odpovídající dané kombinaci statorových fází je zobrazen modře a je kolmý na fázi A.

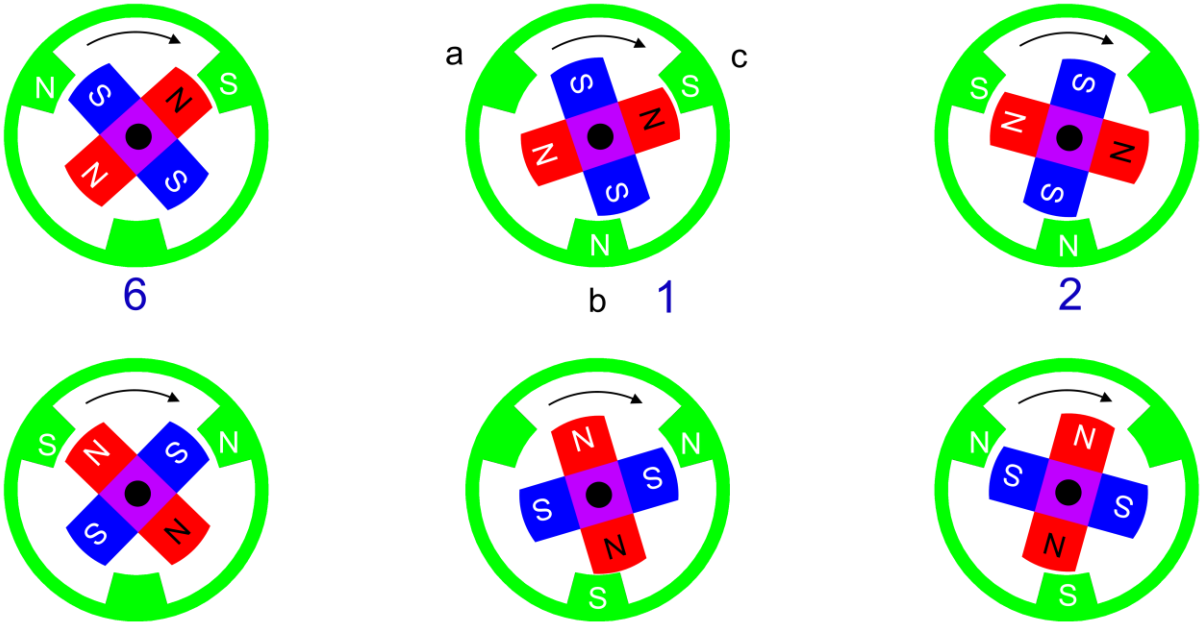
Kompletní spínací vzory pro jednu elektrickou otáčku zobrazuje Obrázek 2.4.



Phase voltages

Obrázek 2.4 Spínací vzory pro jednu elektrickou otáčku

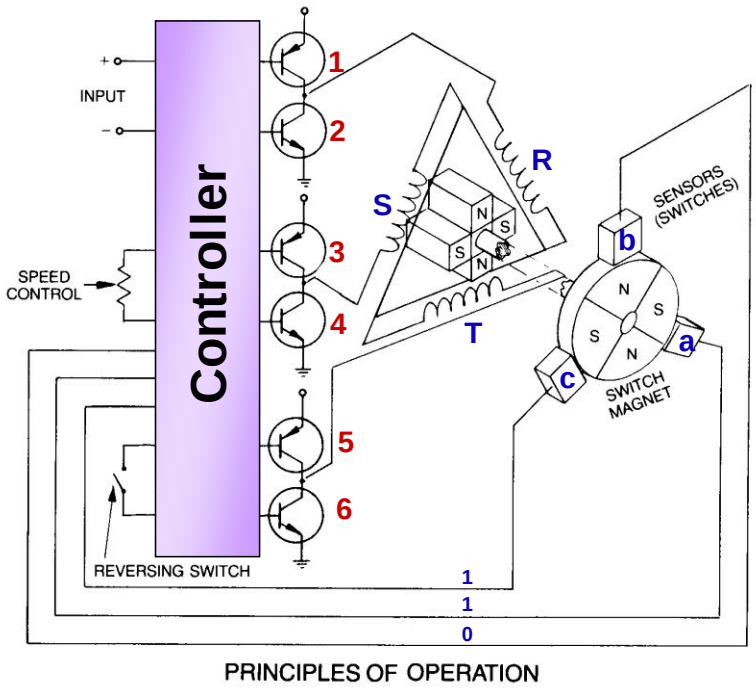
Grafické znázornění jednotlivých poloh rotoru a tomu odpovídající magnetické pole generované statorovým vinutím pro případ jedné elektrické otáčky je znázorněno na následujícím obrázku.



Obrázek 2.5 Komutace BLDC motoru - Jedna elektrická otáčka

2.1. BLDC komutace na základě Hallových sensorů

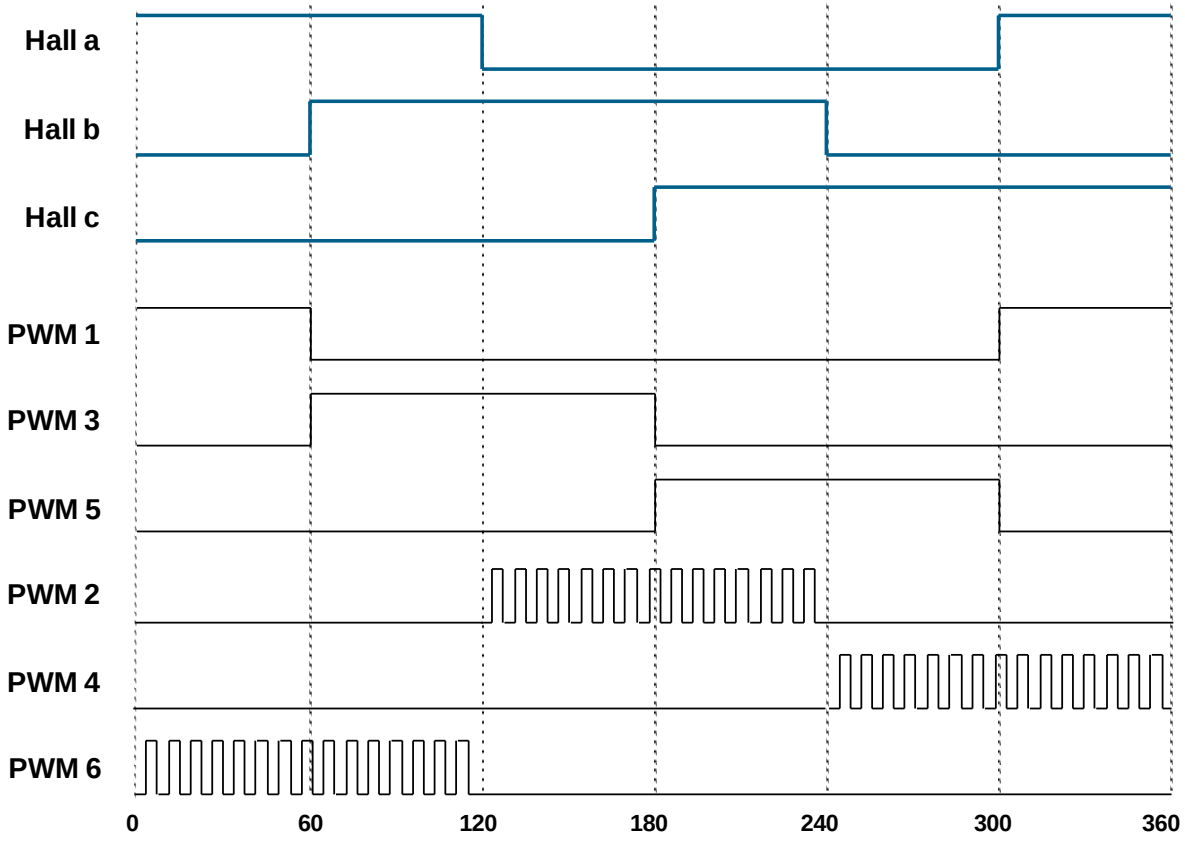
Principiální schéma řízení BLDC motoru na základě informace z Hallových sensorů ukazuje Obrázek 2.6.



Source: Eastern Air Devices, Inc. Brushless DC Motor Brochure

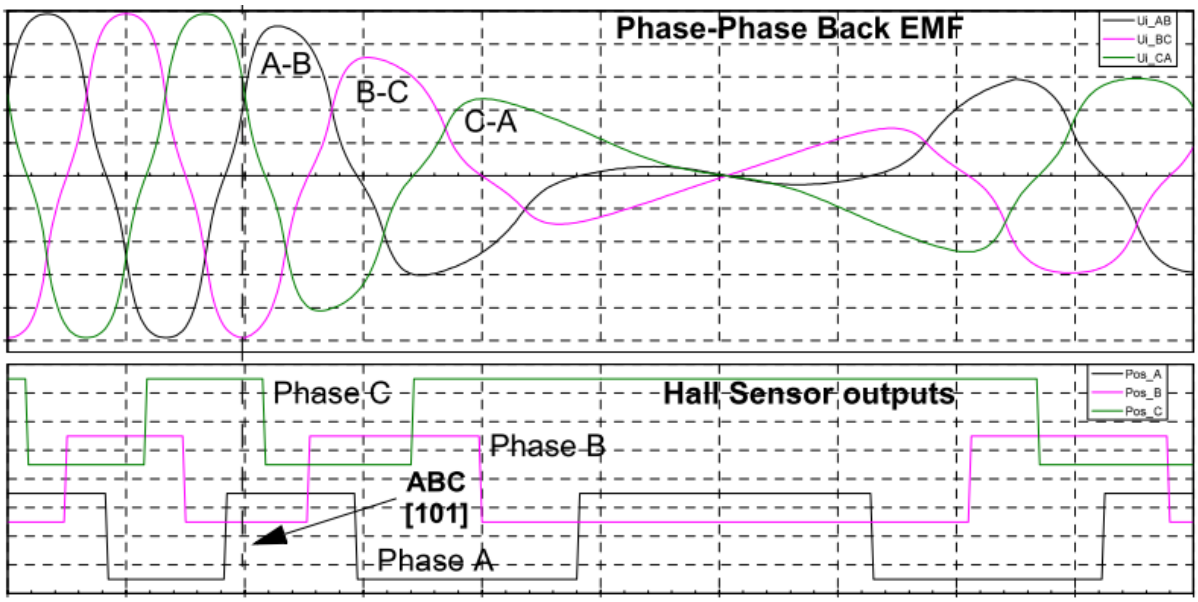
Obrázek 2.6 Princip řízení BLDC motoru na základě Hallových sensorů

Tři Hallovy snímače jsou spojeny s rotorem tvořeným permanentními magnety a zarovnány takovým způsobem, aby odpovídaly sdruženému indukovanému napětí. Pak lze použít informace z Hallových snímačů pro rozpoznání aktuální polohy rotoru s přesností +/- 30° elektrických. Řídicí algoritmus zpracovává informace z Hallových sensorů a na základě komutační tabulky naprogramované v mikropočítači pak následně volí odpovídající kombinace fází pro sepnutí. Stav jednotlivých Hallových sensorů a jim odpovídající PWM signály generované pro jednotlivé tranzistory ukazuje Obrázek 2.7.



Obrázek 2.7 Stav Hallových sensorů a odpovídající generace PWM signálu

Relaci mezi signály generovanými Hallovými sensory a sdruženým indukovaným napětím ukazuje Obrázek 2.8.



Obrázek 2.8 Stav Hallových sensorů vs. sdružené indukované napětí

2.1. BLDC motor - bezsensorové řízení, princip a metody

Pojem bezsensorové řízení je obecně chápán tak, že se předpokládá řízení motoru bez použití snímače polohy rotoru, případně rychlosti. Konfigurace systému bez měření polohy rotoru výrazně zlevňuje hardwarové řešení celého systému, jelikož nepoužijeme snímač polohy, což vede k dalším zjednodušením a zlevněním. Není zapotřebí mít rozhraní pro zpracování signálů ze snímače, kabeláž, konektory, napájení snímače, atd. To vše rovněž vede ke zvýšení spolehlivosti celého systému podle hesla "co v systému není, nemůže se pokazit".
Pro řešení problému bezsensorového řízení existuje řada možných řešení, více či méně vhodných pro konkrétní aplikaci. Jedny z nejpoužívanějších metod jsou založeny na měření nebo odhadu indukovaného napětí a metody založená na principu změny indukčnosti v závislosti na poloze rotoru

2.1.1. *Bezsensorové řízení - metody založené na indukovaném napětí*

Hlavním předpokladem těchto metod je, že indukované napětí je dostatečně velké co do amplitudy, aby bylo možno dosahovat dobrých výsledků i pro malé otáčky. Rozsah otáček, které je možno zpracovat těmito metodami je zhruba v rozsahu od 5%-10% až po 100% nominálních otáček.

Používané techniky:

- Založené na měření průchodu indukovaného napětí nulou (BEMF Zero Crossing methods)
- Pokročilé techniky založené na odhadu indukovaného napětí

2.1.2. *Bezsensorové řízení - metody založené na principu změny indukčnosti na poloze rotoru*

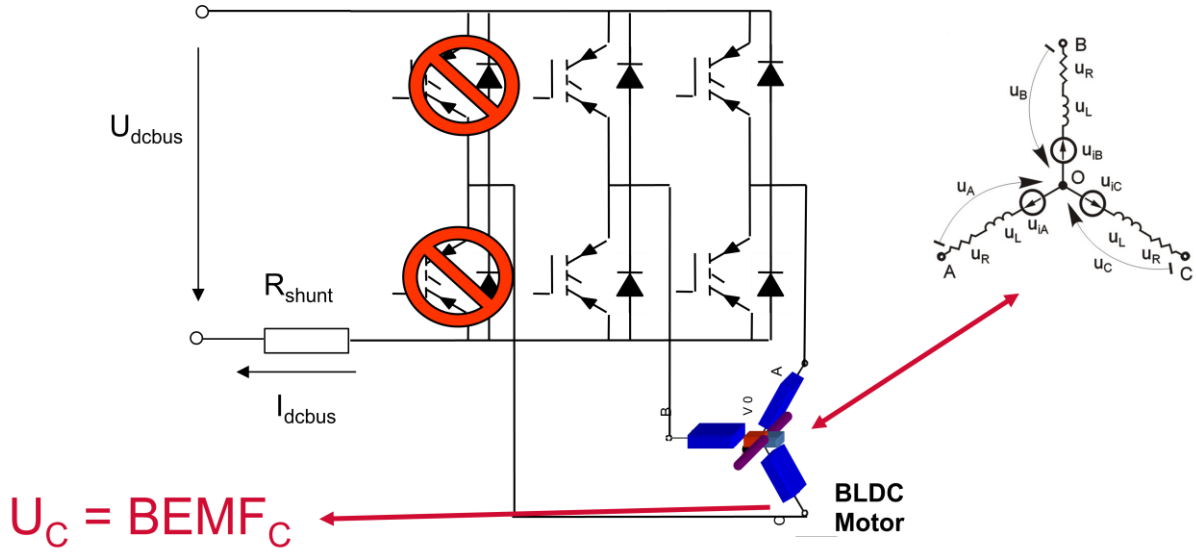
Tyto metody předpokládají, že dochází ke změnám parametrů motoru v důsledku jejich konstrukce (např. motor s vyniklými póly). Rozsah otáček, pro které lze tyto metody použít, se pohybuje od stojícího rotoru, tedy nulových otáček až po zhruba 20% nominálních otáček. Maximální dosažitelné otáčky touto metodou závisí na konstrukci motoru (elektrických a mechanických vlastnostech), proto 20% nominálních otáček je hrubý odhad.

Používané techniky:

- Techniky založené na měření přechodového děje proudu
- Pokročilé techniky založené na injektování proudu. Tyto techniky předpokládají poměrně výkonné jádro mikrokontroleru a přesné měření proudu

2.2. BLDC motor - bezsensorové řízení založené na měření průchodu indukovaného napětí nulou

Princip této metody spočívá v identifikaci okamžiku, kdy indukované napětí prochází nulou. Z toho plyne podmínka, která říká, že musíme být nějakým způsobem schopni měřit indukované napětí v odpovídající fázi motoru. Z toho plyne, že nemůžeme zároveň generovat napětí do fáze, na které předpokládáme měření indukovaného napětí. Z hlediska řídicího algoritmu je zapotřebí, aby vždy jedna fáze nebyla napájena, což vede k vypnutí tranzistorů v dané větvi a to jak horního, tak i dolního viz. Obrázek 2.9.



Obrázek 2.9 Stav střídače umožňující měření indukovaného napětí

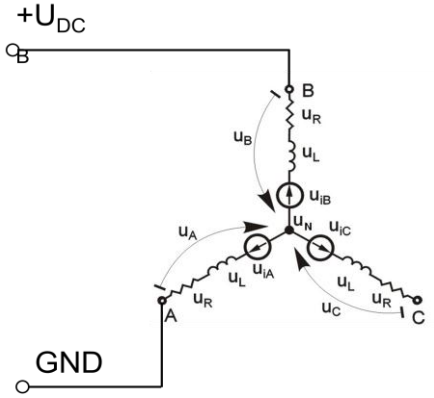
Předpoklady pro měření indukovaného napětí:

- Měřená fáze je odpojena od napájení
- Příslušnou fází neprotéká žádný proud

Měřené indukované napětí odpovídá v našem případě napětí fázovému. To všem činí potíže při implementaci měření, jelikož střed hvězdy vinutí motoru ve většině případů nebývá vyveden.

