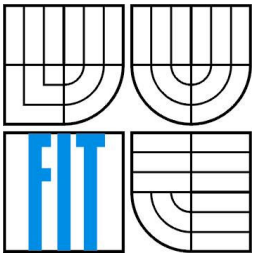
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OVLÁDÁNÍ ROBOTICKÉHO MANIPULÁTORA MIKROKONTROLÉREM

ROBOTIC MANIPULATOR CONTROLLING USING MICROCONTROLLER

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

KRISTIÁN LENGYEL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RICHARD RŮŽIČKA, Ph.D.

BRNO 2007

Zadanie

Ovládání robotického manipulátora mikrokontrolérem

1. Seznamte se s robotickým manipulátorem ROB1-3 a jeho ovládáním servomotory.
2. Seznamte se s mikrokontroléry HC08, které jsou ve školní laboratoři IMP.
3. Připojte robotický manipulátor a běžný joystick k mikrokontroléru.
4. Navrhněte program pro mikrokontrolér, který bude ovládat robotický manipulátor podle pokynů operátora joystickem.
5. Program odlaďte a demonstrujte funkčnost.

Kategorie: Vestavěné systémy

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Práca sa zaoberá návrhom a implementáciou algoritmu na ovládanie servomechanizmu robotického manipulátora. V prvej fázi bolo potrebné pripojiť robotický manipulátor ROB 1-3 a bežný joystick k mikrokontroléru rady HC08 od firmy Motorola používaný v školských laboratóriach. V druhej fázi som navrhol a implementoval samotný program pre mikrokontrolér MC68HC908LJ12 s využitím jazyka C a vývojového prostredia Freescale CodeWarrior. Úlohou programu bolo previesť signály prijímané z operátora joysticku na pokyny pre servomotory.

Kľúčová slová

Robotický manipulátor ROB 1-3, servomechanizmus, PWM signál, joystick, mikrokontrolér MC68HC908LJ12, jazyk C, Freescale CodeWarrior.

Abstract

This thesis deals with design and implementation of an algorithm for operating the servomechanism of robotic manipulator. In first phase it was necessary to attach the robotic manipulator ROB 1-3 and common joystick to HC08 microcontroller from Motorola Corp., which is also used in our school laboratories. In second phase I have designed and implemented the program for MC68HC908LJ12 microcontroller using C programming language and Freescale CodeWarrior development studio. The task of this program was to convert signals received from joystick to commands for servo motors.

Keywords

Robotic manipulator ROB 1-3, servomechanism, PWM signal, joystick, MC68HC908LJ12 microcontroller, C language, Freescale CodeWarrior.

Citácie

Kristián Lengyel: Ovládanie robotického manipulátora mikrokontrolérom, bakalárska práca, Brno, FIT VUT v Brně, 2007

Ovládanie robotického manipulátora mikrokontrolérom

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Richarda Růžičku, Ph.D.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Kristián Lengyel
22. apríla 2007

Poďakovanie

Touto cestou by som sa chcel poďakovat svojmu konzultantovi Ing. Richardovi Růžičkovi, Ph.D. za užitočné rady a odbornú pomoc pri tvorbe mojej bakalárskej práce.

Obsah

Úvod

Človek sa už od dávnych čias snaží dosiahnuť čo najväčší komfort vo svojom živote. Toto stáročia trvajúce úsilie vyústilo práve k vytvoreniu jedného z najkomplexnejších a najzložitejších systémov, bez ktorých si už v dnešnej dobe vie len ťažko človek predstaviť život. Týmito systémami sú počítače a práve ony prešli najrýchlejším vývojom za posledných 10-20 rokov. Sú obľúbené práve z toho dôvodu, že zvládajú široký sortiment úloh náročných na výpočetný výkon a dávajú tiež možnosť pripojenia rôznych periférií ako sú monitor, klávesnica či tlačiareň. Užívateľovi tým poskytujú pohodlný spôsob ovládania a možnosť vidieť výstup užitočných údajov. Nie vždy sú však počítače vhodnou variantou pre riešenie zložitých systémov a prečo?

Zoberme si za príklad ďalšie velmi rýchlo sa rozvýjajúce systémy ako napr. digitálna kamera. Tento systém a mnoho iných podobných potrebuje tiež k svojej činnosti výpočetnú jednotku, tá však nemusí v sebe zahŕňať veľké množstvo operácií, ale stačí, ak sa špecializuje na určitú úlohu ako napr. ovládanie funkcií. Táto výpočetná jednotka je súčasťou väčšieho komplexnejšieho systému a tak sa dostávame práve k pojmu vstavané systémy alebo po angl. *embedded systems*. Tieto systémy dokážu bez použitia počítača vykonávať špecializované úlohy a reagovať na rôzne vonkajšie udalosti. Údaje, ktoré pritom spracovávajú, nie sú priamo viditeľné pre užívateľa.

Využitie vstavaných systémov si vyžaduje špeciálnu jednotku, ktorú nazývame mikrokontrolér. Mikrokontroléry sa od klasických mikroprocesorov odlišujú tým, že ich čip je okrem základnej procesorovej jednotky CPU vybavení aj rôznymi druhmi periférií. Poznáme rôzne druhy mikrokontrolérov, napr. 8, 16 a 32 bitové a rôznych výrobcov ako Atmel, Microchip alebo v našom prípade používaný mikrokontrolér od firmy Motorola. Človek venuje stále viac pozornosti rozvoju vstavaných systémov, či už vo sfére vedy, techniky alebo aj zábavy. Veď ako príklad môžme uviesť aj mobilné telefóny, kopírky, faxy, digitálne fotoaparáty, číslicové meradlá a tiež nachádzajú svoje miesto aj v oblasti robotiky a práve tou sa zaoberám ja v mojej bakalárskej práci.

V dnešnej dobe sa svet čoraz viac zaujíma o problematiku robotiky a to nielen v oblasti priemyslu, ale tiež aj výskumu a zábavy. Spomeňme si napr. na lunárne moduly a rôzne iné robotické zariadenia využivané vo vesmírnom výskume alebo z oblasti zábavy si určite ešte mnohí spomíname na robotického psa AIBA od firmy Sony.

Rovnako treba spomenúť inteligentných robotov, ktoré nachádzajú svoje uplatnenie najmä v oblasti priemyselnej robotiky vďaka tomu, že sú schopné samostatne vykonávať nejakú činnosť bez väčšieho zásahu človeka. Automatizovaná činnosť týchto robotov je zabezpečená rôznymi hmatovými čidlami či kamerami.

V mojej bakalárskej práci sa pokusím priblížiť práve oblasť robotiky na jednoduchom robotickom manipulátorovi bez umelej inteligencie ovládateľnom povelmi operátora joysticku. Prvá kapitola sa venuje charakteristike robota, jeho konštrukcií a technickým parametrom. Podrobne sa

budem zaoberať najmä použitým servomechanizmom, spôsobom ovládania servomechanizmu robota a pulzne-šírkovo modulovaným signálom.

Druhá kapitola bude opisovať použitý mikrokontrolér, ktorým bol mikrokontrolér MC68HC908LJ12 od firmy Motorola používaný v školských laboratóriách. V tejto stati sa zameriam najmä na použité periférie, konkrétne na časovací subsystém a analógovo-digitálny prevodník.

V tretej kapitole sa oboznámime so samotným návrhom, implementáciou programu a spôsobom ovládania robotického manipulátora. Pár slovami spomeniem joystick, ktorý som používal pri testovaní programu. Rovnako v nej zhrniem najdôležitejšie rutiny programu.

V závere popíšem dosiahnuté výsledky, znalosti získané počas tvorby tejto práce a možnosti ďalšieho vývoja.

1 Robotický manipulátor ROB 1-3

V súčasnosti sa objavuje nový fenomén, a to domáca robotika. Na internete máme možnosť nájsť rôzne druhy robotov od kráčajúcich, pojazdných až po manipulátory, akým je práve aj náš robotický manipulátor ROB 1-3. Všetky tieto roboty majú jednu spoločnú vlastnosť, a to tú, že ich pohyb je ovládaný modelárskymi servomotormi. ROB 1-3 je jednoduchým predstaviteľom trojosého robotického manipulátora. Viac sa o jeho konštrukcií, servomechanizme a technických parametroch dozvieme v ďalších sekciách.

1.1 Základné informácie

Robotický manipulátor ROB 1-3 je ovládaný modelárskymi servomechanizmami Hitec HS-311, ktoré patria v súčasnosti medzi najlacnejšie na trhu. Pohyb robota vo všetkých troch osiach je riadený programom s mikrokontroléra MC68HC908LJ12, ktorý je programovateľný cez vývojové prostredie Freescale CodeWarrior.

Robot sa pohybuje v troch osiach – otáčanie základne, zdvih ramena a otváranie resp. zatváranie klieští. Pohyb servomotorov je riadený zo spomínaného elektronického modulu MC68HC908LJ12, ktorý je programovateľný pomocou prípravku JANUS. Programátor komunikuje s počítačom cez sériovú linku RS 232 (na strane počítača COM1).

Konštrukcia robota bola podriadená požiadavku na použitie bežne dostupných dielov, pre výrobu nie je potreba žiadneho strojového obrábania s výnimkou vŕtania a rezania. Všetky komplikovanejšie diely sú dodané hotové, kúpené v predajniach s elektronickými súčiastkami a v modelárskych predajniach.

Prototyp robota bol vyrezaný laserom z doskového organického skla hrúbky 3 mm, nič však nebráni aj ručnej výrobe. Diely sú tvarovo velmi jednoduché a nepotrebujú zvláštnu starostlivosť pri obrábaní. Ako stavebný materiál je možné použiť aj sklený laminát alebo kuprextit.

1.2 Konštrukčné riešenie

Upínacie kliešte sú riešené ako paralelogram, skladajúci sa zo šiestich zhodných ramien a dvoch ramien dvojzvratných. Ťahom serva za tieto dvojzvratné ramená sa kliešte otvárajú alebo zatvárajú. Otočné body ramien klieští sú vytvorené prevlečením skrutiek M3, u zadných ramien sú na skrutky ešte navlečené izolačné podložky IB2 (GM), ktoré tu slúžia ako distančné podložky a umožňujú ľahké zasunutie tiahla od ovládacieho serva. Tiahlo od serva je vyrobené z upravenej kancelárskej

sponky (vidlice) a niekoľko centimetrového oceľového pocínovaného drôtu (v núdzi narovnaná kancelárska sponka). Celé tiahlo je pájkované bežnou cínovou pájkou a pre dosiahnutie efektívnejšieho vzhľadu je potiahnuté teplom zmrštitelnou bužírkou o priemere 1,6 mm. Na dotykových plochách klieští je sekundovým lepidlom nalepené 15 mm pryžového tesnenia do okien.

Servo pohybu klieští je vsadené do vyrezaného otvoru a upevnené štyrmi skrutkami M3. Servo pohybu ramena je upnuté medzi dosku ramena a prítlačnú dosku. Správnu vzdialonosť oboch dosiek zaisťujú 4 poliamydové distančné stĺpiky KDI6M3x20. Poloha serva je zaistená jeho zasunutím medzi tieto stĺpiky. Jeden otočný bod ramena je vytvorený priamo osou serva, nosný tanier serva je upevnený skrutkami M1,6 na bočnicu základne. Za servo je vložená doštička so zalisovaným distančným stĺpikom KDR 12, ktorý tvorí druhý otočný bod ramena. Doštička s čapom a zadná strana serva je spojená kúskom obojstranne lepiacej pásky. Tým je zaistená stabilná poloha serva a pomocnej doštičky. Ak by sa kvôli výrobným toleranciám predsalen servo posúvalo, dá sa ho upevniť čo najtenšou obojstranne lepiacou páskou, najlepšie Scotch, ktorú bežne nájdete v papiernictve.

Bočnice základne sú spojené tromi distančnými stĺpikmi KDI6M3x50. K otočnej doske sú bočnice upevnené plastovými konzolami MPJ 2621 a skrutkami M3. Otočná doska kĺže po hlavách šiestich plastových záslepiek F715HP-08, ktoré sú zatlačené do otvoru o priemere 8 mm v základnej doske.

Pod základnou doskou je upevnený štyrmi skrutkami M3 na pomocnej doske servomotor, ktorý otáča rameno. Pomocná doska je uchytená štyrmi skrutkami M3 a vo správnej vzdialenosti od základnej dosky držaná distančnými stĺpikmi KDR 07.

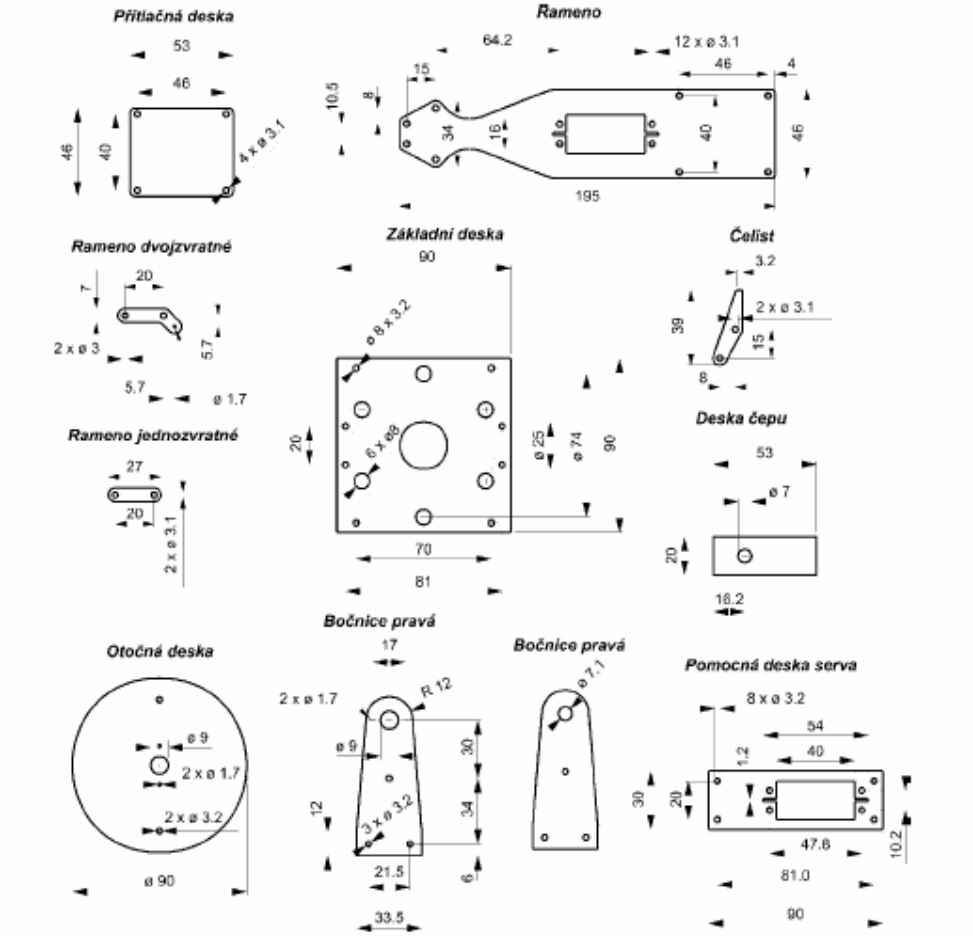
Pri zostavovaní základne robota musíme najprv upevniť dvomi skrutkami M1,6 (MPJET 0205) nosný tanier z príslušenstva serva na otočnú dosku, potom treba zostaviť pomocnú a základnú dosku spoločne so servom a nosný tanier serva s pripevnenou otočnou doskou nasunieme na hriadeľ serva. Otočná doska musí ľahko dosadnúť na hlavy plastových záslepiek, nesmie sa však príliš silno pritlačiť, aby pri otáčaní ramena robota nebol motorček serva nadmerne namáhaný. Z bezpečnostných dôvodov je preto vhodné natrieť hlavy plastových záslepiek vrstvou vazelíny, aby sa otočná doska ľahšie kĺzala po nich. V tejto polohe celú zostavu zaistíme ľahkým dotiahnutím skrutiek, ktoré prechádzajú do osi serva.

