Pedagogická fakulta
Faculty of Education
Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra Informatiky

Bakalářská práce

Soubor úloh postavených na jednočipech Atmel AVR

Vypracoval: Jakub Homola
Vedoucí práce: Ing. Michal Šerý, Ph.D.

České Budějovice 2013

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub HOMOLA**
Osobní číslo:	**P10324**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie ve vzdělávání**
Název tématu:	**Soubor úloh postavených na jednočipech Atmel AVR**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

1. Stručný popis rodiny jednočipů Atmel AVR
2. Programování ve vyšších programovacích jazycích - srovnání možností
3. Popsat vývojové prostředí BASCOM a programovací jazyk
4. Navržení zhruba 20 úloh a ověření funkčnosti ukázkových kódů
5. Zpracovaní podkladů pro jednotlivé úlohy
6. Zhodnocení

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **Hrbáček, J.: Komunikace mikrokontroléru s okolím 1. BEN, 1999, ISBN 80-86056-36-8.**
2. **Váňa, V.: Programování v jazyce BASCOM, BEN, 2004 ISBN: 80-7300-115-2.**
3. **Šubrt, V., Mikrokontroléry Atmel AVR - vývojové prostředí, BEN 2002, 95 s. ISBN: 80-7300-055-5.**
4. **Bumba, J.: Programování mikroprocesorů, COMPUTER PRESS, 2011.**
5. **BASCOM-AVR - MCS Electronics. BASCOM AVR [online]. 2012, 24.4.2013 [cit. 2013-04-24]. Dostupné z: http://www.mcselec.com/?option=com_content&task=view&id=14&Itemid=41**
6. **BASCOM AVR Tutorial. In: Bahrambaba2.persiangig.com/other/bascom avr tutorial.pdf [online]. 2012 [cit. 2013-04-24]. Dostupné z: http://bahrambaba2.persiangig.com/other/bascom avr tutorial.pdf**

Vedoucí bakalářské práce: **Ing. Michal Šerý**
Katedra aplikované fyziky a techniky

Datum zadání bakalářské práce: **16. dubna 2013**

Termín odevzdání bakalářské práce: **30. dubna 2014**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 16. dubna 2013

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. V platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 6. 1. 2014

Jakub Homola

Anotace

Práce pojednává o jednočipech typu Atmel AVR. Dále provedu porovnání prostředí vyšších programovacích jazyků. Zaměřím se převážně na prostředí programovacího jazyka BASCOM. Hlavním smyslem práce je navrhnout soubor úloh, vytvořit protokoly a sestavit vzorová řešení. V první úloze se zaměřím na zpracování digitálních vstupů / výstupů. Další úlohy budou zaměřeny na displeje, analogové vstupy / výstupy, motory, sběrnice a komunikaci mezi mikrokontroléry.

Abstract

The paper deals with the microcontroller Atmel AVR type. Then I´m going to compare the environment of higher programming languages. I will mostly focus on the environment of programming language BASCOM. The main purpose of this paper is to suggest a set of tasks, create reports and compile a sample solution. In the first task I will focus on the processing of digital inputs / outputs. Other tasks will be focus on the display, analog inputs / outputs, engines, bus and communication between microcontrollers.

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce panu Ing. Michalu Šerému, Ph.D. za odbornou a vstřícnou pomoc, poskytnutí cenných rad a veškerý čas, který mi věnoval.

Dále bych rád poděkoval rodině a blízkým za podporu během zpracování této práce.

Obsah

1 Úvod

V dnešní elektronické době je stále více rozšířena miniaturizace zařízení a maximalizace funkcí zařízení.

Tato bakalářská práce seznámí čtenáře s jednočipy obecně. Dále se čtenář seznámí s programovacím jazykem BASCOM a jeho vývojovým prostředím.

Cílem práce je sestavit výukový soubor úloh, ve kterých jsou zařazeny nejjednodušší úlohy, které demonstrují a vysvětlují základní funkce jednočipů.

Dalším cílem je sestavit vzorová řešení jednotlivých úloh.

Posledním stanoveným cílem je vytvoření pracovních protokolů, kde je vždy popsána funkce nově používané instrukce.

2 Teoretická část

2.1 Co je to jednočipový mikropočítač?

Počítač jako takový se vždy skládá z řídící jednotky (procesoru), vždy má nějakou operační paměť (paměť programu), rozhraní pro práci s periferiemi (I/O obvody) a samozřejmě nějakou základní desku, která zprostředkovávala komunikaci mezi vším výše zmíněným. Když se podívám do minulosti, tak první počítače byly sálové počítače (prakticky nepřenosné). Následovaly první domácí desktopové počítače, všechny nutné prvky se tedy musely zmenšit. Bylo potřeba používat počítač i pro jednoduché úkony, jako jsou například základní výpočty, které dnes jsme schopni provádět kalkulačkou, proto musel vzniknout jednočipový mikropočítač, který obsahuje vše výše zmíněné.

Srdcem jednočipového mikropočítače je mikroprocesor, který se stará o vykonávání aritmetických a logických operací podle daného programu. Program se skládá z instrukcí, které jsou uloženy v paměti programu. Mikroprocesor zajišťuje správné zpracování instrukcí programu, zpracování dat v paměti a také ovládání ostatních částí mikropočítače, jako jsou vstupní / výstupní zařízení.