2.2.1. *Měření indukovaného napětí v případě, že oba tranzistory v diagonále jsou sepnuty*

Za tohoto předpokladu je jedna fáze motoru přivedena na GND a druhá na napětí U_{dc} viz. Obrázek 2.10.



Obrázek 2.10 Svorková napětí motoru při sepnutí tranzistorů v diagonále

Za těchto podmínek se dá odvodit závislost měřeného svorkového napětí odpojené fáze na indukovaném napětí téže fáze.

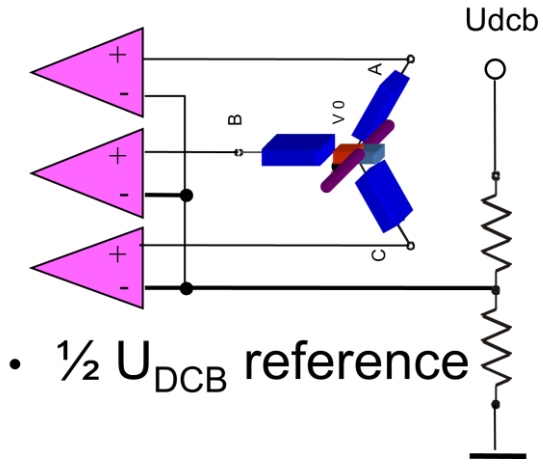
Platí:

$$u_C = \frac{3}{2} \cdot u_{IC} + \frac{U_{dc}}{2}$$

V případě, kdy indukované napětí prochází 0, je u_{IC} = 0 a tedy napětí, které měříme je:

$$u_C = \frac{U_{dc}}{2}$$

Z toho plyne, že je zapotřebí porovnávat měřenou hodnotu svorkového napětí s $1/2U_{dc}$, jak ukazuje Obrázek 2.11.



Obrázek 2.11 Realizace měření fázového indukovaného napětí vůči $1/2U_{dc}$

Tato metoda je vhodná pro:

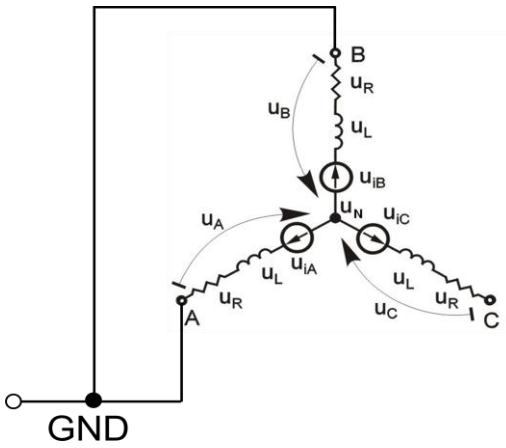
- Bipolární metodu spínání
- Unipolární metodu spínání v případě, že jsou sepnuty tranzistory v diagonále.

Nevýhody:

- Obtížné měření při malých rychlostech - krátká perioda pro měření
- Chyba způsobená různými rezistory v děliči

2.2.2. *Měření indukovaného napětí v případě, že jsou sepnuty pouze dolní tranzistory*

Za tohoto předpokladu jsou obě fáze motoru připojeny na GND viz. Obrázek 2.12.



Obrázek 2.12 Svorková napětí motoru při sepnutí dvou dolních tranzistorů střídače

Za těchto podmínek se dá odvodit závislost měřeného svorkového napětí odpojené fáze na indukovaném napětí téže fáze.

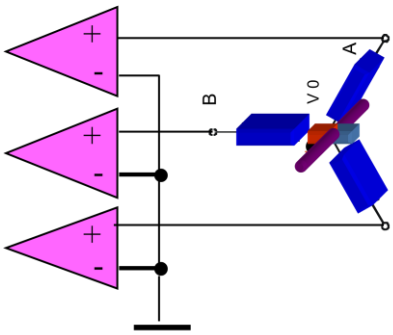
Platí:

$$u_C = \frac{3}{2} \cdot u_{IC}$$

V případě, kdy indukované napětí prochází 0, je $u_{IC} = 0$ a tedy napětí, které měříme je:

$$u_C = 0$$

Z toho plyne, že je zapotřebí porovnávat měřenou hodnotu svorkového napětí s GND, jak ukazuje Obrázek 2.13.



- GND reference

Obrázek 2.13 Realizace měření fázového indukovaného napětí vůči GND

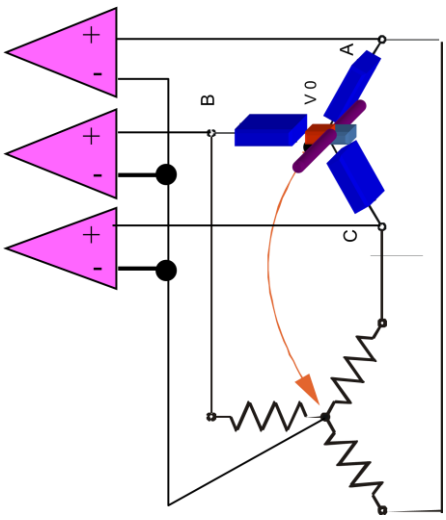
Tato metoda je vhodná pouze pro unipolární metodu spínání v případě, že jsou sepnuty spodní tranzistory.

Nevýhody:

- Obtížné měření při vysokých rychlostech - krátká perioda pro měření.
- Obtížné měření při malém indukovaném napětí - offset error.

2.2.3. *Měření indukovaného napětí pro případ virtuálního středu vinutí tvořeného externím elektrickým obvodem*

U tohoto řešení vytváříme virtuální střed 3-fázového vinutí motoru viz. Obrázek 2.14.



Obrázek 2.14 Topologie měření indukovaného napětí s virtuálním středem

Tato topologie vede ke stejným výsledkům jako topologie s porovnáváním vůči $1/2U_{dc}$.

Výhody:

- Vhodné pro unipolární i bipolární spínání a všechny kombinace septuní tranzistorů

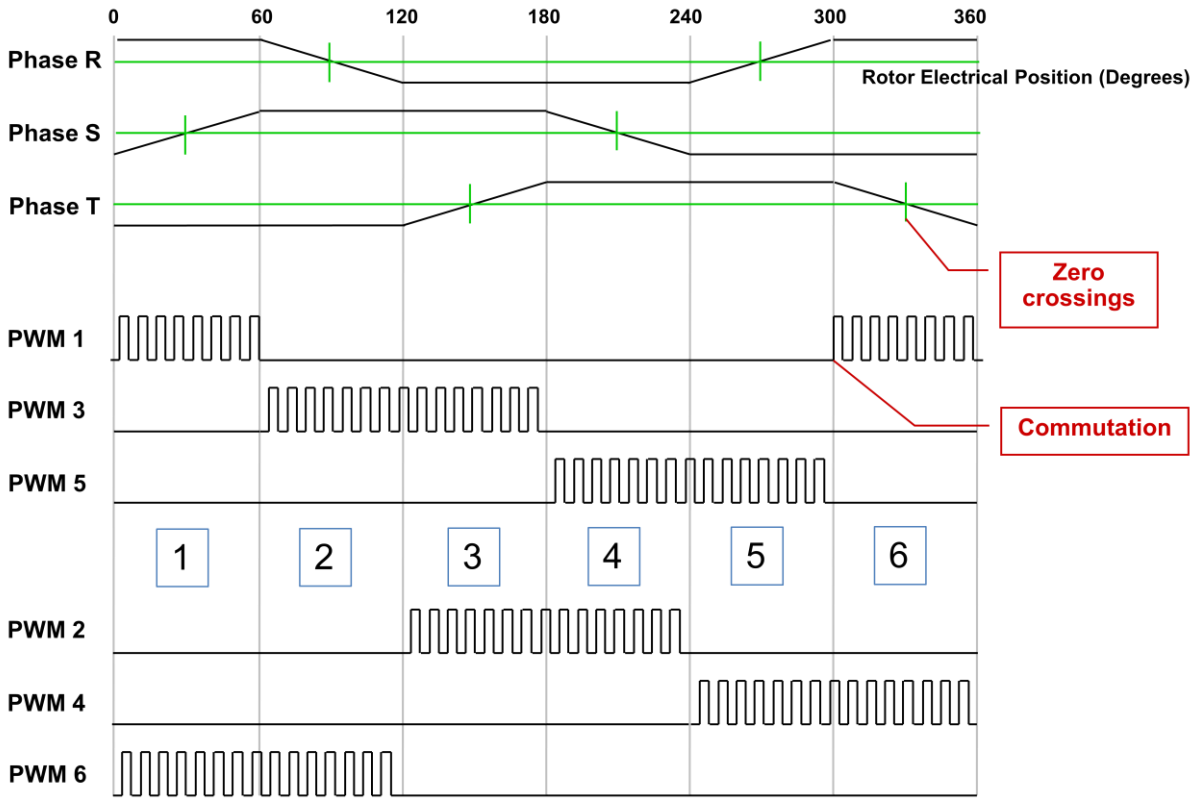
Nevýhody:

- Reference je zarušená
- Offset error - za předpokladu, že se proud uzavírá přes spodní tranzistor a spodní zpětnou diodu.

2.2.4. *Bezsensorové řízení založené na měření idukovaného napětí - princip*

Samotné řízení založené na principu měření průchodu indukovaného napětí 0 předpokládá, že jedna fáze vinutí motoru je vždy odpojena od zdroje. Měření indukovaného napětí pak probíhá právě na této fázi. V okamžiku kdy dojde k detekci průchodu 0, zaznamenáme čas této události. Ze znalosti času předchozího okamžiku, kdy došlo k průchodu 0, vypočítáme okamžik následující komutace a fyzicky přepneme tranzistory. Tento algoritmus stále opakujeme pro jiné fáze.

Obrázek 2.15 ukazuje princip řízení pro jednu elektrickou otáčku



Obrázek 2.15 Princip komutace na základě měření indukovaného napětí

V sektoru 1 vidíme, že průchod 0 indukovaného napětí nastal pro fázi S. To znamená, že tranzistory ovládané PWM signály PWM3 a PWM4 jsou vypnuté a tedy fáze S je odpojena od zdroje a umožňuje měřit indukované napětí. Fáze R má kladné napětí, tudíž horní tranzistor ovládaný signálem PWM1 je spínán, zatímco dolní tranzistor ovládaný signálem PWM2 je vypnut po celou komutační periodu (unipolární spínaní). Fáze T má záporné napětí, tudíž dolní tranzistor ovládaný signálem PWM6 je spínán, zatímco horní tranzistor ovládaný signálem PWM5 je vypnut po celou komutační periodu. Na konci komutační periody 1 dojde k přepnutí a fáze R je odpojena od zdroje, zatímco fáze S je opět připojena na U_{dc} pomocí šířkově pulsní modulace. Stav fáze T se nemění a fáze R je připravena pro měření indukovaného napětí (komutační perioda 2).

3. EMBEDDED MIKROPOČÍTAČ FREESCALE MC56F8006 - HARDWAROVÁ PODPORA PRO ŘÍZENÍ POHONŮ

3.1. Rodina mikropočítačů Freescale MC56F800x

Rodina mikropočítačů Freescale DSC56F800x čítá dva zástupce. Liší se ve velikosti a typu pouzdra, velikosti paměti FLASH, v počtu GPIO pinů a analogových vstupů ADC převodníku. Oba mikropočítače jsou založeny na jádře DSP56800E. Napájecí napětí je 3.3V. Hodiny mikropočítače běží na frekvenci 32MHz. Standardní ladící rozhraní je JTAG/Enhanced OnCE.

Do rodiny mikropočítačů Freescale DSC56F800x patří následující zástupci:

- MC56F8002 – pouzdro je 32 pinové nebo 48 pinové LQFP nebo 28 pinové SOIC, programová/datová Flash – 12kB, programová/datová RAM – 2kB
- MC56F8006 – pouzdro je 32 pinové nebo 48 pinové LQFP nebo 28 pinové SOIC, programová/datová Flash – 16kB, programová/datová RAM – 2kB

3.2. DSC56F8006/56F8002 – základní parametry

Mikropočítače DSC56F8006/56F8002 jsou založeny na jádře DSP5800E , které kombinuje výpočetní výkon DSP procesoru a výhodami klasického mikrokontroléru. Doplněním o flexibilní sadu výkonných periferních modulů integrovaných spolu s pamětí RAM a Flash na jednom čipu nabízí mikropočítače DSC56F8006/2 cenově výhodné řešení pro širokou škálu aplikačního využití. Jsou vhodné aplikace v průmyslu, řízení pohonů, domácích spotřebičů, zpracování inteligentních sensorů, bezpečnostní systémy, spínané zdroje, střídače pro obecné použití, UPS systémy, lékařské aplikace, atd.

Základní vlastnosti:

- Napájení digitální – 3.3 V
- Napájení analogové – 3.3 V

	56F8002	56F8006
Performance	32MHz/MIPs	32MHz/MIPs
Temperature Range (V)	-40C~105C	-40C~105C
Voltage Range	1.8V - 3.6V	1.8V - 3.6V
Voltage Regulator	On-Chip	On-Chip
Program/Data Flash	12KB	16KB
Program/Data RAM	2KB	2KB
Program Security	Yes	Yes
On Chip Relaxation Osc.	Yes	Yes
PLL	Yes	Yes
COP (Watchdog)	Yes	Yes
PWM (96 Mhz Clock)	1 x 6ch	1 x 6ch
PWM Fault Inputs	4	4
12-bit ADCs	2 x 8ch	2 x 12ch
12-bit DACs	0	0
Analog Comparator	3	3
Prog Gain Amp	2	2
16-bit Timers	3	3
Prog. Interval Timers	1 (RTC)	1 (RTC)
GPIO (max) (+/-8mA)	23	40
IIC	1	1
SCI (UART) / LIN Slave	1 - SCI	1 - SCI
SPI (Synchronous)	1 - SPI	1 - SPI
CAN	No	No
JTAG/EOnCE	Yes	Yes
Power Consumption	IDD = 45.6mA;IDDA = 4.5mA	
Package	32LQFP (.8p)	28SOIC 32LQFP 32SDIP 48LQFP

Obrázek 3.1 Přehled mikropočítačů rodiny MC56F800x a jejich vlastnosti

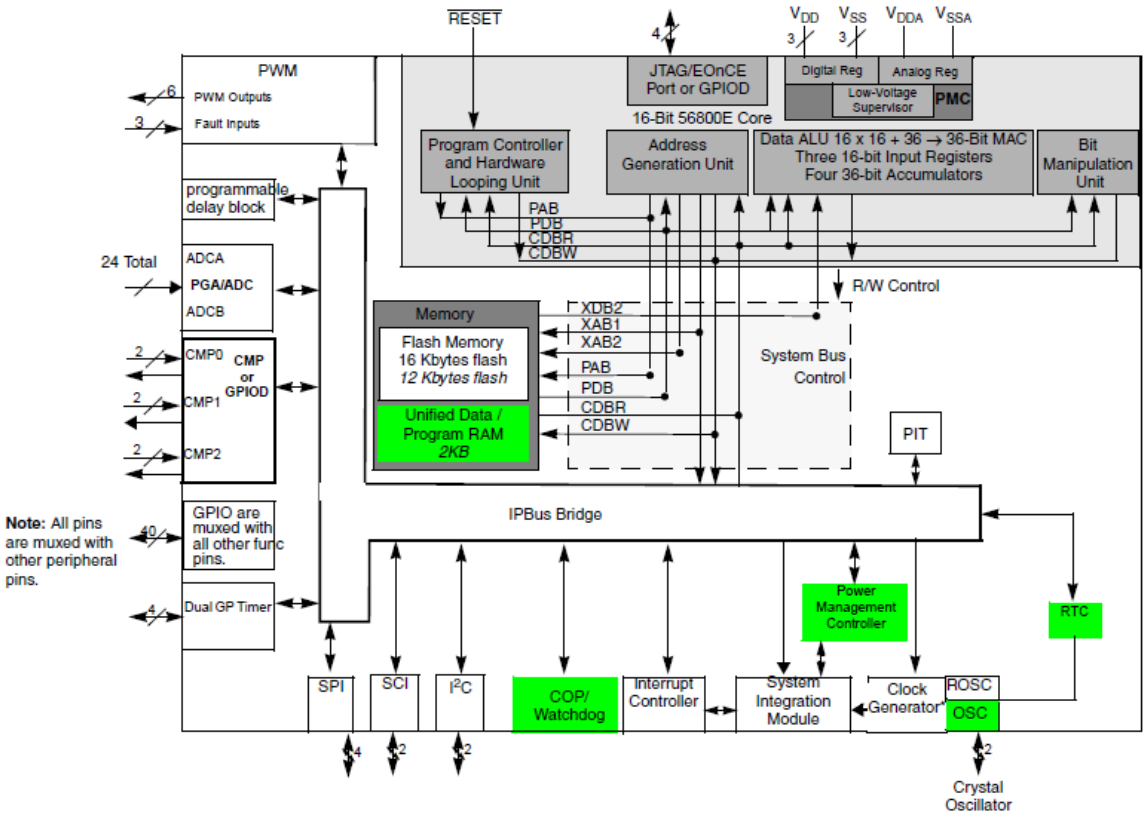
- Frekvence jádra – 32 MHz, což odpovídá výpočetnímu výkonu maximálně 32MIPS
- 56F8006 - 16KB (8K x 16) Program Flash
- 56F8002 - 12KB (6K x 16) Program Flash
- 56F8006 - 2KB (1K x 16) Unified Data/Program RAM
- 56F8002 - 2KB (1K x 16) Unified Data/Program RAM
- PWM modul – flexibilní a velmi komplexní periferie s 6 PWM kanály

- Dva 28-kanálové 12-bitové A/D převodníky (ADCs)
- Dva programovatelné zesilovače (PGA - Programmable Gain Amplifier) se zesílením až 32x
- Tři analogové komparátory
- Jeden programovatelný časovač – PIT (Programmable Interval Timer)
- Jeden modul SCI (Serial Communication Interface) s LIN slave funkčností
- Jeden modul SPI (Serial Peripheral Interfaces)
- Jeden modul 16-bitových časovačů – Quad Timer (TMR)
- Jeden modul I2C (Inter-Integrated Circuit)
- Jeden modul programovatelného zpoždění - PDB (Programmable Delay Block)
- Watchdog – COP (Computer Operating Properly)
- Relaxační oscilátor integrovaný na čipu
- Integrovaný Power-On Reset (POR) a Low-Voltage Interrupt (LVI)
- JTAG/Enhanced On-Chip Emulation (OnCE™)
- Až 40 GPIO

3.3. Organizace Freescale dokumentace

Firma Freescale dělí dokumentaci pro jednotlivé mikropočítače do několika ucelených dokumentů:

- Data sheet – dokument k jednotlivým mikropočítačům. Obsahuje specifické údaje pro daný čip jako např.: napájení, frekvence jádra, mapa pamětí, velikosti Flash a RAM pamětí, typy periferií a jejich bázové adresy, rozsahy vstupních a výstupních napětí, různé důležité údaje specifické pro daný čip, atd.