Základňa je postavená na štyroch gumových podložkách kruhového tvaru o priemere 30 mm.

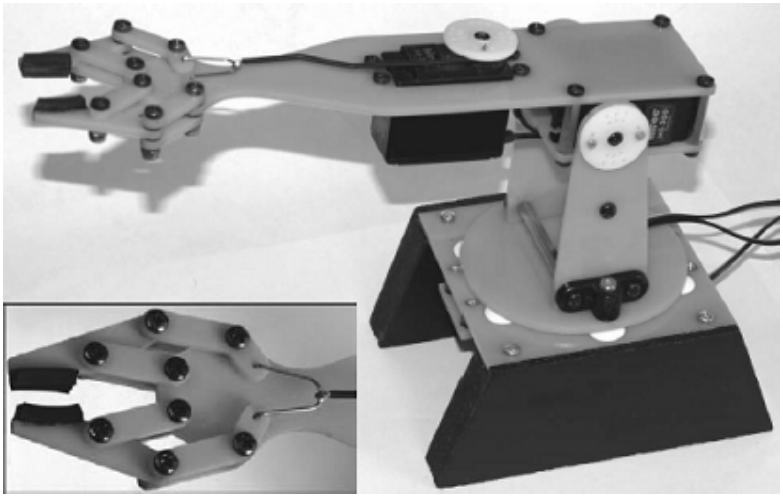
Zoznam súčiastok:

- Distančný stĺpik KDR 07, 4 ks, GM
- Distančný stĺpik KDR 12, 1 ks, GM
- Distančný stĺpik KDI6M3x20, 4 ks, GM
- Distančný stĺpik KDI6M3x50, 3 ks, GM
- Izolačná podložka IB2, 4 ks, GM

- Záslepka Ø 8 mm F1715HP-08, 6 ks, GM
- Upevňovacia konzola Ø 3 mm MPJ 2621, 2 ks, MP Jet a modelárske predajne
- Skrutka M1,6 x 8 aj s maticou, MPJ 0202, 4 ks, MP Jet a modelárske predajne
- Skrutka M3 x 16 so zápustnou hlavou, 6 ks, Fabory a železiarske predajne
- Skrutka M3 x 16 s polguľatou hlavou, 4 ks, Fabory a železiarske predajne
- Skrutka M3 x 12 s polguľatou hlavou, 8 ks, Fabory a železiarske predajne
- Skrutka M3 x 8 s polguľatou hlavou, 18 ks, Fabory a železiarske predajne
- Matica M3, 14 ks, Fabory a železiarske predajne
- Podložky Ø 3,2, 14 ks, Fabory a železiarske predajne



Obr. 1.1: Jednotlivé diely robotického manipulátora



Obr. 1.2: Robotický manipulátor ROB 1-3

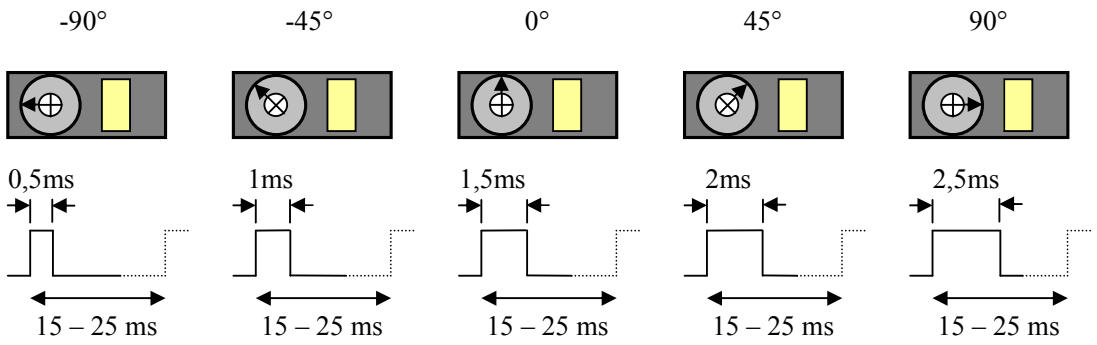
1.3 Ovládací servomechanizmus

Modelárske servo je miniatúrny elektromotor s prevodovkou. Poloha otočného výstupného hriadeľa prevodovky je pritom snímaná spätnoväzobným členom (väčšinou potenciometrom) a zavedená späť do riadiacej jednotky vstavanej v serve. Táto elektronika má na starosti ovládanie rýchlosti a smeru otáčania pohonného motorku. Požadovaná poloha výstupného hriadeľa je do riadiacej elektroniky zadávaná ako pulzne - šírkovo modulovaný signál alebo tiež nazývaný PWM (pulse-width modulated) signál. Tento signál sa musí privádzať s opakovacou frekvenciou 50 Hz pre správnu činnosť servomotorov. Nastavovaním šírky tohoto impulzu v rozsahu od 1 do 2 ms je definovaná poloha výstupného hriadeľa v rozsahu 90°. Pre nastavenie strednej polohy serva je potrebné vysielať kladný impulz o dĺžke 1,5 ms. Dôležitým faktom je, že tento PWM signál musí byť do serva vysielaný nepretržite, v opačnom prípade sa uvoľní spätná väzba serva a nie je zaručená správna poloha výstupného hriadeľa.

Servá v základnom prevedení umožňujú otočenie výstupného hriadeľa o 90°. Niektoré servá, ako aj naše, však umožňujú zväčšenie rozsahu otáčania výstupného hriadeľa až na 180°. Dá sa to docieliť vysielaním kladného riadiaceho PWM impulzu vo väčšom rozsahu od 0,5 do 2,5 ms. Riadenie týmto spôsobom je však možné len u niektorých druhov servomotorov resp. u servomotorov niektorých výrobcov a je treba ho odskúšať najprv na vybranom kuse servomotora, aby sme predišli poškodeniu prevodovky opakovanými nájazdmi na koncové dorazy.

Servá je tiež možné upraviť na trvalé otáčanie, PWM signálom sa potom neriadi poloha hriadeľa servomotora, ale rýchlosť a smer jeho otáčania. Takto upravené servomotory sa používajú v mobilných robotoch k pohonu kolies. Bližšie podrobnosti o úprave je možné nájsť na stránke http://www.rotta.cz .

Servá je možné zakúpiť v modelárskych predajniach v mnoho rôznych prevedeniach a veľkostiach. Pri použití serv s väčším prúdovým odberom však treba vhodne dimenzovať napájací zdroj, ktorými sú napájané servomotory.



Obr. 1.3: Nastavovanie polohy hriadeľa servomotora

1.3.1 Modelársky servomotor HS-311

Servomotor HS-311 je malé zariadenie, ktoré ako väčšina iných servomechanizmov obsahuje šesť základných častí, ktorými sú:

- hnacie kolieska
- motor
- potenciometer
- riadiaci obvod
- puzdro serva
- výstupný hriadeľ

Používa sa pritom motor na jednosmerné napätie, z ktorého vedú 3 vodiče – čierna predstavuje zem (GND), červená napájanie (VCC) a žltá spĺňa úlohu signálového vodiča (PWM). Hriadeľ serva je možné nastaviť na požadovanú pozíciu posielaním kódovaných signálov na signálový vodič. Do doby, kým na tento vodič posielame signály, bude servo udržiavať aktuálnu pozíciu hriadeľa. Ak sa kódovaný signál zmení, zmení sa aj pozícia hriadeľa.

Výhodou týchto servomotorov je, že napriek svojim rozmerom sú extrémne výkonné a privádzajú výkon rovnomerne do mechanickej záťaže. Z toho dôvodu je zrejmé, že málo zaťažený servomotor nespotrebuje veľa energie. Servá sú klasifikované z dvoch hľadísk, a to **rýchlosť**

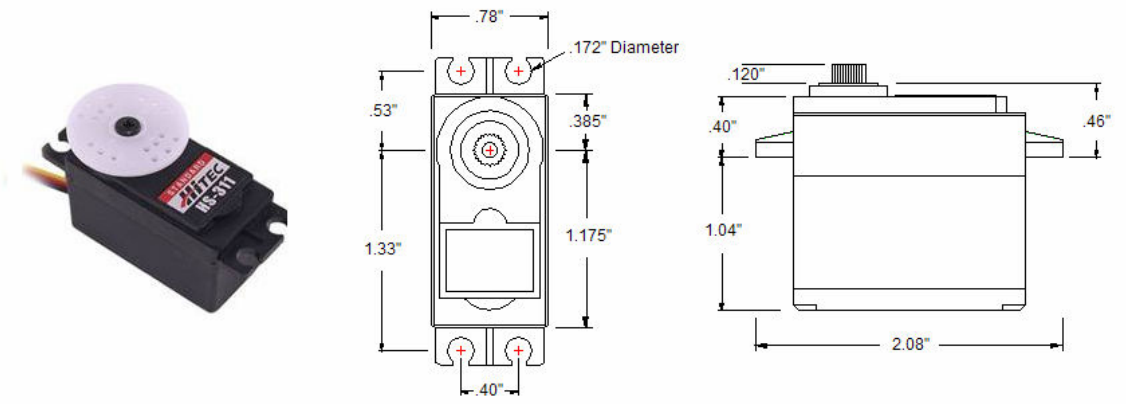
a **krútiaci moment**. Servomotor môže byť prispôsobený len na preferovanie jedného z nich, teda buď vyššia rýchlosť a menší krútiaci moment alebo naopak.

Najdôležitejšie časti servomotora sú motor, potenciometer a riadiaci obvod. Potenciometer umožňuje riadiacemu obvodu sledovať aktuálny uhol natočenia servomotora. Motor otáča výstupný hriadeľ a potenciometer súčasne cez sériu ozubených kolies. Potenciometer je napojený na riadiaci obvod serva a keď riadiaci obvod zistí, že je pozícia správna, zastaví motor. Ak riadiaci obvod zistí, že uhol nie je správny, otočí motor v správnom smere, kým nenájde požadovaný uhol natočenia.

Spravidla je motor prispôsobený na otáčanie v rozsahu od 0° do 90°, príp. od 0° do 180°. Je mechanicky nemožné otočiť ho viacej kvôli mechanickej zarážke zabudovanej na hlavnom hnacom kolese. Výkon potrebný na otočenie motora serva je rovnomerný so vzdialenosťou , ktorú musí motor prejsť. Teda ak sa hriadeľ musí otočiť o veľkú vzdialenosť, motor na to využije plnú rýchlosť. Ak sa potrebuje pohnúť len o malú vzdialenosť, tak motor pobeží na menšej rýchlosti.

Detailná špecifikácia HS-311:

- Riadiaci systém: Pulzne-šírkovo modulovaný signál – 1500µsek neutrál
- Požadovaný pulz: 3-5 V, obdĺžnikový signál
- Prevádzkové napätie: 4,8 – 6,0 V
- Rozsah teplôt: -20 až +60 °C
- Prevádzková rýchlosť (4,8 V): 0,19 sek/60° pri žiadnej záťaži
- Prevádzková rýchlosť (6,0 V): 0,15 sek/60° pri žiadnej záťaži
- Krútiaci moment (4,8 V): 42 oz/in (3,0 kg/cm)
- Krútiaci moment (6,0 V): 49 oz/in (4,5 kg/cm)
- Spotreba elektrického prúdu (4,8 V): 7,4 mA/v nečinnosti, 160mA bez záťaže
- Spotreba elektrického prúdu (6,0 V): 7,7 mA/v nečinnosti, 180mA bez záťaže
- Šírka pásma necitlivosti: 5µsek
- Prevádzkový uhol: 40° jednosmerný prenos impulzov 400µsek
- Smer: V smere hodinových ručičiek/Prenos impulzov od 1500 do 1900 µsek
- Prevodovka potenciometra: 4 jazdcový
- Typ súkolia: Nylon
- Modifikovateľný na 360°: Áno
- Dĺžka spojovacieho kábla: 11,81” (300 mm)
- Váha: 1,52 oz (43 g)



Obr. 1.4: Modelárske servo HS-311 a jeho technické parametre

V predošlom texte bolo často spomínané, že k ovládaniu servomotorov je potrebný tzv. pulzne-šírkovo modulovaný signál. V ďalšej časti textu sa dozvieme, čo to vlastne ten PWM signál je a ako sa generuje.

1.4 Pulzne šírkovo modulovaný signál

Pulzne šírková modulácia signálu je technika generovania impulzného priebehu s konštantnou periódou a premennou dobou pracovného cyklu. Tento druh signálu sa používa predovšetkým na prenos informácii cez komunikačné kanály alebo na kontrolu množstva výkonu posielaného do záťaže.