Mikroprocesor tedy pro svou funkci potřebuje 2 typy paměti - paměť pro program a paměť pro data. Mikroprocesor také potřebuje vstupní a výstupní obvody nutné pro komunikaci s okolními připojenými periferiemi.[1]

2.2 Struktura jednočipového mikropočítače

2.2.1 Mikroprocesor

Nejdůležitější součást jednočipového mikropočítače. Řídí vykonávání instrukcí programu a synchronizuje činnost všech připojených periferií.[3]

2.2.2 Paměť

Existují 2 typy paměti, které jsou využívány (samozřejmě jejich obměny se také využívají). Jedná se o typ paměti ROM a RWM.[3]

2.2.2.1 Paměť ROM (Read Only Memory)

Tento typ paměti je vhodné využívat pouze jako paměť programu, protože je možné z takové paměti pouze číst, nikoliv do ní zapisovat. Nemůže se tedy stát, že by někdo přepsal program. Existuje hned několik typů pamětí ROM. Nejzákladnější typ je naprogramován při výrobě a nelze již nikdy přeprogramovat. Jiné typy například EEPROM (Elektric Erasable Programmable ROM) je paměť, kterou lze mazat elektricky. Další typy jsou PROM (Programmable ROM) a EPROM (Erasable Programable ROM). Paměť typu PROM je možná naprogramovat, ale uložený program již nelze smazat, zatímco EPROM lze jak naprogramovat, tak smazat, ale mazání je možné pouze ultrafialovým světlem namířeným přímo na paměť.[3]

2.2.2.2 Paměť RWM (Read / Write Memory)

Tento typ paměti je specifický v tom, že z ní lze jak číst, tak do ní i zapisovat. Pro udržení informace v paměti je nezbytné napájení, jakmile se napájení odpojí, informace se ztratí. Tyto paměti se využívají jako dnešní operační paměti (označované jako RAM [Random Access Memory]). Existují ještě další dva typy těchto pamětí a to statické a dynamické. Statické paměť nepotřebuje obnovovací signál, zatímco dynamická paměť jej potřebuje s určitou periodicitou (tzv. refresh).[3]

2.2.3 Sběrnice

Sběrnice je komunikační kanál, který zprostředkovává komunikaci mezi všemi částmi mikropočítače.[3]

2.2.4 Brány, tzv. porty

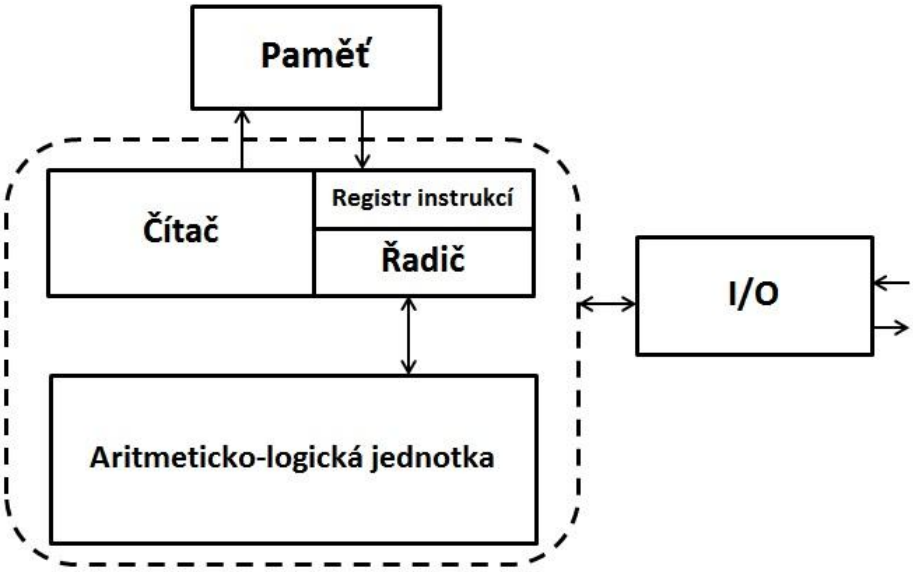
Jsou to obvody, které slouží pro připojení periférií a zajišťují komunikaci s nimi.[3]

2.2.5 Periferie

Periferie mohou sloužit, jak pro obsluhu mikropočítače (například: klávesnice, terminál), tak pro vykonávání funkce určené mikropočítači (například: televizor, tiskárna).[3]

Existují dvě architektury mikroprocesoru.

2.2.5.1 Von Neumanova architektura



Řadič je řídící jednotka, řídící veškerou činnost.

Registr instrukcí načítá instrukce z paměti do registrů, které pak řadič dále posílá ke zpracování.

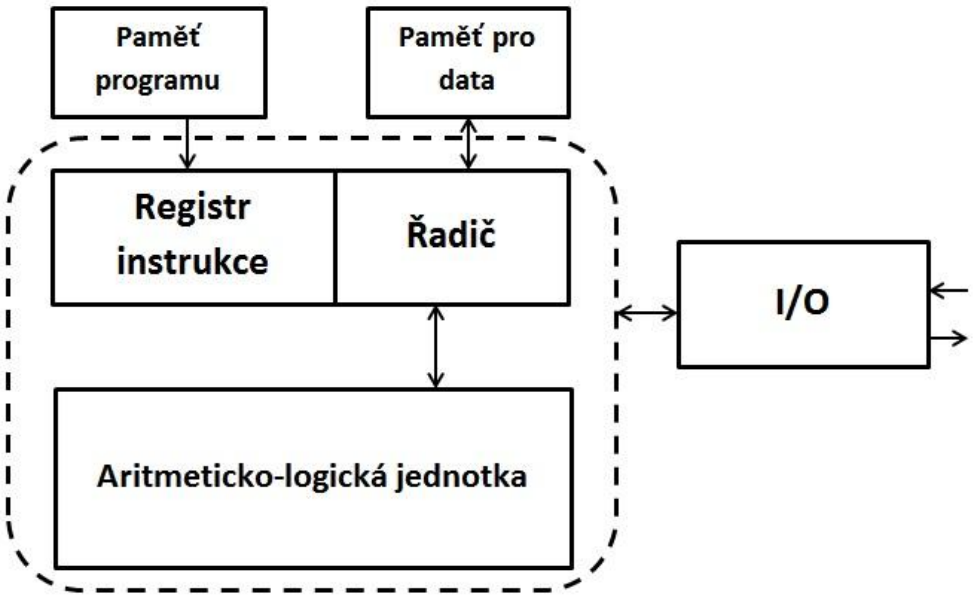
Čítač zapisuje data do paměti.