Obrázek 3.2 Blokové schema mikropočítače MC56F8006/MC56F8002

- Peripheral Reference Manual – dokument popisující periferie mikropočítače pro danou rodinu.
- Reference Manual – vyčerpávající popis jádra mikropočítače
- Chip Errata – seznam zjištěných chyb a jejich případná řešení

3.4. PIT – Programmable Interval Timer

PIT timer modul je velmi jednoduchý časovací modul sloužící pro nenáročné časování aplikace. Mikrokontrolér rovněž nabízí sofitikovaný modul časovačů/čítačů, který se jmenuje Dual Timer (DTMR). DTMR je určen pro komplexnější použití. Byla by škoda používat DTMR modul pouze pro generování periodických událostí. DTMR je z hlediska plochy křemíku výrazně větší než modul PIT.

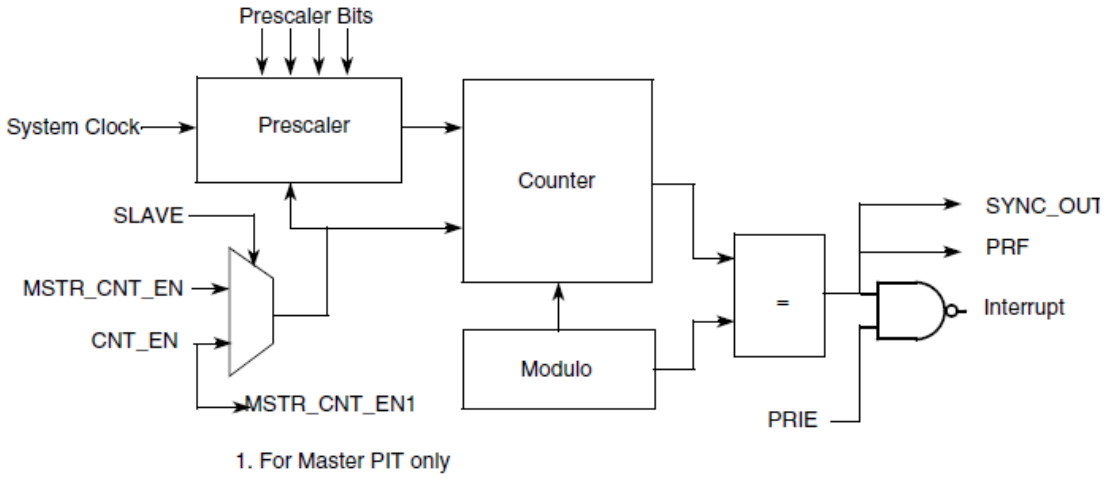
PIT timer obsahuje 16-bitový up counter, modulo registr a řídicí registr. Modulo a řídicí registry jsou pro zápis i pro čtení, zatímco counter registr je pouze pro čtení.

Modulo registr je naplněn číslem, do kterého bude čítač čítat. Pokud čítač dočítá do hodnoty modulo, zresetuje se na číslo 0x0000 a začne nové čítání. V případě dosažení hodnoty modula se nastaví flag, který může generovat přerušení, pokud je povoleno.

3.4.1. *PIT – vlastnosti*

- 16-bitový časovač/čítač
- Programovatelné čítání modulo
- Slave mode – umožňuje synchronizaci povolení několika PIT modulů

3.4.2. *PIT – Blokový diagram*



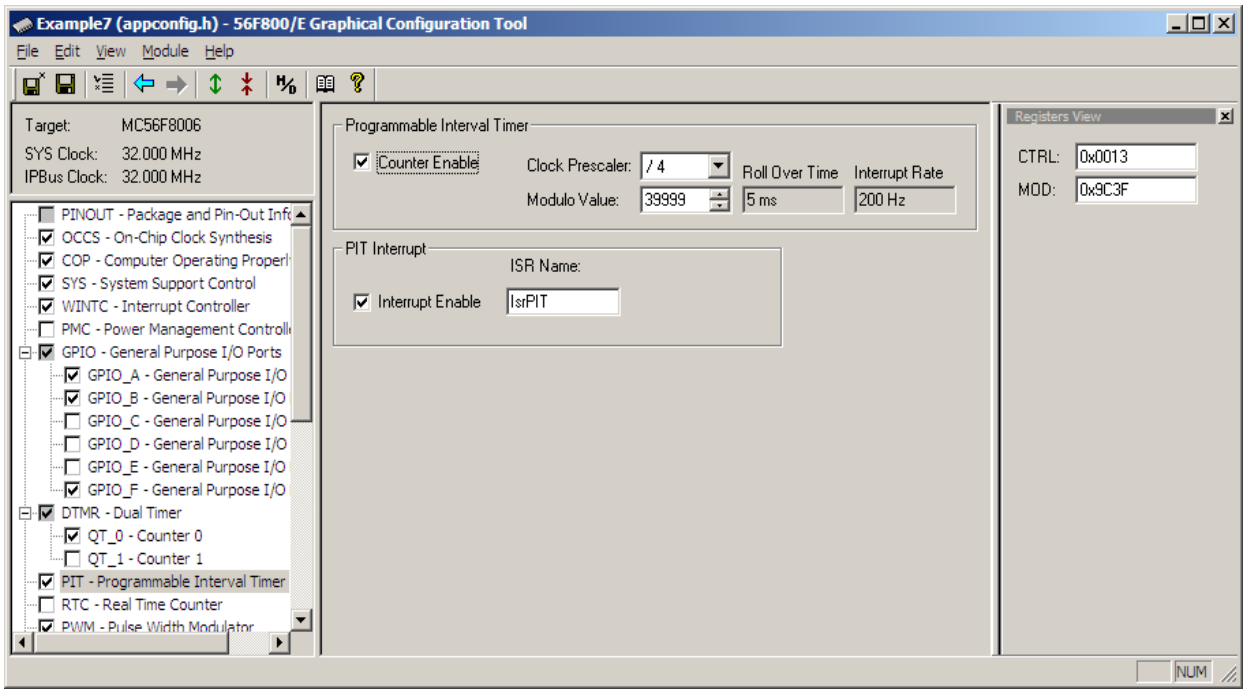
Obrázek 3.3 Blokový diagram PIT modulu

3.4.3. *PIT – typické nastavení pomocí konfiguračního nástroje GCT*

Předpokládaná funkčnost:

- Periodicky volané přerušení
- Perioda - 5 ms
- Jméno přerušovací funkce - IsrPIT

Nastavení provedené pomocí GCT je zcela intuitivní formou grafického rozhraní, příklad viz níže.



Obrázek 3.4 Nastavení PIT module pomocí GCT

Z grafického nastavení vidíme které parametry PIT modulu bylo zapotřebí nastavit pro dosažení předpokládané funkčnosti.

- Systémové hodiny (SYS Clock) a hodiny periferních modulů (IPBus Clock) závisí na globálním nastavení mikrokontroléru, které najdeme v modulu OCCS (On-Chip Clock Synthesis). V našem případě - 32 MHz
- Předdělička PIT modulu (Clock Prescaler) je nastavena na hodnotu - 4
- Hodnota modulo registru (Modulo Value) definuje periodu čítání PIT modulu - 39999
- Je zapotřebí povolit PIT modul zatržením "tick boxu" Counter Enable.
- Dále je nutno přiřadit jméno obslužné funkce ve vektoru přerušení, v našem případě do pole (ISR Name) zapíšeme jméno funkce - IsrPIT a v uživatelském programu pak nadefinujeme funkci "void IsrPIT(void)"
- Nesmíme zapomenout povolit příslušné přerušení zatržením "tick boxu" Interrupt Enable
- V pravé části okna nástroje GCT vidíme seznam registrů a jejich obsah, který se mění na základě předchozích nastavení

- Control registr (CTRL) obsahuje - 0x0013
- Modulo register (MOD) obsahuje číslo v hexadecimálním vyjádření - 0x9C3F

3.5. ADC – Analog-to-Digital Converter

ADC modul se skládá ze dvou nezávislých ADC převodníků, z nichž každý obsahuje samostatný sample and hold (S/H) obvod. Převodníky dokáží zpracovat až 28 (2 x 14) externích analogových vstupů plus 7 interních vstupů. Analogové vstupy jsou jednoduše konfigurovatelné díky vstupnímu multiplexeru. ADC kromě analogové části obsahuje i výkonnou řídicí logiku umožňující efektivní a snadné nastavování a přepogramovávání v reálném čase

3.5.1. *ADC - vlastnosti*

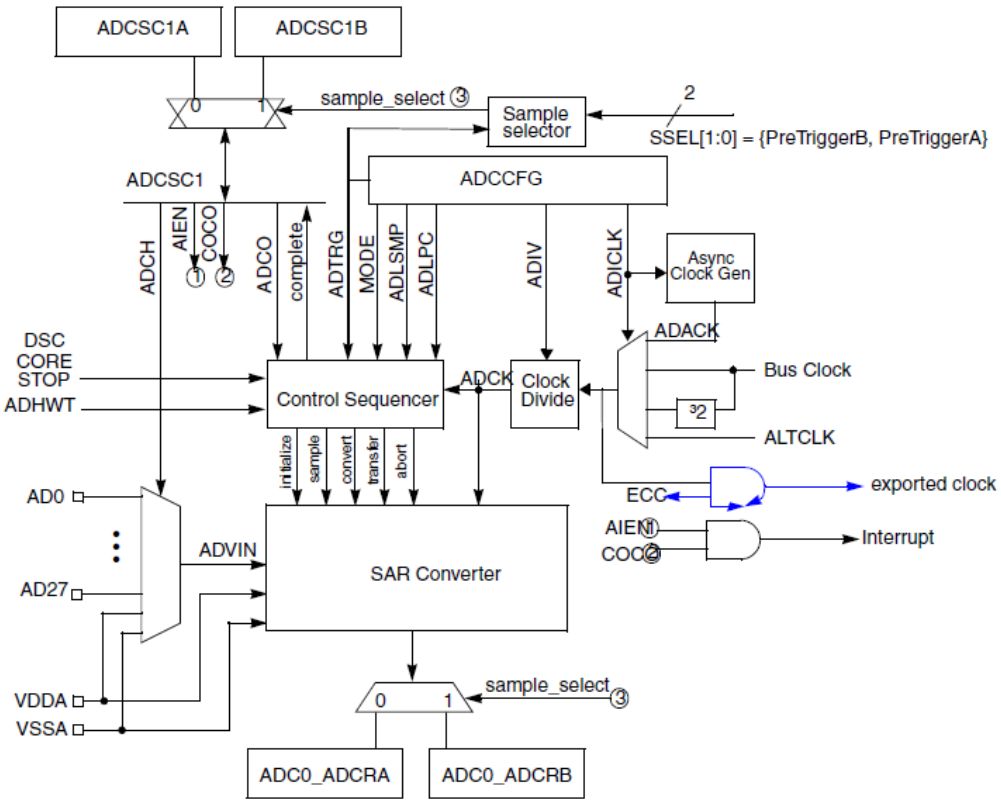
ADC modul charakterizují následující vlastnosti:

- Rozlišení – 12-bitů
- ADC umožňuje konfiguraci převodu pro případy 12-, 10- nebo 8-bitového výsledku
- Výstupní formát dat uložený do 16-bitového registru (bity 0 až 15) je následující:
  - 12-bitový výsledek - ADCR[14:3] obsahuje 12-bitový výsledek převodu, ADCR[2:0] - spodní 3 bity jsou vždy nulové, ADCR[15] je rovněž nulový.
  - 10-bitový výsledek - ADCR[12:3] obsahuje 10-bitový výsledek převodu, ADCR[2:0] - spodní 3 bity jsou vždy nulové, ADCR[15:13] jsou rovněž nulové.
  - 8-bitový výsledek - ADCR[10:3] obsahuje 8-bitový výsledek převodu, ADCR[2:0] - spodní 3 bity jsou vždy nulové, ADCR[15:11] jsou rovněž nulové.
- Možnost synchronizace ADC s PWM modulem – propojeno vnitřně na čipu, je zapotřebí správně nastavit moduly PWM, ADC a PDB

- Schopnost sekvenčně převést a uložit až 4 měření
- Každý nezávislý převodník obsahuje 2 result registry A a B, kde jsou uloženy výsledky převodu. Je nutné vyčíst výsledky převodu před ukončením převodu následujícího, aby nedošlo k přepsání registrů novými daty. Ukončení převodu je indikováno nastavením bitu COCO (Conversion Complete Flag). COCO = 0 znamená, že převod není ukončen, COCO = 1 znamená, že převod byl ukončen a je možno vyčíst data z result registrů.
- Podpora pro převádění v simultánním režimu - oba převodníky se spouštějí ve stejném okamžiku - sample & hold + převod
- Teplotní sensor - připojený na kanály ANA26 a ANB26
- Ping-pong mode - hardwarová synchronizace umožňující převod 4 analogových signálu

3.5.2. *ADC – blokový diagram*

Následující Obrázek 3.5 demonstruje konfiguraci ADC modulu



Obrázek 3.5 Blový diagram ADC modulu

3.6. PDB – Programmable Delay Block

Aplikace vyžadující synchronizaci PWM signálu s okamžikem snímání analogových veličin využívají pro přesné nastavení okamžiku spouštění ADC modulu právě PDB modul. PWM modul generuje synchronizační signál SYNC. Primární funkce PDB modulu je vytvoření programovatelného zpoždění relativně k signálu SYNC a generování signálů pro spouštění ADC modulu a modulu PGA.

Aletrnativní funkce PDB modulu je časování samplovacího okna pro modul komparátoru.

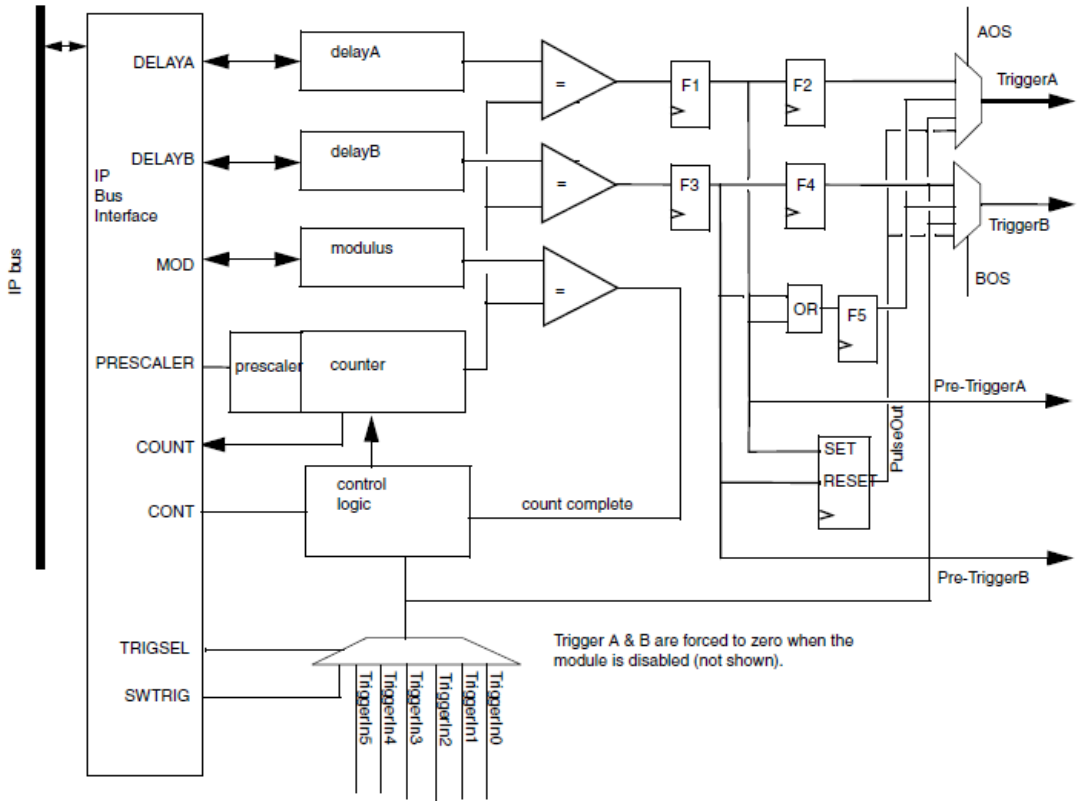
3.6.1. *PDB - vlastnosti*

PDB modul charakterizují následující vlastnosti:

- Nástupná hrana signálu trigger_in spouští čítač
- Podpora dvou trigger_out signálů. Každý z nich umožňuje nezávisle řídit zpoždění oproti sync_in signálu
- Výstupní spouštěcí signály mohou být zpracovány logickou funkcí OR, což umožňuje plánovat 2 převody iniciované jednou spouštěcí událostí
- Podporuje módy "continuous trigger" a "single-shot"
- Každý výstupní spouštěcí signál může být povolen/zakázán nezávisle

3.6.2. *PDB – blokový diagram*

Obrázek 3.6 demonstruje konfiguraci PDB modulu.



