VŠB – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
KATEDRA KYBERNETIKY A BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

SIMULÁTOR PRO VÝUKU VIRTUÁLNÍ INSTRUMENTACE

SIMULATOR FOR VIRTUAL INSTRUMENTATION COURSES

2014 MAREK ZIKA

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Marek Zika**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2601R004 Měřicí a řídicí technika

Téma: Simulátor pro výuku virtuální instrumentace
Simulator for Virtual Instrumentation Courses

Zásady pro vypracování:

1. Seznámení se s technologií základních měřicích přístrojů.
2. Rozbor možností komunikace s měřicími přístroji.
3. Popis architektury mikrokontroléru Freescale Kinetis L.
4. Návrh simulátoru a funkcionality.
5. Realizace softwaru pro mikrokontrolér.
6. Testování simulátoru a zhodnocení výsledků práce.
7. Zpracování návodu pro práci se simulátorem.

Seznam doporučené odborné literatury:

[1] VAN SICKLE, Ted. *Programming microcontrollers in C.* 2nd ed. Eagle Rock, Calif.: LLH Technology Pub., c2001, xvi, 454 p. ISBN 1-878707-57-4.
[2] ZÁHLAVA, Vít. *Návrh a konstrukce desek plošných spojů: principy a pravidla praktického návrhu.* 1. vyd. Praha: BEN - technická literatura, 2010, 123 s. ISBN 978-80-7300-266-4.
[3] *Elektrotechnická měření.* 1. vyd. Praha: BEN - technická literatura, 2002, 255 s. ISBN 978-80-7300-0, EAN 9788073000226.
[4] HRDINA, Zdeněk a František VEJRAŽKA. *Signály a soustavy.* Vyd. 1. Praha: ČVUT, 1998, 234 s. ISBN 80-01-01726-5.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Stankuš**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta:

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

Dne 7.5.2014 Marek Zika.....

Poděkování

Rád bych poděkoval Ing. Martinu Stankušovi za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Tato bakalářská práce se zabývá návrhem a realizací simulátoru pro výuku virtuální instrumentace. Tento simulátor je zkonstruován na bázi mikrokontroléru, který obstarává hlavní funkčnost celého zařízení a je naprogramován tak, aby uměl vykonávat funkce napětím řízeného oscilátoru, napětím řízeného PWM, simulátoru servomotoru a generátoru impulzů. Částí této práce je i vytvoření řídícího programu, který komunikuje s mikrokontrolérem pomocí sériové linky a umožňuje parametrizaci a nastavení funkce simulátoru.

Klíčová slova

Mikrokontrolér, Processor Expert, napětím řízený oscilátor, simulátor, sériová komunikace, servomotor, PWM.

Abstract

This bechelor thesis describes the design and implementation of simulator for the teaching of virtual instrumentation courses. This simulator is based on microcontroller, which performs the main functionality of the device and is programmed to be able to perform the functions of a voltage-controlled oscillator, voltage controlled PWM, the simulator of servo motor and a pulse generator. Part of this work is the creation of a application that communicates with a microcontroller via a serial line and allows the configuration and settings of the simulator.

Key words

Microcontroller, Processor Expert, voltage-controlled oscilator, simulator, seriál communication, servo motor, PWM.

Seznam použitých zkratek a symbolů

PE – Processor Expert
UART – Universal Asynchronous Receiver/Transmitter - univerzální asynchronní přijímač/vysílač
A/D – analogově/digitální
D/A – digitálně/analogový
NI – National Instruments
TPM – Timer/PWM module – čítač/časovač
I/O – input/output – vstup/výstup
PWM – pulse width modulation – pulzně šířková modulace
USB – universal serial bus – univerzální sériová sběrnice
ELVIS - Educational Laboratory Virtual Instrumentation Suite - výukový virtuální laboratorní systém
PC – personal computer – osobní počítač
MSB – most significant bit – bit s nejvyšší hodnotou
LSB – least significant bit – bit s nejnižší hodnotou
FIFO – first in, first out – paměť typu fronta

Obsah

1. Úvod

Bakalářská práce se zabývá návrhem simulátoru pro výuku kurzů virtuální instrumentace. Tento simulátor je postaven na bázi mikrokontroléru od společnosti Freescale.

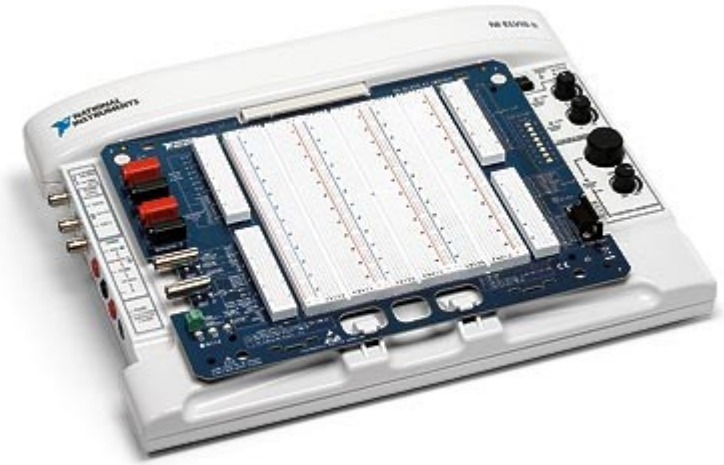
V těchto kurzech pracují studenti na výukové platformě firmy National Instruments s označením ELVIS. Součástí této platformy je i nepájivé kontaktní pole. Do tohoto kontaktního pole je přes vstupní piny zapojen samotný mikrokontrolér. Toto zapojení pak následně umožňuje studentům připojovat vstupní analogové či digitální signály a na výstupech z mikrokontroléru pak měřit jednotlivé výstupní signály podle definované funkce simulátoru.

Simulátor bude tedy umět vykonávat následující funkce. Za prvé půjde o napětím řízený oscilátor. U této funkce pak bude možné definovat minimální a maximální periodu oscilátoru, dále pak tvar výstupního signálu a to konkrétně sinus, pila a obdélník a půjde definovat tvar charakteristiky závislosti vstupního napětí na periodě a to jestli bude lineární nebo exponenciální. Další možnou funkcí je napětím řízené PWM, simulátor servomotoru a nakonec generátor impulzů.

Samotná práce je tedy složena z hardwarové a softwarové části, kde hardwarová část práce se zabývá návrhem propojení měřící karty s mikrokontrolérem a softwarová část se zabývá návrhem programu mikrokontroléru, který obstarává hlavní funkčnost celého simulátoru. Dalším bodem softwarové části je i vytvoření řídícího programu, který slouží k ovládání simulátoru. Tento program pak běží na PC, ke kterému je mikrokontrolér též připojen.

2. Měřicí přístroje

Část simulátoru, která zajišťuje měření analogových či digitálních signálu poskytovaného jeho funkcí je NI ELVIS obr. 2.1.



obr. 2.1 NI ELVIS [1]

NI ELVIS II Series (National Instrument Educational Laboratory Virtual Instrumentation Suite II Series) je výukový virtuální laboratorní systém pro navrhování a testování elektronických obvodů. Základem je odnímatelný modul s nepájivým polem, na kterém lze navrhovat a testovat různá elektronické zapojení. Na modulu je 8 analogových vstupů, 2 analogové výstupy, programovatelné funkční vstupy/výstupy, 24 digitálních vstupů/výstupů, 8 uživatelsky definovatelných I/O.

3. Mikrokontrolér KL25Z

Jak již tedy bylo zmíněno v úvodu této práce, funkčnost simulátoru bude zprostředkována pomocí mikrokontroléru. Konkrétně byl vybrán mikrokontrolér společnosti Freescale s označením KL25Z. Základ mikrokontroléru tvoří ARM procesor o maximální frekvenci 48 MHz. Mikrokontrolér dále obsahuje řadu periferií, kde pro tuto práci jsou stěžejní následující periferie. Jedná se konkrétně o analogově digitální převodník dále pak digitální vstupy a výstupy, digitálně analogový převodník, čítače/časovače a UART. Jednotlivé zmíněné periferie pak budou detailněji popsány v kapitolách níže.

Obr. 3.1. Použitý vývojový kit Freescale FRDM-KL25Z [2]

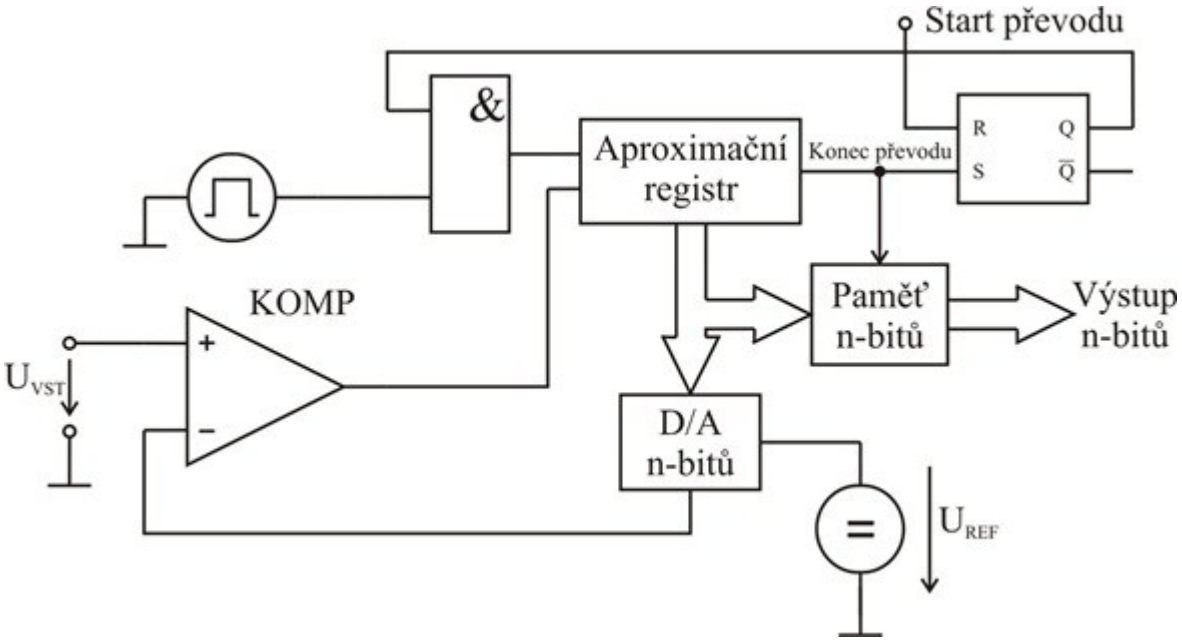
3.1. Periferie

3.1.1. A/D převodník

Jednou z dostupných periferií tohoto mikrokontroléru, které je v této práci využívána je A/D převodník, který slouží k měření vstupního analogového (spojitého) signálu a jeho následné převedení do digitální podoby se kterou pak může mikrokontrolér ve svém programu pracovat. Tento A/D převodník má pak následující vlastnosti:

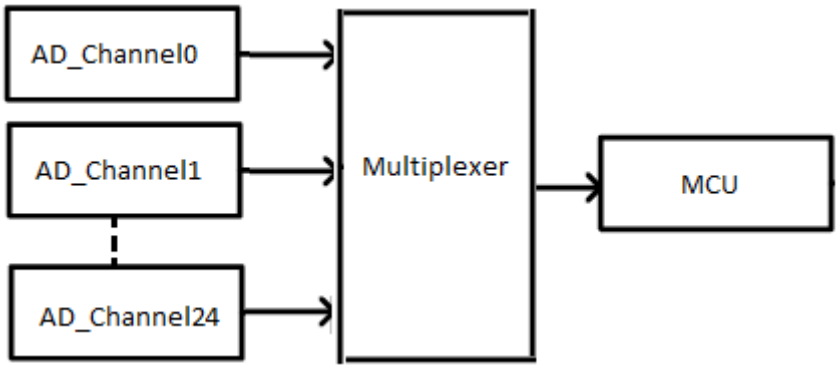
- Rozlišení až 16 bitů, to znamená, že může pracovat i v režimech s nižším rozlišením jako jsou režimy 14, 12 a 8 Bitů.
- Možnost připojení až 24 tzv. single ended vstupů kde je měřeno vstupní napětí na jednom vodiči vůči zemi anebo možnost připojení až 4 diferenčních převodníků, kdy je převáděn rozdíl dvou napětí přivedené na vstupní vodiče.
- Rozsah vstupního napětí 0-3,3 V.