1.4.1 Matematický princíp

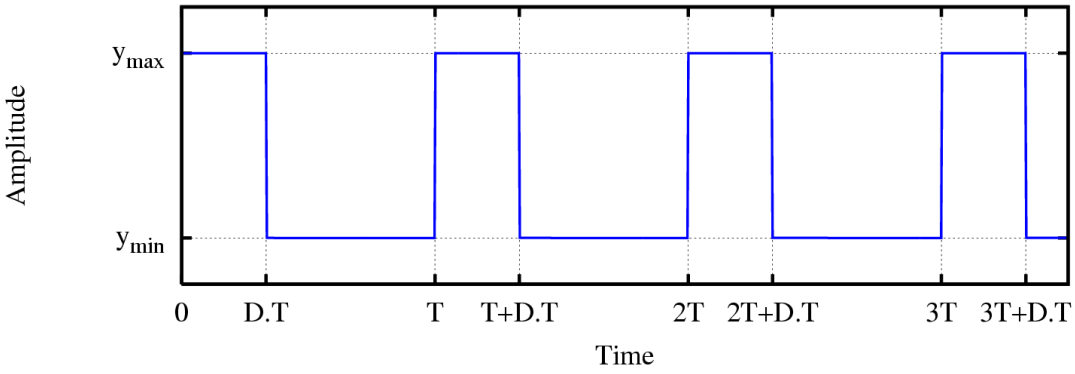
Pulzne šírková modulácia používa obdĺžnikový signál, ktorého doba pracovného cyklu je modulovaná na základe zmeny priemernej hodnoty priebehu. Ak vezmeme do úvahy obdĺžnikový priebeh $f(t)$ s najnižšou hodnotou $y_{\min}$, s najvyššou hodnotou $y_{\max}$ a s dobou pracovného cyklu D (viď. Obr. 1.5), tak priemerná hodnota priebehu je daná ako:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt$$

Keďže $f(t)$ má obdĺžnikový priebeh, tak hodnota $y_{\max}$ dosahuje svoje hodnoty pre interval $0 < t < D.T$ a $y_{\min}$ pre interval $D.T < t < T$. Po dosadení do horného vzorca tak dostávame:

$$\bar{y} = \frac{1}{T}\left(\int_0^{DT} y_{max}\,dt + \int_{DT}^{T} y_{min}\,dt\right) = \frac{D.T.y_{max} + T(1-D).y_{min}}{T} =$$
$$= D.y_{max} + (1-D).y_{min}$$

Tento druhý výraz môže byť značne zjednodušený v prípade, že $y_{min} = 0$ a $\bar{y} = D.y_{max}$. Z toho je zrejmé, že priemerná hodnota signálu $\bar{y}$ je priamo úmerne závislá na dobe pracovného cyklu D.



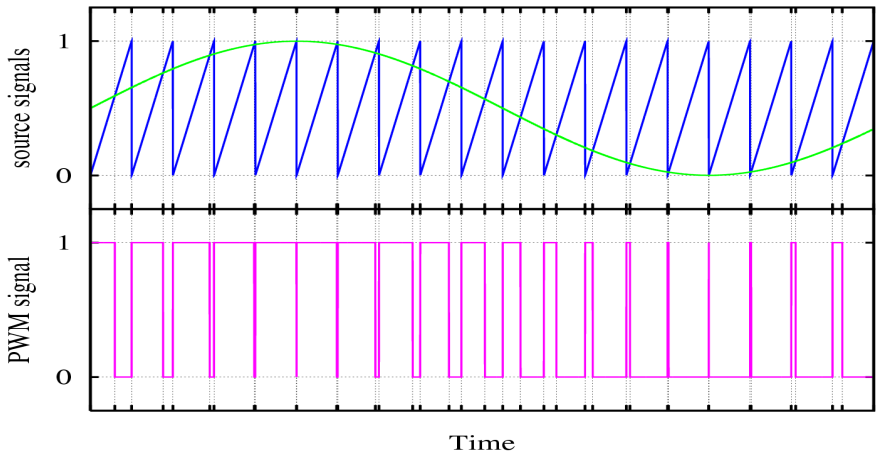
Obr. 1.5: Obdĺžnikový priebeh zobrazujúci popis y_{min}, y_{max} a D

1.4.2 Princípy generovania

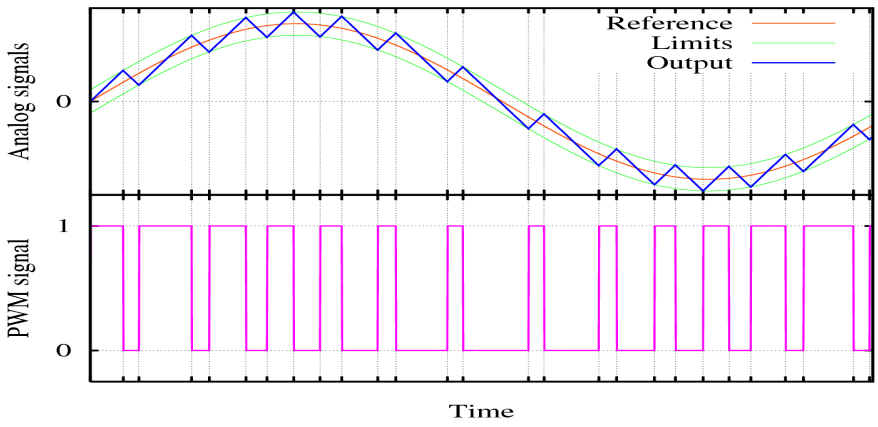
Existujú 4 spôsoby generovania pulzne - šírkovo modulovaného signálu, ktoré si bližšie popíšeme.

- **Pretínajúca (intersective)** – je najjednoduchším možným spôsobom generovania PWM signálu, ktorý vyžaduje iba *pílovitý* alebo *trojuholníkový priebeh* (ľahko generovateľný obyčajným oscilátorom) a komparátorom. Keď hodnota referenčného signálu (zelený sinusový priebeh na Obr. 1.6) je väčšia ako hodnota modulovaného priebehu (modrý), tak PWM signál (ružový) je vo vysokej hladine, v opačnom prípade je v hladine nízkej.
- **Delta** – výstupný signál je porovnávaný s medznými hodnotami, ktoré sa zhodujú s referenčným signálom o konštantnú odchýlku. Zakaždým, keď výstupný signál dosiahne jednu z medzných hodnôt, PWM signál zmení stav (viď Obr. 1.7).
- **Sigma-Delta –** výstupný signál je odčítaný z referenčného signálu na tvar signálu chyby. Táto chyba je integrovaná a keď integrál chyby prekročí medzné hodnoty, výstup zmení svoj stav (viď Obr. 1.8).
- **Číslicová (digitálny) –** mnoho číslicových obvodov ako napr. aj mikrokontroléry majú PWM výstup. Tento fakt má v našom prípade veľký význam práve kvôli možnosti ovládania servomotorov. Spravidla na to používajú čítač, ktorý sa periodicky inkrementuje (je napojený

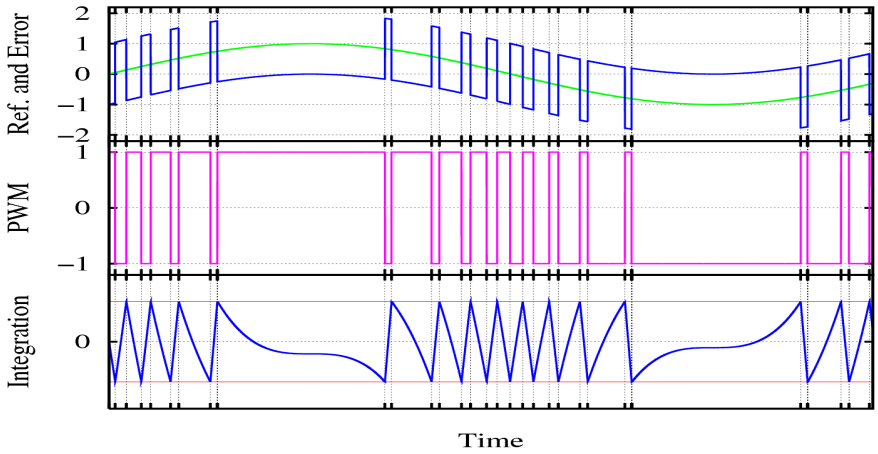
priamo či nepriamo na zdroj hodinového signálu) a je resetovaný pri každej perióde PWM signálu. Keď je hodnota čítača väčšia ako referenčná hodnota, PWM výstup zmení svoj stav z vysokej hladiny na nízku.



Obr. 1.6: Pretínajúca (intersective) metóda generovania PWM signálu



Obr. 1.7: Delta metóda generovania PWM signálu



Obr. 1.8: Sigma-Delta metóda generovania PWM signálu

1.4.3 Typy PWM modulácie

Existujú 3 možné spôsoby PWM modulácie.

1. Stred pulzu je sústredený v strede časového rámca a k zmene šírky pulzu dochádza sťahovaním alebo rozpínaním obidvoch hrán pulzu.
2. Nástupná hrana je držaná v čele časového rámca a zostupná hrana je modulovaná.
3. Zostupná hrana je držaná a nástupná hrana je modulovaná

1.4.4 Použitie PWM signálu

Využitie PWM signálu je široké. Okrem ovládania pohonu motorov nachádza svoje uplatnenie aj v nasledujúcich oblastiach:

- Telekomunikácie – šírky pulzov zodpovedajú hodnotám na jednej strane zakódovaným a na druhej strane dekódovaným
- Dodávka energie – znižuje množstvo výkonu dodávaného do záťaže bez strát, čo je spôsobené tým, že množstvo dodanej energie je úmerné modulácií pracovnej doby cyklu PWM.
- Regulácia napätia – prepínaním napätia do záťaže s vhodnou dobou pracovného cyklu môžeme dostať na výstupe približné napätie požadovanej úrovne.
- Audio efekty a zosilňovanie – PWM sa v tomto prípade využíva na zvukovú syntézu, vytváranie zvukových efektov.

2 Mikrokontroléry HC08

2.1 Úvod

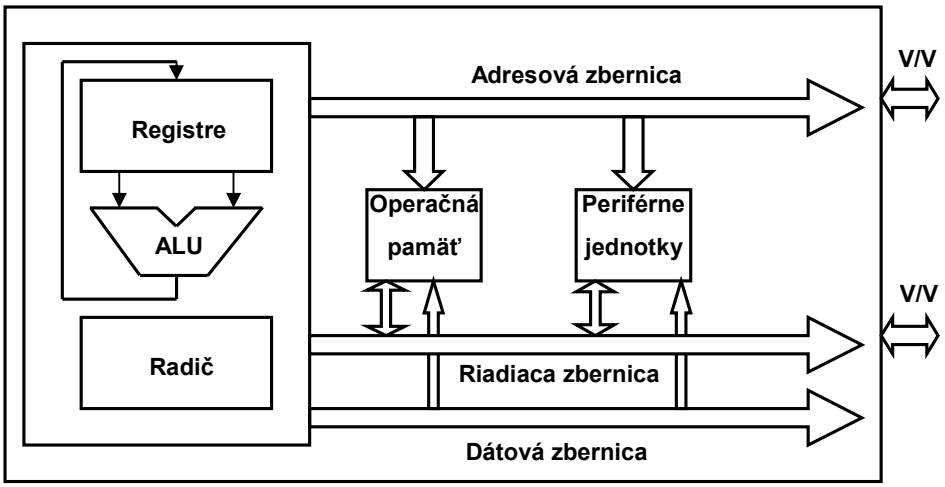
S masívnym nástupom osobných počítačov a vstavaných systémov do bežného života došlo k rýchlemu rozvoju mikroprocesorov a mikrokontrolérov, ktoré tvoria srdce týchto zložitých systémov. V tejto kapitole popíšem rozdiel medzi týmito dvoma výpočetnými jednotkami a zameriam sa najmä na mikrokontroléry, keďže tie tvoria hlavnú časť tejto práce. Väčšia časť kapitoly sa bude venovať použitým periférиám mikrokontroléra, ktoré tvorili neodmysliteľnú časť riadiaceho mechanizmu servomotorov.

2.1.1 Mikroprocesor

Mikroprocesor je základná procesorová jednotka CPU, ktorá je osadená na čipe. Obsahuje základné komponenty ako aritmeticko-logickú jednotku ALU, inštrukčný dekóder, registre a radič. Radič má na starosti chod celého mikroprocesoru a je tvorený súborom inštrukcií, obvodom na dekódovanie inštrukcie a riadiacim obvodom.

2.1.2 Mikrokontrolér

Mikrokontrolér sa od mikroprocesoru líši v tom, že okrem CPU, radiča a ALU má na čipe osadené aj ďalšie komponenty, ktorými sú operačná pamäť, periférne jednotky a zbernica. Preto sa tiež niekedy nazýva aj ako **mikropočítač**. Existuje mnoho druhov a výrobcov mikrokontrolérov. Dôležitým charakteristickým znakom je šírka zbernice, podľa ktorej delíme mikrokontroléry na 8 bitové, 16 bitové a 32 bitové.



Obr. 2.1: Štruktúra mikrokontroléru

2.2 Architektúra mikrokontrolérov HC08

Rodina 8-bitových mikrokontrolérov 68HC08 je pokročilou modifikáciou rodiny 68HC05 na báze mikroprocesoru M6800. Ich výhodou je jednoduchšia štruktúra a malý počet vývodov. Sú vhodné najmä na riešenie menších riadiacich aplikácií.

Tieto mikrokontroléry využívajú **von Neumannovskú architektúru** a pamäťovo mapované vstupy/výstupy. Obsahujú 5 CPU registrov, ktoré nie sú súčasťou pamäte. Týmito registrami sú 8-bitový akumulátor **A**, 16-bitový index register **H:X**, 16-bitový ukazateľ zásobníka **SP**, 16-bitový programový čítač **PC** a 8-bitový stavový register **CCR**.

Rodina mikrokontrolérov 68HC08 ako aj ostatné rodiny z rady mikrokontrolérov Motorola patria medzi tzv. CISC procesory, pre ktoré je charakteristické:

- veľké množstvo inštrukcií
- veľké množstvo adresovacích módov
- inštrukcie pre prácu s operandami v pamäti
- mnoho a zložité spôsoby adresovania

2.2.1 Mikrokontrolér MC68HC908LJ12

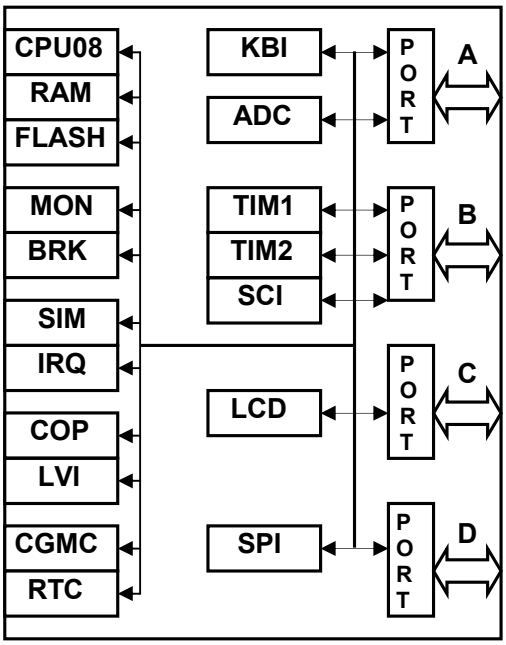
Mikrokontrolér MC68HC908LJ12, ktorý bol využitý pre našu aplikáciu, je členom rodiny 8-bitových mikrokontrolérov rady 68HC08. Jedná sa o vysoko výkonný a lacný mikrokontrolér, ktorý sa hodí na menšie riadiace aplikácie.