Obrázek 3.6 Blokový diagram PDB modulu

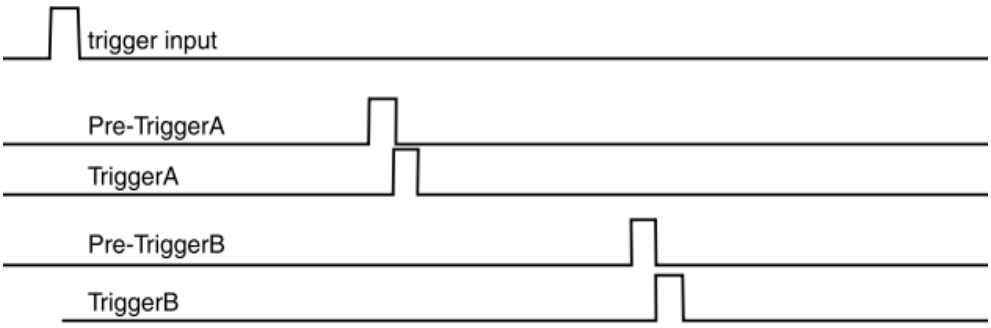
3.6.3. *PDB – módy*

- Disabled - čítač je zakázaný (neběží) a oba signály TriggerA a TriggerB jsou nulové
- Enabled OneShot - čítač je povolený (běží) a je restartován po dočítání do 0 a příchodu pozitivní hrany na trigger input (TriggerIn0, TriggerIn1, ..., TriggerIn6). Signály TriggerA a TriggerB generují pouze jeden výstupní trigger na jeden vstupní trigger.
- Enabled Continuous - čítač je povolený (běží) a je restartován automaticky při každém dočítání do 0. Čítač začíná čítat od 0, po dosažení hodnoty čítače, který je roven hodnotě v PDB_MOD registru (PDB Modulus Register) dojde k restartování čítače a cyklus se opakuje. Tento režim umožňuje kontinuální generaci výstupních signalů TriggerA a TriggerB jako výsledek jednoho vstupního trigger signálu.
- Bypassed - vstupní trigger se vyhne logice PDB modulu. Je možno aplikovat bypass pouze na jeden ze dvou výstupních trigger signálů. Tento mód je možno kombinovat s jakýmkoliv módem zmíněném výše.

- TwoShot a Continuous TwoShot módy - v módech Enabled OneShot a Enabled Continuous lze výstupy Delay A a Delay B komparátorů kombinovat takovým způsobem, že dva ADC převody mohou být trigrovány jednou vstupní událostí.
- Single Pulse a Continuous Pulse módy - v módech Enabled OneShot a Enabled Continuous lze výstupy Delay A a Delay B komparátorů kombinovat takovým způsobem, že lze generovat výstupní puls s přesným řízením nástupné a sestupné hrany pulsu.

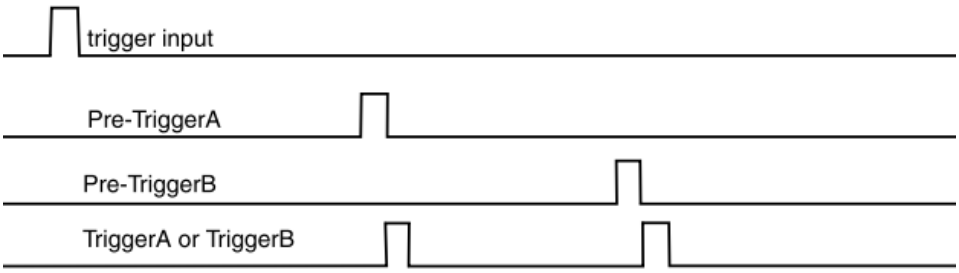
Z blokového diagramu Obrázek 3.6 je zřejmé, že se generují dva signály TriggerA a TriggerB, jejichž zpoždění se definují pomocí PDB DelayA a PDB DelayB registry. Zároveň se generují dva signály Pre-TriggerA a Pre-TriggerB. Tyto dva signály se generují 1 peripheral bus clock před signály TriggerA a TriggerB a slouží jako řídicí signály pro moduly ADC a PGA. Pre-Trigger signály (Pre-TriggerA a Pre-TriggerB) selektují mezi ADC result registry v závislosti na aktuálně generovaném Trigger signalů (TriggerA a TriggerB).

Obrázek 3. 7 demonstruje nezávislé generování signálů TriggerA a TriggerB s použitím OneShot módu. Vstupní trigger signál iniciuje čítání čítačů a registry DelayA a DelayB defunují zpoždění vůči okamžiku, kdy byl vygenerován vstupní trigger signál. Z obrázku je vidět, kdy se generují signály Pre-TriggerA a Pre-triggerB.



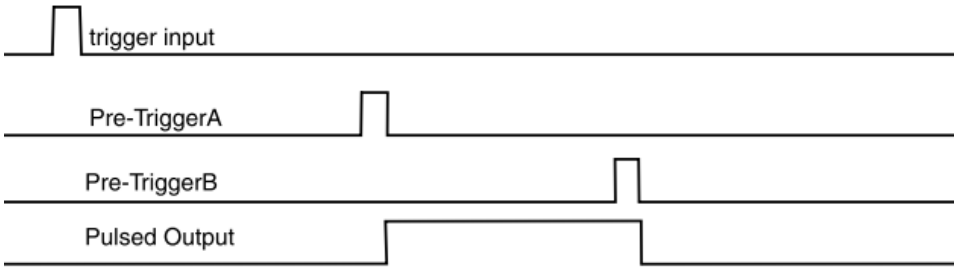
Obrázek 3. 7 Generování nezávislých TriggerA a TriggerB signálů

Obrázek 3.8 demonstruje TwoShot mód. V tomto případě jsou oba ADC převodníky spouštěny jak na signál TriggerA, tak i na signál TriggerB. Tato funkčnost předpokládé správné nastavení ADC převodníku. Toto nastavení vede k tomu, že dostaneme 4 ADC výsledky s použitím dvou trigger signálů a 4 result registrů ADC modulu.



Obrázek 3.8 Tzv. Ping-Pong mód

Obrázek 3.9 demonstruje Triggered Pulsed mód. Signály Pre-TriggerA a Pre-TriggerB přesně definují nástupnou a sestupnou hranu pulsu.



Obrázek 3.9 Triggered Pulsed mód

3.1. PGA – Programmable Gain Amplifier

PGA modul předpokládá spolupráci s ADC modulem. Samostatné využití PGA modulu nemá smysl. Slouží k zesilování analogových signálů následně pak převáděných ADC modulem

3.1.1. *PGA - vlastnosti*

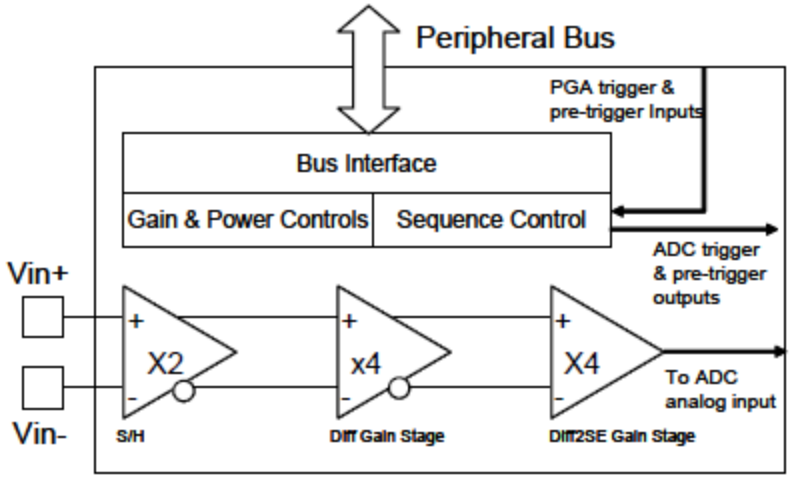
PGA modul charakterizují následující vlastnosti:

- Několik programovatelných zesílení - 1x, 2x, 4x, 8x, 16x a 32x
- Samplovací čas je možno přesně řídit - méně než 0,1 us
- Samplovací okamžik je možno synchronizovat s PWM modulem a PDB modulem
- PGA není analogový zesilovač
- Rozsah vstupního napětí - rail-to-rail

- "Single-ended" výstup je přímo vedený na ADC kanály ANA15 a ANB15
- Trigrování - software nebo hardware
- Zahrnuje kalibrační vlastnosti umožňující kalibraci PGA modulu

3.1.2. *PGA – blokový diagram*

Obrázek 3.10 demonstruje konfiguraci PGA modulu



Obrázek 3.10 Blokový diagram PGA modulu

3.2. PWM – Pulse Width Modulator

PWM modul je komplexní a poskytuje celou řadu komplikovaných nastavení v závislosti na připojeném externím zařízení. PWM modul umožňuje generovat tři komplementární výstupy, šest nezávislých (independent) výstupů nebo jejich kombinace jako např.: jeden komplementární a čtyři nezávislé výstupy. Oba módy edge- a center-aligned dovolují řídit šířku pulsu od 0% až do 100%. PWM modul pro generaci PWM signálu používá 15-bitový čítač a to pro všech 6 kanálů. Rozlišení PWM signálu závisí na zvoleném módu a pro edge-aligned mód činí jeden hodinový signál, v případě center-aligned módu je to pak dva hodinové signály. Hodinový signál PWM modulu je volitelný a může být roven systémovým hodinám nebo trojnásobku systémových hodin. V případě generování signálu v komplementárním módu je automaticky vkládaný deadtime (řízeno hardwarově). Každý PWM výstup může být volitelně řízen PWM generátorem, časovačem, výsledkem ADC převodu, GPIO pinem nebo

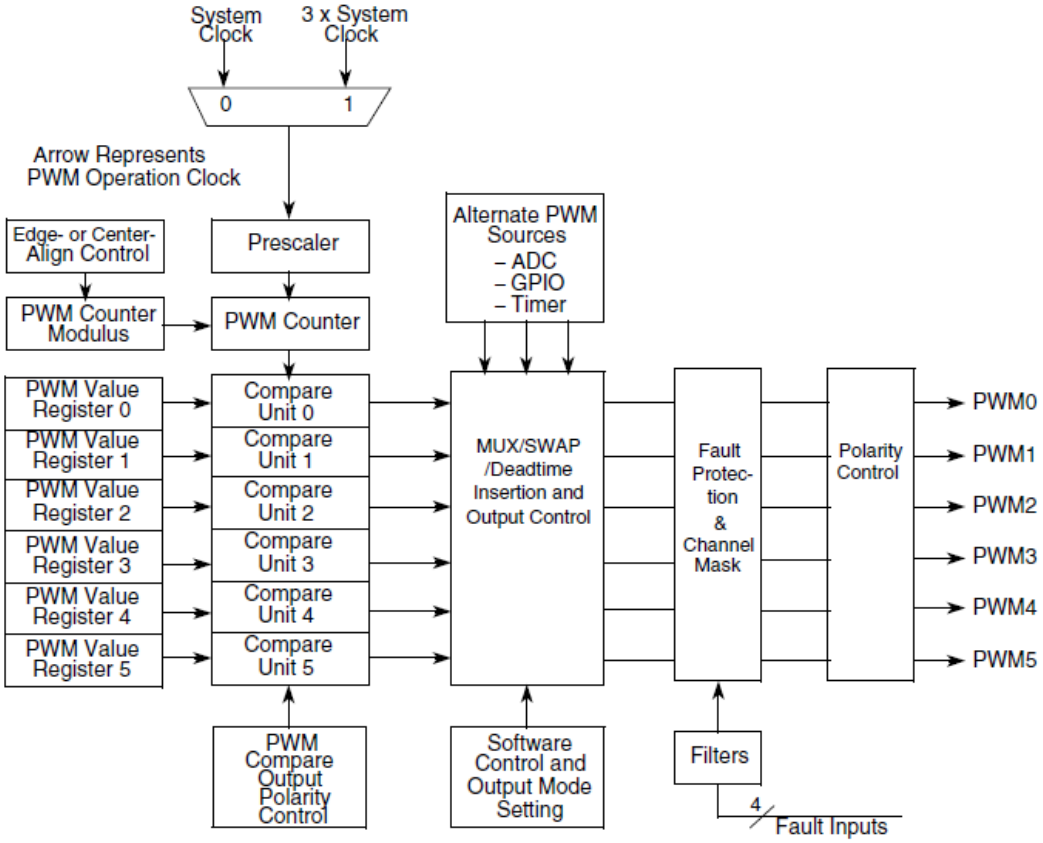
manuálně softwarem. Nezávisle lze řídit výstupní polaritu PWM signálu pro horní a dolní výstupy. Rovněž lze generovat nesymetrický PWM výstup.

3.2.1. *PWM – vlastnosti*

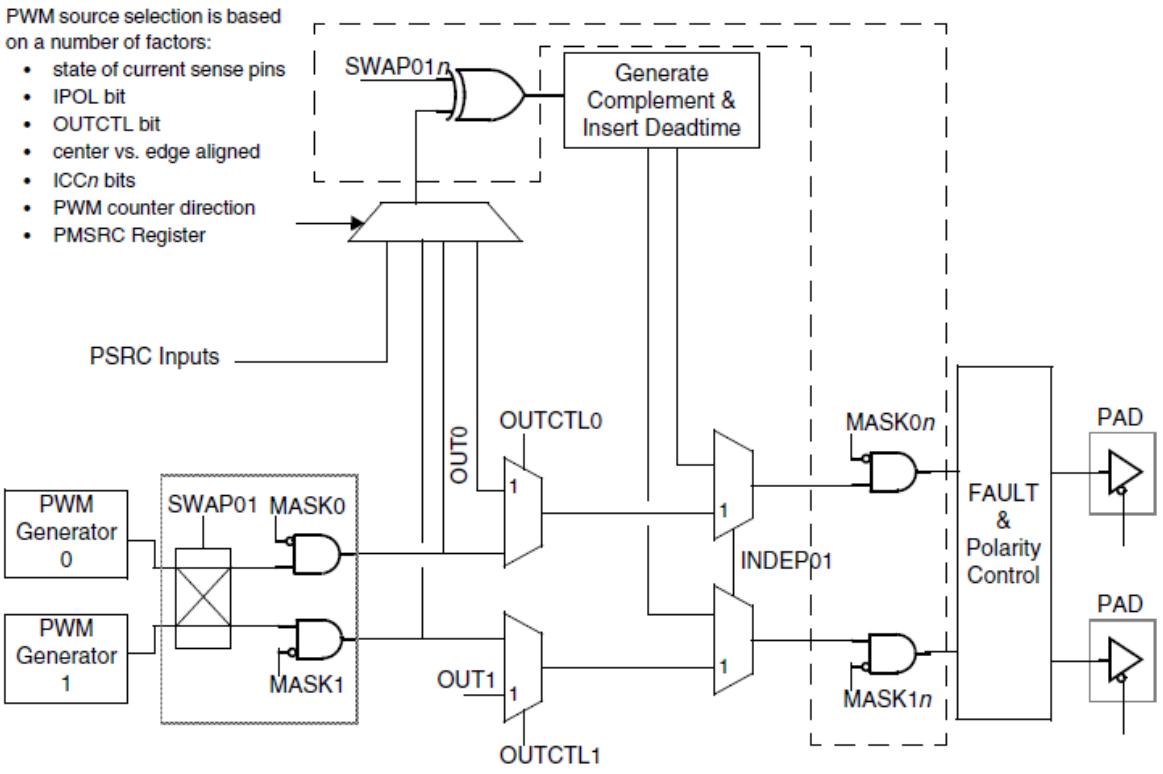
- PWM operační hodiny běží buď na stejné frekvenci jako systémové hodiny nebo na trojnásobku systémových hodin.
- 6 PWM výstupních signálů
  - Všechny nezávislé (independent)
  - Tři komplementární páry (Complementary pairs)
  - Kombinace nezávislých a komplementárních párů
- Vlastnosti PWM výstupů v případě generování komplementárních párů
  - Generovaní deadtime nezávisle pro nástupnou a sestupnou hranu
  - Oddělené generování korekce šířky pulsu pro horní a spodní výstupy prostřednictvím software
  - Asymetrický PWM výstup v případě center align módu
  - Oddělené řízení polarity výstupů pro horní a dolní PWM výstupy
  - Edge- nebo center-alligned PWM signály
  - 15-bitové rozlišení
  - Reload schopnost uprostřed PWM cyklu (half-cycle reload)
  - Integrální reload rate od 1 do 16
  - Možnost řízení PWM výstupů softwarem
- Programovatelná fault ochrana
- Řízení polarity PWM výstupů
- Push-Pull nebo open drain módy pro PWM piny
- Programovatelný vstupní filtr pro chybové piny