Konkrétně se jedná o aproximační převodník, jehož princip spočívá v tom, že se nastaví jednotlivé váhové bity a převod se provádí postupně, od nejvyššího MSB bitu směrem k nejnižšímu LSB bitu metodou půlení intervalu. Na začátku cyklu převodu se nastaví hodnota převodu výstupu aproximačního registru na 10000000. Této hodnotě pak odpovídá polovina referenčního napětí. Hodnota tohoto napětí je posléze porovnána se vstupním napětím. Je-li vstupní napětí větší než referenční napětí nechá se hodnota MSB bitu v aproximačním registru nastavena na 1, pokud ne nastaví se do nuly. V dalším cyklu se nastaví nižší váhový bit opět do 1 a porovná se příslušná napěťová úroveň referenčního napětí se vstupním napětím a podle toho tento bit zůstane v 1, nebo se nastaví do 0. Takto převod pokračuje až k LSB. Princip převodníku je zobrazen na obrázku 3.2. [3]



obr. 3.2. Princip A/D převodníku [3]

Jak již bylo tedy řečeno, tento A/D převodník umožňuje připojení až 24 tzv. „single ended“ vstupů. Tyto vstupy se též označují kanály. Z důvodu připojení tolika vstupů k jednomu samostatnému A/D převodníku je nutno tyto vstupy multiplexovat. Multiplexování znamená, že je rychle přepínáno mezi jednotlivými kanály a tím je možné připojit k jednomu fyzickému A/D převodníku více vstupů. Toto uspořádáni je pak zobrazeno na obrázku 3.2.



obr. 3.2. Princip multiplexování A/D převodníku

3.1.2. D/A převodník

Další používanou periferií je pak D/A převodník, který jak už z názvu vypovídá, slouží k převedení digitální hodnoty na analogovou (spojitou). Tento konkrétní D/A převodník má rozlišení 12 bit a v práci je používán ke generování výstupních signálu a tvaru sinus a pila.

3.1.3. Časovač

Důležitou periferií jsou pak časovače. Kdy mikrokontrolér obsahuje tři TPM moduly s několika kanály, které mohou pracovat v režimech input capture, output compare a PWM. Tyto jednotlivé režimy jsou pak popsány níže.

Input capture:

Pomocí funkce input capture může časovač zachytit okamžik, kdy dojde k vnější události. Okamžik je zaznamenán v případě detekce hrany na vstup časovače. V tomto módu je i možné generovat přerušení. Tento mód pak může zachytit tyto okamžiky:

- Detekce vzestupné hrany
- Detekce sestupné hrany
- Detekce vzestupné i sestupné hrany

Output compare:

Funkce output compare umožňuje generovat časové pulzy s programovatelnou pozicí, polaritou, trváním a frekvencí. Když čítač dosáhne požadované hodnoty v příslušném kanálu, tak časovač provede na výstupu z časovače jednu z následujících akcí:

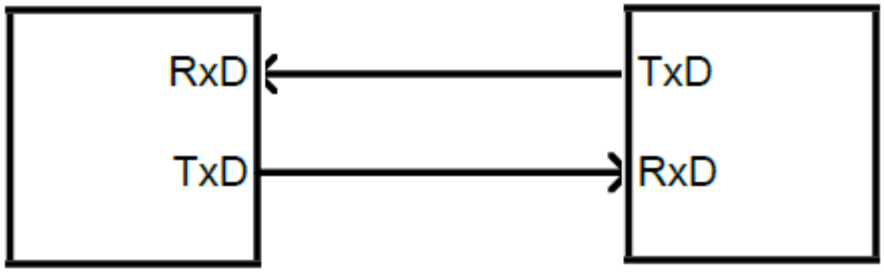
- Překlopení výstupu
- Nastavení logické nuly na výstupu čítače/časovače
- Nastavení logické jedničky na výstupu čítače/časovače

PWM:

Funkce PWM je speciální případ režimu output compare, který umožňuje generování výstupního obdélníkového signálu s programovatelnou frekvencí a střídou. [4]

3.1.4. UART

Další využívanou periferií je pak UART. Ten slouží ke komunikaci po sériové lince mezi PC a mikrokontrolérem. Kde sériová komunikace je sekvenční proces přenosu dat, kdy data jsou posílána po jednotlivých bitech pomocí komunikačního kanálu.



obr. 3.4. Sériová komunikace

Zkratka UART znamená v češtině univerzální asynchronní přijímač a vysílač. Jelikož se jedná o asynchronní přenos dat tak jak samotný vysílač, tak i přijímač musí pracovat na stejné frekvenci.

Přenosová rychlost:

Z nutnosti práce přijímače i vysílače pracovat na stejné frekvenci se zde zavádí pojem přenosové rychlosti.

Přenosová rychlost udává počet změn za sekundu a nese název podle francouzského vynálezce a matematika J.M.E. Baudota. Jelikož u sériových rozhraní jsou změny signálu v časové oblasti stále stejné tak je přenosová rychlost (bps) rovna počtu přenesených bitů za sekundu.

3.1.5. Vstupně výstupní digitální porty

Tento mikrokontrolér obsahuje čtyři vstupně výstupní porty. Jedná se o paralelní porty, tzn., že každý jednotlivý port je tvořen více piny. Jak už teda vyplývá ze samotného názvu, tak každý pin může pracovat ve dvou režimech jak ve vstupním tak ve výstupním, ale nikdy ne v obou režimech najednou. Každý pin pak může zastávat nějakou alternativní funkci. Jedná se o:

- Vstupy pro jednotlivé kanály A/D převodníku.
- Výstup D/A převodníku
- Vstupy a výstupy pro komunikaci po sériové lince
- Vstupy a výstupy časovačů

U jednotlivých pinů, je pak možno ve vstupním režimu připojit interní pull up rezistor. Kde když není k pinu nic připojeno, je na jeho vstupu hodnota logické jedničky a k zapsání logické nuly dochází jeho uzemněním.

3.2. Processor Expert

K naprogramování samotného mikrokontroléru je přímo využíváno vývojového prostředí od společnosti Freescale s názvem CodeWarrior a jejich softwarovým nástrojem nazvaným Processor Expert.

Hlavním rysem systému Processor Expert je sjednocený přístup k návrhu, který uživatele odstiňuje od hardwaru daného embedded systému. Tento přístup je založen na použití komponent. Processor Expert obsahuje základní sadu těchto komponent, které zapouzdřují funkčnost některého z mikrokontroléru. To umožňuje, aby byl uživatel při návrhu aplikace ušetřen nutnosti studovat odlišnosti jednotlivých procesorů.

Použití Processor Expertu pak přináší mnoho výhod, které jsou využívány i v této práci a jedná se zejména o tyto následující:

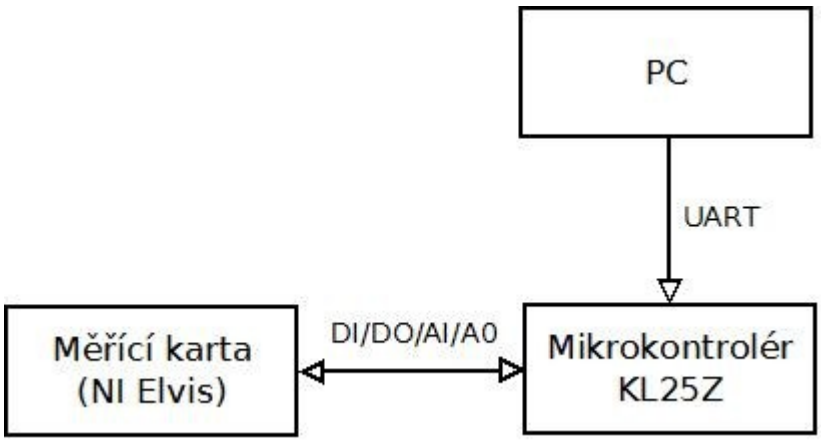
- Díky tomuto speciálnímu nástroji je možný rychlý vývoj aplikací pro mikrokontroléry bez nutnosti detailní znalosti jejich vnitřní architektury a s tím spojené nastavování systémových registrů.

- Jako další výhoda je pak znovu-použitelnost vytvořené aplikace i pro jiné mikrokontroléry, než na které byla aplikace primárně vyvíjena

- A jako jedna z hlavních výhod je pak to, že s vygenerovaným kódem dostáváme k dispozici metody (funkce), které nám pak následně umožňují práci s jednotlivými periferiemi.

4. Návrh simulátoru a funkcionality

Samotný simulátor bude pracovat tak, že mikrokontrolér, který bude obstarávat funkčnost celého zařízení, který bude umět vykonávat funkce napětím řízeného oscilátoru, napětím řízeného PWM, simulátoru servomotoru a generátoru impulzů. Mikrokontrolér bude připojen do nepájivého pole, které je součástí NI ELVIS. Do tohoto nepájivého pole bude připojen přes precizní piny, které budou napájeny na jeho vstupy. Vstupy a výstupy mikrokontroléru budou pak přes toto nepájivé pole připojeny ke vstupům a výstupům měřící karty, která je k tomuto nepájivému poli připojena. Samotný mikrokontrolér pak bude přes USB kabel připojen k PC, ve kterém poběží řídící aplikace, pomocí které bude možno přes sériovou linku posílat data mikrokontroléru a tím i řídit jeho funkci. Následující kapitola se pak zabývá propojení mikrokontroléru s měřící kartou. Schéma tohoto zapojení je pak zobrazeno na obr. 4.1.

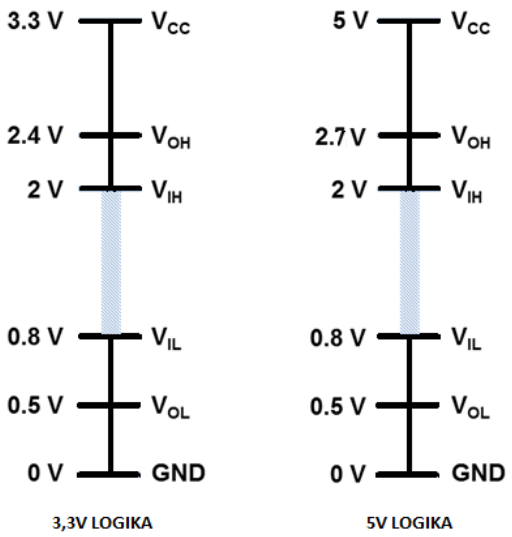


obr. 4.1. Návrh simulátoru

4.1. Návrh zapojení

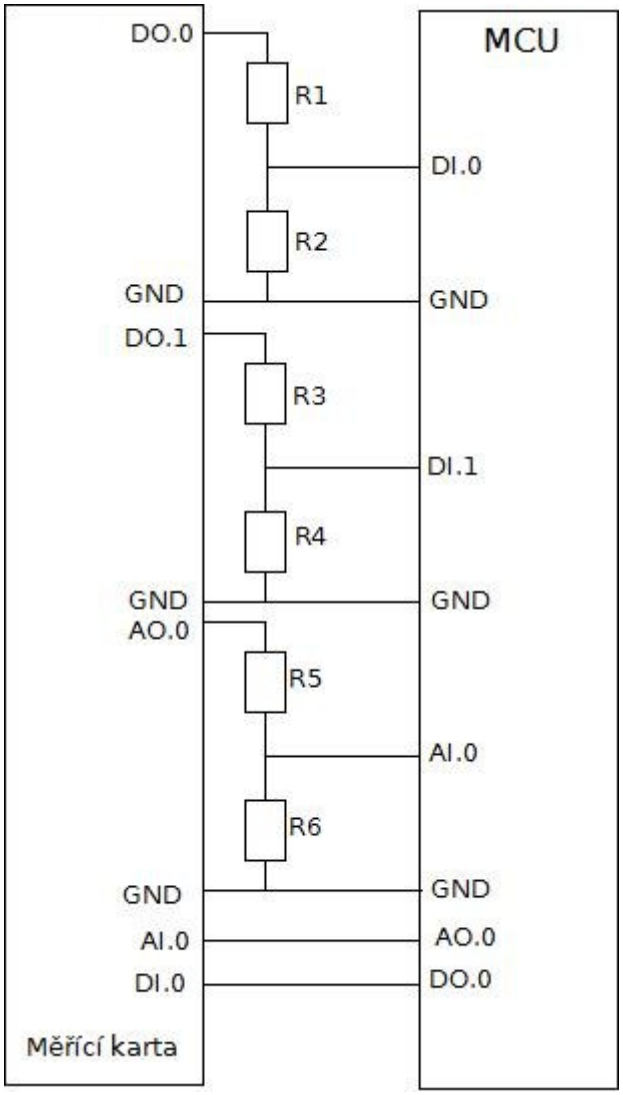
Jelikož mikrokontrolér a měřící karta jsou zařízení, která pracují s rozdílnými napěťovými úrovněmi, je nutné zajistit správné propojení mezi těmito zařízeními, aby se navzájem nemohla zničit.