ALU je aritmeticko-logická jednotka, která zpracovává aritmetické a logické operace.

Paměť je zde společná. Je v ní tedy uložen jak program, tak data a výsledky výpočtů.

I/O jsou rozhraní vstupních a výstupních zařízení.[2]

2.2.5.2 Harvardská architektura



Tato architektura využívá na rozdíl od Von Neumanovi dvě paměti a to paměť programu a paměť určenou pro data. Paměti však nelze využívat pro jiný účel než ten určený.[2]

2.3 Porovnání vyšších programovacích jazyků

2.3.1 Porovnání

2.3.1.1 BASCOM-AVR

1. Výhody
   - Strukturovaný BASIC s návěštími[4]
   - Rychlý strojový kód[4]
   - Velké množství rutin, obsažených již ve výrobcem definovaných knihovnách[5]
   - Možnost využití direktivy Assembleru[9]
2. Nevýhody
   - Výsledný soubor pro procesor (HEX) je asi o 20% větší než u programovacího jazyka ASSEMBLER.[5]

2.3.1.2 Assembler

1. Výhody
   - Je možné naprogramovat prakticky cokoliv
   - Vysoká rychlost programů[6]
2. Nevýhody
   - Jazyk je obecně poměrně složitý
   - Zdrojový kód je při složitějším programu dlouhý a hůře přehledný.[6]

2.3.1.3 CodeVisionAVR

1. Výhody
   - Snadné nastavení komponent - program automaticky vygeneruje část kódu.

- Spolehlivá spolupráce s programátorem připojeným přes USB rozhraní.[7]

2. Nevýhody

- Placená licence
- Volně šiřitelná licence má omezenou délku zdrojového kódu.[7]

2.3.1.4 Závěr

Vybral jsem si programovací jazyk Bascom, protože má jednoduché prostředí, vychází z jazyka Basic, tudíž je jednoduchý na pochopení. Má rychlý zdrojový kód a velké množství rutin. Samozřejmě využití direktivy programovacího jazyka Assembler je také nespornou výhodou. Pro účely tohoto souboru úloh není příliš podstatná nevýhoda většího *.hex souboru.

2.3.2 Popis jazyka BASCOM a jeho vývojového prostředí

Kompilátor BASCOM je navržen pro platformu Windows. Je podporován na všech jeho starších verzích, dokonce funguje i ve Windows 7. O podpoře na nejnovější verzi (Windows 8) je na oficiálním webu zmínka pouze u některých verzí kompilátoru.[8]

Vývojové prostředí BASCOM složí především pro editaci zdrojových kódů, překlad, spojování, debugging atd., ale také pro spouštění simulátoru nebo programátoru. Prostředí vypadá jako běžný program běžící v operačním systému Windows. Většina běžných akcí jako je otevření nebo zavření souboru, editace souboru, tisk a obecná nastavení vlastního prostředí odpovídá standardnímu rozložení, jako u jiných programů pro Windows. Úmyslně je prostředí takto situováno, aby ho dokázal ovládat běžný uživatel, který je schopný pracovat v operačním systému Windows. Téměř všechny ovládací prvky včetně nastavení a menu se nacházejí v základním okně.[9]

Ke spuštění pomocných nástrojů BASCOMU slouží menu Tools. Nacházejí se zde položky Terminal emulator, LCD designer, LIB Manager, Graphic Converter, Auto Update, Stack Analyser a PlugIn Manager (poslední 3 se běžně nevyužívají).[9]

Terminal emulator spouští terminálové okno, které slouží k sériové komunikaci prostřednictvím sběrnice RS232.[9]

LCD designer je editor, který nám umožňuje navrhnout si vlastní uživatelské znaky LCD displeje.[9]

LIB Manager slouží k přidávání vlastních knihoven, například pro využití speciálních knihoven jazyka Assembler.[9]

Graphic Converter je nástroj, který se využívá pro spolupráci s grafickým LCD displejem.[9]

V menu následuje záložka Options. Po zvolení této záložky se spustí nové okno, v kterém máme celou řadu možných nastavení prostředí.

Programovací jazyk BASCOM vychází z programovacího jazyka Basic. Využívá strukturované programování typu[8]:

a. IF-THEN-ELSE-END IF
b. DO-LOOP
c. WHILE-WEND
d. SELECT-CASE
e. FOR-NEXT

Jazyk BASCOM využívá tyto elementární datové typy.[9]

Typ	Velkost (bitů)	rozsah
Bit	1	0, 1
Boolean	1	0, 1
Byte	8	0 až 255
Integer	16	-32 768 až 32 767
Word	16	0 až 65 535
Long	32	-2 147 483 648 až 2 147 483 647
Single	32	$1,5 . 10^{-45}$ až $3,4 . 10^{38}$
String	až 254*8	Každý znak je uložen v jednom byte, následuje byte obsahující nuly

2.4 Programátory

Programátor je elektronické zařízení, které dokáže nahrát program nebo data do obvodu. Tím obvodem může být mikropočítač, paměť nebo hradlové pole, atd.[10]