- Ochrana PWM modulu ve formě registrů s ochranou proti zápisu

3.2.2. *PWM – blokový diagram*



Obrázek 3.11 Blokový diagram PWM modulu



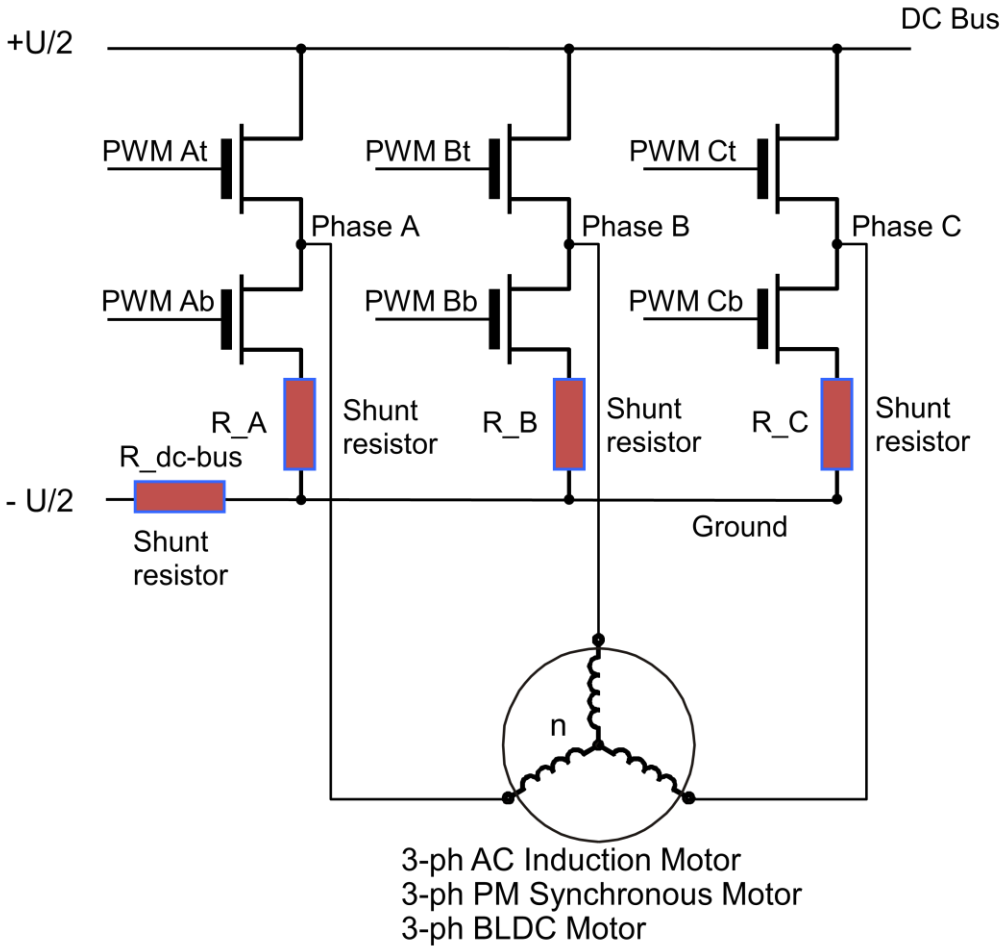
Obrázek 3.12 Detailní blokový diagram funkcí MUX, Swap and Deadtime

3.1. ADC a PWM synchronizace – princip měření proudu

Pro celou řadu aplikací řízení elektrických pohonů je zapotřebí měřit okamžitou hodnotu proudu protékajícího vinutím elektrického stroje a následně použít tuto informaci pro zlepšení dynamických vlastností řízeného pohonu a další pomocné výpočty důležité pro konkrétní aplikaci. Příkladem může být momentové řízení, podpora pro odbuzování pohonu, výpočet momentu na hřídeli, detekce nerovnoměrnosti zatěžovacího momentu, atd.

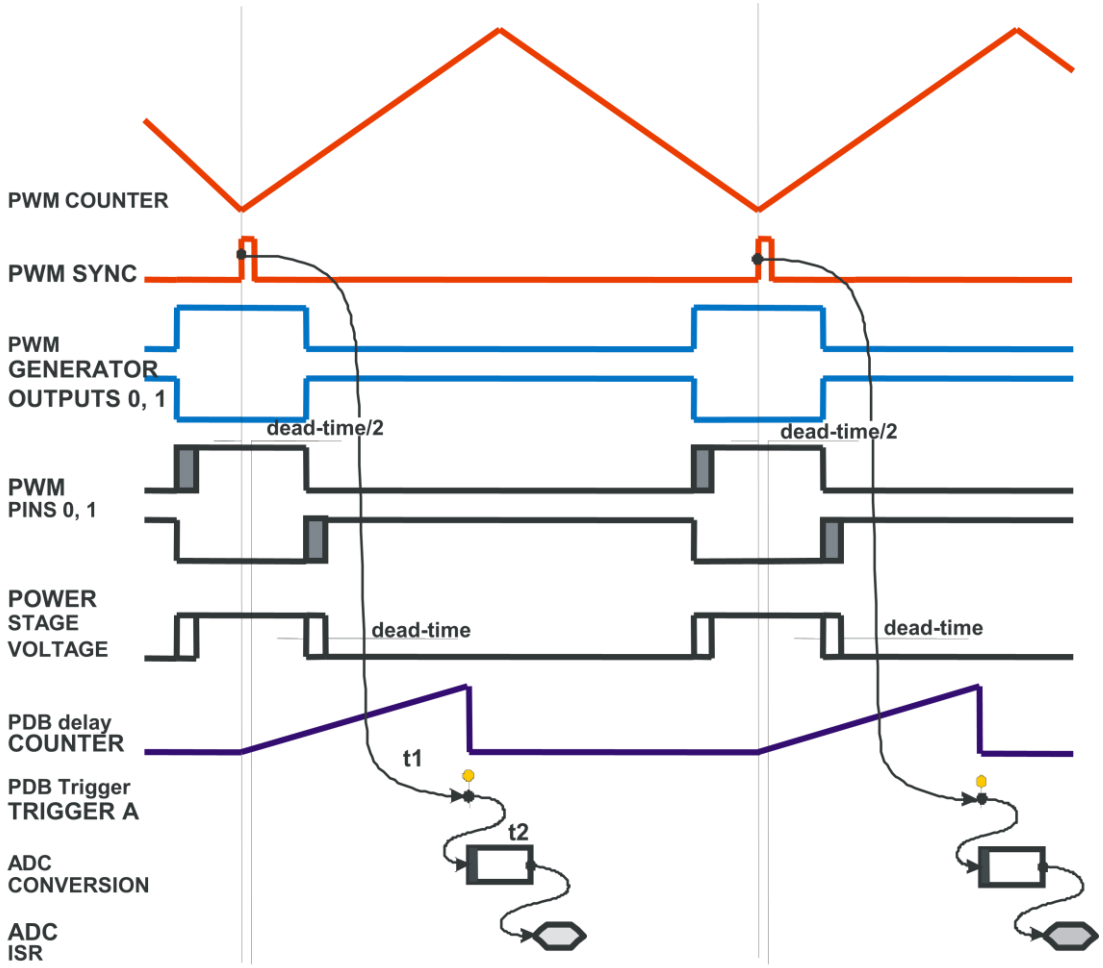
Existuje celá řada možností měření proudu od relativně jednoduchého měření úbytku napětí na snímacím rezistoru, přes proudové transformátory až po více či méně komplikované snímače. Jeden z faktorů, který rozhoduje o volbě měřicí metody, je cena řešení. Z tohoto hlediska je měření s použitím rezistorů preferovanou volbou, která však přináší celou řadu úskalí při samotné implementaci. Jeden z omezujících faktorů použití je volba počtu rezistorů a jejich umístění v měniči. To pak vede k volbě strategie měření. Z topologie měniče, umístění jednotlivých rezistorů a řídicí metody pak získáme informaci o tom, v jakých časových okamžicích proud protéká daným rezistorem a kdy na rezistoru nic nezměříme. Jelikož pro generaci napětí pro pohon používáme

šírkově pulzní modulaci generovanou PWM modulem, dokážeme přesně určit okamžiky měření proudu relativně k signálu generovanému PWM modulem. Samotné měření analogového signálu napětí se provádí s použitím ADC modulu. Tato úvaha nás vede k nutnosti nějakým způsobem synchronizovat moduly PWM a ADC. Podle typu měřicí metody je okamžik měření konstantní vůči referenčnímu okamžiku generování PWM signálů nebo se může měnit v každém PWM cyklu. Tato variabilita vede k nutnosti hardwarové podpory ze strany mikrokontroleru.



Obrázek 3.13 Topologie střídače s měřením proudu pomocí měřicích rezistorů

Z topologie střídače je zřejmé, že proud můžeme měřit na rezistorech umístěných v každé větvi (R_A, R_B, R_C), čímž získáme informaci o proudech v jednotlivých fázích nebo můžeme měřit proud tekoucí do motoru pomocí rezistoru R_dc-bus. Princip měření zahrnující všechny důležité signály a jejich časování je zobrazen na následujícím obrázku

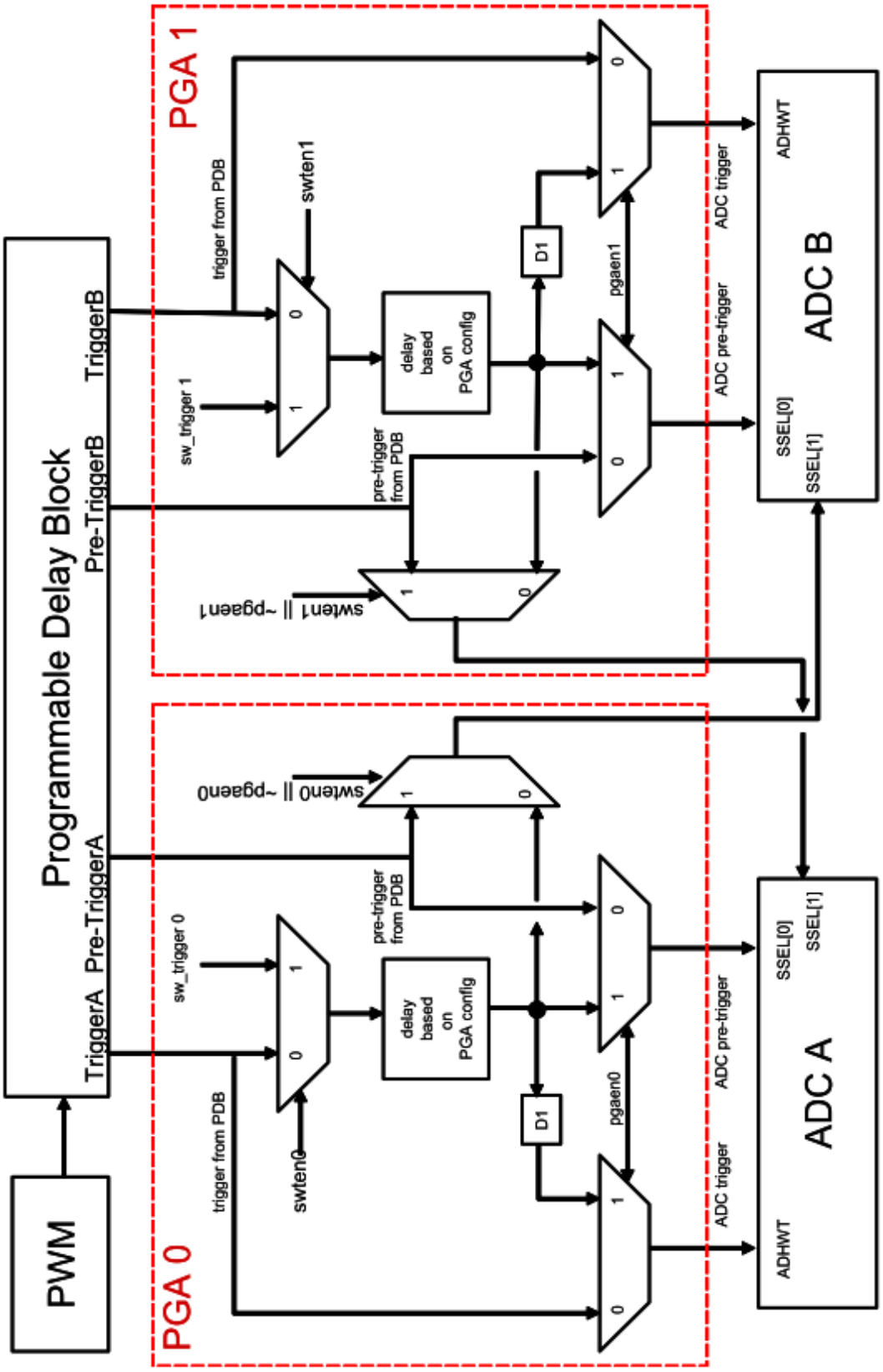


Obrázek 3.14 ADC - PWM synchronizace - časování signálů

3.2. ADC a PWM synchronizace – podpora mikrokontroleru MC56F800x

Mikrokontroléry MC56F800x poskytují poměrně komplexní podporu pro měření proudu. K tomuto účelu slouží tři moduly: PWM, PDB a ADC s řadou interních signálů, které při vhodném propojení umožňují plně hardwarovou podporu měření.

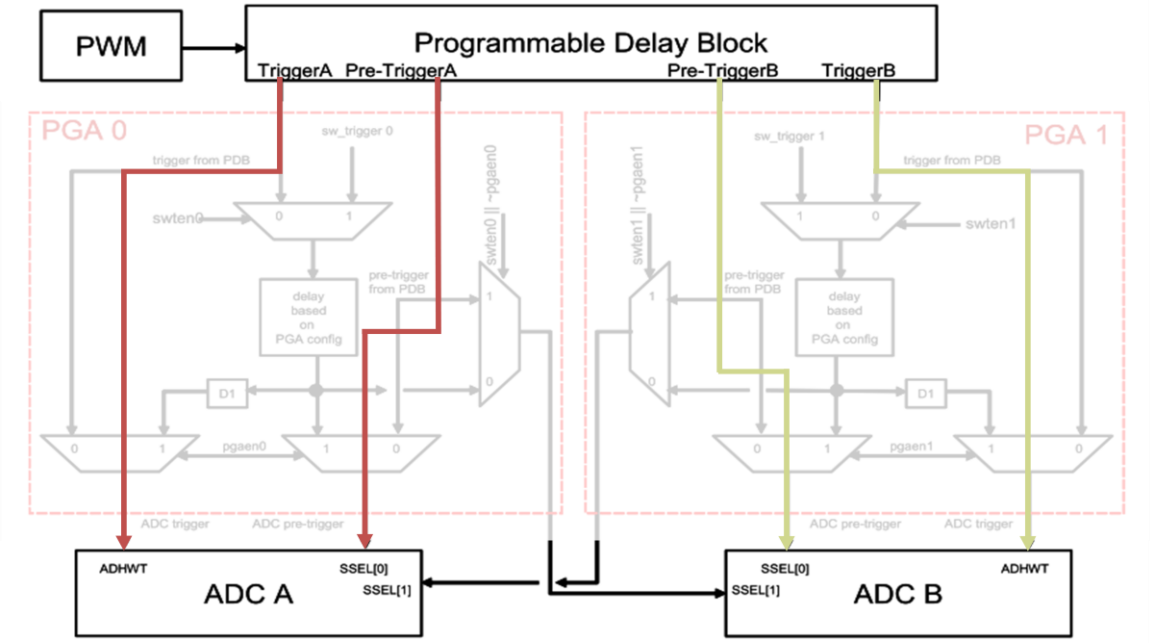
Obrázek 3.108 ukazuje blokové schéma propojení modulů PWM, PDB, PGA a ADC za účelem plně hardwarové synchronizace. V případě, že nepoužíváme modul PGA (Programmable Gain Amplifier), lze moduly PWM, PDB a ADC propojit bez použití PGA modulu.