Špecifikácia mikrokontroléra LJ12:

- 12kB FLASH pamäť s možnosťou vnútro obvodového programovanie (in-circuit programming ISP)
- 512B RAM pamäť
- radič displeja LCD
- sériové asynchrónne rozhranie SCI a sériové synchrónne rozhranie SPI
- interný modul reálneho času RTC (Real Time Clock)
- 6-kanálový, 10-bitový AD prevodník
- dva 16-bitové dvojkanálové časovače
- 32 univerzálnych V/V vývodov
- 64 vývodové umelohmotné ploché štvorcové puzdro
- maximálna frekvencia zbernice – 8 MHz (5V) a 4 MHz (3,3V)
- KBI (Keybord Interrupt) – modul pre obsluhu ôsmych maskovateľných prerušení, určených pre obsluhu klávesnice

- BRK – modul na ladenie priamo v aplikácií (in-circuit debugging)
- SIM (System Integration Modul) – kontrola CPU, časovanie internej zbernice, riadenie prerušenia a resetu
- IRQ (Interrupt Request) – modul vonkajšieho maskovateľného prerušenie
- COP (Computer Operating Properly) – modul správneho behu programu, tiež nazývaný Watchdog
- LVI (Low voltage inhibit) – modul pre zabránenie poklesu napätia

Mikrokontrolér MC68HC908LJ12 zdieľa svoje vývody s periférіami, dajú sa však použiť aj ako obecné vývody. Port A zdieľa svoje vývody s modulmi KBI a ADC, port B s modulmi TIM1, TIM2 a SCI, port C plne zdieľa všetky svoje vývody s radičom displeja LCD a port D poskytuje svoje vývody pre moduly KBI a SPI.

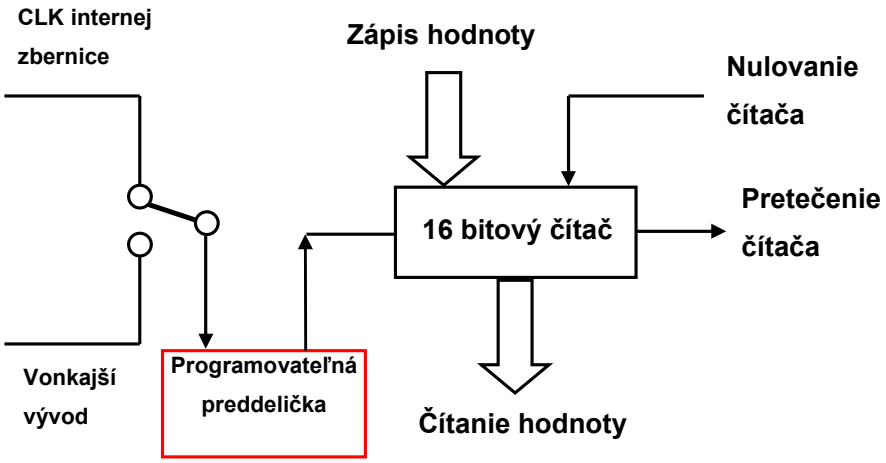


Obr. 2.2: Bloková schéma mikrokontroléra MC68HC908LJ12

MC68HC908LJ12 bol vhodným mikrokontrolérom pre vytvorenie našej aplikácie. Jeho pamäť má síce menšiu kapacitu (len 12kB) napr. v porovnaní s verziou GP32 (32kB), avšak pre naše potreby bola táto pamäť postačujúca. Oproti tomu ale obsahuje moduly ako AD prevodník a časovače TIM1 a TIM2, ktoré boli nevyhnutné pre vytvorenie ovládacích signálov pre servomotory. Ako velmi užitočný modul mi prišiel aj radič LCD displeja ako výstup testovacích údajov.

2.3 Časovací subsystém

Ak sa povie pojem časovací subsystém alebo v skratke časovač, myslí sa tým vlastne dvojica čítač/časovač. Táto jednotka je základnou časovacou jednotkou a slúži na odmeriavanie času.
Čítač je špeciálny register, ktorý okrem funkcií čítania a zápisu hodnoty zvyšuje alebo znižuje svoju hodnotu o jednotku po každej perióde hodinového signálu. Spravidla sa používa na čítanie udalostí. Časovač je v úzkom slova zmysle čítač, ktorého čas je tiež zvyšovaný o jednotku hodinovým signálom a jeho úlohou je odmeriavať čas.



Obr. 2.3: Štruktúra časovača s použitím programovateľnej preddeličky

Mikrokontroléry HC08 obsahujú prídavný register, ktorým môžeme meniť cyklus čítača tým, že do neho zapíšeme novú hodnotu a táto hodnota je porovnávaná s hodnotou čítača. Keď sú hodnoty rovnaké, čítač sa vynuluje. Cyklus časovača môžeme zväčšovať programovateľnou preddeličkou vstupného hodinového signálu. Výhodou je jednoduchšie detekovanie dlhších impulzov.

2.3.1 Časovací subsystém MC68HC908LJ12

Mikrokontrolér MC68HC908LJ12 obsahuje dva časovacie moduly TIM1 a TIM2, ktoré poskytujú funkcie pre jednotku záchytu hrany (input capture), jednotku výstupnej komparácie (output compare) a s ním spojenú a pre nás významovo najdôležitejšiu funkciu pre generovanie pulzne - šírkovo modulovaného signálu.

2.3.2 Vlastnosti časovacieho modulu TIM

Časovací modul mikrokontroléra MC68HC908LJ12 má nasledujúce vlastnosti:

- 2 kanály pre funkcie jednotky záchytu hrany/ výstupnej komparácie
  - jednotka záchytu nástupnej hrany, zostupnej hrany, obidvoch hrán
  - nastavenie, nulovanie alebo preklopenie pri výstupnej komparácií
- pulzne - šírková modulácia bez použitia alebo s využitím vyrovnávacej pamäte
- programovateľná preddelička s možnosťou 7-ich frekvencií internej zbernice
- preklopenie hodnoty na vývode po pretečení
- možnosť zastavenia a resetovania časovača

2.3.3 Konvencie pri nazývaní vývodov

V predchádzajúcom texte sme sa dozvedeli, že port B zdieľa svoje vývody s časovačmi TIM1 a TIM2, poskytujúc nám tak výstup funkcií jednotky záchytu hrany / výstupnej komparácie. Nasledujúca tabuľka zobrazuje presne, ktoré bity portu B sú zdieľané s jednotlivými kanálmi časovačov TIM1 a TIM2.

Obecný názov TIM vývodov:		**T[1,2]CH0**	**T[1,2]CH1**
Plné názvy TIM vývodov:	TIM1	PTB2 / T1CH0	PTB3 / T1CH1
	TIM2	PTB4 / T2CH0	PTB5 / T2CH1

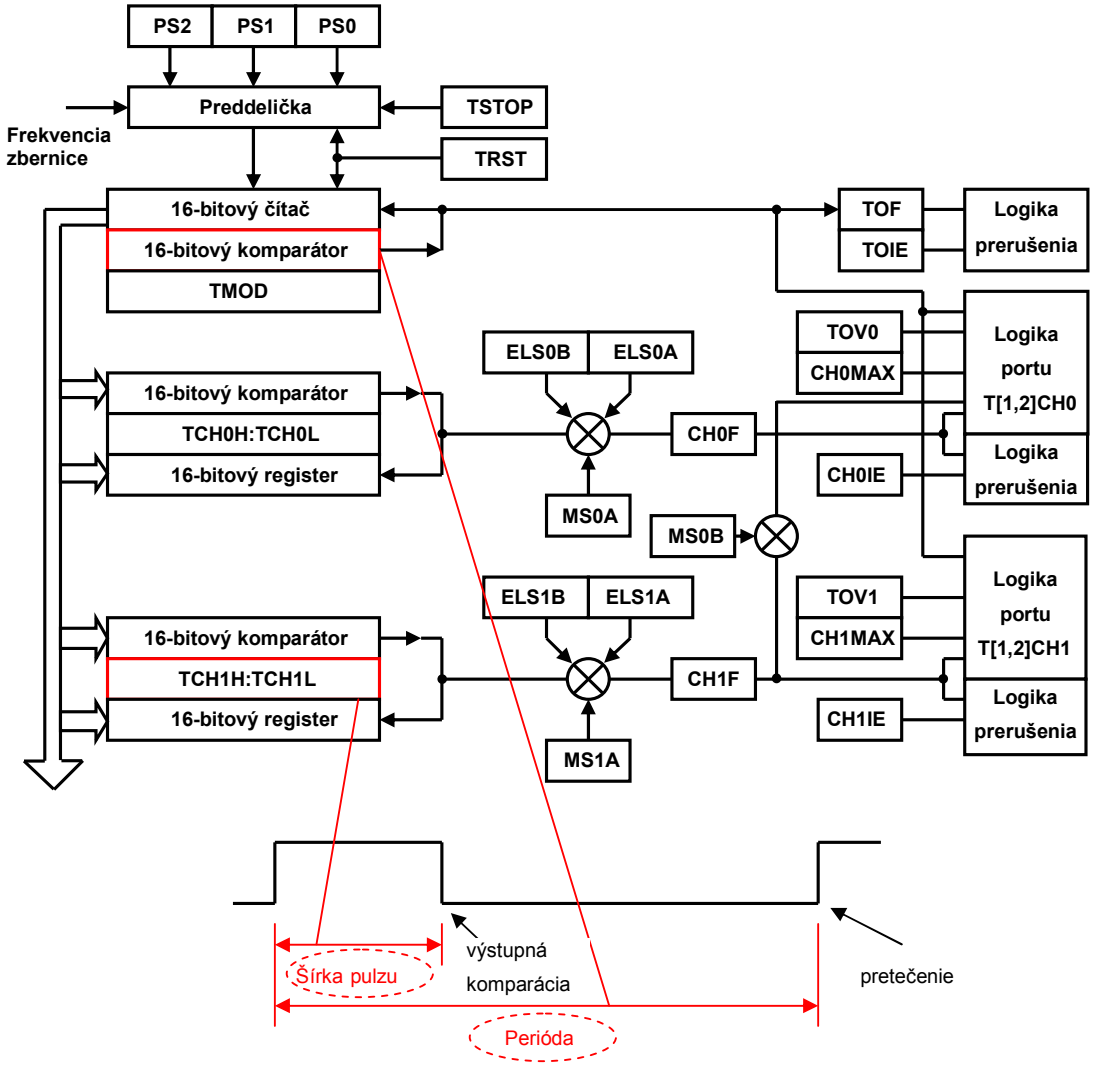
Tab. 2.1: Konvencie pri názvoch vývodov TIM1 a TIM2

2.3.4 Generovanie PWM bez použitia vyrovnávacej pamäte

Za pomoci funkcie preklopenia pri pretečení (toggle-on-overflow) máme možnosť generovať PWM signál na hociktorom kanáli realizujúcom funkciu výstupnej komparácie. Periódu signálu reprezentuje hodnota zapísaná v modulo registri čítača TIM a hodnota v kanálovom registri predstavuje šírku pulzu PWM signálu. Hodnota sa preklopí na vývode kanálu v momente, keď čítač dosiahne hodnotu zapísanú v modulo registri časovača. Čas medzi pretečeniami je perióda signálu a čas medzi pretečením a výstupnou komparáciou je šírka pulzu. Ak nastavíme, aby sa pri výstupnej komparácií vynulovala hodnota na vývode kanála časovača, dostaneme tak plne funkčný PWM signál.

Na zmenu frekvencie PWM výstupu slúži okrem modulo registra ešte hodnota nastaviteľná v preddeličke vstupného hodinového signálu.

Hociktorý z kanálov výstupnej komparácie dokáže generovať PWM signál bez použitia vyrovnávacej pamäte po angl. unbuffered PWM signal generation. Znamená to, že pri požiadavku na novú hodnotu šírky pulzu musíme prepísať touto hodnotou obsah kanálového registra. Tento spôsob generovania PWM signálu bol využitý aj v našom prípade.



Obr. 2.4: Blokový diagram časovača TIM

2.4 AD prevodník

Analógovo - číslicový prevodník je technický prostriedok, ktorý slúži na prevod hodnoty analógovej veličiny v danom čase na číselnú podobu. Samotnému AD prevodníku predchádza obvod **Sample and Hold** alebo **Track and Hold,** ktorý si uchováva hodnotu analógovej veličiny do doby, kým je AD prevod dokončený. Pre správne navzorkovanie hodnoty musí byť splnená podmienka **Shanonovho vzorkovacieho teorému**, podľa ktorej platí, že vzorkovací kmitočet AD prevodníka musí byť aspoň 2x väčší než najvyšší kmitočet obsiahnutý vo vzorkovanom analógovom signále. Preto býva tomuto obvodu ešte predradený ďalší prvok filter, ktorý slúži ako dolný priepust a ten má za úlohu vyfiltrovať vysokofrekvenčné zložky z analógového signálu.

2.4.1 Vlastnosti AD prevodníka

Mikrokontrolér MC68HC908LJ12 obsahuje 6 kanálový 10-bitový AD prevodník, ktorý používa lineárnu postupnú aproximáciu LPA na prevod analógovej hodnoty na číselnú reprezentáciu. Podrobná špecifikácia AD prevodníka:

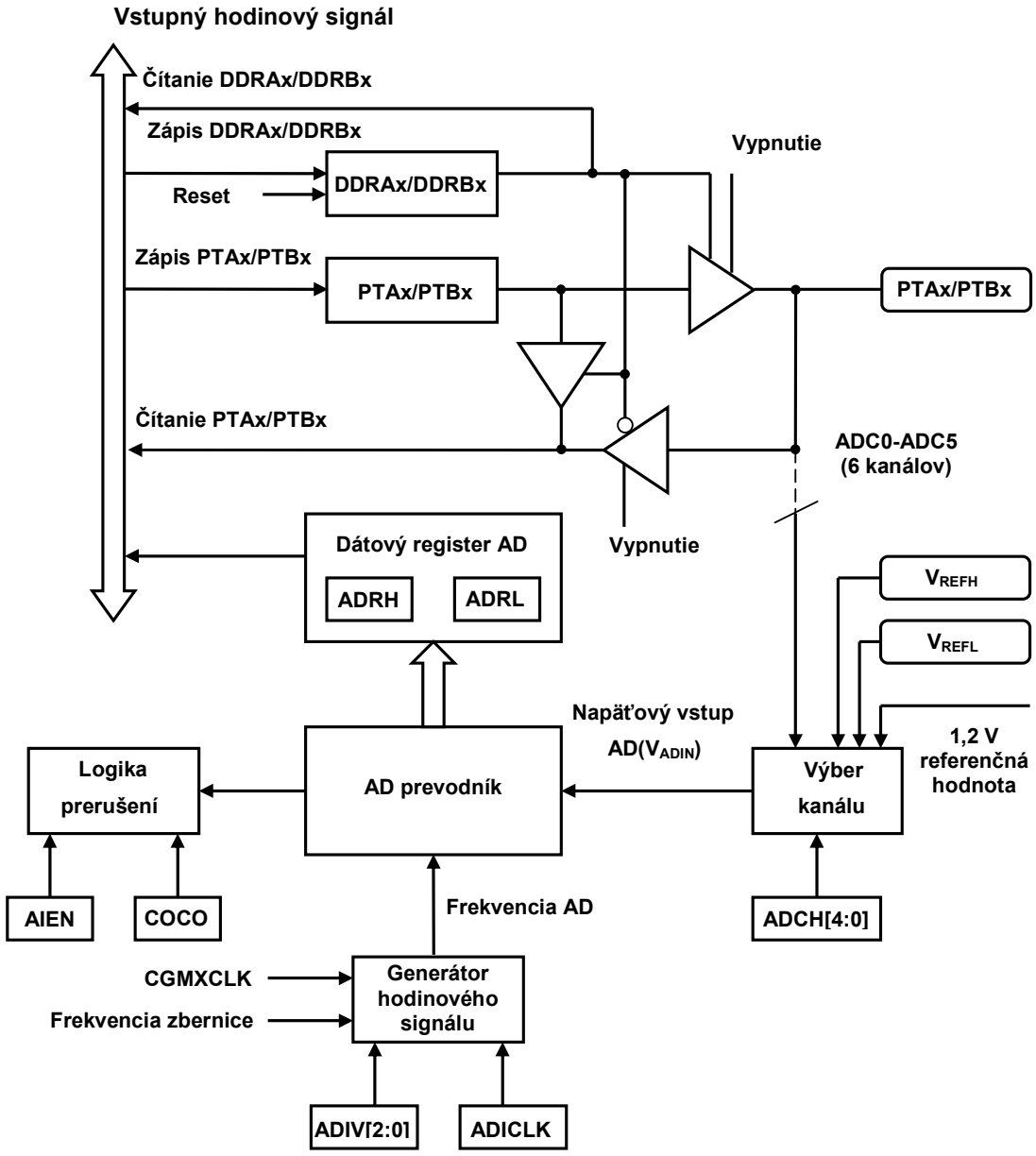
- 6 časovo multiplexovaných AD kanálov
- 8 alebo 10 bitová presnosť
- LPA – Lineárna postupná aproximácia
- jednorázový alebo nepretržitý AD prevod
- možnosť generovania prerušenia po dokončení AD prevodu
- možnosť programového testovania príznaku „AD prevod dokončený“
- nastaviteľná frekvencia AD prevodu
- zaokrúhľovanie výsledku
  - orezanie na 8 bitov
  - zarovnanie doprava
  - zarovnanie doľava
  - znamienkové zarovnanie doľava

2.4.2 Popis funkcie

AD prevodník poskytuje 6 vývodov na vzorkovanie vonkajších zdrojov analógového signálu, ktorými sú PTA4/ADC0 – PTA7/ADC3 a PTB6/ADC4 – PTB7/ADC5. Pomocou prvku zvaného multiplexor máme možnosť výberu jedného zo šiestich AD kanálov ako napäťový vstup AD (V_{ADIN}). V_{ADIN} je konvertovaný pomocou lineárnej postupnej aproximácie. Prevedená hodnota sa potom uloží do

dátových registrov AD prevodníka a nastaví sa vlajka ukončenia prevodu alebo sa vygeneruje prerušenie.

Vývody PTA4-PTA7 a PTB6-PTB7 sú obecne využiteľné porty, ktoré sú zdieľané s kanálmi AD prevodníka. Nastavením niektorého z vývodov pre AD prevod je okamžite pozastavené možnosť obecného použitia tohto vývodu.



Obr. 2.5: Blokový diagram AD prevodníka

2.4.3 Prevodná funkcia

Vstupné napätie V_{IN} musí byť v intervale referenčných napätí V_{REFL} a V_{REFH}. Pri plnom 10-bitovom vzorkovaní platí, že ak sa vzorkované napätie $V_{IN} = V_{REFH}$, tak AD prevodník prevedie túto hodnotu na vzorku 03FFh. V opačnom prípade ak sa $V_{IN} = V_{REFL}$, tak vznikne vzorka 0000h.
Obecne platí:

$$V_{dig} = [V_{in} / (V_{REFH} - V_{REFL})] * (2^{10} - 1)$$

2.4.4 Doba prevodu

Konverzia začína po zápise do riadiaceho registra AD prevodníka a trvá zhruba 16 až 17 cyklov hodinového signálu.

$$\text{čas konverzie} = \frac{16 \text{ až } 17 \text{cyklov}}{\text{frekvencia AD}}$$

Frekvenciu AD prevodu dostávame z frekvencie vstupného hodinového signálu, ktorá je rovná f_{BUS} = 2,4576 MHz. Ak máme nastavený deliaci pomer v riadiacom registri AD prevodníka, tak ňou musíme ešte podeliť vstupnú frekvenciu. Zdrojom hodinového signálu môže byť aj externý signál CGMXCLK. Počet cyklov hodinového signálu potom vypočítame ako:

$$\text{takty hodinového signálu} = \text{čas konverzie} \times \text{frekvencia zbernice}$$

2.4.5 Ukladanie výsledku prevodu

Existujú 4 spôsoby formátovania výsledku AD prevodu:

- orezanie na 8 bitov
- zarovanie doprava
- zarovnanie doľava
- znamienkové zarovnanie doľava

V našom prípade bolo využité **zarovnanie výsledku doprava**. Tento režim zarovnania ukladá 2 najvyššie bity výsledku do horného bajtu dátového registra na 0. a 1. bit a a zvyšných 8 bitov do dolného bajtu dátového registra. Takto dostávame presnejší 10-bitový výsledok, pri ktorom musíme pred čítaním nižšieho bajtu registra prečítať najprv vyšší bajt, inak získame zlú hodnotu.

3 Návrh a vlastná implementácia

V tejto kapitole sa budem zaoberať návrhom a vlastnou implementáciou programu na ovládanie robotického manipulátora.

V prvom rade bolo potrebné vytvoriť prípravok na pripojenie servomotorov robota a joysticku k nášmu mikrokontroléru. Toto zariadenie malo vytvárať jednoduché rozhranie medzi jednotlivými prvkami, preto muselo byť čo najmenšie a najmenej komplikované. Jeho návrh bude popísaný v prvej časti tejto kapitoly.

V ďalšom kroku bolo potrebné oboznámiť sa so spôsobom výpočtu ovládacích signálov servomotora, zoznámiť sa s registrami časovača a ich správneho nastavenia tak, aby generovali nami požadovaný PWM signál. Okrem toho bolo potrebné zamerať sa tiež na možnosť softvérového generovania PWM signálu bez použitia časovacieho subsystému mikrokontroléra, ktorý bol využitý pri jednom zo servomotorov.

Ďalšia časť kapitoly bude pojednávať o AD prevodníku, konkrétne o jeho registroch a taktiež o spôsobe programového nastavenia a ovládania.

Posledná sekcia je zameraná na samotnú implementáciu ovládania robotického manipulátora.

3.1 Vývojové prostriedky

Ako implementačné prostredie pre vývoj našej aplikácie bol zvolený Freescale CodeWarrior. Toto vývojové prostredie poskytuje prostriedky pre programovanie širokej škály mikrokontrolérov firmy Motorola.

V prostredí CodeWarrioru máme možnosť programovania v assembleri, jazyku C a C++. Pri vytváraní projektu sa nám automaticky generujú startup súbory, hlavičkové súbory pre daný typ mikrokontroléra a knižnice. Taktiež sa vygeneruje súbor `main.asm` alebo `main.c` podľa zvoleného implementačného jazyka. CodeWarrior poskytuje výber spôsobu ladenia

- Full Chip Simulation
- Mon08 Interface – pre vývojový kit
- P&E Multilink / Cyclone Pro
- SofTec HC08
- MMDS – MMEVS Emulator
- FSICE Emulator

Prostredie CodeWarrioru obsahuje aj simulátor periférií napr. pre 7-segmentový displej, LCD, tlačítkovú klávesnicu, umožňuje nastavenie portov a pod.

Po spustený ladiaceho programu (debugger) sa nám otvorí hlavné okno s ďalšími vnútornými oknami, ktoré zobrazujú zdrojový kód, preložený kód v assembleri, obsahy registrov, mapu pamäte, premenné a procedúry programu a príkazový riadok.

CodeWarrior používa k prekladu zdrojového `*.c` súboru prekladač splňujúci kritériá normy ANSI-C/cC++.

Náš projekt pozostáva zo zdrojového súboru `robot.c`, ktorý obsahuje všetky základné rutiny potrebné k ovládaniu robotického manipulátora.

3.2 Prepojenie komponentov

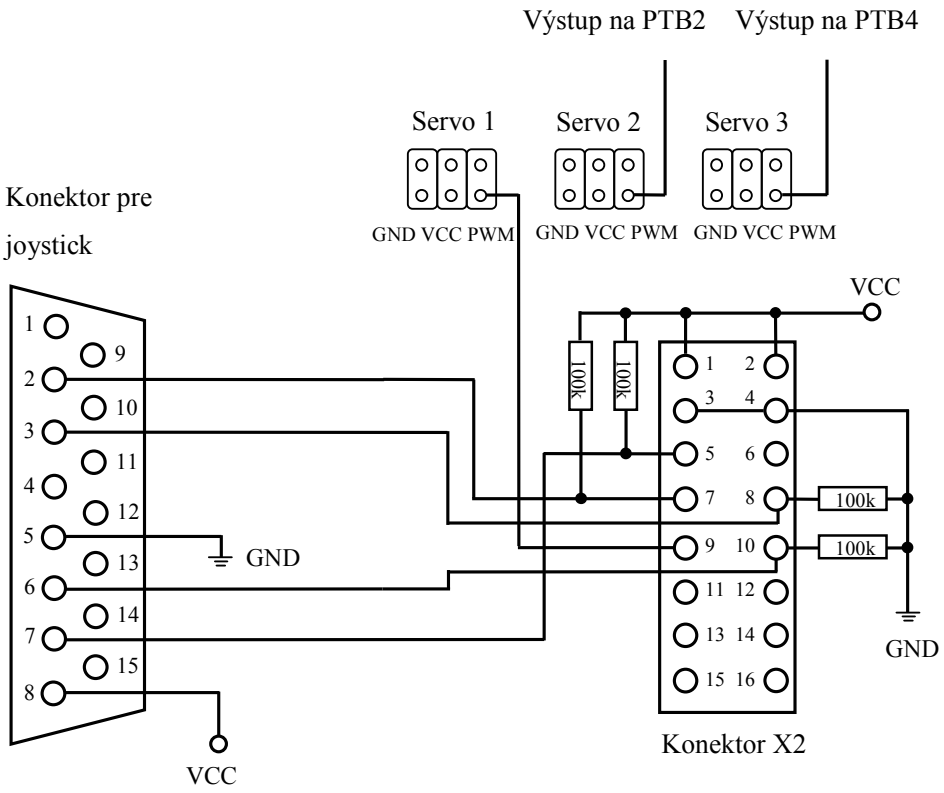
Pre pripojenie joysticku a servomotorov robota k mikrokontroléru bola navrhnutá jednoduchá doska. Na jej vytvorenie sme potrebovali:

- tri 3-kolíkové konektory na pripojenie servomotorov
- PIN D-SUB (samica) – konektor pre pripojenie joysticku
- štyri 100kΩ SMD odpory
- jeden plochý 16-žilový kábel
- jeden plochý 5-žilový kábel

Káble vedené zo servomotorov pripojíme na kolíkové konektory. Signálové vodiče (PWM) z dvoch servomotorov ovládajúcich rameno sa pripoja na vývody PTB2 a PTB4 mikrokontroléra. Vývod PTB2 je výstupom T1CH0 (kanál 0 časovača TIM1) a vývod PTB4 je výstupom T2CH0 (kanál 0 časovača TIM2). Signálový vodič tretieho serva ovládajúceho kliešte je vyvedený na vývod PTD2.

Potenciometre joysticku sú pripojené na dva kanály AD prevodníka, ktorými sú ADC0 a ADC1. Na výstupy potenciometrov sme ešte pripojili dva 100k odpory a dorobili vetvu na zem, aby sme z klasických potenciometrov dostali deliče napätia.

Tlačítka joysticku sme pripojili na porty PTD0 a PTD1. Aj v tomto prípade sme na výstupy tlačidiel pripojili dva 100k odpory. Bolo to potrebné z toho dôvodu, aby pri rozopnutom tlačítku neostala na vývode náhodná hodnota. Týmto odporom pevne definujeme vysokú hladinu (logická 1) na vývode portu v prípade rozopnutého tlačítka. Pri stlačenom tlačítku sa na vývode portu objaví hodnota logickej 0, pretože tlačítka spínajú zem. Na dolnom obrázku vidíme detailnú schému zapojenia.



Obr. 3.1: Schéma zapojenia komponentov

Konektor joysticku							
Pin	**Názov**	**Smer**	**Popis**	**Pin**	**Názov**	**Smer**	**Popis**
1	+5V	OUT	+5 VDC	9	+5V	OUT	+5 VDC
2	/B1	IN	Tlačidlo1	10	/B3	IN	Tlačidlo 3
3	X1	IN	Joystick 1 – X	11	X2	IN	Joystick 2 – X
4	GND	-	Zem	12	GND	-	Zem
5	GND	-	Zem	13	Y2	IN	Joystick 2 – Y
6	Y1	IN	Joystick 1 – Y	14	/B4	IN	Tlačidlo 4
7	/B2	IN	Tlačidlo 2	15	+5V	OUT	+5 VDC
8	+5V	OUT	+5 VDC				

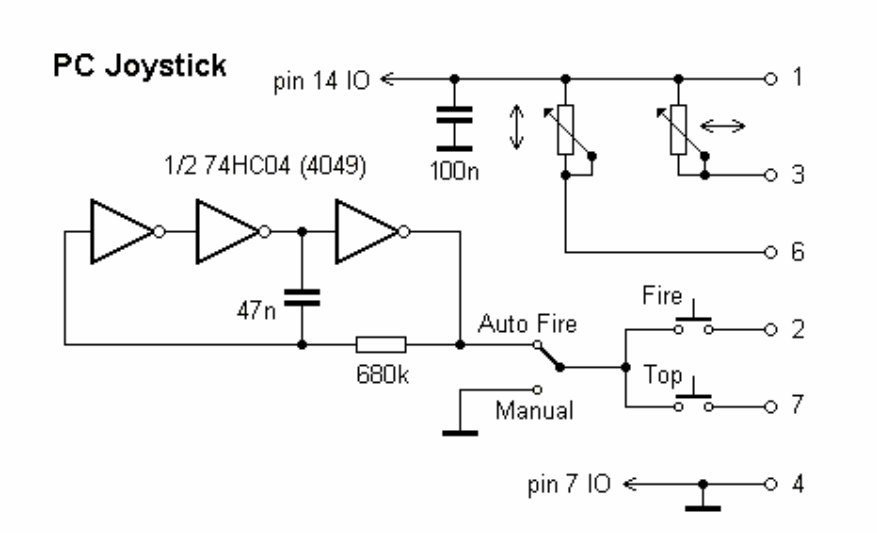
Tab. 3.1: Vývody konektora joysticku

Konektor X2 na strane kitu LJ12EVB			
Vývod	**Popis**	**Vývod**	**Popis**
1	VCC	9	PTD2/MOSI
2	VCC	10	PTA5/ADC1
3	GND	11	PTD3/SPSCK
4	GND	12	PTA6/ADC2
5	PTD0/$\overline{SS}$	13	PTC7/FP26
6	PTA3/KBI3	14	PTA7/ADC3
7	PTD1/MISO	15	TXD_X2
8	PTA4/ADC0	16	RXD_X2

Tab. 3.2: Vývody konektora X2

K testovaniu aplikácie bol využitý klasický dvojosí joystick s dvoma tlačidlami. Pákou joysticku sa dá pohybovať v smere osi Y (hore a dole) a v smere osi X (doprava a doľava). Táto páka je napojená na dva regulovateľné rezistory - potenciometre. Pohybom páky regulujeme odpor na potenciometroch a na výstupe tak dostávame rôzne hodnoty napätia.