2.4.1 STK500

AVR STK500 Starterkit je kompletní balík vývojových nástrojů pro mikroprocesory Atmel AVR. Podporuje všechny druhy programování mikroprocesorů AVR ukládaných do patic i ISP programování v externích zařízeních.[11]

Základní vlastnosti

- Spolupracuje s AVR Studiem.
- Rozhraní RS-232 pro připojení k PC
- Regulovaný napájecí zdroj pro vstupní napětí 10-15Vst

- Sokly pro 8, 20, 28 a 40 vývodové mikroprocesory AVR
- Paralelní a sériové programování vysokým napětím
- Sériové ISP programování
- ISP programování v externích aplikacích
- Přeprogramování z AVR zařízení
- 8 tlačítek pro použití v uživatelské aplikaci
- 8 LED pro použití v uživatelské aplikaci
- Všechny vývody AVR snadno přístupné přes vyvedené konektory
- Druhý RS-232 port pro použití v uživatelské aplikaci
- Rozšiřující konektory pro Plug-in moduly a prototypy
- 2Mbit datové Flash
- Frekvence oscilátoru, napětí a nulování nastavitelné i z AVR Studia[11]

„Vstupně výstupní porty (I/O) jsou uživateli zpřístupněny prostřednictvím konektorů, lze je použít ve spojení s externími zařízeními, nebo je možné využít ovládacích tlačítek a signalizačních LED umístěných přímo na vývojové desce. Samostatný sériový port RS-232 může být připojen k jakýmkoliv I/O vývodům.“[11]

„Pro potřeby vývoje aplikací s různými mikroprocesory je vývojová deska STK500 koncipována jako široce variabilní, takže lze nastavovat prakticky vše od kmitočtu oscilátoru až po napájecí napětí mikroprocesorů a to pomocí zkratovacích propojek. K dispozici je též objímka pro připojení externího krystalu. Při použití interního oscilátoru desky však lze hodinovou frekvenci mikroprocesorů AVR a jejich napájecí napětí rovněž pohodlně nastavovat z prostředí AVR Studia.“[11]

2.4.2 EVB4.3

Konstrukce celého kitu umožňuje velice flexibilní zapojení a konstrukci projektů. Veškeré periferie jsou součástí desky, ale většinou využívají pouze napájecí větve. Jejich vstupy a výstupy jsou vyvedeny na jednotlivé piny stejně jako u mikroprocesoru. Zapojení zůstává na obsluze, kam periferie připojí. Mikrokontrolér má pevně připojený pouze krystalový oscilátor 16MHz, tlačítko RESET a Pull-Up rezistory pro I2C sběrnici o velikosti 10kΩ a napájení. Všechny ostatní porty jsou dostupné na propojovacích pinech. Například LCD displej si může obsluha připojit podle svých zvyklostí.[12]

Vlastnosti a osazení kitu EvBrev. 4.3 je následující:

- Procesor AVR ATMega32 (existují verze s ATMega16 a ATmega 644p), krystal 16MHz
- Obvod reálného času PCF8583
- Paměť EEPROM AT24C02
- Infračervený přijímač TSOP4836
- Teplotní čidlo DS18B20
- Převodník sběrnic RS485/RS232 - SN75176BP
- Patici pro kartu MMC/SD
- 5 tlačítek
- 8 indikačních LED diod
- 2 potenciometry pro nastavení napětí
- 4 x sedmisegmentový LED zobrazovač
- 5 x výkonový výstup s otevřeným kolektorem ULN2003
- podsvícený displej LCD 2x16 znaků (zelený, modrý, černý)
- USB konektor
- ISP programovací konektor[12]

2.5 Stavebnice

2.5.1 Arduino

Arduino je otevřená elektronická platforma, která využívá uživatelsky přívětivý hardware a software. Arduino je určeno pro každého, koho zajímá vytváření interaktivních objektů nebo prostředí.[13]

Arduino je schopné vnímat okolí prostřednictvím celé škály senzorů a čidel. Zároveň může také využívat celou řadu výstupů např. LED diody, motorky s dalšími výstupními periferiemi.[13]

Mikroprocesor na desce se programuje pomocí speciálního Arduino programovacího jazyka (založeného na jazyku Wiring - podobný C). Programování probíhá v originálním Arduino vývojovém prostředí. Projekty založené na Arduinu mohou jednoduše komunikovat se softwarem na stolním počítači nebo notebookem (např. Flash, Processing, MaxMSP).[13]

Desky Arduino je možné ručně stavit a upravovat. Máme však i možnost koupit si desky již hotové a otestované. Software lze stáhnout zdarma z webu. Návrhy plošného spoje jsou k dispozici pod otevřenou licencí a lze je tedy upravovat dle potřeby samotného návrhu.[13]

Hlavní výhody

- jednoduché programování
- jednoduché zapojení
- nízká cena oproti jiným kitům
- spousta návodů
- uživatelská komunita
- platformní nezávislost (Win/Linux/MacOS/...)[13]

2.5.2 MLAB

Autoři této stavebnice uvádějí jako cíl vzniku projektu stavebnice MLAB poskytnout otevřenou, robustní a univerzální technickou stavebnici jak amatérům, tak i profesionálům. Stavebnice je určena k testování a vývoji nových zařízení a je použitelná i k didaktickým účelům a tak je i používána.[14]