Obrázek 3.15 Propojení modulů PWM, PDB, PGA a ADC

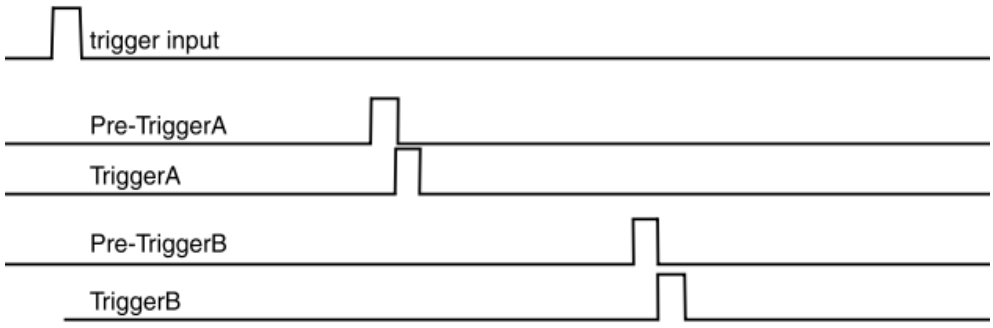
Principiálně je možno celý řetězec nakonfigurovat dvěma základními způsoby a to tak, že ADC moduly jsou nakonfigurovány nezávisle, což předpokládá, že každý ADC modul je ovládaný nezávisle z PDB modulu generováním TriggerA a TriggerB signálů. Druhá možná konfigurace předpokládá, že ADC moduly běží v simultánním módu. Pak vhodným nastavením modulu PDB lze spouštět oba ADC převodníky ve stejném okamžiku, což umožňuje až 4 převody generované hardwarově.

3.2.1. *PWM - ADC synchronizace – nezávislé ovládání ADC převodníků*



Obrázek 3.16 PWM - ADC synchronizace - nezávislé ovládání ADC převodníků

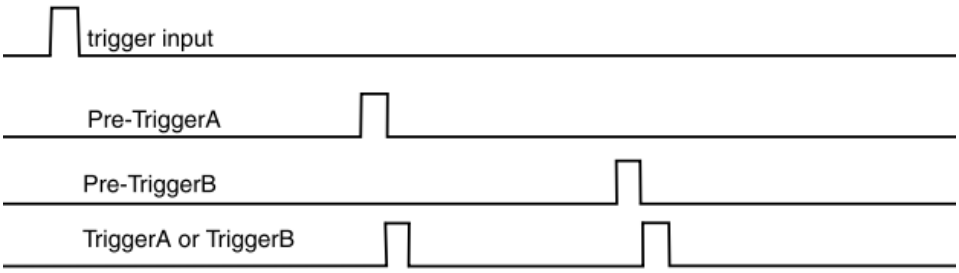
Tento způsob synchronizace předpokládá nastavení PDB modulu podle následujícího obrázku



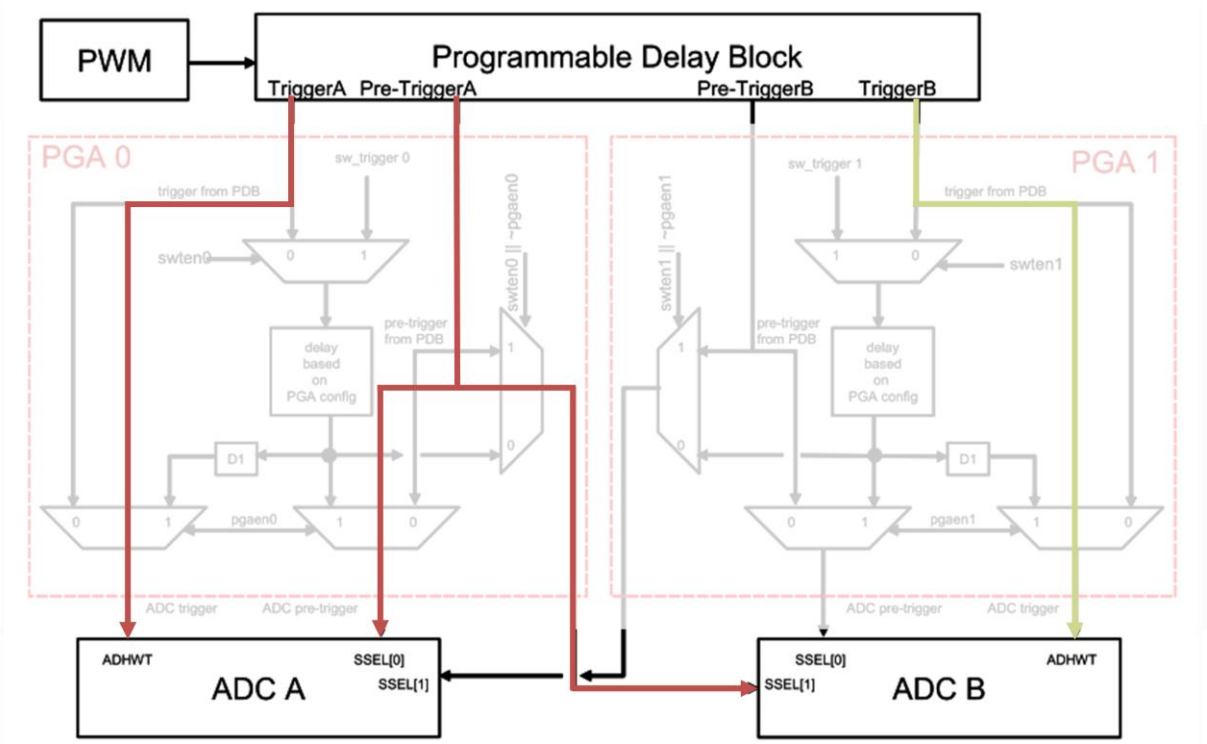
Obrázek 3.17 Generování nezávislých TriggerA a TriggerB signálů

3.2.2. *PWM - ADC synchronizace – simultánní ovládání ADC převodníků (Ored Mode)*

Pro tento typ synchronizace je zapotřebí nastavit PDB modul do tzv. ping-pong módu viz. Obrázek 3.18. Za tohoto předpokladu je možné spouštět oba převodníky simultánně. Pre-Trigger signály pak selektují příslušný result registr odpovídajícího převodníku.



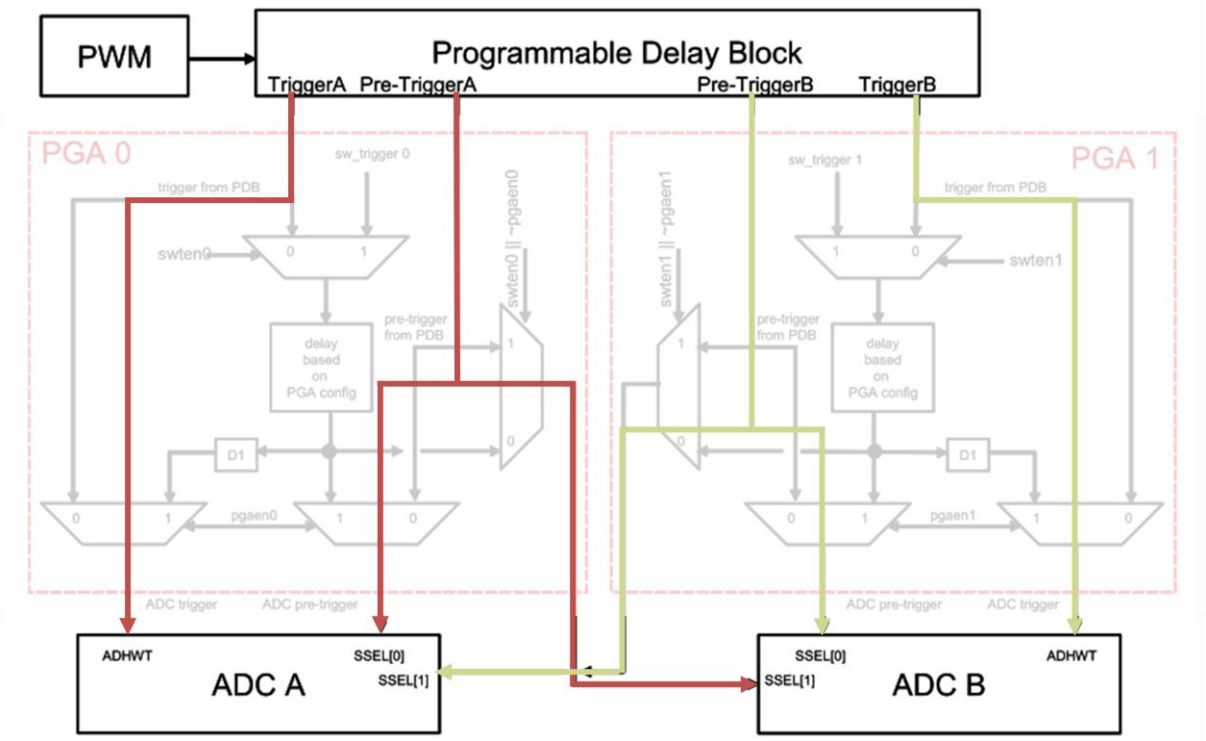
Obrázek 3.18 Tzv. Ping-Pong mód



Obrázek 3.19 PWM - ADC synchronizace - simultánní ovládání ADC převodníků, první převod

Obrázek 3.19 ukazuje, jakým způsobem dochází k volbě result registrů jednotlivých převodníků. V případě převodníku ADC A signál Pre-TriggerA

selektuje result registr 0, zatímco v případě převodníku ADC B se první převod ukládá do result registru 1.



Obrázek 3.20 PWM - ADC synchronizace - simultánní ovládání ADC převodníků, druhý převod

V dalším kroku, který demonstruje Obrázek 3.20, vidíme, že dochází k vygenerování signálu Pre-TriggerB jeden clock před signály Trigger. V tomto případě dojde k výběru result registru 1 pro převodník ADC A a result registr 0 pro převodník ADC B. Od tohoto okamžiku jsou k dispozici výsledky 4 převodů analogových signálů a je zapotřebí výsledky vyčíst dříve, než dojde k následným převodům iniciovaným trigrovacím signálem generovaným periodicky PWM modulem.

4. Quick Start Tool A Graphical Configuration Tool – vývojové prostředí

DSP56800E_Quick_Start je vývojové prostředí, které umožňuje programátorovi rychle a efektivně vytvořit aplikační kód více méně nezávislý na architektuře mikropočítače. Poskytuje softwarovou infrastrukturu pro vytváření kódu, který je snadno přenositelný na jiné platformy. Přenositelnost je samozřejmě omezena funkčností cílového procesoru. Součástí vývojového prostředí Quick_Start je i tzv. GCT (Graphical Configuration Tool), který podporuje efektivní nastavování chování mikropočítače pomocí propracovaného grafického prostředí, které spolupracuje s prostředím Quick_Start.

4.1. Quick Start Tool – vlastnosti

Vývojové prostředí Quick_Start se skládá z několika součástí:

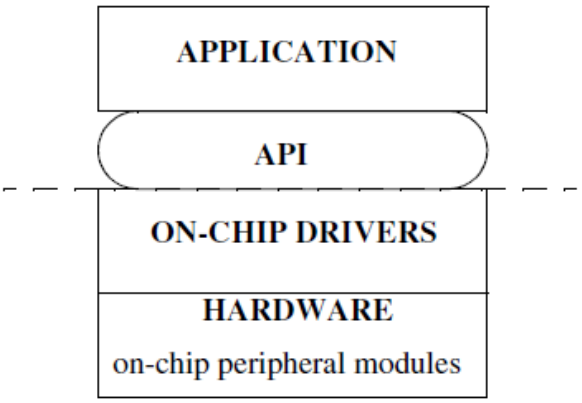
- Infrastruktura celého sytému
- On-chip drivers s jednotným API
- Vzorové aplikace
- GCT (Graphical Configuration Tool)
- Softwarová podpora nástroje FreeMaster

4.1.1. *Infrastruktura sytému Quick_Start*

Infrastruktura sytému vytváří propracovanou podporu pro práci s mikropočítači řady 56F8xxx a umožňuje integraci ostatních součástí prostředí Quick_Start. Součástí infrastruktury je celá řada souborů usnadňujících práci při programování mikropočítače. Mezi infrastrukturu systému Quick_Start patří: definice maker, přenositelná deklarace registrů, mechanismus pro statickou konfiguraci čipu, tabulka přerušení a obsluha přerušovacích vektorů, start-up kód, soubory pro konfiguraci linkeru, project templates, soubory pro inicializaci debuggeru, atd.

4.1.2. *On-chip drivers*

On-chip drivers izolují hardware od softwarové obsluhy. Jasně definované API usnadňuje programování čipu, zvyšuje čitelnost kódu a umožňuje poměrně snadnou přenositelnost výsledného kódu závislého na typu mikropočítače a druhu periferií.



Obrázek 4.1 Softwarová struktura aplikačního kódu

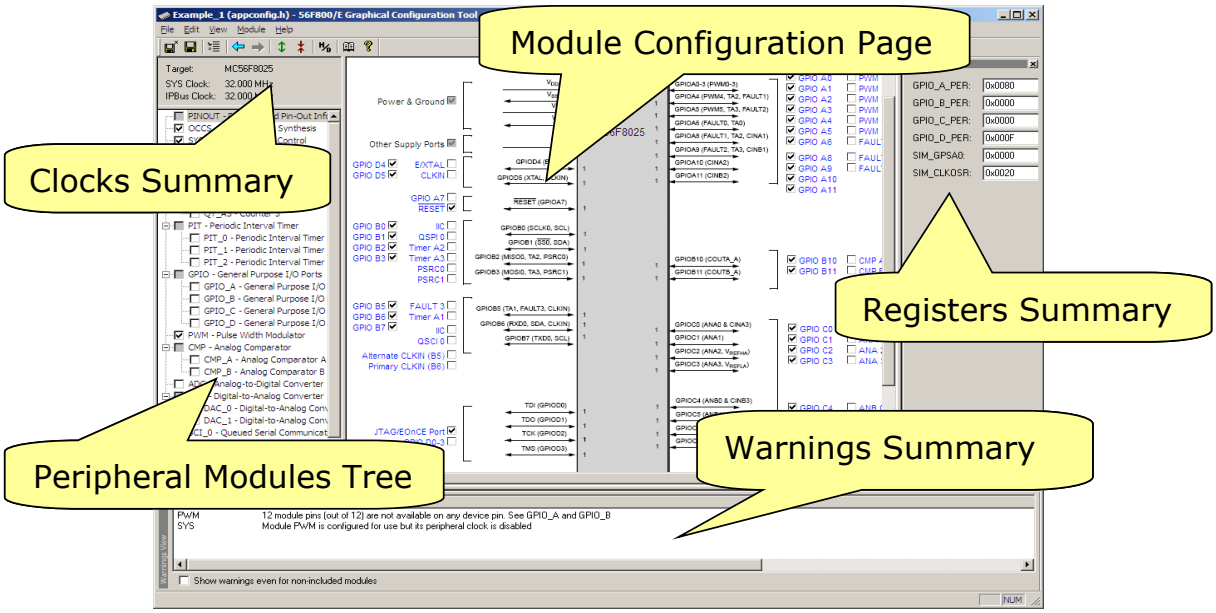
4.1.3. *Vzorové aplikace*

Prostředí Quick Start poskytuje uživateli celou řadu příkladů obsluhy typických periferií, na kterých se demonstruje použití on-chip drivers a zároveň se vysvětluje jak implementovat specifická nastavení mikropočítače. Příklady jsou jednoduché a ilustrativní. Jejich účelem je zkrátit dobu učení a tím zefektivni vývoj uživatelského software.

4.1.4. *GCT – Graphical Configuration Tool*

GCT je grafické rozhraní, které uživateli umožňuje pohodlně a srozumitelně provést statické nastavení čipu. Statické nastavení zahrnuje i nastavení interrupt vektorů pro daný zdroj přerušení a jejich obsluhu (uživatelská service routine).

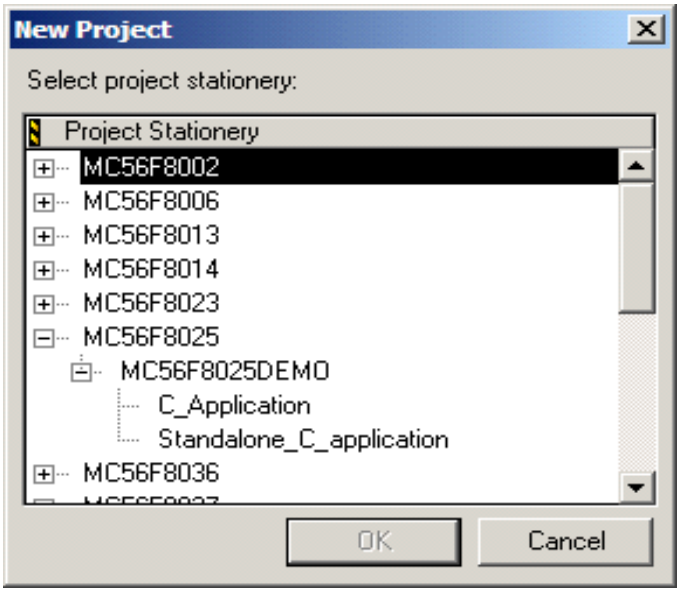
GCT je volitelný nástroj, není bezpodmínečně nutný pro funkci prostředí DSP56800E_Quick_Start. Nicméně tento nástroj výrazně zjednodušuje práci s mikropočítačem a zkracuje dobu nutnou na nastavení čipu na minimu. Proto je velmi doporučován.