U digitálních vstupů a výstupů pracuje mikrokontrolér s 3,3V logikou, kdežto měřící karta pracuje s 5V logikou. Na obr. 4.2. jsou pak zobrazeny napěťové úrovně obou logik.



obr. 4.2. Napěťové úrovně 3,3V a 5V logiky

Z obrázku 4.2. je patrné, že při propojení digitálního výstupu měřící karty s digitálním vstupem mikrokontroléru, by mohlo dojít k zničení mikrokontroléru, neboť výstupní napětí měřící karty při stavu logické „1“ může být v rozsahu od 2,7 V do 5V a vstupní napětí mikrokontroléru pro stav logické „1“ je pouze od 2V do 3,3V. Z tohoto důvodu je nutné výstupní napětí z měřící karty snížit na úroveň, se kterou může mikrokontolér pracovat. Toho je docíleno tím, že se mezi výstup měřící karty a vstup mikrokontroléru zapojí napěťový dělič. Tento dělič je nutné zapojit taky mezi analogový výstup z měřící karty a analogový vstup A/D převodníku mikrokontroléru. Samotné propojení mezi měřící kartou a mikrokontrolérem je zobrazeno na obr. 4.3.



obr. 4.3. Propojení mikrokontroléru s měřící kartou

Hodnoty odporů vychází ze vzorce pro napěťový dělič. A byly vybrány tak aby při vstupním 5V napětí byl výstup děliče 3,3V. Proto byly vybrány odpory o hodnotách R_2= 68 kΩ a R_1=33 kΩ. Hodnoty odporů R3 a R4 jsou totožné s odpory R1 a R2.

$$U_2 = U_1 \cdot \frac{R_2}{R_1 + R_2} = 5 \cdot \frac{68000}{33000 + 68000} = 3{,}3V \tag{4.1}$$

U děliče, který je zapojen mezi analogovým výstupem z měřící karty a vstupem A/D převodníku mikrokontroléru byly vybrány hodnoty odporů R_5=33 kΩ a R_6=68 kΩ. Tudíž při maximálním vstupním napětí 10V bude výstupní napětí rovno 3,3V.

$$U_2 = U_1 \cdot \frac{R_5}{R_5 + R_6} = 10 \cdot \frac{33000}{33000 + 68000} = 3{,}3V \tag{4.2}$$

Při propojování digitálních či analogových výstupů z mikrokontroléru se vstupy měřící karty se mohou tyto zařízení připojit napřímo bez jakékoliv úpravy výstupního signálu, neboť u digitálních výstupů 3,3V logiky napěťové úrovně pro výstupní signál dle obr. 4.2 zapadají do vstupního rozsahu 5V logiky. U analogového výstupu mikrokontroléru je rozsah výstupního napětí od 0 do 3,3 V a vstupní rozsah analogového vstupu u měřící karty je od -10V do 10V tudíž i zde nemusí být signál mezi těmito dvěma zařízeními nijak upraven.

5. Realizace softwaru

Tato část BP se zabývá návrhem softwaru pro mikrokontrolér i následné řídící aplikace. Návrh softwaru vychází z požadavků, které funkce má umět simulátor vykonávat. A to konkrétně funkce napětím řízeného oscilátoru, napětím řízeného PWM, simulátoru servomotoru a generátoru impulzů. Program pro mikrokontrolér byl napsán v jazyce C v programovacím prostředí CodeWarrior od firmy Freescale.

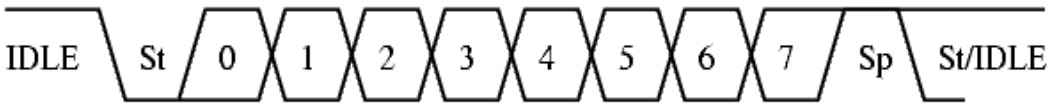
5.1. Mikroprocesorová část

V této části práce bude nejdříve popsána část komunikace po sériové lince, která představuje stěžejní část funkčnosti celého simulátoru, a následně pak budou popsány jednotlivé funkce simulátoru.

5.1.1. Sériová komunikace

Jednou z nejdůležitějších částí této práce byla komunikace po sériové lince, kdy bylo nutno zajistit komunikaci mezi PC a mikrokontrolérem tak, aby bylo možné nastavovat patřičnou funkci kterou má mikrokontrolér vykonávat a s tím i přenést patřičná data, potřebná pro danou funkci.

Komunikace po sériové lince probíhá pomocí rámců (frames). Pokud se nic neděje, tak je linka v klidovém (IDLE) stavu, pro který se používá kladné napětí. Každý rámec začíná start bitem (St), což je změna na nulové napětí na dobu danou rychlostí komunikace (např. pro 9600baud je to 1/9600s, tj. cca 104us). Následují datové bity, kdy logická jednička odpovídá nulovému napětí a logická nula kladnému. Vysílá se od nejméně důležitého bitu (LSB). Celý rámec je zakončen stop bitem (Sp), kdy je linka zase v klidovém, tedy kladném napětí. Po stop bitu může následovat pauza (IDLE) nebo hned start bit (St). [5]



obr. 5.1. Frame

Rámce mohou po sobě hned následovat, takže pokud používáme přenosovou rychlost 9600 baud, tak za 1s můžeme poslat maximálně 9600/10=960 bajtů (číslo 10 odpovídá jednomu start bitu, 8 datovým bitům a jednomu stop bitu).

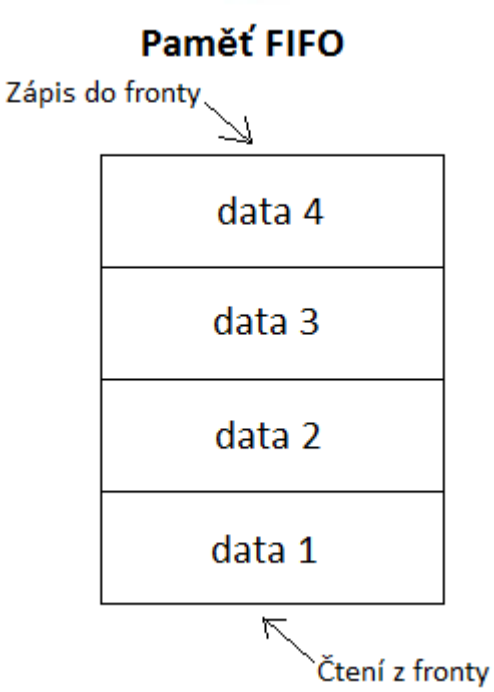
Samotná komunikace byla nastavena následovně:

- 8 datových bitů
- žádný paritní bit
- jeden stop bit

Jelikož během komunikace proudí z PC do mikrokontroléru data v rychlém sledu za sebou je nutno zajistit, aby nebyla příchozí data smazána novými daty dříve, než jsou tyto data zpracována.

Aby tento případ nenastal je v hlavním programu implementována paměť typu fifo, do které jsou data uložena hned při jejich příjmu, a tím je zajištěno, že nedojde ke ztrátě dat při jejich příjmu a jejich následném zpracování.

Paměť typu fifo z anglického „first in first out“ je typ paměti fronta, kde jsou data vyčtena v pořadí, v jakém do fronty přišla. Tento princip je zobrazen na obrázku 5.2.



obr. 5.2. Princip FIFO paměti

Samotná implementace fifo paměti obsahuje 4 proměnné a to samotné pole o velikosti 16 hodnot do kterého jsou příchozí data zapisována a následně počet prvků v paměti, index pozice prvního prvku v paměti a index pozice posledního prvku v paměti. A jsou zde dvě funkce pro práci s pamětí a to funkce pro vložení dat do pamětí a funkce pro vyjmutí dat z paměti.

```
uint8_t fifo_insert(uint8_t data)
{
          if (fifo_cnt == FIFO_SIZE){
                    return 1;
          }
          if (fifo_cnt != 0){
                    fifo_tail++;
                    fifo_tail &= FIFO_MASK;
          }
```

```
            fifo_cnt++;
            fifo[fifo_tail] = data;
            return 0;
}
```

Výpis 1. Funkce pro zápis do paměti FIFO

Při vkládání dat do paměti pomocí funkce fifo_insert jsou této funkci předána data v podobě 8bitového unsigned integeru a pokud je již fifo paměť plná tak je funkce ukončena s návratovou hodnotou 1 což reprezentuje chybný stav. Pokud paměť není zaplněna a počet prvků v paměti je větší než nula zvětší se index pozice posledního prvku v paměti o jedna a následně se tato hodnota tzv. promaskuje. To znamená, že se provede bitový AND hodnoty indexu posledního prvku v poli s prvkem o binární hodnotě 0b00001111. Při této operaci zůstává hodnota pozice posledního prvku nezměněna do doby, dokud nepřesáhne hodnotu 15. Při přesažení této hodnoty je hodnota indexu pozice posledního prvku v paměti vynulována. Touto operací je zajištěno, že jsou data neustále zapisována cyklicky v kruhu. Podmínka kdy ke zvětšení hodnoty indexu posledního prvku dochází pouze, když počet prvku v paměti je větší než nula je zde proto, aby nedocházelo k přepisování dat, protože když je počet dat v paměti nulový není nutno tuto hodnotu zvyšovat a rovnou lze na tuto pozici data zapsat. Na konci funkce pak dochází pouze ke zvětšení počtu prvků v paměti o jedna a následné zapsání dat do paměti. Při úspěšném zapsání pak funkce vrací hodnotu nula.

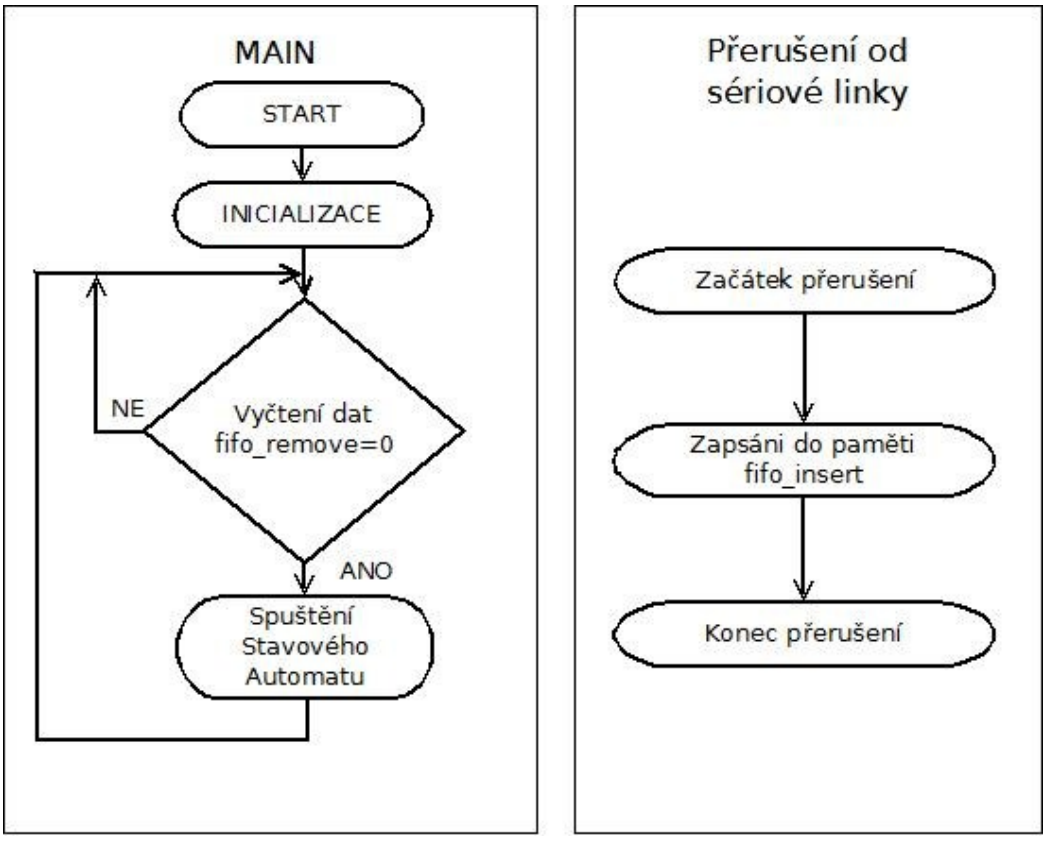
```
uint8_t fifo_remove(volatile uint8_t *data)
{
            if (fifo_cnt == 0){
                        return 1;
            }

            *data = fifo[fifo_head];
            if (fifo_cnt != 1){
                        fifo_head++;
                        fifo_head &= FIFO_MASK;
            }
            fifo_cnt--;
            return 0;
}
```

Výpis 2. Funkce pro čtení z paměti FIFO

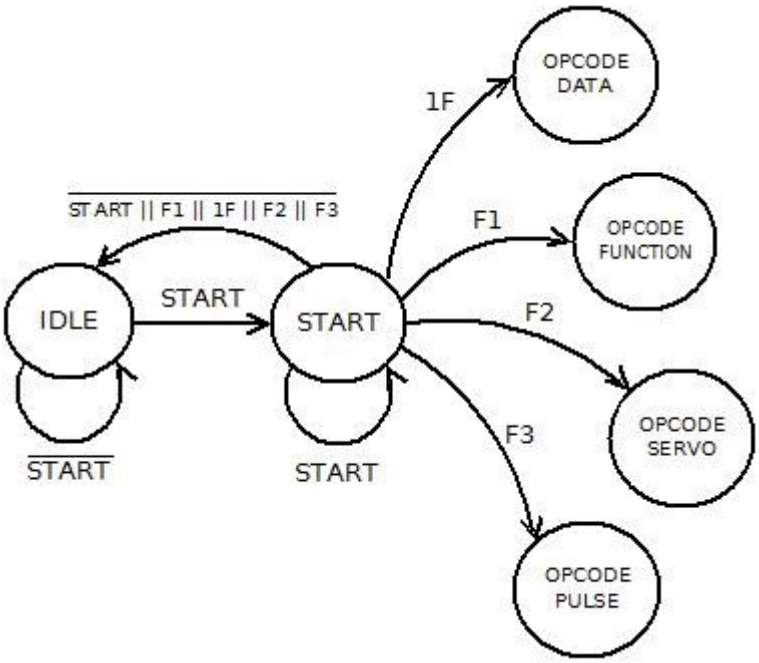
Pro vyčtení dat z paměti slouží funkce fifo_remove. Vstupem této funkce je ukazatel na proměnnou, do které budou data vyčtená z paměti uložena. Pokud paměť neobsahuje žádná data, je funkce ukončena s návratovou hodnotou jedna, což značí chybový stav. Pokud v paměti jsou obsaženy nějaká data tak jsou data zapsána do výstupní proměnné. Následně pokud se počet prvků v poli nerovná jedné dojde ke zvýšení hodnoty indexu prvního prvku v poli o jedna a následnému maskování, které má zde stejnou funkci jako u zapisování do paměti. Podmínka, která toto provede pouze, pokud se počet prvků v poli nerovná jedné je zde z toho důvodu, že pokud je v paměti pouze jeden prvek tak index pozice prvního a posledního prvku v poli mají stejnou hodnotu a proto nesmí být index pozice prvního prvku v paměti inkrementován, neboť by byla jeho hodnota vyšší než hodnota indexu pozice posledního prvku což by byla chyba. Na konci pak dojde ke zmenšení počtů prvků v paměti o jedna a následně dojde k ukončení funkce s návratovou hodnotou rovnou nule.