Joystick má na páke dve tlačidlá, ktorými vieme spustiť nejaký druh akcie. Tieto tlačítka fungujú ako jednoduché zapínacie/vypínacie prepínače. Na dolnom obrázku môžeme vidieť schému použitého joysticku.



Obr. 3.2: Schéma zapojenia joysticku

3.3 Výpočet ovládacích signálov

V predchádzajúcej kapitole sme v časti o časovači spomenuli, že práve tento modul slúži na generovanie požadovaného PWM signálu s danou periódou a šírkou pracovnej doby cyklu. Taktiež už vieme, že k ovládaniu našich servomotorov v rozmedzí -90° až +90° potrebujeme generovať impulzy v intervale od 0,5ms do 2,5ms. Ako však tieto hodnoty reprezentovať v podobe, aby sme ich vedeli zapísať do príslušných registrov časovacie modulu?

K výpočtu číselnej formy periódy a pracovnej doby cyklu musíme vychádzať z frekvencie zbernice f_{BUS}. U mikrokontrolérov rady HC08 má na starosti generovanie hodinového signálu modul **CGM** (Clock Generator Modul). Tento generátor väčšinou pozostáva z dvoch obvodov – z generátoru striedavého periodického signálu, ktorým je oscilátor a z obvodov, ktoré tento signál upravujú. Týmito obvodmi môžu byť rôzne deliče alebo násobiče kmitočtu, fázové závesy a pod. Zvyčajne vstavané systémy používajú na generovanie periodického signálu vlastný kryštálový oscilátor ako aj náš kit, na ktorom je osadený 32kHz oscilátor s fázovým závesom, ktorý zvyšuje kmitočet rádovo až na MHz. V našom prípade však bolo potrebné využiť externý oscilátor osadený na ladiacom rozhraní MiniMON, keďže sme používali režim „monitor“ na programovanie a ladenie aplikácie. CGM generuje frekvenciu 8,3904MHz, tá je však ešte pred výstupom CGMOUT podelená dvomi. Tento signál je v module SIM znovu podelený dvomi, takže tým dostávame frekvenciu zbernice f_{BUS} = 2,4576MHz.

Ďalšia vec, ktorú je potrebné vedieť pred výpočtom šírky impulzu, je doba trvania jedného cyklu hodinového signálu. Túto hodnotu vypočítame jednoducho podľa vzorca:

$$\text{čas trvania cyklu} = \frac{1}{f_{BUS}} = \frac{1}{2{,}4576\text{MHz}} = 407\text{ns}$$

Ak už poznáme čas trvania jedného cyklu, vieme si ľahko určiť počet cyklov potrebných pre generovanie konkrétnej periódy a šírky pulzu. Určime si najprv periódu signálu, ktorú budeme musieť zabezpečiť pre servomotory. Vieme, že táto perióda sa má opakovať s frekvenciou 50Hz, čo znamená, že potrebujeme periódu dĺžky 20ms.

$$\text{perióda} = \frac{f_{BUS}}{50\text{Hz}} = \frac{2{,}4576\text{MHz}}{50\text{Hz}} = 49152\,(C000_{HEX})$$

Už poznáme počet cyklov potrebných pre vytvorenie periódy, teraz musíme zistiť počet cyklov potrebných pre generovanie pracovných dôb PWM signálu. Pre účely aplikácie postačí výpočet troch

najdôležitejších hodnôt, ktorými sú medzné hodnoty pre najnižšiu šírku pulzu 0,5ms, strednú hodnotu 1,5ms a najvyššiu hodnotu impulzu 2,5ms.

$$0{,}5\text{ms} * \frac{10^6\ \text{ns}}{1\text{ms}} * \frac{1\,\text{takt hodinového signálu}}{407} = 1228\,(4\text{CC}_{\text{HEX}})$$

$$1{,}5\text{ms} * \frac{10^6\ \text{ns}}{1\text{ms}} * \frac{1\,\text{takt hodinového signálu}}{407} = 3685\,(\text{E65}_{\text{HEX}})$$

$$2{,}5\text{ms} * \frac{10^6\ \text{ns}}{1\text{ms}} * \frac{1\,\text{takt hodinového signálu}}{407} = 6142\,(17\text{FE}_{\text{HEX}})$$

Z týchto hodnôt vyplýva, že pre správne generovanie periódy musí každých 49152 taktov hodinového signálu dôjsť k pretečeniu čítača a vzapätí nastavenia výstupu do jednotky. Pre ovládanie servomotorov v intervale od 0,5 do 2,5ms musí časovač mikrokontroléra generovať funkciu výstupnej komparácie v rozsahu od 1228 do 6142 taktov hodinového signálu. Neostáva teda nič iné, len správne nastaviť časovače, aby sme dostali periódu a signál požadovanej dĺžky na vývodoch mikrokontroléra.

3.4 Nastavenie registrov časovača

Obidva časovače TIM1 aj TIM2 mikrokontroléra MC68HC908LJ12 pozostávajú z 5-tich registrov, ktorými sú:

- TIM stavový a riadiaci register (TSC)
- TIM čítací register (TCNTH:TCNTL)
- TIM modulo register čítača (TMODH:TMODL)
- TIM kanálový stavový a riadiaci register (TSC0, TSC1)
- TIM kanálový register (TCH0H:TCH0L, TCH1H:TCH1L)

Modifikovateľné sú všetky okrem čítacieho registra TCNTH:TCNTL, ktorý je k dispozícií len na čítanie. Na nastavenie registrov časovača TIM1 a TIM2 slúžia dve rutiny v našej aplikácií

- `rob13_Initialize_TIM1()` – inicializácia registrov časovača TIM1
- `rob13_Initialize_TIM2()` – inicializácia registrov časovača TIM2

Najprv si však treba povedať aké bity jednotlivé registre obsahujú, čo tieto bity nastavujú a aký majú význam v rámci našej aplikácie.

3.4.1 Stavový a riadiaci register TSC

Stavový a riadiaci register (TSC) má nasledovné úlohy:

- Povolenie prerušenia po pretečení z časovača TIM
- Nastavenie vlajky pretečenia
- Zastavenie časovača TIM
- Resetovanie časovača TIM
- Nastavenie deliaceho pomeru preddeličky

Adresa v pamäti: T1SC ($0020) a T2SC ($002B)

	Bit 7	6	5	4	3	2	1	0
Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST				

Obr. 3.3: Stavový a riadiaci register TSC

TOF – Bit pretečenia časovača TIM

Tento bit je nastavený do jednotky, keď časovač dosiahne hodnotu zapísanú v modulo registri časovača. TOF bit sa vynuluje čítaním zo stavového a riadiceho registra a následným zápisom logickej 0 do tohto bitu. Reset nuluje TOF bit. Zápis logickej 1 nemá vplyv na TOF bit.

1 = časovač TIM dosiahol hodnotu zapísanú v modulo registri

0 = časovač TIM nedosiahol hodnotu zapísanú v modulo registri

TOIE – Bit povolenia prerušenia z časovača

Tento bit povoluje prerušenie po pretečení z časovača TIM, keď je bit TOF nastavený do 1. Reset nuluje bit TOIE.

1 = povolenie prerušenia po pretečení z časovača TIM

0 = zakázanie prerušenia po pretečení z časovača TIM

TSTOP – Bit zastavenia časovača TIM

Tento bit zastavuje časovač TIM. Zápis logickej 0 do tohto bitu spúšťa časovač opäť do činnosti. Reset nastavuje bit TSTOP do 1.

1 = časovač TIM zastavený

0 = časovač TIM spustený

TRST – Bit resetu časovača TIM

Nastavenie bitu TRST do jednotky resetuje časovač TIM a preddeličku registra TSC, na iné registre nemá vplyv. Počítanie po resete časovača začína od $0000. Reset nuluje bit TRST.

1 = časovač TIM a preddelička nulovaná

0 = žiaden efekt

PS[2:0] – Bity preddeličky vstupného kmitočtu

Pomocou týchto troch bitov je možné vybrať 1 zo 7-ich výstupov preddeličky vstupného kmitočtu podľa nasledujúcej tabuľky:

PS2	PS1	PS0	Frekvencia časovača TIM
0	0	0	Frekvencia ÷ 1
0	0	1	Frekvencia ÷ 2
0	1	0	Frekvencia ÷ 4
0	1	1	Frekvencia ÷ 8
1	0	0	Frekvencia ÷ 16
1	0	1	Frekvencia ÷ 32
1	1	0	Frekvencia ÷ 64
1	1	1	-

3.4.2 Čítací register TCNT

Tieto 16-bitové registre nie sú modifikovateľné a obsahujú nižší a vyšší byte hodnoty čítača. Reset nuluje čítacie registre časovača TIM, rovnako aj nastavenie bitu TRST do jednotky.

Adresa v pamäti: T1CNTH ($0021) a T2CNTH ($002C)

	Bit 7	6	5	4	3	2	1	0
Read:	15	14	13	12	11	10	9	8
Write:								

Adresa v pamäti: T1CNTL ($0022) a T2CNTL ($002D)

	Bit 7	6	5	4	3	2	1	0
Read:	7	6	5	4	3	2	1	0
Write:								

Obr. 3.4: Čítacie registre T2CNT a T1CNT

3.4.3 Modulo register TMOD

Modulo registre TMOD obsahujú hodnotu čítača TIM. Keď čítač dosiahne hodnotu uloženú v tomto registri, nastaví sa bit TOF do jednotky a čítač začne počítať od $0000 pri ďalšom takte hodinového signálu. Reset nastavuje modulo register TMOD.

Adresa v pamäti: T1MODH ($0023) a T2MODH ($002E)

	Bit 7	6	5	4	3	2	1	0
Read: Write:	15	14	13	12	11	10	9	8

Adresa v pamäti: T1MODL ($0024) a T2MODL ($002F)

	Bit 7	6	5	4	3	2	1	0
Read: Write:	7	6	5	4	3	2	1	0

Obr. 3.5: Modulo registre T1MOD a T2MOD

V rámci našej aplikácie nebolo potrebné využiť obslužnú rutinu prerušenia po pretečení časovača a z toho dôvodu boli zakázané prerušenia z oboch časovačov.

Deliaci pomer preddeličky bol nastavený na pomer 1÷1, čím sme vlastne zabránili možnosti podeliť vstupný kmitočet. Tento krok bol zvolený z jednoduchého dôvodu. V sekcii 3.1 sme sa totiž dozvedeli, že počet cyklov potrebných pre generovanie 20ms periódy je rovný 49152 ($C000_{HEX}$). Táto hodnota sa nám vojde do dvoch bajtov modulo registra TMOD, a teda nie je potrebné ďalšie delenie vstupnej frekvencie zbernice.

Ako posledný krok pre inicializáciu časovačov nám stačí resetovať čítač a následne ho spustiť. Týmto sme vlastne zabezpečili správne generovanie periódy signálu počas celej doby behu programu a ostáva nám teda ešte správne nastaviť stavové a riadiace registre kanálov časovača TIM1 a TIM2 pre vytvorenie korektného PWM signálu a zápis šírky impulzov do kanálových registrov. Opäť si však najprv treba povedať aké bity obsahujú tieto registre, akú majú funkciu a ich význam v rámci programu.

3.4.4 Kanálový stavový a riadiaci register TxSC0 a TxSC1

Každý stavový a riadiaci register kanálov časovača TIM poskytuje nasledovné funkcie

- Nastavenie vlajky pri funkciách záchytu hrany a výstupnej komparácie
- Povolenie prerušenia pri funkciách záchytu hrany a výstupnej komparácií

- Výber funkcie záchytu hrany, výstupnej komparácie alebo PWM operácie
- Nastavenie, nulovanie alebo preklopenie pri výstupnej komparácií
- Nastavenie záchytu hrany na nástupnú hranu, zostupnú hranu alebo obe hrany
- Nastavenie preklopenia výstupu po pretečení
- Výber 0% a 100%-nej pracovnej doby PWM signálu
- Výber funkcie výstupnej komparácie/PWM operácie s využitím alebo bez využitia vyrovnávacej pamäte

Adresa v pamäti: T1SC0 ($0025) a T2SC0 ($0030)

	Bit 7	6	5	4	3	2	1	0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							

Adresa v pamäti: T1SC1 ($0028) a T2SC1 ($0033)

	Bit 7	6	5	4	3	2	1	0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							

Obr. 3.6: Stavový a riadiaci register TSC0 a TSC1

CHxF – Bit vlajky kanála x

Bit CHxF sa nastaví do jednotky pri funkcii výstupnej komparácie, ak sa hodnota zapísaná v čítacom registri rovná hodnote v kanálovom registri časovača. Tento bit sa nastavuje aj pri jednotke záchytu hrany. Reset nuluje bit CHxF. Zápis logickej 1 do CHxF nemá žiaden efekt.

1 = záchyt hrany alebo výstupná komparácia na kanále x

0 = žiaden záchyt hrany alebo výstupná komparácia na kanále x

CHxIE - Povolenie prerušenia na kanále x

Tento bit povoluje prerušenie z kanála x. Reset nuluje bit CHxIE.

1 = povolenie prerušenia na kanále x

0 = zakázanie prerušenia na kanále x

MSxB – Bit B výber režimu

Tento bit slúži pre výber PWM operácie s využitím vyrovnávacej pamäte. Bit MSxB obsahuje len kanálový register 0 časovača TIM1 a TIM2. Reset nuluje bit MSxB.

1 = povolenie PWM operácie s využitím vyrovnávacej pamäte

0 = zabránenie PWM operácie s využitím vyrovnávacej pamäte

MSxA – Bit A výber režimu

Tento bit nám slúži pre výber PWM operácie bez využitia vyrovnávacej pamäte alebo jednotky záchytu hrany v prípade, že bity ELSxB:ELSxA ≠ 0:0. Reset nuluje bit MSxA.