Vlastnosti

Speciální vlastností stavebnice MLAB je její vysoká kompatibilnost. Má možnost využívat různé architektury mikroprocesorů a to nejen AVR nebo PIC. Mohou být využity součástky a mikroprocesory jiných výrobců a navzájem konkurenční technologie. Dále má mnohem pohodlnější použití v aplikacích, kde je vyžadována robustnost a variabilita vytvořeného modulu (Autoři uvádějí jako příklad vědecké aplikace.). V MLABu je nový prototyp řešen komplexně od elektroniky po mechanickou konstrukci. Tím můžeme dosáhnout jednoduchým způsobem vysoké spolehlivosti vzniklého zařízení, které je zpravidla schopno pracovat léta bez poruchy.[14]

„Jedním z těchto systematických postupů je použití speciální konstrukční desky. Tato základna je opatřena rastrem děr pro úchyt jednotlivých elektronických modulů. Rastr byl zvolen 400mils (tedy 10,16mm). Připevnění modulů k základní desce je řešeno pomocí šroubů M3x12 s maticemi na druhé straně desky. K propojování vstupů a výstupů elektronických desek se používají kablíky různých délek opatřené na obou koncích konektory (samicemi) pro kontaktní hřebínky. Používají se standardní hřebínky v rastru 100mils (2,54mm).“[14]

3 Praktická část

3.1 Seznam úloh

3.1.1 Blikající řada diod

Zadání:

Máme řadu osmi diod. Vytvoříme úlohu tak, že budou diody cyklicky běhat z levé strany na druhou a zpět, přičemž maximálně mohou svítit pouze 3 najednou (tzv. Knight rider).

3.1.2 Pole

Zadání:

Napište úlohu, která využije pole o rozsahu dvaceti členů a každý člen bude mít hodnotu rovnou druhé mocnině svojí pozice (tzn. 2. pozice = 2^2). Pro naplnění využijte cyklus FOR.

3.1.3 Sedmisegmentovka

Zadání:

Máme sedmi segmentový display, který naprogramujeme tak, že nám postupně zobrazí všechna čísla (0-9). Změna bude po jedné sekundě, přičemž desetinná tečka bude mít 2 stavy (rozsvícená a zhasnutá) mezi těmito stavy bude rozestup pouze 0,5. Bez využití instrukce Select Case.

3.1.4 Sedmisegmentovka – select case

Zadání:

Máme sedmisegmentový display, který naprogramujeme tak, že nám postupně zobrazí všechna čísla (0-9). Změna bude po jedné sekundě, přičemž

desetinná tečka bude mít 2 stavy (rozsvícená a zhasnutá). Využijte pro úsporu kódu instrukci select case.

3.1.5 Sedmisegmentovka – rozšíření 1

Zadání:

Na panelu 4 sedmisegmentovek rozsviťte pouze na prvním z displejů postupně všechny segmenty ve směru hodinových ručiček s tím, že svítit smí pouze jeden segment. Následně to samé na druhém, třetím a čtvrtém displeji a to vše opakovat v cyklu. Každý z displejů se rozsvítí přivedením 0 na příslušný spínací tranzistor, který je připojený na jeden konkrétní pin portu (např. portc.3), avšak aktivní může být pouze jeden.

3.1.6 Sedmisegmentovka – rozšíření 2

Zadání:

Na panelu 4 sedmisegmentovek rozsviťte čísla 1 - 4 postupně na všech 4 displejích. Opět musíme pro rozsvícení příslušného displeje přivézt 0 na spínací tranzistor a pamatujte, že současně nesmí být 0 na dvou tranzistorech. Nastavte zpoždění tak, aby všechna čísla svítila současně (multiplexování).

3.1.7 Sedmisegmentovka – rozšíření 3

Zadání:

Na panelu 4 sedmisegmentovek rozsviťte čtyřmístné číslo zadané jako řetězec (včetně desetinné tečky, tzn. 5 znaků). Nastavte zpoždění tak, aby všechna čísla svítila současně (multiplexování). Opět platí pravidlo, že aktivní může být pouze jeden spínací tranzistor.

3.1.8 Sedmisegmentovka – rozšíření 4

Zadání:

Na panelu 4 sedmisegmentovek rozsviťte čtyřmístné číslo zadané jako řetězec (včetně desetinné tečky, tzn. 5 znaků). Nastavte zpoždění tak, aby všechna čísla svítila současně (multiplexování). Číslo se bude zadávat přes sběrnici RS232.

3.1.9 LCD panel

Zadání:

Vytvořte úlohu tak, aby se na displeji zobrazil řetězec textu „Hello World.“. Upravte zdrojový kód tak, aby se text nejprve zobrazil na středu a začal se posouvat doprava. Text se následně vysune z levé části a takhle bude cyklicky rotovat.

3.1.10 LCD panel – rozšíření 1

Zadání:

Vytvořte úlohu tak, aby se na displeji zobrazil řetězec textu „Hello World.“. Tento řetězec se bude posouvat směrem doprava a vysune se na levé straně a to vše v cyklu. Proveďte úpravu posunu tak, aby nebylo zpoždění mezi zmizením textu z pravé strany a objevením se textu na levé straně.

3.1.11 Blikající řada diod – volba módů

Zadání:

Vytvoříme úlohu tak, abychom přepínači 1 a 2 (pinb.5 a pinb.6) zvolili příslušnou kombinaci, jak se budou diody rozsvěcet. V první pozici budou diody cyklicky běhat z levé strany na druhou a zpět, přičemž maximálně můžou svítit pouze 3 najednou (tzv. Knight rider). V druhé pozici rozsvítíme jednu diodu na obou stranách a následně je v cyklu posouváme směrem do

středu, kde se „srazí“ a otočí zpátky. Třetí pozicí bude jednoduché blikání celého led panelu. Poslední čtvrtá pozice pouze rozsvítí celý LED panel.