Data do paměti jsou vždy ukládána při přerušení, které je vyvoláno přijmutím jednoho bytu dat od sériové linky. Vyčítání dat z paměti, běží v programu mikrokontroléru v nekonečné smyčce, kdy pokud dojde k úspěšnému vyčtení dat z paměti, tzn., že návratová hodnota funkce fifo_remove je rovna nule, je v programu spuštěn stavový automat, který zajišťuje správnou interpretaci příchozích dat. Princip je pak zobrazen na obr. 5.3.



obr. 5.3. Princip hlavního programu

Jak již bylo zmíněno samotnou komunikaci zajištuje stavový automat. A to z toho důvodu, aby bylo možné rozlišit, co příslušná příchozí data reprezentují, a aby bylo možno zajistit obstarání chyby v přenosu, kdy příchozí data nepřišla, anebo nejsou zcela kompletní.

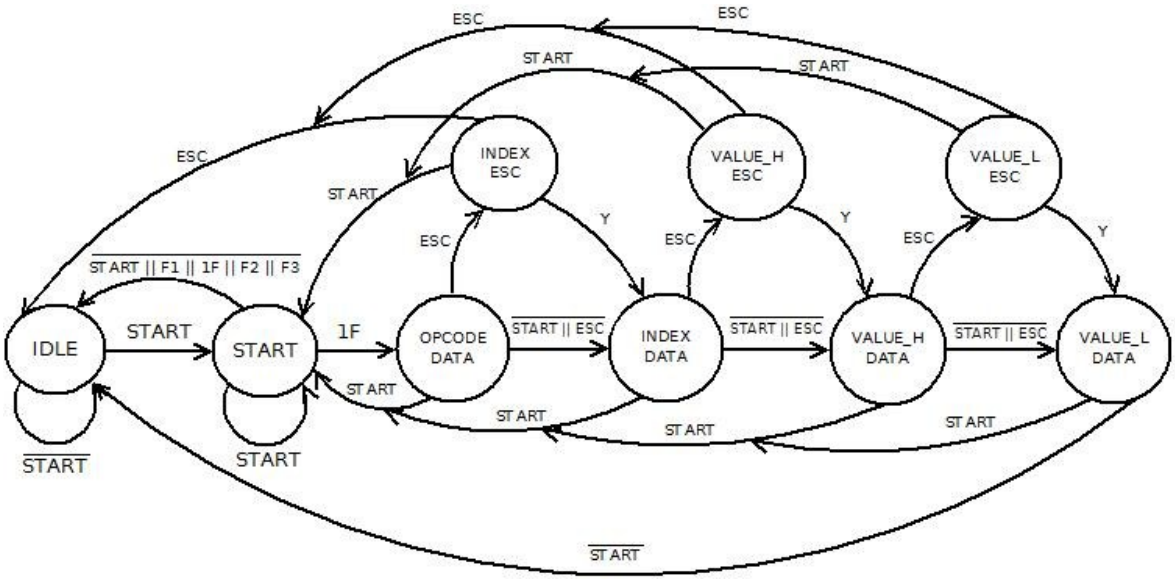


obr. 5.4. Zjednodušený stavový automat pro příjem dat

Na obr. 5.4. je pak zobrazen zjednodušený stavový automat, kde na začátku je automat v IDLE stavu kdy čeká na příchod start bytu ze sériové linky. Tento start byte představuje hexadecimální hodnota 0xFF. Při úspěšném příjmu tohoto startovacího bytu se automat přepne do stavu START, ve kterém čeká na příchod některého z operačních kódů. Operační kódy nabývají následujících hexadecimálních hodnot 0x1F, 0xF1, 0xF2 a 0xF3. Při příjmu některé z těchto hodnot se automat přepne do příslušných stavů, které zajišťují příjem dat, pro jednotlivé funkce simulátoru, anebo pro přepínání mezi funkcemi. Tyto stavy v sobě ukrývají další stavové automaty, které se o jejich správnou funkci starají a budou popsána dále v textu.

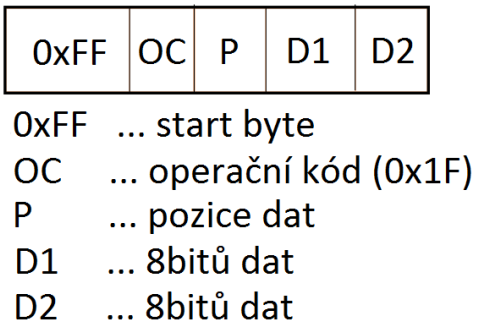
Příjem dat pro look up tabulku:

Většina z funkcí simulátoru potřebuje pro svou správnou funkci mít data uložena v look up tabulce, což je konkrétně pole o velikosti 256 prvků ve kterém jsou obsaženy 16 bitové celočíselné hodnoty. Které mohou například obsahovat hodnoty periody výstupního signálu pro jednotlivá vstupní napětí tak aby je nebylo potřeba za běhu programu počítat, ale stačí je pouze nahrát do paměti.



obr. 5.5. Stavový automat pro příjem dat do look up tabulky

Na obr. 5.5. je pak zobrazen automat, který obstarává příjem těchto dat a jejich následné ukládání do look up tabulky. Data jsou posílána v paketech, kdy každý paket začíná start bytem který má hexadecimální hodnotu 0xFF. Poté následuje byte s operačním kódem, což je v tomto případě 0x1F. Poté následuje byte, který nese hodnotu s indexem místa v look up tabulce do kterého se mají data zapsat. Následně pak po sobě přichází dva byty se samotnými daty. Na obr. 5.6 je pak zobrazena struktura celého paketu.



obr. 5.6. Struktura paketu

Kvůli tomu, že příchozí byty s pozicí a daty mohou obsahovat hodnotu startovací bytu, je nutno zajistit aby docházelo ke správné reprezentaci příchozích dat. To je zajištěno pomocí tzv. escape charakteru, což je byte o hexadecimální hodnotě 0xFE, který zajišťuje to, aby když příchozí data, které mají v sobě obsaženu hodnotu startovacího bytu, nebyly jako tento start byte brány. Toho je docíleno tak, že vždy když data mají v sobě obsahovat start byte nebo escape charakter je vždy při komunikaci nejdříve poslán escape charakter a následně jsou poslány data v podobě, kdy jsou bitově xorovány s hodnotou 0x20 a při jejich příjmu jsou tyto data opět bitově xorována s hodnotou

0x20 tudíž jsou data přijata v jejich původní podobě. Z obrázku se stavovým automatem jsou pak zobrazeny tzv. „ESC“ stavy, které řeší tuto synchronizaci, kdy na vstupech těchto stavů jsou právě escape charaktery a výstupem je pak Y což reprezentuje bitový XOR těchto přijatých dat.

$$Y = přijatá\ data\ XOR\ 0x20$$

Jelikož data, které jsou ukládány do look up tabulky mají podobu 16bit čísla a sériová linka umožňuje posílat data o maximální velikosti 8bit (jeden byte), je nutno tyto data nejprve na straně vysílače (PC) rozdělit a poslat nejdřív prvních 8bitů dat a následně poslat zbylých 8bitů dat a na straně přijímače je pak nutno tyto data opět spojit. K spojení dat na straně přijímače pak slouží následující funkce, které je vždy volána po příjmu dat ve stavu VALUE_L_DATA.

```
void look_up_table_insert(void)
{
      result = 0;
      result = result + value_h;
      result = result << 8;
      look_up_table[indx] = result | value_l;
}
```

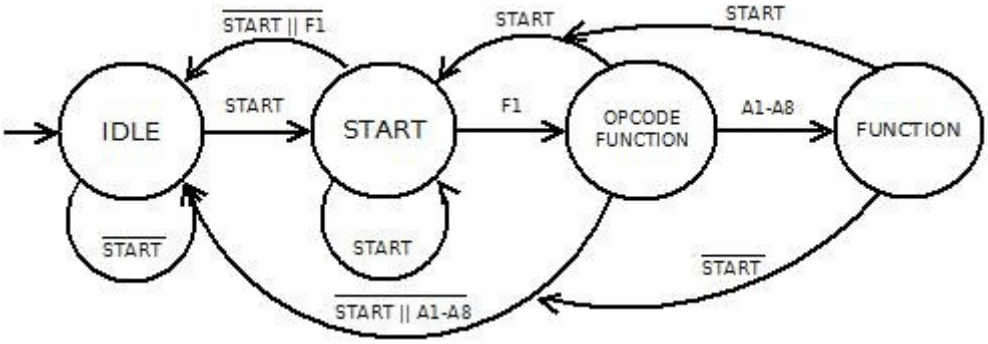
Výpis 3. Funkce zápis dat do look up tabulky

Tato funkce nejdříve byte s vyšší hodnotou přičte k proměnné, která bude obsahovat konečný výsledek a následně je proveden bitový posun doleva o 8 pozic, tudíž tyto data nabudou své původní hodnoty a následně jsou k nim přičtena data s nižší hodnotou a tím jsou získána 16bit data, které byly původně odeslány a jsou následně uloženy do look up tabulky.

Samotný stavový automat pak dále obsahuje ošetření kdy, pokud do jednotlivých stavů vstupují špatná data, tudíž je to bráno jako chybový stav a automat je pak přepnut do IDLE stavu a pokud tyto chybná data obsahují start byte, je automat přepnut do stavu START.

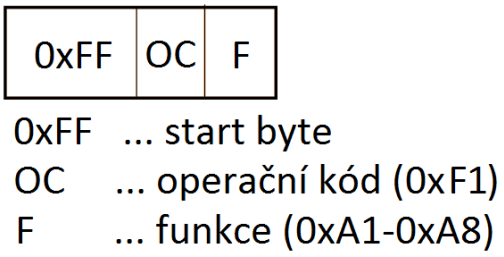
Příjem dat pro výběr funkce:

Jelikož má simulátor několik funkcí, je nutné, aby bylo možné se mezi těmito funkcemi přepínat. K přepnutí pak dochází, pokud po sériové lince přijdu příslušná data k tomu určená. Tuto funkčnost obstarává stavový automat zobrazený na obr. 5.7.



obr. 5.7. Stavový automat pro výběr funkce

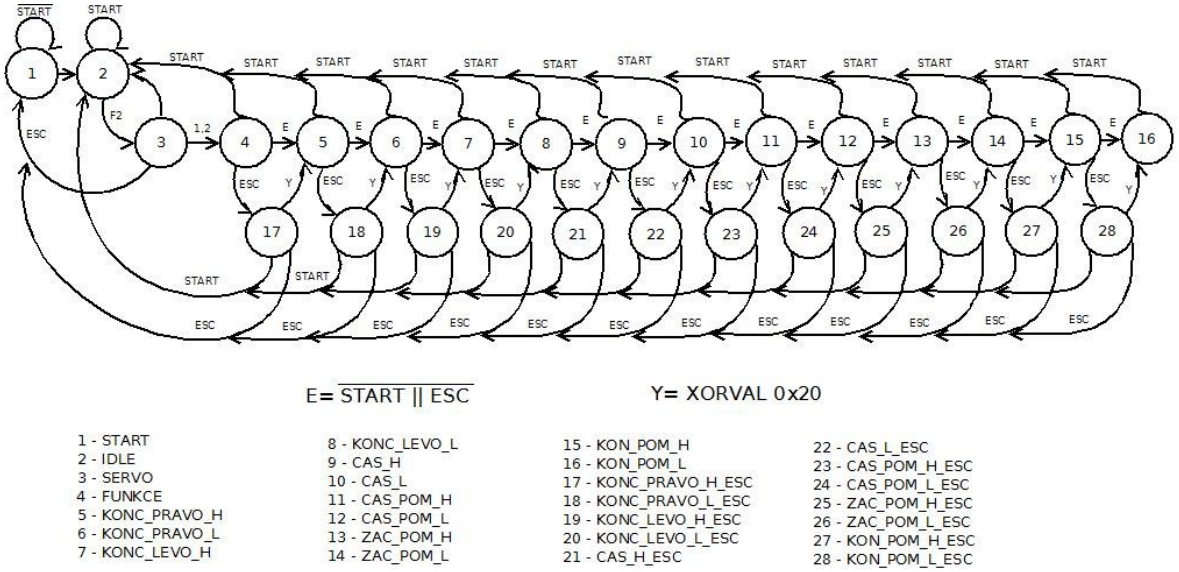
Kde na začátku je automat ve stavu IDLE kde čeká na start byte, který zahájí komunikaci a automat se přepne do stavu START. Po start bytu následuje operační kód, který má v tomto případě hodnotu 0xF1. Pomocí tohoto operačního kódu automat pozná, že bude následovat byte s hodnotou, která udává funkci simulátoru. Po přijetí dat s funkcí simulátoru se automat přepne do stavu FUNCTION, kde v programu mikrokontroléru nastaví patřičnou funkci. Samotná struktura paketu pro nastavení funkce simulátoru je zobrazena na obr. 5.8.



obr. 5.8. Struktura paketu pro nastavení funkce

Příjem dat pro simulátor servomotoru:

Simulátor servomotoru nepotřebuje pro svou správnou funkci data z look up tabulky, ale je zapotřebí pro jeho funkčnost přenést po sériové lince data se sedmi proměnnými kde jedna z těchto proměnných je datového typu bool a zbylých šest proměnných jsou pak 16bit integer. Co jednotlivé proměnné představují a k čemu slouží je pak vysvětleno v kapitole, která se funkcí simulátoru servomotoru zabývá. Na obr. 5.8. je pak zobrazen stavový automat, který zajišťuje, aby byla data přijímána správně.