1 = PWM operácia bez využitia vyrovnávacej pamäte

0 = jednotka záchytu hrany

ELSxB:ELSxA – Bit výberu hrany/hladiny

Význam týchto bitov je úzko spojený s bitmi MSxB:MSxA, preto pre lepšiu názornosť uvedieme v tabuľke, ako sa používajú.

MSxB:MSxA	ELSxB:ELSxA	Režim	Konfigurácia
x 0 x 1	0 0 0 0	Predvoľba výstupu	Vysoká hladina Nízka hladina
0 0 0 0 0 0	0 1 1 0 1 1	Záchyt hrany	Záchyt na nástupnú hranu Záchyt na zostupnú hranu Záchyt oboch hrán
0 1 0 1 0 1	0 1 1 0 1 1	Výstupná komparácia a PWM operácia	Preklopenie výstupu Nulovanie výstupu Nastavenie výstupu
1 x 1 x 1 x	0 1 1 0 1 1	Výstupná komparácia a PWM operácia s využitím vyrovnávacej pamäte	Preklopenie výstupu Nulovanie výstupu Nastavenie výstupu

TOVx – Bit preklopenia výstupu po pretečení

Tento bit má význam len pri výstupnej komparácií, lebo kontroluje chovanie výstupu kanála x po pretečení časovača TIM.

1 = kanál x sa preklopí po pretečení čítača TIM

0 = kanál x sa nepreklopí po pretečení čítača TIM

CHxMAX – Bit maximálnej pracovnej doby

Ak je bit TOVx nastavený do logickej 1, tak nastavenie bitu CHxMAX zabezpečí generovanie 100% pracovnej doby cyklu.

3.4.5 Kanálové registre časovača TxCH0 a TxCH1

Tieto registre obsahujú hodnotu, pri ktorej dochádza k záchytu hrany alebo výstupnej komparácií podľa toho, ktorá funkcia bola nastavená v kanálovom stavovom a riadiacom registri TxSC0 a TxSC1. Po resete je hodnota kanálových registrov neznáma.

Zápis do vyššieho bajtu registra TCHxH v režime výstupnej komparácie zabraňuje ďalšej funkcií výstupnej komparácie do doby, kým nedôjde k zápisu hodnoty do nižšieho bajtu TCHxL.

Adresa v pamäti: T1CH0H ($0026), T1CH1H ($0029)
Adresa v pamäti: T2CH0H ($0031), T2CH1H ($0034)

TCHxH

	Bit 7	6	5	4	3	2	1	0
Read: Write:	15	14	13	12	11	10	9	8

Adresa v pamäti: T1CH0L ($0027), T1CH1L ($002A)
Adresa v pamäti: T2CH0L ($0032), T2CH1L ($0035)

TCHxL

	Bit 7	6	5	4	3	2	1	0
Read: Write:	7	6	5	4	3	2	1	0

Obr. 3.7: Schéma kanálových registrov časovača TIM

Rovnako ako v prípade stavového a riadiaceho registra TSC, kde sme zablokovali prerušenie po pretečení časovača, tak ani v prípade kanálového stavového a riadiaceho registra nepovolujeme prerušenie po výstupnej komparácií. Neni potreba tieto prerušenia využívať, pretože k ovládaniu servomotorov dochádza na strane užívateľa pohybom joysticku a teda stačí klasický „polling“, kedy pri každej zmene polohy operátora joysticku zapíšeme novú hodnotu šírky pulzu.

Na to, aby sme zabezpečili generovanie PWM signálu, musíme správne nastaviť štvoricu bitov MSxA:MSxB a ELSxB:ELSxA. Pomocou prvých dvoch bitov MSxA:MSxB sme nastavili generovanie PWM signálu bez využitia vyrovnávacej pamäte. Princíp tvorby pulzne-šírkovo modulovaného signálu bez vyrovnávacej pamäte bol vysvetlený v sekcii 2.3.4.

Za pomoci ďalších dvoch bitov ELSxB:ELSxA sme zaručili, aby po výstupnej komparácií prešiel výstup do logickej 0. Posledná vec, ktorú musíme zabezpečiť, je preklopenie hodnoty na výstupe kanála po pretečení časovača, k čomu nám slúži bit TOVx.

Takto dostávame plne využitelný PWM signál na vývodoch mikrokontroléra a stačí už len meniť hodnotu v kanálových registroch časovača podľa toho, aký dlhý pulzný priebeh potrebujeme. Hodnoty v kanálových registroch môžeme meniť v intervale od *1228* po *6142*, ako sme uviedli v sekcii 3.2.

3.5 Softvérové generovanie PWM

V sekcii 1.4 sme si vysvetlili, čo to vlastne ten PWM signál je, jeho matematický princíp a spôsoby hardverového generovania. Naskytuje sa však otázka, či je možné PWM signál vytvoriť aj softvérovou emuláciou. Odpoveď na túto otázku nám už čiastočne zodpovedá princíp generovania PWM v mikrokontroléroch.

Veľké množstvo mikrokontrolérov v dnešnej dobe obsahuje v sebe časovací modul, ktorý slúži na generovanie PWM signálu. Využíva k tomu digitálny princíp generovania vysvetlený v sekcii 1.4.2. Čítač/časovač počíta takty hodinového signálu a po každom pretečení začne počítať odznova a nastaví výstup do vysokej hladiny. Ak čítač dosiahne určitú referenčnú hodnotu, nastaví sa opäť úroveň signálu do nízkej hladiny a až do ďalšieho pretečenia čítača ostáva úroveň signálu na tejto hladine. Podobný prístup by sa dal využiť aj pri softvérovo generovanom pulznom priebehu.

Ak sa teda pozrieme na PWM signál z iného uhla pohľadu, zistíme, že ide vlastne o digitálny priebeh, ktorý si po určitú časovú dobu drží vysokú hladinu a po inú časovú dobu nízku hladinu signálu. Vysokú úroveň signálu pritom môžeme reprezentovať hodnotou logickej 1 a nízku úroveň hodnotou logickej 0.

Pri jednom z našich servomotorov som sa rozhodol využiť tieto poznatky a vyskúšať ovládanie serva pomocou emulovaného PWM signálu. Za výber padlo servo ovládajúce otváranie resp. zatváranie klieští. Už sme spomenuli, že vstup tohto serva bol pripojený na PTD2 (bit 2 portu D). Nastavenie vývodu tohto portu ako výstupu nám umožňuje vysielať naň hodnoty logickej 1 a logickej 0. Týmto spôsobom by sme mali možnosť generovať PWM signál na výstupe, ostáva už len vyriešiť otázku, ako udržať tieto hodnoty na vývode portu po konkrétnu časovú dobu.

Z toho dôvodu bola implementovaná rutina `rob13_Delay(unsigned int delay)`, ktorá vytvára oneskorovaciu slučku v hlavnom toku programu. Na to, aby sme vedeli generovať impulzy v intervale 0,5ms až 2,5ms musíme vedieť, aký dlhý oneskorovací cyklus vytvoriť. K tomu je však nutné vedieť, koľko taktov procesora je potrebných k vykonaniu jedného takéhoto cyklu.

Prostredie Freescale CodeWarrior umožňuje previesť zdrojový kód z jazyka C spätne do assembleru s tým, že ku každej inštrukcii doplní aj číslo reprezentujúce počet cyklov procesora potrebných na jeho vykonanie. Spočítaním týchto hodnôt dostávame celkový počet cyklov

potrebných na vykonanie tejto rutiny. Ak ponásobíme toto číslo dobou trvanie jedného cyklu procesora, dostaneme celkový čas potrebný na vykonanie oneskorovacej rutiny.

$$t = 4{,}07 * 10^{-4} * N \ [ms]$$

t – celkový čas trvania

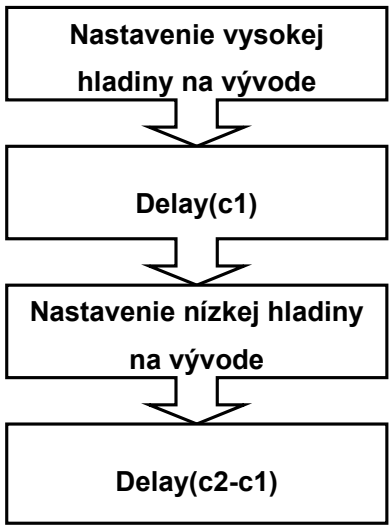
N – počet cyklov

Keď už je známy čas *t,* vieme si určiť počet cyklov potrebných pre vytvorenie požadovaného oneskorenia.

$$PC = \frac{\text{požadovaný čas oneskorenia [ms]}}{t}$$

PC – počet cyklov pre oneskorovaciu slučku

Z experimentálnych pokusov sa však zistilo, že tieto hodnoty nie sú zďaleka vždy presné, a preto bolo treba aj v našom prípade zvyšovať počet cyklov oneskorovacej slučky rádovo až o desiatky pre správny chod servomotora. Nasledujúci diagram zobrazuje spôsob softvérového generovania PWM signálu.



c1 – cyklus potrebný pre generovanie požadovanej šírky pulzu

c2 – cyklus potrebný pre generovanie požadovanej periódy

Obr. 3.8: Blokový diagram algoritmu SW generovaného PWM signálu

3.6 Nastavenie registrov AD prevodníka

Analógovo-digitálny prevodník mikrokontroléra MC68HC908LJ12 pozostáva z dvoch 8-bitových a jedného 16-bitového registra, ktorými sú:

- Stavový a riadiaci register ADSCR
- Register pre časovanie ADCLK
- Dátový register ADRH:ADLR

Na inicializáciu registrov AD prevodníka slúži rutina `rob13_Initialize_ADC()`. Pre pochopenie správneho nastavenia AD prevodníka si najprv treba vysvetliť význam jednotlivých registrov, aké bity obsahujú tieto registre a akú funkciu plnia.

3.6.1 Stavový a riadiaci register ADSCR

Prvá vec, ktorú by sme mali mať na pamäti ohľadom registra ADSCR je, že zápis do tohto registra pozastaví aktuálne prebiehajúcu konverziu a iniciuje novú. Preto treba vždy pred zápisom do registra ADSCR prečítať najprv aktuálnu hodnotu v dátovom registri ADR.

Adresa v pamäti: ADSCR ($003C)

	Bit 7	6	5	4	3	2	1	0
Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
Write:								

Obr. 3.9: Stavový a riadiaci register ADSCR

COCO – Bit dokončenia konverzie

Tento bit slúži iba na čítanie a je nastavený po každom dokončení AD prevodu. Nuluje sa po zápise do registra ADSCR alebo po zápise do registra ADCLK príp. po prečítaní obsahu registra ADRL. Bit COCO je nastavený na logickú 0, ak je nastavený bit AIEN = 1. Reset nuluje bit COCO.

1 = prevod dokončený (AIEN = 0)

0 = prevod nedokončený (AIEN = 0) alebo povolené prerušenie z AD (AIEN = 1)

AIEN – Povolenie prerušenia z AD prevodníka

Keď je bit AIEN = 1, AD prevodník generuje prerušenie po každom dokončení prevodu. Signál prerušenia je nulovaný po zápise do registra ADSCR. Reset nuluje bit AIEN.

1 = povolenie prerušenia z AD

0 = zakázanie prerušenia z AD

ADCO – Bit nepretržitého AD prevodu

Ak je bit ADCO = 1, AD prevodník vzorkuje vstupný signál nepretržite a na konci prevodu aktualizuje hodnotu v registri ADR. Reset nuluje bit ADCO.

1 = nepretržitý AD prevod

0 = jeden AD prevod

ADCH[4:0] – Výber kanála AD prevodu

Pomocou týchto 5-tich bitov vieme vybrať jeden kanál, na ktorom bude prebiehať AD prevod. V nasledujúcej tabuľke vidíme spôsob, ako zvoliť konkrétny kanál AD prevodníka.

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	ADC kanál	Výstup kanála
0	0	0	0	0	ADC0	PTA4
0	0	0	0	1	ADC1	PTA5
0	0	0	1	0	ADC2	PTA6
0	0	0	1	1	ADC3	PTA7
0	0	1	0	0	ADC4	PTB6
0	0	1	0	1	ADC5	PTB7
0	0	1	1	0	ADC6	1,2V ref. napätie
0 ↓ 1	0 ↓ 1	1 ↓ 1	1 ↓ 0	1 ↓ 0	ADC7 ↓ ADC28	Rezervované
1	1	1	0	1	ADC29	V_{REFH}
1	1	1	1	0	ADC30	V_{REFL}
1	1	1	1	1	ADC vypnuté	

V prípade AD prevodníka sme povolili prerušenie po dokončení AD prevodu, pretože z hľadiska implementácie bolo výhodnejšie a elegantnejšie oddeliť túto časť od hlavného toku programu a ošetriť obslužnú rutinu prerušenia namiesto programovej kontroly vlajky ukončenia AD prevodu. Nebolo potrebné využiť ani funkciu nepretržitého AD prevodu, pre naše potreby stačila aj jedna vzorka po dokončení konverzie.

Ostávalo už len vyriešiť možnosť, ako používať kanály AD prevodníka kváziparalelne, keďže potenciometre joysticku boli pripojené na 2 kanály ADC0 a ADC1 a obslužná rutina prerušenia dokáže spracovať naraz len jeden kanál prevodníka. Touto problematikou ako aj významom použitia AD prevodníka k manipulácií s robotom sa zaoberá sekcia 3.6 o implementácií ovládania.

3.6.2 Register pre časovanie ADCLK

Register ADCLK slúži na nastavenie frekvencie AD prevodu a spôsobu uloženia výsledku v dátovom registri ADR.

Adresa v pamäti: ADCLK ($003F)

	Bit 7	6	5	4	3	2	1	0
Read:	ADIV2	ADIV1	ADIV0	ADICLK	MODE1	MODE0	0	0
Write:								R

Obr. 3.10: Register pre časovanie ADCLK

ADIV[2:0] – Bity preddeličky AD prevodníka

Trojica bitov slúži na nastavenie deliaceho pomeru vstupného kmitočtu AD prevodníka. Tato hodnota musí byť v intervale od 32kHz až 2MHz. Nasledovná tabuľka zobrazuje možné prípady konfigurácie.

ADIV2	ADIV1	ADIV0	Deliaci pomer
0	0	0	ADC kmitočet ÷ 1
0	0	1	ADC kmitočet ÷ 2
0	1	0	ADC kmitočet ÷ 4
0	1	1	ADC kmitočet ÷ 8
1	x	x	ADC kmitočet ÷ 16

ADICLK – Bit výberu zdroja hodinového signálu

Bit ADICLK slúži na výber zdroja hodinového signálu, ktorým môže byť buď frekvencia zbernice f_{BUS} alebo CGMXCLK. Pri resete sa nastaví CGMXCLK ako zdroj hodinového signálu.

1 = zdrojom vstupného kmitočtu je f_{BUS}

0 = externý signál CGMXCLK

MODE[1:0] – Režim zarovnania výsledku

Táto dvojica bitov slúži na výber zo štyroch možných režimov uloženia výsledku AD prevodu. V tabuľke môžme vidieť spôsob výberu jednotlivých režimov.