Obrázek 4.2 GCT – Graphical Configuration Tool

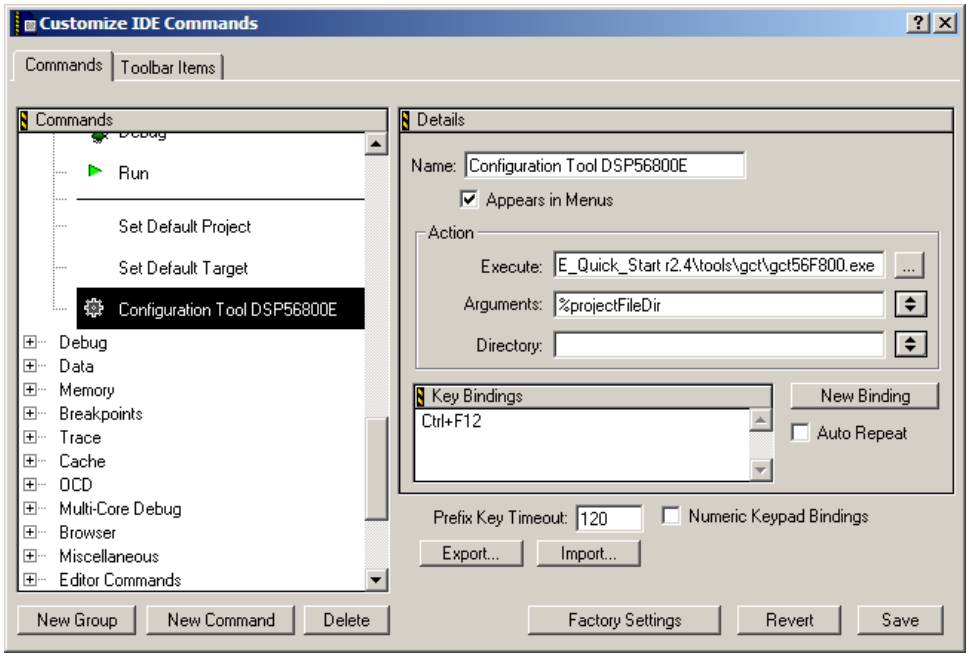
4.2. Quick Start Tool – integrace do prosředí Metrowerks CodeWarrior

Jednotlivé komponenty prostředí DSP56800E_Quick_Start jsou zakomponovány přímo do programovacího prostředí Metrowerks CodeWarrior. Quick Start poskytuje projekt template (project stationery), který po instalaci nástroje naistalován do adresářové struktury CodeWarrioru.



Obrázek 4.3 Vytváření nového projektu založeného na šabloně Quick Start

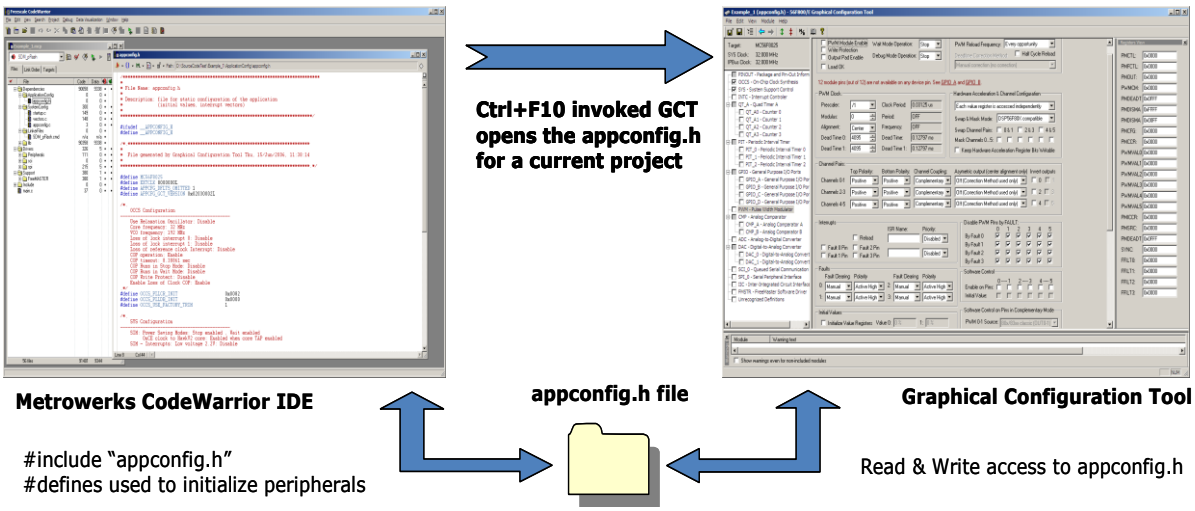
Při vytváření nového projektu je pak automaticky šablona projektu založený na prostředí Quick Start k dispozici. GCT je přímo volatelný z prostředí CodeWarrior, což v důsledku značně zjednodušuje práci s GCT nástrojem. Nastavení demonstruje Obrázek 4.4.



Obrázek 4.4 Intergace GCT do prosředí Metrowerks CodeWarrior

4.3. Spolupráce mezi GCT a nástrojem Quick Start

Spolupráci mezi nástrojem GCT a Quick Start demonstruje následující Obrázek 4.5.



Obrázek 4.5 Spolupráce mezi nástroji GCT a Quick Start

Z obrázku je zřejmé, jak funguje nastavování vlastností čipu pomocí nástroje GCT. Prostředí Metrowerks CodeWarrior z běžícího projektu otevře nástroj GCT. Ten vezme aktuální nastavení uložené v souboru „`appconfig.h`“ a zobrazí nastavení v grafické podobě. V případě, že „appconfig.h“ neobsahuje nastavení dané periferie, GCT zobrazí defaultní nastavení. V tomto okamžiku lze nastavit jakoukoliv funkčnost dostupnou na čipu prostřednictvím nástroje GCT. Uložením aktuálního nastavení dojde k přepsání souboru „appconfig.h“, který pak uchovává poslední nastavení. Prostředí Quick Start pak pracuje s tímto nastavením v závislosti na aplikaci. Nastavení uložená v souboru „appconfig.h“ je pak možno použít pro nastavení čipu pomocí „`ioctl()`“ příkazů.

4.4. ArchIO struktura

Jedná se o strukturu definovanou v ANSI C, která obsahuje všechny registry jádra a periferií. Počáteční adresa struktury je pak přiřazena takovým způsobem, aby struktura byla namapovaná přes všechny registry

`ArchIO` struktura je deklarovaná v souboru `arch.h` a je zde deklarovaná jako `extern` proměnná. Adresa proměnné typu struktura je pak přiřazena přímo v souboru „linker command file“.

Prostředí Quick Start využívá této struktury pro přístup ke všem registrům mikropočítače.

Ukázka definice části `ArchIO` sruktury – `arch.h` file:

```
typedef volatile struct
{
    arch_sTimer     TimerA;   /* TMRA_BASE     0xF000 */
    arch_sTimer     TimerB_unused;
    arch_sADC       Adc;     /* ADC_BASE      0xF080 */
    arch_sPWM       Pwm;      /* PWM_BASE      0xF0C0 */
    arch_sIntc      Intc;     /* INTC_BASE     0xF0E0 */
    arch_sSIM       Sim;     /* SIM_BASE      0xF100 */
```

```
    arch_sCOP       Cop;      /* COP_BASE      0xF120 */
    arch_sPLL       Pll;      /* PLL_BASE      0xF130 */
    arch_sLVI       Lvi;      /* LVI_BASE      0xF140 */
    UWord16         reserved4[0xFF0600];
    arch_sEOnCE     EOnCE; /* EOnCE_BASE    0xFFFF00 */
} arch_sIO;
```

`arch.h` file rovněž obsahuje deklaraci

```
Extern arch_sIO ArchIO
```

Linker command file pak obsahuje následující definici:

```
FArchIO = ADDR(.x_onchip_peripherals);
```

Tímto příkazem je pak struktuře přiřazena adresa a můžeme s ní pracovat v rámci prostředí Quick Start a uživatelského kódu.

Příklad použití ArchIO struktury – read/write operace:

```
UWord16 RegValue;     // variable definition

RegValue = ArchIO.TimerA.Channel0.HoldReg;    // read
register

ArchIO.TimerA.Channel0.CompareReg1   =   0x8000;      //
write number to reg
```

4.5. ioctl command – Input Output Control

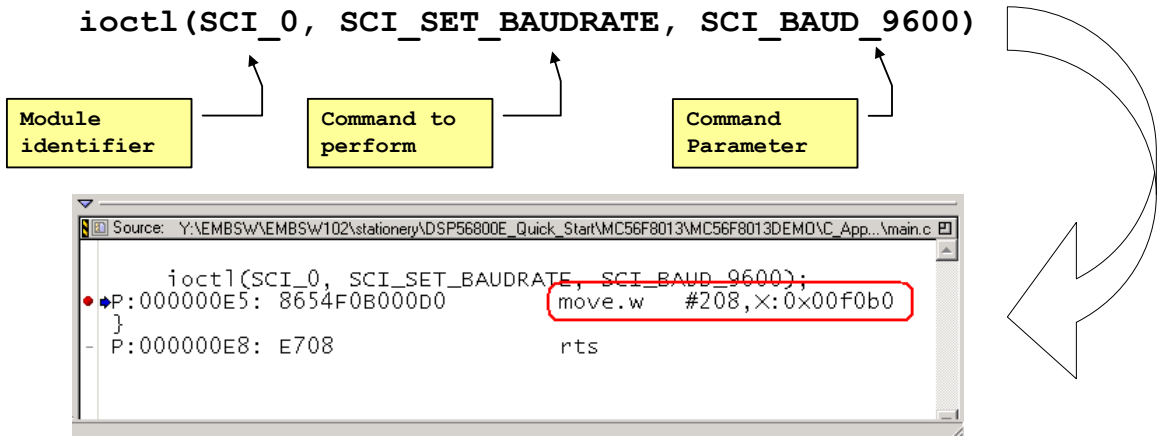
Jedním ze základních pilířů nástroje Quick Start jsou tzv. ioctl commands. Jedná se o low-level drivery, pomocí nichž řídíme funkčnost mikropočítače za běhu programu.

Vlastnosti „ioctl commands“:

- Unifikuje přístup do registrů mikropočítače
- Umožňuje plně řídit všechny zdroje mikropočítače

- Aplikační kód nesahá na registry přímo, ale pomocí „ioctl commands“
- Volání „ioctl commands“ je kompilátorem překládáno optimálně
- Zvyšuje čitelnost kódu
- Zjednodušuje přenositelnost aplikace

Formát „icotl command“ je následující:



Obrázek 4.6 ioctl command a jeho zpracování překladačem

5. FreeMASTER Tool – vývojové prostředí

Nástroj FreeMASTER je neocenitelný při vývoji aplikací běžících v reálném čase. Nástroj FreeMASTER poskytuje uživateli celou řadu vlastností, které velmi výrazně urychlí čas potřebný pro vývoj komplexní aplikace. Je hodnocen zákazníky jako vynikající nástroj, který již po zběžném osahání nechtějí dát z ruky.

5.1. FreeMASTER tool – vlastnosti

Nástroj FreeMASTER slouží především jako výkonný ladící nástroj v reálném čase. Jeho možnosti se v průběhu času neustále zlepšovaly a stále nové a nové vlastnosti byly dodávány. V současné době nástroj může sloužit jako:

- Monitor v reálném čase
- Grafický řídicí panel
- Platforma vhodná k demonstračním účelům
- Nástroj podporující prodej mikropočítačů

5.1.1. *FreeMASTER tool – monitor v reálném čase*

Nástroj FreeMASTER umožňuje přístup do libovolného paměťového místa s pevnou adresou známou při překladu a linkování. To znamená, že nepodporuje monitorování proměnných na zásobníku. Nástroj FreeMASTER po linkování dokáže analyzovat ELF soubor (Executable File), analyzuje DWARF ladící informace obsažené v ELF souboru. Na základě této analýzy pak zná adresy globálních a statických proměnných. Zná rovněž jejich velikost, typ, velikost pole a jména, takže umožňuje uživateli velice jednoduše přistupovat k proměnným definovaným v aplikačním kódu.

Nástroj FreeMASTER má dvě strany

- Embedded strana – běží na procesoru
- PC strana – běží na PC v prostředí Windows

Aby bylo možnost spolupráce obou stran nástroje FreeMASTER je zapotřebí mít komunikační kanál, kterým si obě strany budou předávat data. V tomto ohledu nástroj FreeMASTER podporuje všechny možné komunikační kanály dostupné na daném čipu.

- SCI, UART
- JTAG/EonCE (56F8xxx)
- BDM (HCS08, HCS12)
- CAN Calibration Protocol

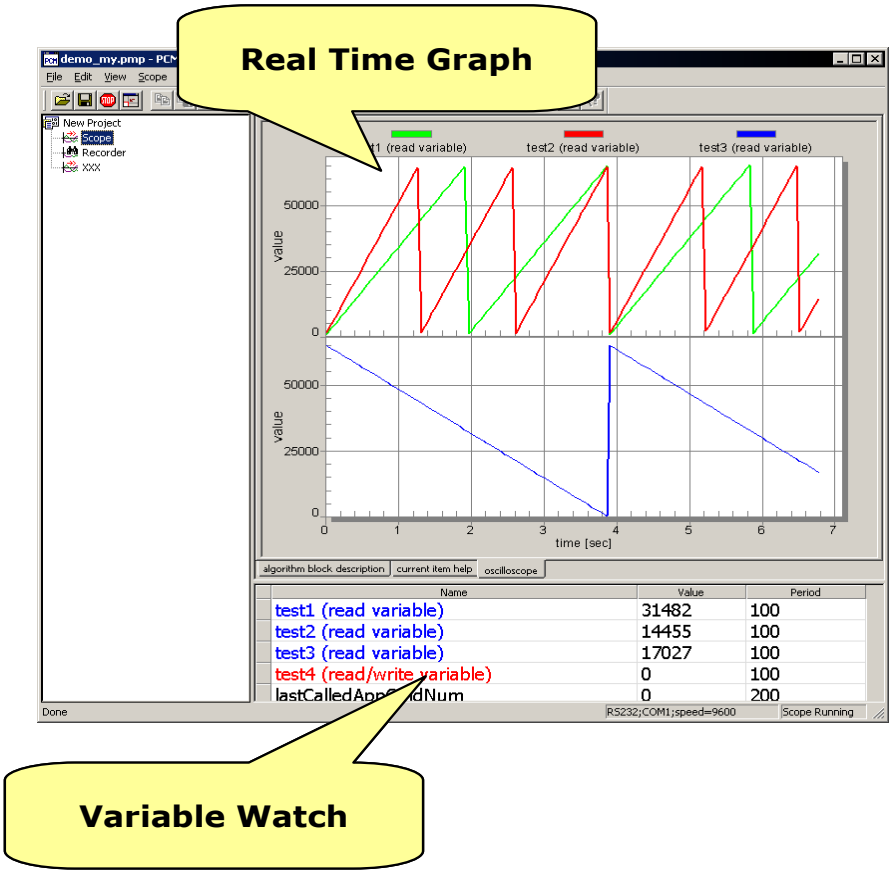
Nástroj FreeMASTER dokáže zobrazovat proměnné flexibilním způsobem v závislosti na nastavení. Možnosti jsou následující:

- Text
  - Zobrazí jméno proměnné
  - Hodnotu v různých formátech (hex, dec nebo bin)
  - Minimální a maximální hodnotu během monitorování
- Průběhy v reálném čase – režim „oscilloscope“
  - Zobrazení až 8 proměnných zároveň v režimu „oscilloscope“
  - Grafická podoba odpovídá grafu s časovou osou
- Vysokorychlostní záznam dat – režim „recorder“
  - Zobrazení až 8 proměnných zároveň v režimu „recorder“
  - Poskytuje přesné vzorkování
  - Data dočasně uložena v paměti procesoru – data buffer
- Transformace proměnných
  - Možnost transformace dat z formátu procesoru do uživatelky přijatelného zobrazení – např.: místo fractional hodnot můžeme pracovat přímo s napětím ve [V], proudem v [A], otáčkami v [rad/s],

s teplotou ve [°C], atd. Transformace funguje obousměrně a to jak z formátu procesoru do formátu skutečné veličiny, tak i naopak, což nám umožňuje manipulovat s veličinami programu v jejich přirozených fyzikálních jednotkách

- Možnost ochrany části paměti proti nechtěnému přepsání
  - Definice proměnných, které jsou pro FreeMASTER viditelné
  - Deklarace proměnných jako read-write nebo jen read-only. Přístup k těmto proměnným je garantovaný driverem na embedded straně
- Aplikační komandy

Ukázka typického zobrazení FreeMASTERu:



Obrázek 5.1 FreeMASTER – typické zobrazení

6. FREESCALE LIBRARY

Freescale poskytuje volně šiřitelné knihovny funkcí. Algoritmy jsou členěné do oddělených knihoven podle typu implementovaných algoritmů. Algoritmy jsou napsány v assembleru a jejich interface vypadá jako standardní C funkce. Dodaná knihovna obsahuje *.lib file, *.h file a dokumentaci v pdf formátu. Algoritmy jsou rozděleny do čtyř knihoven:

- MCLIB – 56800E_MCLIB.lib
- GFLIB - 56800E_GFLIB.lib
- GDFLIB - 56800E_GDFLIB.lib
- ACLIB - 56800E_ACLIB.lib

6.1. Freescale library – MCLIB

Knihovna MCLIB obsahuje algoritmy optimalizované pro řízení elektrických pohonů různých typů od stejnosměrného motoru, přes asynchronní motor, synchronní motor s permanentními magnety až po spínaný reluktanční motor.