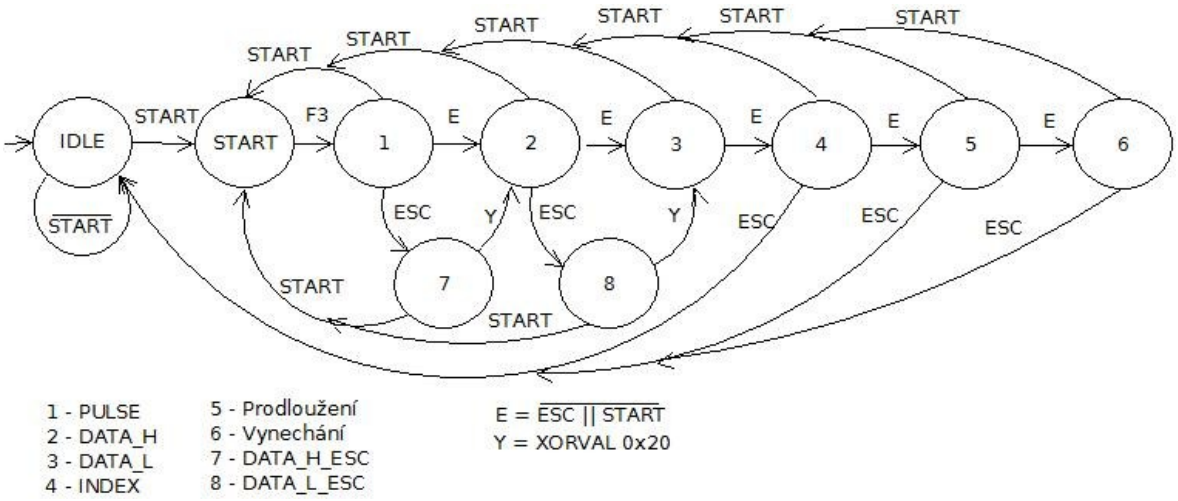


obr. 5.8. Stavový automat pro příjem dat pro funkci simulátoru servomotoru

Automat se na začátku opět nachází v IDLE stavu, kde čeká na příchod start bytu. Po úspěšném přijmutí start bytu se automat přepíná do stavu start, kde čeká na příchod operačního kódu, který zahájí příjem dat pro funkci simulátoru servomotoru a přepne se do stavu SERVO. Operační kód má v tomto případě hodnotu 0xF2. Poté se čeká na přijmutí hodnoty 0x1 nebo 0x2, které reprezentují hodnotu TRUE nebo FALSE pro proměnnou datového typu BOOL. Poté následuje příjem dat s hodnotami zbylých šesti proměnných, kde každá proměnná je rozdělena na dvě části. U tohoto automatu funguje stejný princip s escape charakterem, jako je tomu u příjmu dat u look up tabulky.

Příjem dat pro generátor impulzů:

Pro funkci generátoru impulzů je nutné přenést po sériové lince data se čtyřmi proměnnými, kde jedna proměnná je datové typu 16bit integer, tudíž je poslána ve dvou částech. Zbylé tři proměnné jsou typu 8bit integer. Význam jednotlivých proměnných je vysvětlen v kapitole zabývající se funkcí generátoru impulzů. Na obr. 5.9. je zobrazen stavový automat zajišťující příjem těchto dat.



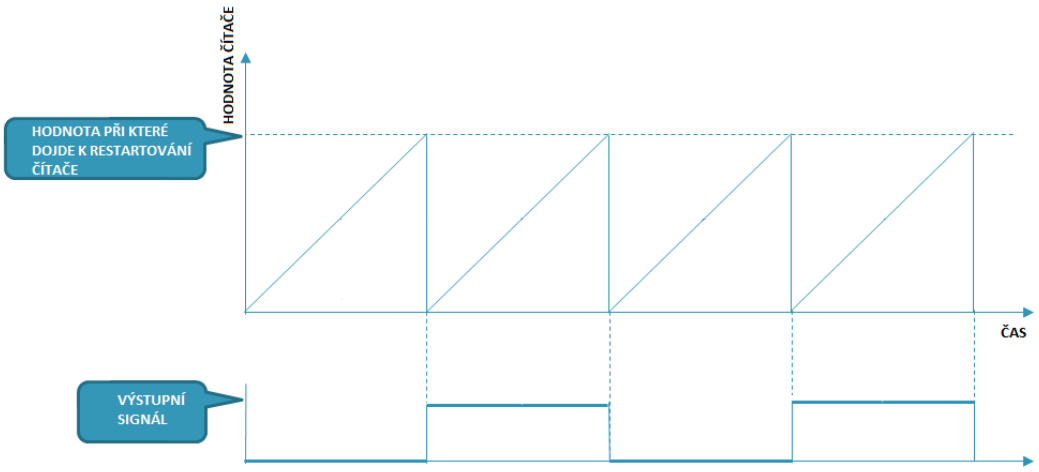
obr. 5.9. Stavový automat pro příjem dat pro funkci generátor impulzů

Princip příjmu dat je pak totožný jako u příjmu dat pro look up tabulku, či pro simulátor servomotoru s tím rozdílem, že u stavů, které slouží pro příjem tří 8bitových proměnných, nejsou escape stavy a to z toho důvodu, že hodnoty těchto proměnných nenabývají hodnot start bytu.

5.1.2. Popis funkcí simulátoru

Napětím řízený oscilátor

První z funkcí, kterou simulátor umí vykonávat je napětím řízený oscilátor, kde výstupní signál má tvar obdélníku se střídou v poměru 1:1 tzn., že během doby cyklu jedné periody je čas, ve kterém je výstup ve stavu logické „1“ stejný jako čas kdy je výstup ve stavu logické „0“. Ke generování signálu s těmito parametry je použit jeden z kanálů, z dostupných TPM modulů, který pracuje v režimu output compare tzn., že když čítač dosáhne požadované hodnoty v tomto kanálu dojde k překlopení hodnoty výstupního signálu viz. obr. 5.10.



obr. 5.10. Princip změny periody

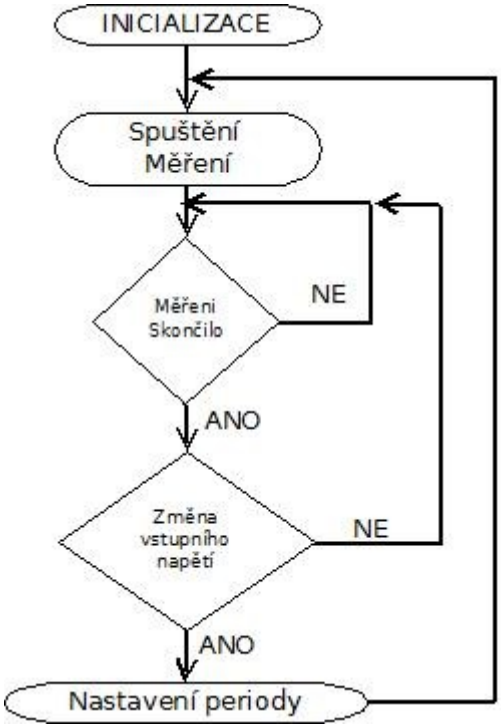
Nastavení požadované hodnoty pro překlopení, tím pádem příslušné periody je provedeno pomocí metody setperiodticks poskytnutou PE. Tato metoda umožnuje nastavení periody v tzv. taktech a jelikož TPM modul je 16 bitový je možné tuto hodnotu nastavit v rozmezí 1-65535. Samotný TMP modul je nastaven na takt 375kHz a z obr. 5.10. lze vidět, že pro vygenerování jedné periody signálu je zapotřebí dvou překlopení. Lze pak generovat výstupní signál o frekvenci v rozmezí od 2,861Hz do 187,5kHz.

$$f_{min} = \frac{375000}{2 \cdot 65535} = 2{,}861Hz \tag{5.1}$$

$$f_{max} = \frac{375000}{2 \cdot 1} = 187{,}5kHz \tag{5.2}$$

Jelikož je možno tento oscilátor parametrizovat tzn., že lze měnit hodnoty minimální a maximální frekvence a také lze měnit tvar závislosti výstupní periody na vstupním napětí a to zda tato závislost bude lineární nebo exponenciální je nutno při každém nastavení nebo změně této funkce, která přichází z řídící aplikace (PC) poslat data po sériové lince tak aby bylo zajištěna správná funkce dle nastavených parametrů. K tomu je potřeba zaslat 256 16bitových hodnot, která jsou uložena v look up tabulce a ve kterých jsou obsaženy hodnoty, při kterých dojde k překlopení výstupu. Jelikož výstupní frekvence signálu je dána vstupním napětím, které je přivedeno na vstup A/D převodníku a tento převodník je nastaven jako 8bitový, tudíž může nabývat hodnot 0-255. Tzn., že tato hodnota z A/D převodníku pouze indexuje pozici v look up tabulce, ve které jsou uloženy hodnoty při kterých dojde k překlopení výstupu. Tímto způsobem je zajištěno, že lze tento oscilátor parametrizovat, jelikož lze hodnoty posílané přes sériovou linku, které jsou uloženy v look up tabulce měnit.

Samotný algoritmus zajišťující funkci tohoto napětím řízeného oscilátoru je zobrazen na obr. 5.11. Kde na při startu dochází nejprve k inicializaci, tzn., že se nastaví parametry A/D převodníku a TMP modulu. Následně pak dochází ke spuštění měření signálu, který je přiveden na vstup A/D převodníku a čeká se, dokud není získán výsledek z měření. Následně se pak provede porovnání hodnoty vstupního napětí z A/D převodníku získaného v tomto a předešlém cyklu a pokud se tyto hodnoty liší, dojde k nastavení nové periody. Toto porovnání je zde z toho důvody, aby nedocházelo ke zbytečnému znovu-nastavování stejné periody, když hodnota vstupního napětí je pořád stejná. Nakonec je pak znovu spuštěno měření a tento cyklus se stále dokola opakuje.



obr. 5.11. Vývojový diagram napětím řízeného oscilátoru

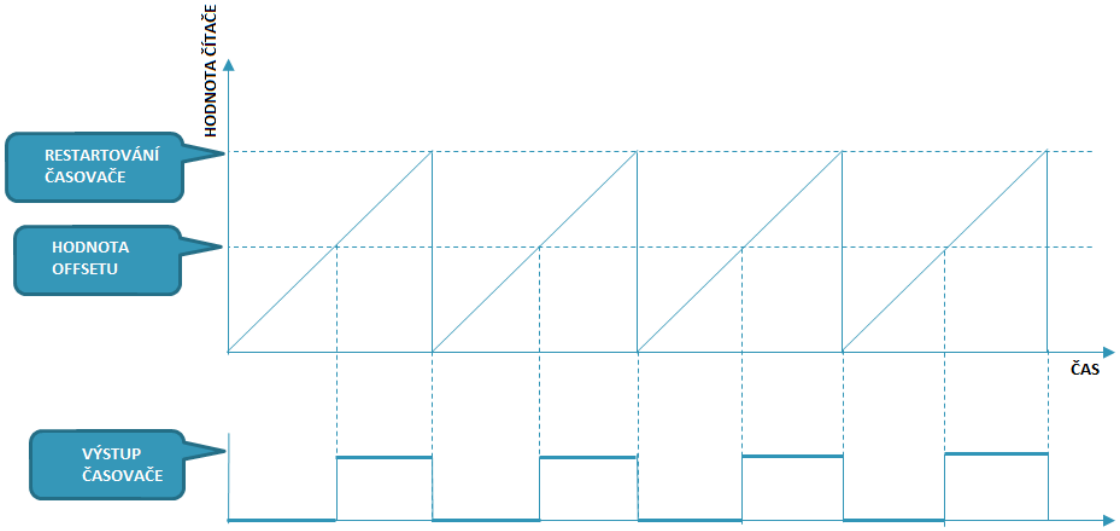
Napětím řízené PWM

Další z funkcí simulátoru je funkce napětím řízeného PWM, kde na základě vstupního napětí se mění střída výstupního obdélníkového signálu, což je poměr času T_H ve kterém je signál ve stavu logické „1“, ku celkové periodě T výstupního signálu. Při vynásobení tohoto poměru stem, se dostane střída v procentech.