MODE1	MODE0	Režim uloženia výsledku
0	0	Orezanie výsledku na 8 bitov
0	1	Zarovnanie výsledku doprava
1	0	Zarovnanie výsledku doľava
1	1	Znamienkové zarovnanie doľava

Pri nastavení registra ADCLK sme opäť museli vychádzať z frekvencie zbernice, ktorú sme zvolili ako zdroj hodinového signálu v bite ADICLK. Vieme, že frekvencia AD prevodu musí byť v rozmedzí od 32kHz do 2MHz. Frekvencia zbernice je f_{BUS} = 2,4576MHz. Táto frekvencia je vyššia ako povolená horná hranica, preto sme ju podelili dvomi, čím sa frekvencia AD prevodu dostala do požadovaného intervalu.

V sekcii 2.4.4 bol vysvetlený spôsob výpočtu doby konverzie a počet taktov hodinového signálu pre jeden AD prevod.

$$\text{čas konverzie} = \frac{16 \text{ až } 17}{2{,}4576\text{MHz} \div 2} = 13\,\mu\text{s až } 13{,}8\,\mu\text{s}$$

$$\text{cykly hodinového signálu} = 13\mu\text{s} \times 2{,}4576\text{MHz} = 32 \text{ až } 34 \text{ cyklov}$$

Z týchto údajov vyplýva, že jeden prevod bude trvať 13 až 13,8μs a na vykonanie bude treba 32 až 34 cyklov procesora.

Výsledok AD prevodu sa ukladá do dátových registrov v režime *zarovnania výsledku doprava*, pri ktorom dostávame presnejší 10-bitový výsledok. Pri 8-bitovom orezaní výsledku boli totiž hodnoty velmi nestále, čo vykazovalo silnú vibráciu servomotorov.

3.6.3 Dátový register ADRH:ADRL

Dátový register ADR pozostáva z dvoch 8-bitových registrov ADRH (vyšší byte) a ADRL (nižší byte). Do tohto registra sa ukladá 10-bitový výsledok prevodu podľa požadovaného režimu uloženia. V sekcii 2.4.5 sme si už vysvetlili spôsob uloženia výsledku v jednotlivých režimoch.

Adresa v pamäti: ADRH ($003D)

	Bit 7	6	5	4	3	2	1	0
Read:	0	0	0	0	0	0	0	0
Write:	R	R	R	R	R	R	R	R

Adresa v pamäti: ADRL ($003E)

	Bit 7	6	5	4	3	2	1	0
Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
Write:	R	R	R	R	R	R	R	R

Obr. 3.11: Dátové registre ADRH a ADRL

3.7 Ovládanie robotického manipulátora

V doterajších sekciách 3. kapitoly boli uvedené prvotné postupy návrhu ovládania robota, ktoré zahŕňali správnu inicializáciu časovacieho subsystému, spôsob výpočtu cyklov časovača potrebných pre korektný PWM signál , metódu softvérového generovania PWM a správnu inicializáciu AD prevodníka. Nevyriešenými otázkami ostávajú:

- ako zaručiť permanentné generovanie PWM
- ako bude analógový signál z joysticku po AD prevode reprezentovať zmenu šírky pulzu
- ako súčasne obslúžiť 2 kanály AD prevodníka
- v ktorej chvíli zapísať novú hodnotu šírky pulzu
- kedy testovať stlačenie tlačítok joysticku

Pre správny chod servomotorov musíme permanentne vysielať na signálový vodič PWM signál z mikrokontroléra. Tento stav sme docielili tým, že sme hlavný program nechali bežať v nekonečnom cykle. Chod programu je možné zastaviť potom len hardvérovým resetom.

```
void main(){
      /*
        ...
      */
      for(;;){
      }
}
```

Kľúčovými rutinami aplikácie sú:

- `void rob13_SetVerticalPosition()` – nastavenie Y-ovej osi prvého serva ramena
- `void rob13_SetHorizontalPosition()` – nastavenie X-ovej osi druhého serva ramena
- `void rob13_SetJawsPosition()` – nastavenie pozície serva klieští
- `interrupt 17 void rob13_Manipulate()` – obslužná rutina prerušenia z AD prevodníka

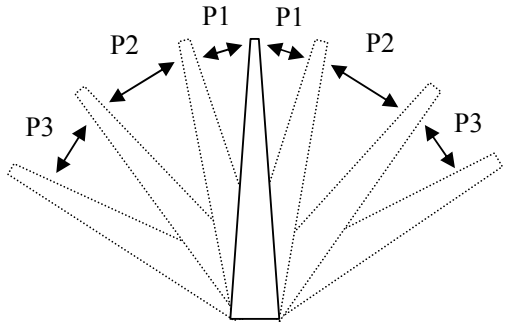
Najdôležitejšiu časť aplikácie tvorí obslužná rutina prerušenia po dokončení AD prevodu. Aby sme mohli využívať podsystém prerušení, musíme v programe zaviesť makro `EnableInterrupts`, ktoré volá inštrukciu assembleru `cli` (clear interrupt mask) a povolí tým globálnu masku prerušení. V tejto rutine dochádza nielen k výpočtu novej hodnoty šírky pulzu, ale aj k samotnému nastaveniu servomotorov do požadovanej polohy. Takto máme istotu, že každá zaznamenaná hodnota prevodu ovplyvní zmenu polohy serva ak je to potrebné. Keby sme ponechali zápis šírky pulzu na hlavný tok programu, mohlo by dôjsť k tomu nežiadúcemu javu, že sa prerušenie zavolá skôr, než sa stihne zapísať stará hodnota na nastavenie serva.

V sekcii 3.6.2 sme si povedali, že kvôli presnejšiemu AD prevodu budeme ukladať hodnoty v režime10-bitového zarovnania výsledku doprava. Aj napriek tomu sa môže vyskytnúť prípad, že hodnoty nebudú presné, ale budú sa skokovo meniť. Tento jav je potrebné zamedziť, a z toho dôvodu sme si vytvorili jednoduchý digitálny filter.

```
#define FILTER_SAMPLES     32
/*
  ...
*/
unsigned int dFilter[FILTER_SAMPLES];
```

Ako vidíme, filter je realizovaný ako jednoduché pole s počtom prvkov definovaných v makre `FILTER_SAMPLES`. Hodnoty poľa sú typu `unsigned int`. Úlohou tohto filtra je zobrať určitý počet výsledkov AD prevodu a urobiť z nich priemer. Tak dosiahneme vyššiu presnosť výsledných hodnôt.

Konečne sa dostávame k hlavnej otázke, a tou je ako vplývajú hodnoty AD prevodu na zmenu šírky pulzu PWM signálu. Pre ľahšie pochopenie si to zobrazme na obrázku.



P1 – interval, v ktorom sa servo nehýbe vôbec

P2 – interval, v ktorom sa servo pohybuje pomaly

P3 – interval, v ktorom sa servo pohybuje rýchlejšie

Obr. 3.12: Spôsob ovládania šírky PWM signálu pákou joysticku

Pohybom páky joysticku meníme napätie na vstupe AD prevodníka. Interval P1 z obr. 3.12 ukazuje interval, v ktorom sa nemení šírka pulzu, teda v tomto intervale sa nezapisuje žiadna hodnota do kanálových registrov časovača. Keď sa pákou dostaneme do intervalu P2, čo znamená, že máme na vstupe AD kanála napätie z tohto intervalu, začne sa zvyšovať alebo znižovať (podľa smeru pohybu páky) hodnota v kanálovom registri o zvolenú konštantu. Veľkosť tejto konštanty ovplyvňuje rýchlosť otáčania servomotora, preto sme zvolili menšie číslo, aby sme vedeli rameno pohodlne nastaviť na zvolenú pozíciu. Natočením páky do intervalu P3 dostávame vyššiu rýchlosť otáčania, pretože zvyšujeme/znižujeme hodnotu v kanálovom registri časovača o väčšiu hodnotu. Tento postup platí ekvivalentne pre obe osi otáčania ramena.

Pre nastavenie polohy serva ovládajúceho kliešte bol použitý softvérovo emulovaný PWM signál vysvetlený v sekcii 3.5. Kliešte sa ovládajú tlačítkami joysticku, pričom tlačítko bližšie k palcu slúži na otváranie a tlačítko bližšie k ukazováku na zatváranie klieští. Tlačítko otvárania je pripojené na port PTD1 a tlačítko zatvárania na port PTD0. Aby tieto porty správne fungovali, musíme ich nastaviť ako vstupné. Na to slúžia registre nastavenia smeru toku dát (data direction registers). Pre nastavenie vývodu ako vstup musíme zapísať na príslušný bit portu logickú nulu.

```
DDRD_DDRD0 = 0;
DDRD_DDRD1 = 0;
```

Tlačítka treba pri zopnutom stave testovať na logickú nulu, pretože spínajú zem. Musíme však mať na zreteli ešte jeden nežiadúci efekt, ktorými sú *zákmity*. Tento jav vzniká kvôli tomu, že prechod z pokojovej polohy alebo do pokojovej polohy nie je nikdy ideálny skok, ale k ustáleniu dochádza postupne. Zákmity sa dajú odstrániť jednoduchým spôsobom, kedy otestujeme tlačítko na zopnutý stav, počkáme nejakú dobu a znova otestujeme, či je tlačítko stále zopnuté. Na vytvorenie čakacej doby nám opäť prišla vhod naša rutina `rob13_Delay(unsigned int delay)`. Oneskorenie by malo byť dlhé asi 50ms, v našom prípade sme však zvolili omnoho kratšiu dobu, aby sme zbytočne nespomaľovali chod serva. Ak teda dôjde k stlačeniu tlačítka, zvýšime resp. znížime šírku pulzu o konštantnú hodnotu a začneme generovať softvérový PWM signál na vstup serva, ktorým je vývod PTD2. Ten musí byť nastavený ako výstup, čo dosiahneme zápisom logickej 1 do registra nastavenia smeru toku dát.

```
DDRD_DDRD2 = 1;
```

Na konci obslužnej rutiny musíme znovu povoliť lokálnu masku prerušení v registri ADSCR. Keďže táto rutina dokáže spracovať naraz len jeden kanál AD prevodníka, museli sme prepínať medzi kanálmi ADC0 a ADC1, aby sme dosiahli súbežné spracovanie prerušení z oboch kanálov.

Záver

Cieľom bakalárskej práce bolo navrhnúť a implementovať algoritmus na ovládania servomotorov robotického manipulátora pomocou joysticku s použitím mikrokontroléra HC08 od firmy Motorola. Pri jej tvorbe som narazil na rôzne komplikácie hardvérového aj softvérového charakteru, no napokon sa podarilo splniť všetky vytýčené ciele.

Návrh ovládania si žiadal vymyslieť taký spôsob, aby bol čo najviac užívateľsky pohodlný a aby sa pomocou joysticku dala jednoducho nastaviť požadovaná pozícia ramena. Prvá verzia implementácie ovládacieho mechanizmu nebola vhodná, pretože rameno nebolo možné udržať na stanovenom mieste. Bolo to zapríčinené tým, že každá hodnota AD prevodu mala vplyv na polohu servomotorov. Tým pádom bolo ťažké a takmer nemožné udržať rameno na požadovanej pozícií.

Druhá verzia ovládania priniesla vylepšenie v nastavovaní polohy vďaka tomu, že k pohybu ramena mohlo dôjsť len pri súčasnom stlačení oboch tlačítok joysticku a tak sme už mali možnosť zabezpečiť stabilnú polohu. Avšak ani tento spôsob ovládania nepriniesol komfort v ovládaní, naopak sa javil ako dosť nepraktický.

Tretia verzia (viď kapitola 3.7) priniesla do mechanizmu ovládania to, vďaka čomu sa stala manipulácia s robotom jednoduchou, intuitívnou a pohodlnou činnosťou. Principiálna schéma je znázornená na obrázku 3.12. Výhodou tohto prístupu je hlavne to, že v oblasti strednej polohy joysticku nedochádza k zápisu novej šírky pulzu. Užívateľ má potom možnosť jednoduchého nastavenia polohy tak, že pohne joystickom na niektorú stranu a keď už je rameno na požadovanej pozícií, vráti sa s joystickom späť do strednej polohy.

Naša aplikácia žial nezaručuje správne nastavenie polohy servomotorov pre iný druh joysticku ako ten, na ktorý bol dimenzovaný. Je to spôsobené tým, že každý joystick obsahuje potenciometre s rôznou rezistivitou, čím dostávame iné hodnoty po AD prevode.

Robotické manipulátory ako je tento majú dnes široké uplatnenie pri montážnych činnostiach, lakovaní, v logistike a v iných odvetviach priemyslu, kde nachádzajú svoje miesto v automatizovanej činnosti.

Ďalší vývoj nášho robotického manipulátora by si žiadal zopár konštrukčných zmien. Velmi perspektívnou myšlienkou sa mi javí napr. mobilizácia robotického manipulátora. Tým by sme dosiahli, že by robot nevykonával činnosť staticky na jednom mieste, ale dokázal by sa presúvať na väčšie vzdialenosti a prenášať pritom predmety. Neskôr pridaním senzorov, hmatových čidiel alebo kamery by sa dal zdokonaliť robot natoľko, že by túto činnosť dokázal vykonávať samostatne bez zásahu človeka. Automatizovaná činnosť by zahŕňala schopnosť robota vyhnúť sa prekážkam alebo ísť po vyznačenej trase.

Literatura

[1] SCHWARZ, J., RŮŽIČKA, R., STRNADEL, J.: *Studijní opora k předmetu IMP*. 2006

[2] Motorola: Katalógové listy MC68HC908LJ12. 2002, [Online, navštívené 1. 5. 2007].
URL http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC68HC908LJ12.pdf

[3] Wikipedia: Pulse-width modulation. 2007, [Online, navštívené 1. 5. 2007].
URL http://en.wikipedia.org/wiki/Pulse-width_modulation

[4] Wikipedia: Joystick. 2007, [Online, navštívené 1. 5. 2007].
URL http://en.wikipedia.org/wiki/Joystick

[5] RAMPON, A.: PWM Generation with the HC08 Timer. 2004, [Online, navštívené 1. 5. 2007].
URL http://www.freescale.com/files/microcontrollers/doc/app_note/AN2701.pdf

[6] BARR, M.: Introduction to Pulse Width Modulation (PWM) Modulation. 2001, [Online, navštívené 1. 5. 2007]
URL http://www.netrino.com/Publications/Glossary/PWM.php

[7] BELZA, J.: PC Joystick. 2002, [Online, navštívené 1. 5. 2007]
URL http://www.belza.cz/pcfan/pcjoys.htm

[8] COVARRUBIAS, L.: Programming the Analog-to-Digital Converter on M68HC08 Microcontrollers. 2005, [Online, navštívené 1. 5. 2007].
URL http://www.freescale.com/files/microcontrollers/doc/app_note/AN2984.pdf