Použití funkcí je snadné. Dokumentace obsahuje popis API, popis algoritmu, popis vstupů a výstupů včetně rozsahů a ke každé funkci je uveden příklad

Seznam funkcí v MCLIB knihovně:

- `MCLIB_ClarkTfr` – Clarkova transformace
- `MCLIB_ClarkTfrInv` – inverzní Clarkova transformace
- `MCLIB_ParkTfr` – Parkova transformace
- `MCLIB_ParkTfrInv` – inverzní Parkova transformace
- `MCLIB_SvmStd – SVM` (Space Vector Modulation)
- `MCLIB_SvmU0n` - SVM (Space Vector Modulation)
- `MCLIB_SvmU7n` - SVM (Space Vector Modulation)
- `MCLIB_SvmAlt` - SVM (Space Vector Modulation)

- `MCLIB_SvmSci` - SVM (Space Vector Modulation)
- `MCLIB_PwmIct` – obecná sinusová modulace
- `MCLIB_DecouplingPMSM` – eliminace cross-coupling napětí v rotačním souřadném systému, odstranění nelinearity řízení
- `MCLIB_ElimDcBusRip` – transformace vypočtených napětí v závislosti na aktuální velikosti napětí na mezilehlém obvodu a eliminace zvlnění
- `MCLIB_VectorLimit` – limitace napěťového vektoru

6.2. Freescale library – GFLIB

Tato knihovna je nejrozsáhlejší a obsahuje celou řadu užitečných matematických funkcí optimalizovaných v assembleru. Výčet funkcí je rozsáhlý. Seznam a detailní popis algoritmů můžete najít v dokumentaci ke knihovně.

V praktické části semináře využijeme některé funkce z této knihovny. Podrobněji se zde proto zmíníme o dvou funkcích a to, PI regulátoru a funkce rampa.

6.2.1. *PI regulátor*

Aby bylo možno funkci v uživatelském programu použít, je nutno přidat do projektu samotnou knihovnu "56800E_GFLIB.lib". Dále je zapotřebí do uživatelského kódu vložit správný hlavičkový soubor, což je "gflib.h".

Samotné jméno knihovní funkce je - GFLIB_ControllerPIp.

Hlavička funkce je následující:

Frac16 **GFLIB_ControllerPIp**(Frac16 f16InputErrorK, GFLIB_CONTROLLER_PI_P_PARAMS_T *pudtPiParams, const Int16 *pi16SatFlag)

Argumenty funkce				
Jméno	Typ In/Out	Formát	Rozsah	Popis
f16InputErrorK	In	SF16	0x8000 0x7FFF	Chyba v kroku k zpracovávaná proporcionální a integrační částí regulátoru
pudtPiParams	In/Out	-	-	Ukazatel na strukturu "GFLIB_CONTROLLER_PI_P_PARAMS_T " parametrů PI regulátoru.
pi16SatFlag	In	-	-	Ukazatel na 16-bitovou integer proměnnou. V případě, že proměnná je rovna 0, integrační část regulátoru je limitována na limity regulátoru nastavené ve struktuře parametrů. Pokud je proměnná různá od 0, integrační část je okamžitě zmražena

GFLIB_CONTROLLER_PI_P_PARAMS_T				
Jméno položky struktury	In/Out	Format	Rozsah	Popis
f16PropGain	In	SF16	0x0000... 0x7FFF	Proporcionální zesílení
f16IntegGain	In	SF16	0x0000... 0x7FFF	Integrační zesílení
i16PropGainShift	In	SI16	0...13	Shift proporcionálního zesílení
i16IntegGainShift	In	SI16	0...13	Shift integračního zesílení
f32IntegPartK	In/Out	SF32	0x80000000... 0x7FFFFFFF	Stavová proměnná, stav integrálu v kroku k-1

f16UpperLimit	In	SF16	0x0000... 0x7FFF	Horní limit regulátoru
f16LowerLimit	In	SF16	0x0000... 0x7FFF	Dolní limit regulátoru
i16LimitFlag	Out	SI16	0 nebo 1	Limitační flag, pokud je flag = 1, výstup regulátoru narazil na horní nebo dolní limit regulátoru, jinak je flag = 0

Popis funkce:

Funkce počítá algoritmus PI regulátoru podle rovnice viz níže. Implementace algoritmu umožňuje nastavovat P a I parametry nezávisle. Výstup regulátoru je limitován parametry "f16UpperLimit" a " f16LowerLimit". Algoritmus vrací flag (i16LimitFlag). Pokud algoritmus PI regulátoru dosáhne hodnoty jednoho z limitů, algoritmus vrací i16LimitFlag = 1, jinak i16LimitFlag = 0.

Anti-windup algoritmus je implementován limitací integrační části regulátoru dvěma způsoby:

- Stav integrační části je limitován na limity "f16UpperLimit" a "f16LowerLimit"
- Pokud je proměnná "i16SatFlag" nastavena uživatelským algoritmem na hodnotu různou od nuly, integrační složka algoritmu je zmražena.

PI algoritmus ve spojité oblasti je definován následovně:

$$u(t) = K\left[e(t) + \frac{1}{T_I}\int_0^{\tau} e(t)dt\right]$$

Kde:

e(t) - chyba regulátoru

u(t) - výstup regulátoru

T_I - integrační časová konstanta - [s]

Diskrétní verze regulátoru:

$$u(k) = K \cdot e(k) + u_I(k-1) + K_I \cdot e(k)$$

$$u_I(k) = u_I(k-1) + T \cdot e(k)$$

Kde:

e(k) - chyba regulátoru v kroku k

u(k) - výstup regulátoru v kroku k

K - proporcionální zesílení

K_I - integrační zesílení v diskrétní oblasti

T - vzorkovací perioda - [s]

$$K_I = K \cdot \frac{T}{T_I}$$

Úprava diskrétní formy regulátoru to rozsahu 16-bitů s využitím výhod fractional aritmetiky.

$$u_f(k) = K_{sc} \cdot e_f(k) + u_{If}(k-1) + K_{Isc} \cdot e_f(k)$$

Kde

$$u_f(k) = u(k)/u_{max}$$

$$e_f(k) = e\,(k)/e_{max}$$

$$K_{sc} = K \cdot \frac{e_{max}}{u_{max}}$$

$$K_{Isc} = K \cdot \frac{T}{T_I} \cdot \frac{e_{max}}{u_{max}} = K_I \cdot \frac{e_{max}}{u_{max}}$$

Kde

e_{max} - maximální rozsah vstupní fyzikální veličiny

u_{max} - maximální rozsah výstupní fyzikální veličiny

$$f16PropGain = K_{sc} \cdot 2^{-i16PropGainShift}$$

$$f16IntegGain = K_{Isc} \cdot 2^{-i16IntegGainShift}$$

$$0 \leq f16PropGain < 1$$

$$0 \leq f16IntegGain < 1$$

$$0 \leq i16PropGainShift < 14$$

$$0 \leq i16IntegGainShift < 14$$

Příklad použití:

K_{Isc} = 2.4

Číslo 2.4 nelze přímo interpretovat jako fractional hodnotu, protože fractional rozsah je <-1; 1). Bude zapotřebí použít "shift" parametr a pomocí něho dostat f16IntegGain parametr do rozsahu <0; 1)

Řešení:

Nejpřesnější výsledek docílíme, pokud se podaří parametr "f16IntegGain" dostat do rozsahu <0.5; 1).

$$\frac{\log(K_{Isc}) - \log(0.5)}{log2} \geq i16IntegGainShift$$

$$\frac{\log(2.4) - \log(0.5)}{log2} \geq i16IntegGainShift$$

$$2.26 \geq i16IntegGainShift$$

$$\frac{\log(K_{Isc}) - \log(1)}{log2} < i16IntegGainShift$$

$$\frac{\log(K_{Isc})}{log2} < i16IntegGainShift$$

$$\frac{\log(2.4)}{log2} < i16IntegGainShift$$

$$1.26 < i16IntegGainShift$$

A tedy platí:

$$1.26 < i16IntegGainShift < 2.26$$

Poněvadž parametr "i16IntegGainShift" je integer hodnota, je tedy

$$i16IntegGainShift = 2$$

Pak vypočteme hodnotu parametru "f16IntegGain" následujícím způsobem:

$$f16IntegGain = K_{Isc} \cdot 2^{-i16IntegGainShift}$$

$$f16IntegGain = 2.4 \cdot 2^{(-2)} = 0.6$$

Parametry regulátoru jsou:

$$f16IntegGain = 0.6$$

$$i16IntegGainShift = 2$$

Funkce PI regulátoru vrací 16-bitovou fractional hodnotu v rozsahu

$$f16LowerLimit \leq i16IntegGainShift \leq f16UpperLimit$$

Příklad použití v uživatelském kódu:

```
#include "gflib.h"
static Frac16        mf16DesiredValue;
static Frac16        mf16MeasuredValue;
static Frac16        mf16ErrorK;
static Int16         mi16SatFlag;
static Frac16        mf16ControllerOutput;
/* Controller parameters */
static GFLIB_CONTROLLER_PI_P_PARAMS_T mudtControllerParam;
void Isr(void);
```

```
void main(void)
{
/* Controller parameters initialization */
mudtControllerParam.f16PropGain    =       FRAC16(0.5);
mudtControllerParam.f16IntegGain   =       FRAC16(0.032);
mudtControllerParam.i16PropGainShift =    1;
mudtControllerParam.i16IntegGainShift =   0;
mudtControllerParam.f32IntegPartK_1 =     0;
mudtControllerParam.f16UpperLimit =       FRAC16(0.8);
mudtControllerParam.f16LowerLimit =       FRAC16(-0.7);

/* Desired value initialization */
mf16DesiredValue          =       FRAC16(0.5);
/* Measured value initialization */
mf16MeasuredValue      =       0;
/* Saturation flag initialization */
mi16SatFlag                =        0;
}

/* Periodical function or interrupt */
void Isr(void)
{
/* Error calculation */
```

```
mf16ErrorK = mf16DesiredValue - mf16MeasuredValue;

/* Controller calculation */

mf16ControllerOutput = GFLIB_ControllerPIp(mf16ErrorK,

&mudtControllerParam, &mi16SatFlag);

}
```

6.2.2. *Rampa*

GFLIB knihovna obsahuje čtyři implementace funkce rampa.

- GFLIB_Ramp16
- GFLIB_Ramp32
- GFLIB_DynRamp16
- GFLIB_DynRamp32

Podrobněji si vysvětlíme implementaci "GFLIB_Ramp32"

Aby bylo možno funkci v uživatelském programu použít, je nutno přidat do projektu samotnou knihovnu "56800E_GFLIB.lib". Dále je zapotřebí do uživatelského kódu vložit správný hlavičkový soubor, což je "gflib.h".

Samotné jméno knihovní funkce je:

GFLIB_Ramp32

Hlavička funkce je následující:

Frac32 **GFLIB_Ramp32**(Frac32 f32Desired, Frac32 f32Actual, const

GFLIB_RAMP32_T *pudtParam)

Argumenty funkce:

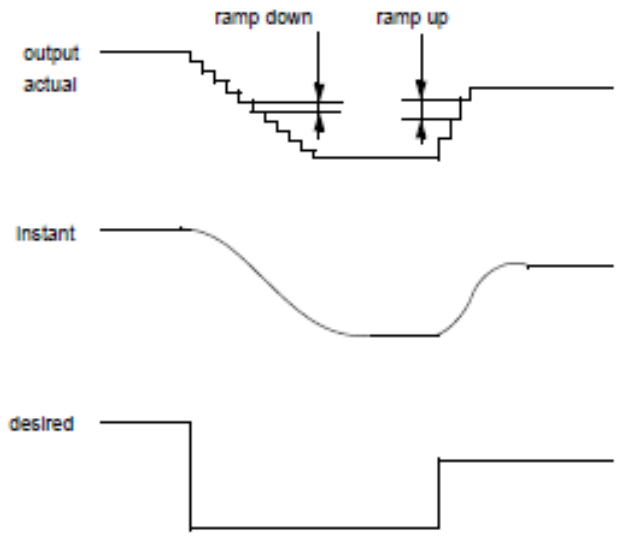
Argumenty funkce				
Jméno	Typ In/Out	Formát	Rozsah	Popis
f32Desired	In	SF32	0x80000000... 0x7FFFFFFF	Žádaná hodnota, na kterou se rampa dostane za definovaný časový úsekl
f32Actual	In	SF32	0x80000000... 0x7FFFFFFF	Aktuální hodnota výstupu rampy
pudtParam	In	-	-	Ukazatel na strukturu parametrů rampy

GFLIB_RAMP32_T				
Jméno položky struktury	In/Out	Format	Rozsah	Popis
f32RampUp	In	SF32	0x80000000... 0x7FFFFFFF	Inkrement rampy směrem nahoru
f32RampDown	In	SF32	0x80000000... 0x7FFFFFFF	Inkrement rampy směrem dolů

Popis funkce:

Funkce počítá algoritmus rampy ve 32-bitovém rozlišení. Dochází k přičítání nebo odečítání parametrů rampy "f32RampUp" a "f32RampDown" k aktuální hodnotě rampy.

Pokud je hodnota proměnné f32Desired > f32Actual, dochází k přičítání parametru f32RampUp. Rampa dosáhne maximálně hodnoty f32Desired. V případě f32Desired < f32Actual, dochází k odečítání parametru f32RampDown.



Obrázek 6.1 Diagram popisující funkci algoritmu rampa

Příklad použití v uživatelském kódu:

```
#include "gflib.h"

static Frac32           mf32DesiredValue;

static Frac32           mf32ActualValue;

/* Ramp parameters */

static GFLIB_RAMP32_T  mudtRamp32;


void Isr(void);


void main(void)

{

/* Ramp parameters initialization */

mudtRamp32.f32RampUp       = FRAC32(0.25);

mudtRamp32.f32RampDown   = FRAC32(0.25);
```

```
/* Desired value initialization */
mf32DesiredValue                = FRAC32(1.0);
/* Actual value initialization */
mf32ActualValue                 = 0;
}
/* Periodical function or interrupt */
void Isr(void)
{
/* Ramp generation */
mf32ActualValue = GFLIB_Ramp32(mf32DesiredValue,
mf32ActualValue, &mudtRamp32);
}
```

6.3. Freescale library – GDFLIB

GDFLIB obsahuje několik algoritmů realizovaných digitálních filtrů. Kód je napsaný v asembleru. Knihovna obsahuje následující algoritmy:

- `GDFLIB_FilterIIR1` – filtr IIR prvního řádu
- `GDFLIB_FilterIIR1Init` – inicializační funkce filtru IIR prvního řádu
- `GDFLIB_FilterIIR2` - filtr IIR druhého řádu
- `GDFLIB_FilterIIR2Init` - inicializační funkce filtru IIR druhého řádu

- `GDFLIB_FilterMA32` – rekurzivní forma filtru váženého průměru
- `GDFLIB_FilterMA32Init` - inicializační funkce rekurzivní formy filtru váženého průměru

6.4. Freescale library – ACLIB

Tato knihovna je zaměřená na pokročilé techniky řízení synchronních motorů s permanentními magnety bez použití sensoru polohy tzv. „sensorless control“. Použití těchto funkcí vyžaduje poměrně rozsáhlé znalosti v oblasti bezsnímačového řízení pohonů. Knihovna obsahuje:

- `ACLIB_PMSMBemfObsrvAB` – pozorovatel odhadující indukované napětí ve stojících α, β souřadnicích
- `ACLIB_PMSMBemfObsrvDQ` – pozorovatel odhadující indukované napětí v rotujících souřadnicích d,q
- `ACLIB_AngleTrackingObsrv` – estimátor uhlové rychlosti a polohy vstupního sinusového signálu
- `ACLIB_TrackingObsrv` - estimátor uhlové rychlosti a polohy vstupního signálu
- `ACLIB_Itegrator` - numerická integrace

SEZNAM POUŽITÉ LITERATURY

[1] MC56F8006RM.pdf - 56F8006 Peripheral Reference Manual

[2] MC56F8006.pdf - 56F8006/56F8002 Data Sheet

[3] DSP56800ERM.pdf - DSP56800E Reference Manual

[4] DSP56800E_Quick_Start_Users_Manual.pdf – DSP56800E_Quick_Start User's Manual, Targeting Freescale 56F8xxx Platform

[5] 56800E_aclib.pdf – Advanced Control Library for 56800E, User Reference Manual

[6] 56800E_mclib.pdf – Motor Control Library, User Reference Manual

[7] 56800E_gflib.pdf – General Functions Library, User Reference Manual

[8] 56800E_gdflib.pdf – General Digital Filters Library for 56800E, User Reference Manual

[9] Zdeněk Caha, Miroslav Černý: Elektrické pohony. Praha, SNTL 1990

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

Ústav automatizace a měřicí techniky
VUT v Brně
Kolejní 2906/4
612 00 Brno
Česká Republika

http://www.crr.vutbr.cz

info@crr.vutbr.cz