$$střída = \frac{T_H}{T} \cdot 100\ [\%; s; s] \tag{5.3}$$

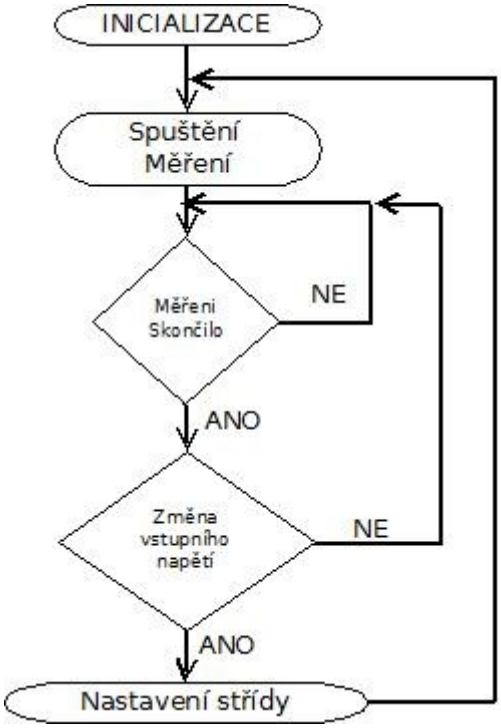
K této funkci jsou použity dvě periferie mikrokontroléru a to A/D převodník pro měření vstupního napětí podle kterého se nastavuje střída výstupního signálu. Ke generování výstupního signálu se pak využívá jednoho z kanálů časovačů, který pracuje v režimu output compare, kdy časovač periodicky čítá od 0 do hodnoty 255 po které dojde k restartování časovače a ten začne celý cyklus čítáni opět od začátku. V tomto režimu se pak nastaví tzv. offset, což je hodnota, při které časovač provede příslušnou operaci na výstupu časovače. V tomto případě je to nastaveno tak, že při restartování časovače se nastaví hodnota výstupu z časovače na logickou nulu, a když čítač dosáhne hodnoty offsetu tak se výstup časovače nastaví do logické jedničky. Tímto způsobem je pak možno dosáhnutí změny střídy výstupního signálu na jednom z kanálu časovače, protože hodnota offsetu lze měnit pomocí funkce setoffsetticks poskytnutou PE. Tudíž na základě hodnoty vstupního napětí z A/D převodníku která může být v rozmezí od 0 do 255 lze přímo pomocí této

hodnoty nastavovat offset tím pádem i střídu neboť i tuto hodnotu lze nastavit v rozmezí od 0 do 255. Princip nastavování střídy je pak zobrazen na obrázku 5.11.



obr. 5.12. Princip změny střídy výstupního signálu

Samotný algoritmus, který vykonává funkci napětím řízeného PWM je vyobrazen na následujícím obrázku 5.13, z něhož je patrné, že samotný algoritmus má velice obdobnou funkci, jak u napětím řízeného oscilátoru tzn., že opět hned na začátku dochází k inicializaci, tedy nastavení patřičných parametrů jednotlivých periferií tedy A/D převodníku a časovače. Posléze je spuštěno samotné měření a čeká se, dokud není získán výsledek měření od A/D převodníku a posléze se zjistí, jestli hodnota vstupního napětí se změnila a podle toho se pak určí, jestli dojde k novému nastavení střídy výstupního signálu.



obr. 5.13. Vývojový diagram napětím řízeného PWM

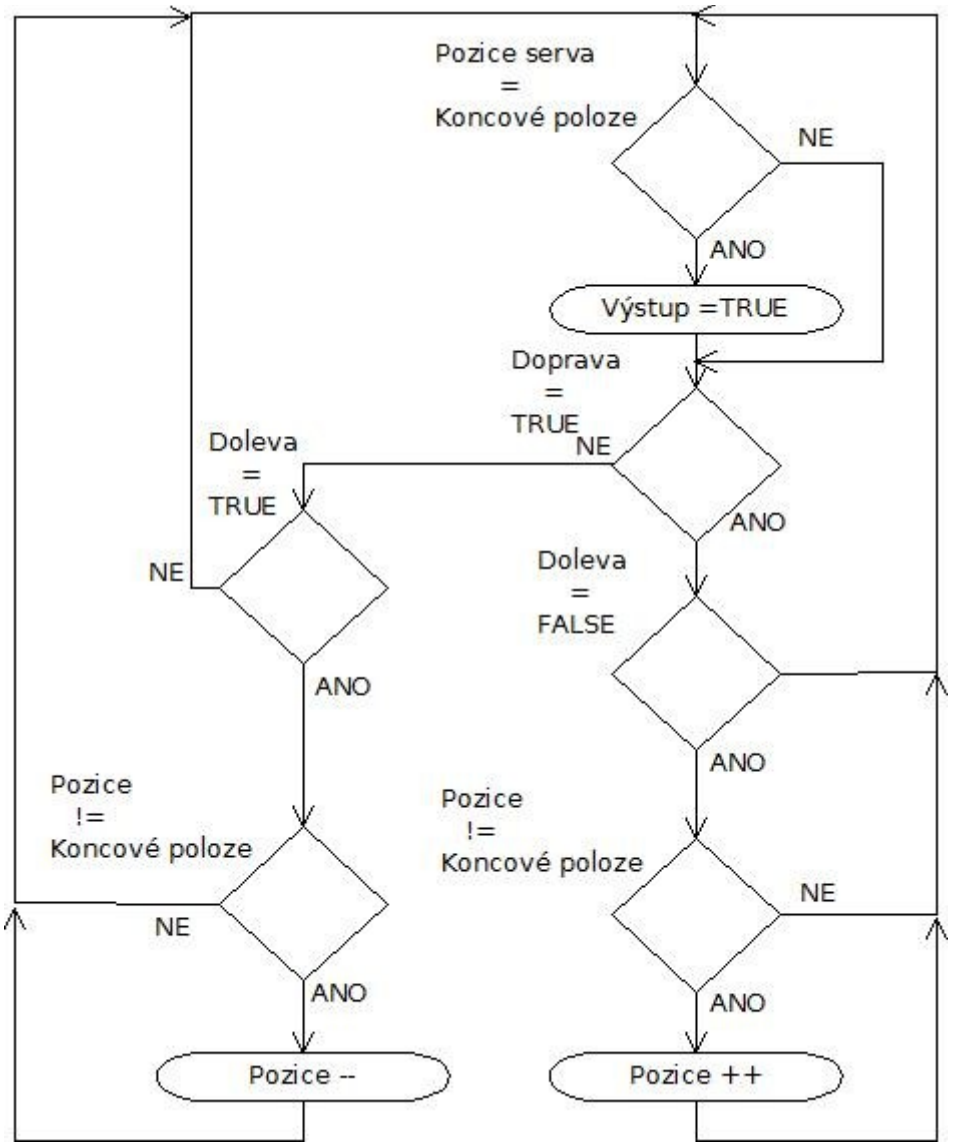
Simulátor servomotoru

Tato funkce simuluje servomotor pro ovládání ventilu topení v automobilu kdy rozsah pohybu tohoto servomotoru je 180°. Pomocí dvou digitálních vstupů mikrokontroléru je poskytována informace o tom zda se má servomotor otáčet doprava či doleva. Digitální výstup mikrokontroléru pak poskytuje informaci, zda se servomotor nachází v koncové poloze. Mikrokontrolér pomocí jeho analogového výstupu poskytuje informaci o poloze servomotoru, kdy rozsah pohybu 0 až 180° odpovídá výstupnímu napětí od 0 do 3,3V. U této funkce lze parametrizovat čas, za jakou dobu projede servomotor z jedné krajní polohy do druhé, polohu koncových spínačů a také část dráhy kterou projede servomotor pomaleji.

Při nastavení této funkce jsou přijata data s proměnnými potřebné pro tuto funkci. A to konkrétně proměnná datového typu bool, která určuje, zda když nejsou koncové pozice nastaveny na 0° a 180° se servomotor zastaví či nikoliv. Dále pak 16bitová celočíselná proměnná, která nese informaci o čase, za kterou má servomotor projet z jedné krajní polohy do druhé. Další dvě proměnné nesou hodnoty o krajních polohách servomotoru. Poslední tři proměnné pak souvisí s funkcí, kdy má servomotor projekt část své dráhy pomaleji, kde jedna obsahuje čas, který bude servomotoru trvat tuto část dráhy projet a zbylé dvě pak konec a začátek této části dráhy kterou projede servomotor pomaleji.

Na obr. 5.14. je pak zobrazen vývojový diagram popisující tuto funkci. Kdy vždy při přerušení, které přichází od jednoho z TMP modulu a nastává vždy po 10µs se inkrementuje hodnota

proměnné a pokud hodnota této proměnné odpovídá času, kdy se má servomotor pohnout je volán program zobrazený na obr. 5.14. Při každém přerušení je též zjištěna logická hodnota digitálních vstupů



obr. 5.14. Vývojový diagram simulátoru servomotoru

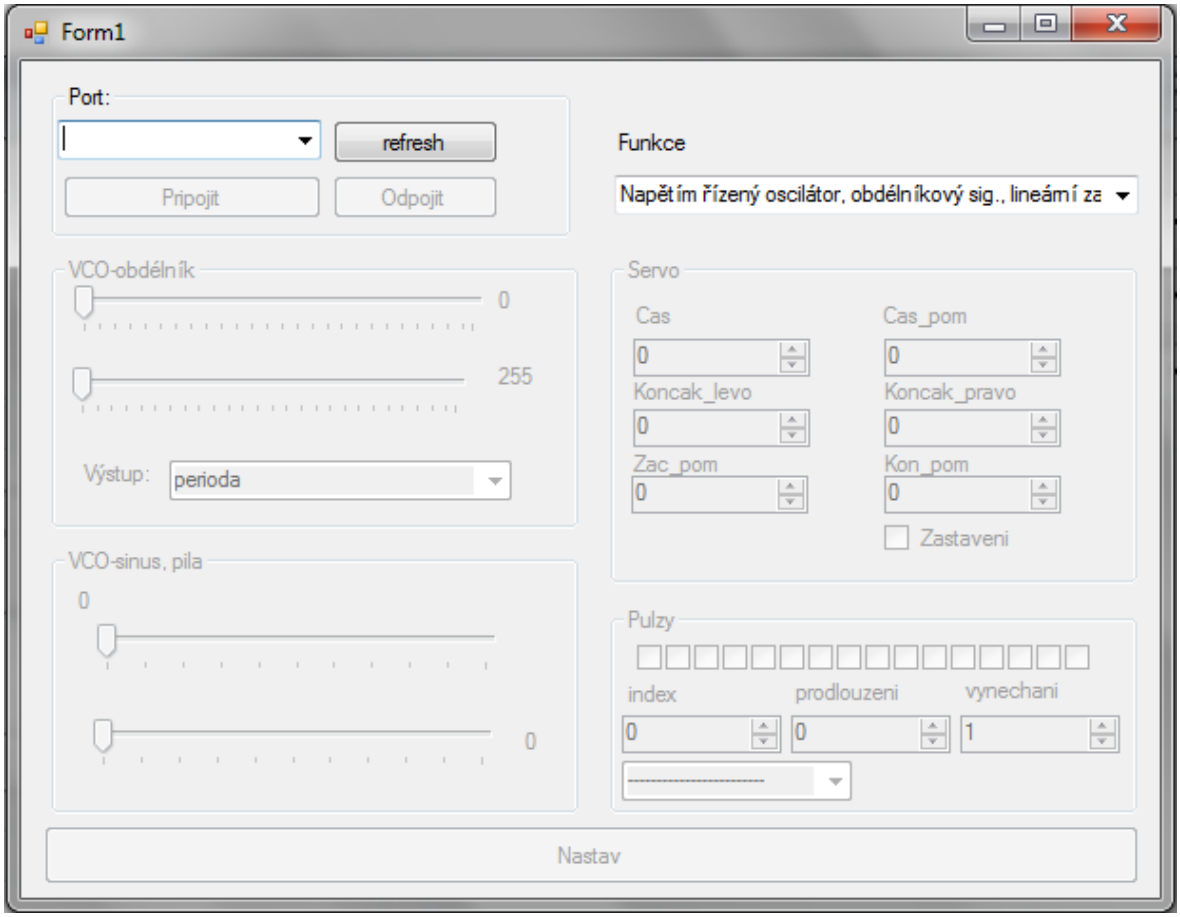
Generátor impulzů

Pro tuto funkci je zapotřebí jednoho čítačového modulu a jeden digitální výstup. Při nastavení této funkce je přijata jedna 16bit proměnná ve které jsou uloženy hodnoty nastaveného sledu impulzů a pomocí níž je vytvořené pole datového typu bool, které slouží pro generování výstupního signálu. Zbylé tři proměnné, které jsou přijaty při nastavení této funkce, slouží k úpravě tohoto pole podle požadavků na vynechání nebo prodloužení části pulzu.

Princip programu pak spočívá v tom, že vždy při přerušení, vyvolané čítačovým modulem se porovná prvek pole s následujícím a pokud se jejich hodnoty nerovnají, dojde k překlopení digitálního výstupu.

5.2. Řídící program

Program pro ovládání simulátoru je napsaná v programovacím jazyce C# a jeho podoba je zobrazena na obr. 5.15. Samotná aplikace teda umožňuje nastavení příslušné funkce simulátoru a její parametrizaci a s tím i spojené nahrání dat do mikrokontroléru, které jsou pro tuto funkci nezbytné.



obr. 5.15. Řídící aplikace

První část programu obstarává připojení se k sériovému portu, ke kterému je mikrokontrolér připojen. K tomu slouží tlačítka připojit a odpojit. Tlačítko refresh pak vypíše seznam dostupných fyzických portů.