3.1.12 Print

Zadání:

Napište úlohu, která bude tisknout na terminál následující údaje.

1. Několik řetězců najednou
2. Řetězec prostřednictvím proměnné
3. Přímí tisk znaku
4. Tisk znaku podle ascii kódu
5. Tisk řetězce čísla typu word (0-65535) zadaného do terminálu a odeslaného po stisknutí enter
6. Tisk textu zadaného do terminálu - klasický typ string. Také odeslat stisknutím enter.

3.1.13 And, Or

Zadání:

Napište úlohu tak, abychom provedli 4 testování proměnných. Máme 3 proměnné B, B1, B2. Nejprve testujeme proměnnou B, zda je v rozmezí 3 až 9 (využijeme AND). Dále testujeme Proměnné B a B1 zda alespoň jedna z nich splňuje podmínku. Podmínky jsou B je menší nebo rovno 4 a B1 je větší nebo rovno 6. Ve třetím testování musí být všechny hodnoty proměnných (B, B1, B2) větší než 1. V posledním testování zkoumáme, zda je proměnná B v intervalu 4 až 9 nebo proměnná B1 menší/rovna 4. V případě splnění podmínky tiskneme na terminál „Pravda“ a v případě nesplnění „Nepravda“.

3.1.14 Timer0

Zadání:

Napište úlohu, která bude počítat sestupné hrany, které vzniknou stiskem tlačítka. Počet zaznamenávejte a tiskněte na terminál.

3.1.15 Timer1

Zadání:

Napište úlohu, která bude počítat frekvenci čítače. Frekvenci spočítáte tak, že budete na běžícím čítači počítat sestupné hrany. Dobu, po kterou bude čítač běhat, nastavte na 999 ms. Následně vytiskněte výsledek na terminál s připojením hodnoty (Hz).

3.1.16 Formát

Zadání:

Napište úlohu tak, abychom do výsledné proměnné zapsali přetypovanou *proměnnou zdrojovou* (z integer do string) a následně upravovali formát zobrazení na terminálu. Využijte pomocného zápisu, kdy formátovanou hodnotou přepíšete původní hodnotu *proměnná výsledná*.

Formáty:

1. „*0000*" – v případě kratšího řetězce se zobrazí nuly, maximální řetězec o délce 4.
2. „ „ – 4x mezera, v případě kratšího řetězce jsou mezery ignorovány.
3. „*00.00*" – vložení desetinné tečky na zvolené místo.
4. „ *0.00*" – 1x mezera na začátku, opět bude ignorována, půjde-li o třímístný řetězec.

5. „+ „ - + a 3x mezera, před třímístný řetězec se připíše znaménko +
6. PrintFormat(*proměnná výsledná*, " 0.000") – způsob formátování přímého při zobrazování na terminál(1x mezera na začátku)

3.1.17 Funkce

Zadání:

Napište úlohu tak, aby se proměnná (délka strany) zadávala přes terminál s potvrzením enterem. Vytvořte funkci, která bude počítat obsah čtverce a jako proměnnou (stranu) využije zadanou proměnnou z terminálu. Výsledek opět tisknete na terminál. V hlavní části úlohy tedy bude pouze zadání proměnné, volání funkce a tisk na terminál.

3.1.18 Podprogram

Zadání:

Napište úlohu, kde bude blikat dioda buď 4x za sekundu, nebo 1x za sekundu. Tato volba bude probíhat pomocí tlačítek, přičemž při stisku tlačítka se vyvolá podprogram, který určí frekvenci blikání. Je nutné vytvořit dva podprogramy. Jeden pro „pomalý“ běh a druhý pro „rychlý“ běh.

3.1.19 Watchdog

Zadání:

Napište úlohu, kde vyvoláte přerušení typu watchdog. Nejprve vytiskněte na terminál „Start“ a spusťte watchdog. Dále v cyklu tiskněte na terminál „průběh“ a nastavte zpoždění tak, aby tisk proběhl 4x, když je watchdog nastaven na hodnotu 2048.

3.1.20 Tlačítko

Zadání:

Napište úlohu, kde bude neustále blikat dioda. Připojte do úlohy ještě tlačítko, kterým se bude regulovat rychlost blikání diody a to tak, že při stisknutém tlačítku bude dioda blikat rychleji a při rozepnutém tlačítku pomaleji. Napište úlohu bez využití podprogramu.

3.1.21 Uart

Zadání:

Napište úlohu, kde vytisknete na terminál řetězec textu. Nejprve přímý tisk z programu (pouze pevné zobrazení textu), následně nastavit řetězec do proměnné typu string a onu proměnnou vytisknout na terminál. Dále vytisknout text, který zadáme přes terminál, nastavte potvrzení odeslání řetězce na enter. Nakonec v cyklu načítejte znaky zadávané přes terminál (bez potvrzení). Využijte cyklus for o rozsahu 1 000 000 průběhů.

3.1.22 Hodiny

Zadání:

Napište úlohu, která bude zobrazovat čas na terminálu. Využijte podprogramů pro zobrazení a počítání času zvlášť. Mějte na paměti, že podle nastavení speciálního přerušení compare1a se nám počítají sekundy v určitém poměru. Zobrazení času bude klasické hod: min: sec.

3.1.23 Přerušení INT1

Zadání:

Napište program tak, aby tisknul hodnotu na portb.0, která je nastavena na hodnotu 1 a to při běhu programu na terminál. Pokud ovšem dojde k přerušení, vytisknout hodnotu portu na terminál a vymazat příznak přerušení.

Přerušení je možné vyvolat změnou hodnoty na portb.0, tedy uzemněním portu.