Po připojení se k sériovému portu, je uživateli umožněno vybrat jednu z následujících funkcí:

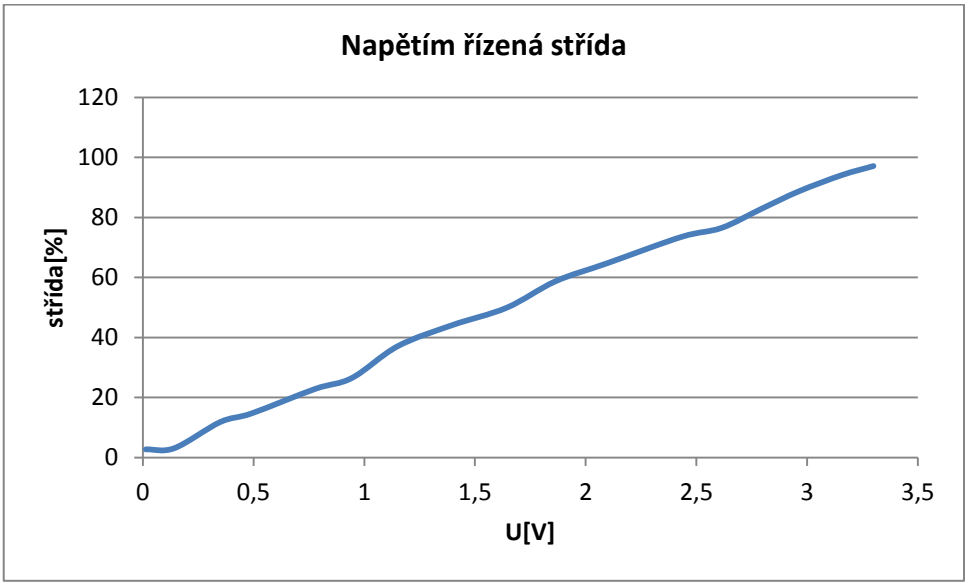
1. Napětím řízený oscilátor pro výstupní obdélníkový signál, kde závislost periody výstupního signálu na vstupním napětí má lineární průběh. U této funkce je možno nastavit minimální a maximální periodu v rozmezí od 5,33µs do 349,52ms.
2. Napětím řízený oscilátor pro výstupní obdélníkový signál, kde závislost periody výstupního signálu na vstupním napětí má exponenciální průběh. A i zde jako v předchozím případě je možné nastavit minimální a maximální periodu v rozmezí od 5,33µs do 349,52ms.
3. Napětím řízené PWM, kde závislost střídy výstupního signálu na vstupním napětí má lineární průběh.
4. Napětím řízené PWM, kde závislost střídy výstupního signálu na vstupním napětí má exponenciální průběh.
5. Simulátor servomotoru, u kterého lze nastavit čas, za který projede servomotor z jedné krajní polohy do druhé. Dále pak polohu koncových spínačů, jestli se servomotor v koncových polohách zastaví nebo ne. A dále je zde možné nastavit, že servomotor projede kus své dráhy jinou rychlostí.
6. Napětím řízený oscilátor pro výstupní signál s tvarem buď sinus, nebo pila, u kterého lze nastavit závislost periody na vstupním napětí, zda bude lineární, nebo exponenciální.
7. Generátor impulzů, u kterého lze nastavit sled definovatelných impulzů, prodloužení některé části impulzů nebo vynechání některé části impulzu.

Samotný program pak obstarává, aby data byla vysílána v daném sledu, ve kterém je mikrokontrolér očekává, a jsou popsány v kapitole, která se zabývá příjmem dat pro jednotlivé funkce simulátoru.

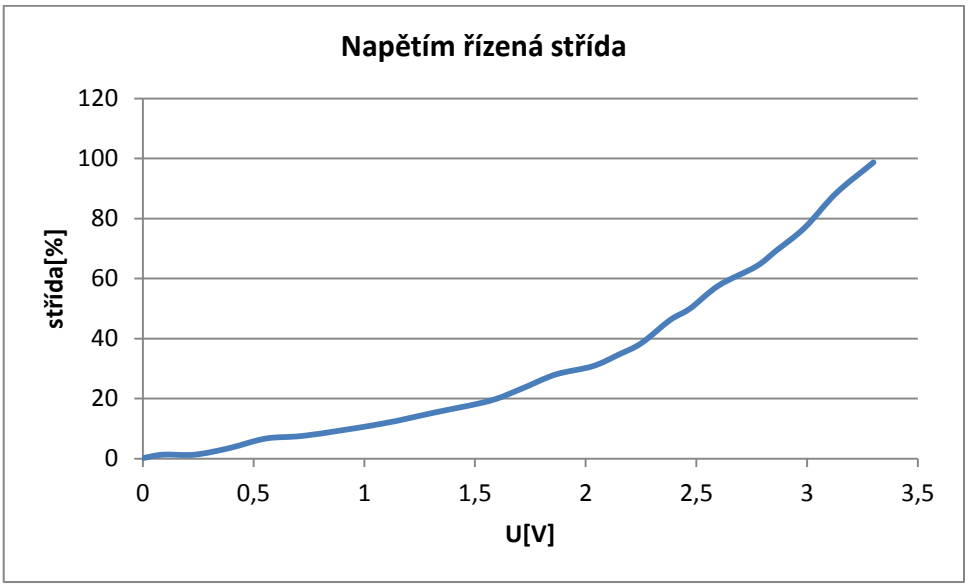
6. Testování simulátoru

Měření bylo provedeno pomocí měřící karty 6210 od firmy National Instruments a za tímto účelem byla v programu Labview, což je software poskytovaný firmou National instruments pro měření pomocí těchto karet, vytvořena aplikace, kterou byly ověřeny jednotlivé funkce simulátoru.

Na obrázcích 6.1 a 6.2 je zobrazena závislost střídy na napětí, kdy byla měřena velikost vstupního napětí a střída výstupního napětí pro lineární a exponenciální nastavení závislosti.

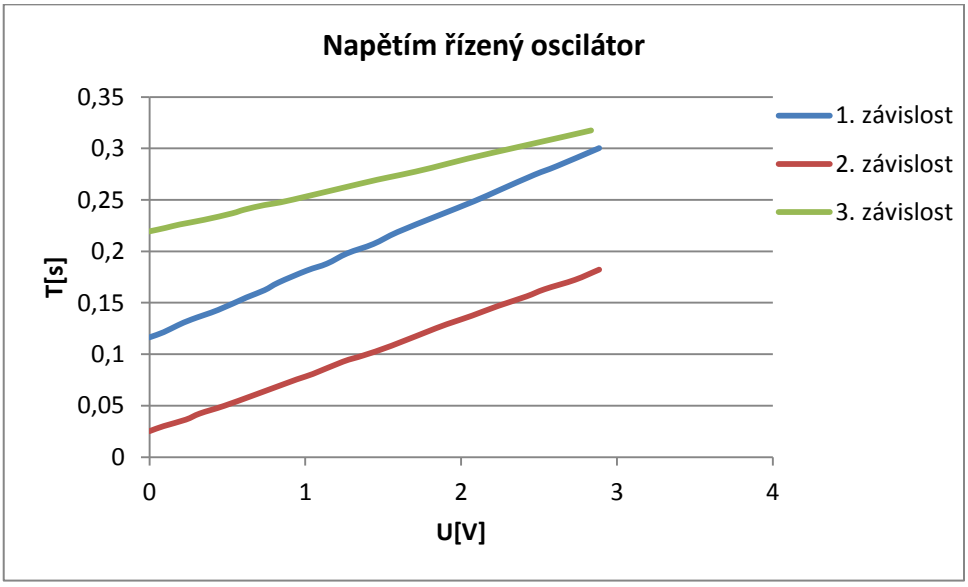


obr. 6.1. Lineární závislost napětím řízeného PWM

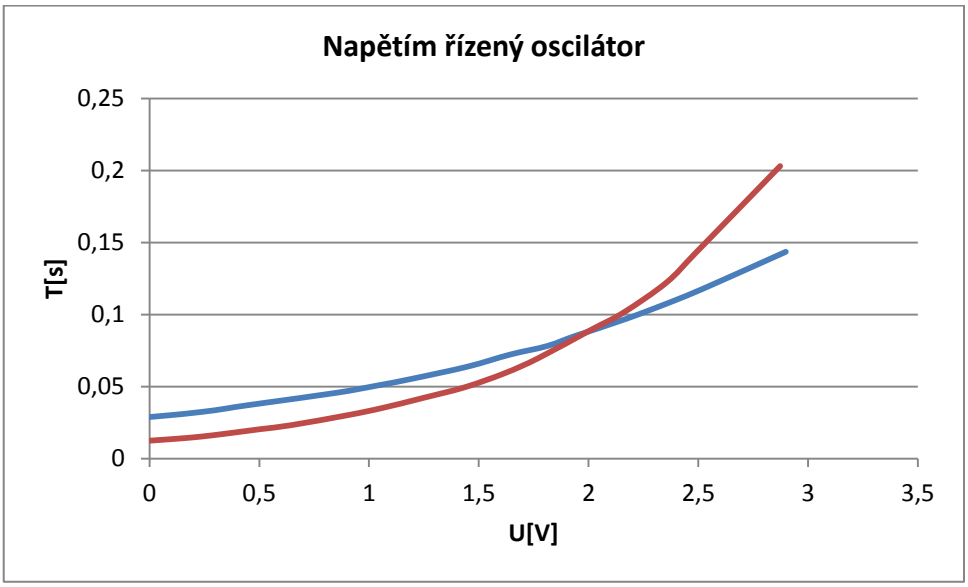


obr. 6.2. Exponenciální závislost napětím řízeného PWM

Na obrázku 6.3 jsou zobrazeny tři naměřené závislosti periody na napětí, kde u každého měření byla nastavená jiná minimální a maximální perioda. U tohoto měření byla nastavena lineární závislost periody na napětí. Na obr. 6.4 jsou změřené dvě závislosti pro různé minimální a maximální periody, kde závislost periody na napětí byla nastavena na exponenciální. U obou dvou měření byl tvar výstupního signálu nastaven na obdélník.



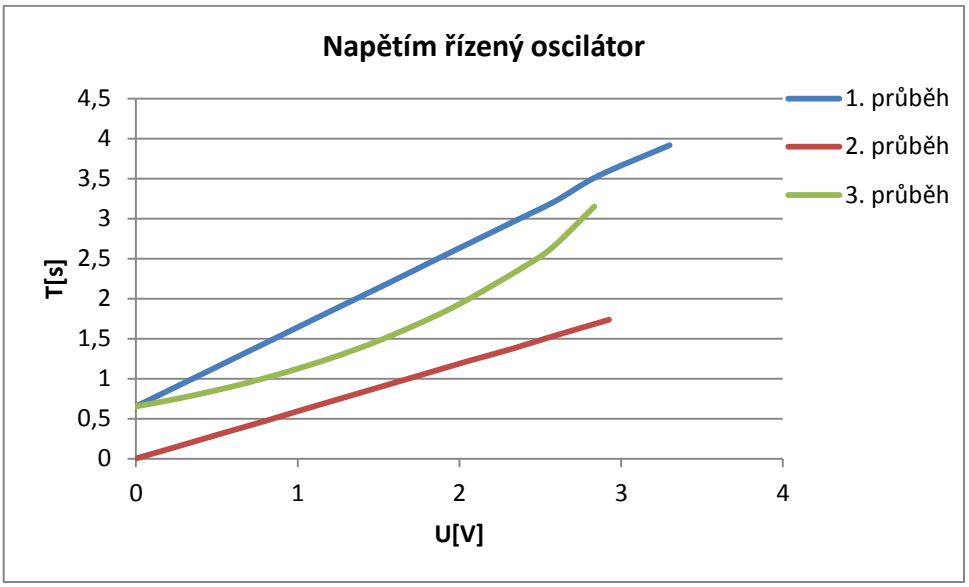
obr. 6.3. Lineární závislost napětím řízeného oscilátoru pro výstupní signál tvaru obdélník



obr. 6.4. Exponenciální závislost napětím řízeného oscilátoru pro výstupní signál tvaru obdélník

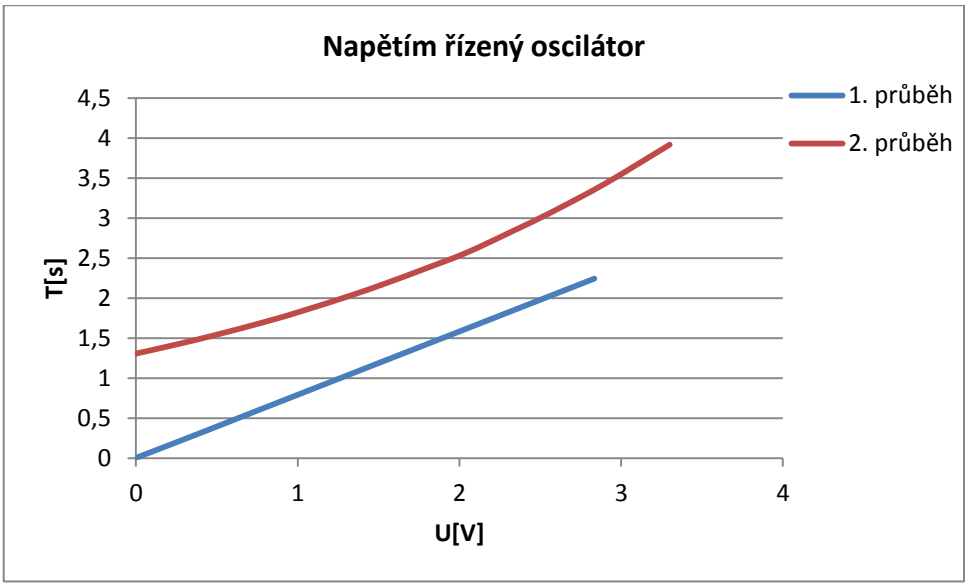
Na obr. 6.5. jsou zobrazeny tři změřené závislosti periody na napětí pro výstupní signál tvaru sinus, kde první a druhý průběh měl nastavenou lineární závislost periody na napětí, kdežto třetí

průběh mněl tuto závislost nastavenu na exponenciální. U všech měření pak byla nastavená jiná minimální a maximální frekvence.



obr. 6.5. Závislost napětím řízeného oscilátoru pro výstupní signál tvaru sinus

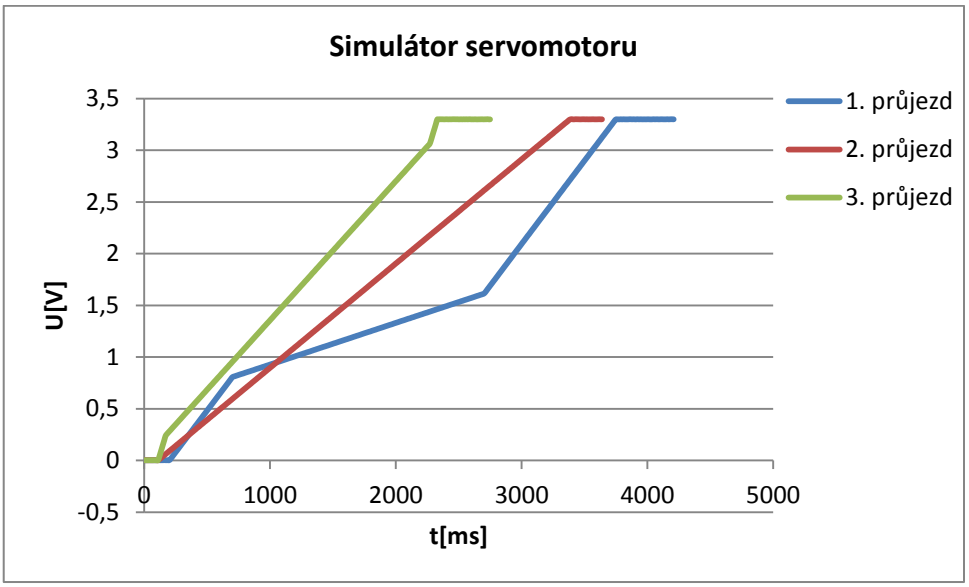
Obrázek 6.6. zobrazuje změřené závislosti periody na napětí pro výstupní signál tvaru pila, kde první průběh byl změřen při nastavené lineární závislosti periody na napětí, kdežto druhý průběh pro exponenciální.



obr. 6.6. Závislost napětím řízeného oscilátoru pro výstupní signál tvaru pila

Při testování funkce simulátoru servomotoru, byl změřen průběh při průjezdu servomotoru z jedné krajní polohy do druhé, při různých konfiguracích servomotoru. Na obr. 6.7. je pak

zobrazen tento průběh, kdy u prvního a třetího průjezdu je nastavena část dráhy, kterou projede servomotor pomaleji a u druhého ne.



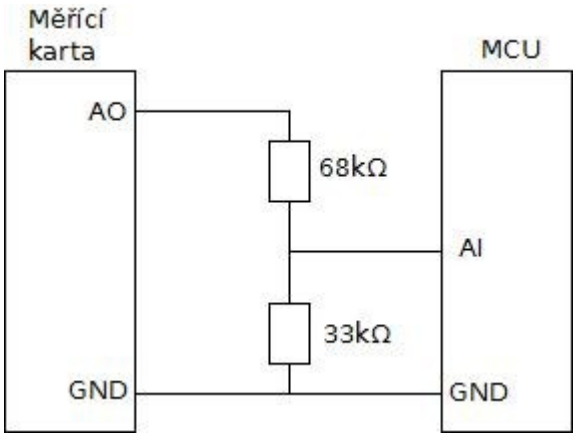
obr. 6.7. Průběh simulátoru servomotoru

7. Návod pro práci se simulátorem

Tato část práce se zabývá návodem pro práci se simulátor a je rozdělena na dvě části. První část je HW část, která se zabývá připojením mikrokontroléru k výukové platformě NI ELVIS a následné propojení s měřící kartou a druhá část je SW část, která řeší nastavení funkcí simulátoru pomocí řídící aplikace.

HW část:

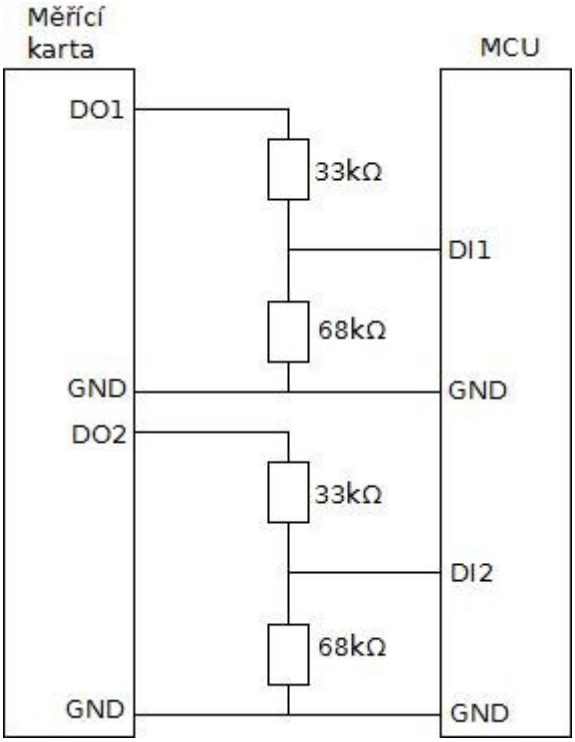
Zaprvé je nutné přípravek s mikrokontrolérem zapojit do nepájivého pole. Následně je pak nutné propojit vstupy a výstupy mikrokontroléru se vstupy a výstupy měřící karty dle funkce, kterou bude mít simulátor vykonávat. U funkcí napětím řízeného oscilátoru a napětím řízeného PWM je nutné propojit analogový výstup z měřící karty s analogovým vstupem mikrokontréru. Při tomto propojení je nutno mezi zařízení dát napěťový dělič, viz obr.7.1.



obr. 7.1. Zapojení analogového vstupu a výstupu

Výstupy mikrokontroléru u těchto funkcí se mohou propojit přímo se vstupy měřící karty.

U funkce simulátoru servomotoru je nutné propojit dva digitální výstupy měřící karty s digitálními vstupy mikrokontroléru. Těmito vstupy mikrokontroléru se pak ovládá pohyb servomotoru doprava nebo doleva. Při tomto propojení je mezi zařízení nutné zapojit napěťový dělič viz obr. 7.2. Výstup z mikrokontroléru, který poskytuje informaci o krajní poloze, je možné přímo spojit s vstupem měřící karty.



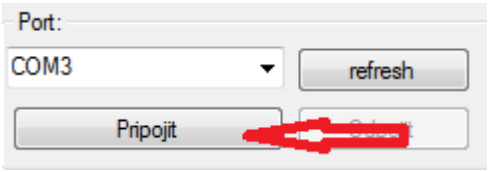
obr. 7.2. Zapojení digitálních vstupů a výstupů

Funkce generátoru impulzů vyžaduje pouze propojení výstupu mikrokontroléru se analogovým vstupem měřící karty.

SW část:

Poté co je mikrokontrolér propojen s měřící kartou a pomocí USB kabelu je spojen s PC je zapotřebí pustit aplikaci pro ovládání simulátoru. Aplikace má název simulator.exe

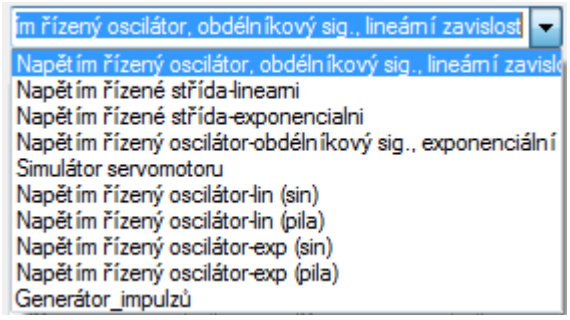
Po spuštění se zobrazí samotné okno aplikace, ve které je nejdříve nutno se připojit k sériovému portu, ke kterému je simulátor připojen obr. 7.3.



obr. 7.3. Připojení

To k jakému portu je simulátor připojen lze zjistit ve správci zařízení, kde v záložce porty je simulátor pod názvem OpenSDA.

Po připojení se k simulátoru je pak možné vybrat samotnou funkci zobrazenou na obr. 7.4.



obr. 7.4. Výběr funkce

U funkce napětím řízeného oscilátoru pro výstupní obdélníkový signál kde závislost periody na vstupním napětí je lineární je možné pomocí dvou posuvníků možné nastavit minimální a maximální periodu. Při tomto nastavování musí být maximální perioda vždy větší, než ta minimální jinak tato funkce nepůjde nastavit. Dále je možné nastavit, jestli v závislosti na vstupním napětí se bude lineárně měnit perioda nebo frekvence výstupního signálu. Doporučuje se však pracovat s periodou, neboť u té je zaručena přesnost nastavení.

Další funkce jsou funkce napětím řízené PWM pro buď lineární, nebo exponenciální závislost střídy na vstupním napětí. U těchto funkcí se žádné parametry nenastavují.

U funkce simulátoru servomotoru je nutné nastavit správné parametry. Zaprvé je to čas, za který projde servomotoru z jedné krajní polohy do druhé. Dále je pak nutné nastavit hodnoty krajních poloh servomotoru. Tyto hodnoty jsou v rozmezí od 0° do 180°. Při nastavení těchto krajních poloh musí být hodnota koncové polohy vpravo větší než koncová poloha vlevo. Dále je pak možné nastavit část dráhy servomotoru, která bude projeta jiným časem. Při této volbě je nutné, aby byl nastaven začátek a konec dráhy, kterou má servomotor projet jiným časem v intervalu daným krajními polohami. Nakonec je zde nastavení zda se servomotor zastaví v krajních polohách či ne.

U generátoru impulzů se pouze nastaví pomocí šestnácti zaškrtávacích polí sled impulzů, kde zaškrtnuté pole představuje impulz v poloze logické „1“.

8. Závěr

V této bakalářské práci se zabývám návrhem simulátoru pro výuku virtuální instrumentace a to jak po stránce fyzické realizace tak též po stránce návrhu softwaru.

V části fyzické realizace jsem při návrhu zařízení využil toho, že součástí výukové platformy NI ELVIS je i nepájivé pole, do kterého bylo možné mikrokontrolér i s ochrannými obvody zapojit. Toto řešení jak pak výhodné z toho důvodu, že není potřeba celé zařízení umísťovat na desku plošného spoje a tudíž je toto řešení výhodnější z hlediska úspory ceny.

Další částí bakalářské práce bylo navržení softwaru. Návrh se skládal z části pro mikrokontrolér a z části vytvoření řídící aplikace. V části pro mikrokontrolér byl navržen software, který umí generovat požadované funkce a to napětím řízený oscilátor, napětím řízené PWM, simulátor servomotoru a generátor impulzů a s tím spojený příjem dat po sériové lince. Dále pak byla vytvořena řídící aplikace, umožňuje nastavovat parametry a funkci simulátoru.

Nedílnou součásti práce bylo i její testování. To bylo provedeno pomocí měřící karty, kdy byly testovány jednotlivé funkce simulátoru při různých parametrech těchto funkcí. Tímto testováním byla ověřena správná funkčnost celého zařízení.

Nakonec je v práci zpracován návod pro práci s tímto zařízením.

Seznam použité literatury

[1] *elvis_l.jpg* [online] http://sine.ni.com/images/products/us/elvis_l.jpg [citováno 2. 4. 2014]

[2] *frdm.jpg* [online] http://www.freescale.com/ FRDM-KL25Z [citováno 2. 4. 2014]

[3] A/D převodník. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-02]. Dostupné z: http://cs.wikipedia.org/?title=A/D_p%C5%99evodn%C3%ADk

[4] PRAUZEK, M. *Číslicová a mikroprocesorová technika*. Ostrava, 2013. 117 s. VŠB-TUO

[5] Sériová komunikace. In: *Robotika* [online]. 2011 [cit. 2014-05-02]. Dostupné z: http://robotika.cz/guide/comm/cs

Seznam příloh

Obsah disku CD-ROM

- Zdrojový kód pro mikrokontrolér
- Zdrojový kód pro řídící aplikaci