3.1.24 Procedura

Zadání:

Napište program, který prostřednictvím procedury zrcadlově otočí pevně daný řetězec. V hlavním programu bude definován pouze řetězec, který se bude otáčet. Bude volána procedura, která řetězec otočí a samozřejmě tisk otočeného řetězce na terminál.

3.1.25 Tabulka – lookup

Zadání:

Napište program, který bude využívat tabulku. Naplňte tabulku patnácti hodnotami (provádí se mimo hlavní program). V programu zadejte přes terminál pozici prvku tabulky (1-15) a zadání nastavte na potvrzení enterem. Získanou hodnotu zapište do proměnné a následně vytiskněte na terminál.

3.2 Komplexní úlohy

3.2.1 Akvárium

Řízení akvária

3.2.1.1 obsluha digitálních vstupů

V této části je důležité podotknout, že se zde nepracuje s daty, ale s obsluhou portů.

V sekci digitálních vstupů můžeme mít celou řadu různých senzorů. Záleží na nás, jak bychom si představovali propracované a zabezpečené akvárium. Může zařadit světelné čidlo, které by se staralo o osvětlení, aby zbytečně nesvítilo, pokud jej není třeba, např.: přes den. Jakmile by čidlo zaznamenalo

dostatečné světlo v okolí, provede se vypnutí osvětlení akvária a naopak. Dále je možné také zahrnout teplotní čidlo, které bude kontrolovat přihřívání vody, pokud to budou vyžadovat naše rybičky. Můžeme například mít v akváriu různé exotické druhy ryb, které potřebují teplejší vodu nebo naopak, které potřebují vodu studenější, přičemž by mohlo docházet k ohřevu vody z teploty v místnosti. V těchto případech je nutné zařadit teplotní čidlo, které bude udržovat konstantní teplotu vody. Posledním čidlem, které bych zařadil, ale to už by bylo možná až nadbytečné, by mohlo být čidlo, které hlídá obsah kyslíku ve vodě. Takové čidlo by bylo programově propojeno s čerpadlem, které by se v případě poklesu obsahu kyslíku pod nastavenou hranici zapnulo a jinak zůstávalo vypnuté, čímž by byla zabezpečena určitá energetická úspora.

3.2.1.2 obsluha digitálních výstupů

V sekci digitálních vstupů je hned několik dílčích částí. Ve své podstatě jsou zde nastaveny a pojmenovány jednotlivé porty a každý port se stará o jinou funkci. Musíme počítat s portem, který ovládá topné tělísko. Dále port, který se stará o osvětlení, ať už je osvětlení provedeno žárovkou, zářivkou nebo LED páskou a samozřejmě poslední port, který se stará o regulaci čerpadla, tzn. regulování přísunu vzduchu, okysličování vody. Můžeme ještě připojit například LCD displej, kde bychom zobrazovali aktuální teplotu a procentuální obsah kyslíku ve vodě. Měli bychom tedy přehled o tom, zda vše funguje jak má. V případě, že by například čerpadlo mělo poruchu, můžeme programově nastavit spodní hranici obsahu kyslíku ve vodě, která by byla pod hranicí, kdy se zapíná čerpadlo. V případě, že by došlo k oné spodní hranici, zobrazila by se na displeji varovná hláška. V případě poruchy čerpadla by bylo možné programově zařadit zvukovou signalizace nebezpečí, která by byla řízena časovačem. Tím získáváme další možnou periferii a tou je speaker neboli reproduktor.

3.2.1.3 Komunikace po sběrnici 1-wire

Komunikací přes sběrnici můžeme vyřešit velkou spoustu problémů, které nám vznikají při realizaci. Nejprve bych se zaměřil na teplotu a její regulaci. Sběrnicí 1WIRE můžeme propojit teplotní čidlo s jednočipem a následně s topným tělískem. Pokud bychom připojili touto sběrnicí ještě LCD displej a např. maticovou klávesnici, mohli bychom měnit nastavení teploty, popřípadě i citlivost čidla. Mohli bychom také zařadit jinou verzi řešení, kde bychom pouze zvolili typ nastavení, který by nesl číselné označení, čímž bychom vyvolali hodnotu (požadovanou teplotu) uloženou na zvolené pozici v poli teplot. Dále bychom mohli připojit na sběrnici ještě čidlo hlídající obsah kyslíku ve vodě. Opět bychom tím získali možnost určitého nastavení. Tím máme na mysli výše zmiňované hranice, kdy dojde k zapnutí čerpadla a v případě jeho poruchy, k signalizaci nebezpečí. Další možností by bylo zapínání a vypínání samotného čidla, které hlídá obsah kyslíku ve vodě. V případě vypnutí čidla, což by bylo provedeno přes maticovou klávesnici a volbu na LCD displeji, by čerpadlo běželo nepřetržitě. Něco podobného by bylo možné udělat i s čidlem, které se stará o osvětlení akvária. Mohli bychom mít opět volbu, kdy čidlo vypneme a to by znamenalo, že osvětlení bude nepřetržitě zapnuté.

3.2.1.4 Hodiny

Hodiny jsou v tomto případě možné využít například u osvětlení akvária. Vyřadilo by se světlocitlivé čidlo a osvětlení by se zapínalo/vypínalo podle hodin, které by běžely v pozadí. Dala by se připojit možnost změny letní/zimní nastavení, abychom neměli akvárium osvětlené při denním světle a naopak měli osvětlené v noci. Změna mezi těmito stavy by byla možná 2 způsoby, buď pevné nastavení programově pomocí čítače, respektive samotných hodin. Toto řešení by ale nepočítalo s přestupným rokem, což je ve výsledku lehce

nepřesné. Další možností by bylo ruční přepínání možností přes LCD displej a maticovou klávesnici.

3.2.1.5 Podprogramy

Program by měl být postavený tak, aby pouze kontroloval čidla a v případě určitého stavu využil patřičný podprogram. Podprogramů by v tomto případě byla celá řada. Musíme vytvořit podprogram pro udržování konstantní teploty vody, který bude muset být rozdělen do dvou na sobě nezávislých částí a to na ohřev vody a ochlazení vody. Další podprogram sledující stav na světlocitlivém čidle (ve verzi, kdy jej chceme využívat a nepoužíváme verzi s hodinami). Dalším podprogramem budeme obsluhovat čidlo pro spouštění čerpadla a na tento podprogram vytvořit další navazující, který bude vyvolán pouze v případě, kdy dojde k poruše čerpadla a čidlo pro hlídání obsahu kyslíku ve vodě zaznamená pokles pod spodní „kritickou“ hranici. Další podprogram by mohl být na verzi, kdy budeme využívat hodiny na nastavení osvětlení a tím pádem samotné hodiny jako podprogram. Ještě bychom mohli zařadit podprogram pro verzi, kdy se bude osvětlení chovat podle toho, jaký je měsíc (každý měsíc se postupně mění délka dne a noci, v závislosti na tom by se měnila i doba, kdy je osvětlení zapnuto a kdy vypnuto). Poslední a možná i trochu bizarní podprogram by mohl opět využívat pole, kde bychom měli nastaveny hodnoty teploty vody podle druhů ryb, které se aktuálně nacházejí v akváriu.

3.2.1.6 Pole

Jak výše zmiňuji, jsou zde možné využít dvě varianty polí. Tou první variantou by bylo pole, které by mělo 24 hodnot. Každý měsíc by měl 2 pozice a to pozici pro rozsvícení osvětlení a druhou pro zhasnutí osvětlení. Každá pozice by obsahovala hodnotu času, kdy má dojít k vyvolání podprogramu zapnutí/vypnutí osvětlení. Druhou variantou využití pole v tomto příkladu by bylo teplotní nastavení. Pokud máme například 10 druhů ryb a každá potřebuje

jinou teplotu vody, tak bychom měli vytvořit pole, kde by každá pozice nesla hodnotu teploty vody a každá pozice pole by znamenala jeden druh ryb. Následně bychom pomocí maticové klávesnice zvolili, jaký typ ryb aktuálně máme v akváriu a pomocí podprogramu bychom z pole získali hodnotu teploty, která se bude konstantně udržovat v akváriu.

3.2.2 Panákovač

Tato komplexní úloha je inspirována úlohou, kterou vytvořil pan Pavel Janík.[15]

3.2.2.1 obsluha digitálních vstupů

Dávkovač alkoholu má hned několik digitálních vstupů. Asi tím nejdůležitějším digitálním vstupem je ovládání přes určitou skupinu tlačítek. Záleží opět na tvůrci, jaký si zvolí způsob ovládání.

První způsob ovládání by byl možný pomocí tlačítek, přičemž každé tlačítko by mělo svojí volbu. Prvním tlačítkem by se provádělo nastavení počtu panáků, které jsou připraveny k obsloužení. Nejlepším řešením by bylo nejspíše připojení dvou tlačítek, kde by jedno zvyšovalo počet o 1 a druhé naopak snižovalo. Další skupina dvou tlačítek by se starala o volbu obsahu alkoholu v panáku. Bylo by možné vytvořit tedy dvě tlačítka, kde by jedno znamenalo „malý panák“ s obsahem alkoholu 0,1 cl a druhé tlačítko by znamenalo „velký panák“ s obsahem alkoholu 0,4 cl. Poslední skupinou tlačítek by mohl být ruční posun tácku. Opět záleží na tom, jakým způsobem bychom toto chtěli řešit. Můžeme si nastavit nulovou polohu tácku, kdy první panák přesně uprostřed přední části je krásně přístupný k odběru. Po stisku tlačítka by se do polohy nula přesunul panák číslo dvě, aby si jej vzal jeho majitel. Toto otočení by bylo možné řešit i automatizovaně, tedy váhovým čidlem. Když by byl předchozí panák odebrán, došlo by k otočení tácku. Pokud bychom potřebovali panák „přeskočit“, byl by možný ruční tlačítkový posun. Při tomto řešení musíme bohužel

vždy projít celé kolo, než se dostaneme k přeskočenému panáku. Řešením by bylo přidat ještě tlačítko, které by otáčelo opačným směrem.

Dalším způsobem ovládání by bylo využití dálkového ovládání, kde bychom využili infračerveného vysílače a přijímače. Stiskem tlačítka se vyšle specifický kód určité instrukce, na přijímači dojde k dekódování specifického kódu a podle tabulky se vyhodnotí, který podprogram bude obsloužen.

Posledním způsobem by mohla být kombinace obou předchozích způsobů. Je nutné zdůraznit, že je nezbytné tuto možnost programově ošetřit tak, že se bude vykonávat požadovaný podprogram v případě stisku tlačítka na panelu nebo po stisku tlačítka na dálkovém ovladači.

Mezi další vstupy je nutné zařadit senzory, které budou hlídat, zda je na tácku umístěn panák nebo není. Pokud bychom tuto kontrolu nevytvořili, mohlo by docházet k rozlévání alkoholu. V případě, že zapomeneme vložit panáka nebo vložíme malého panáka a přitom nastavíme, že chceme nalít velkého panáka, tak senzor zaznamená chybu a vyvolá přerušení. Dále je ještě nutné zařadit celou řadu váhových senzorů. V první řadě senzor na každou pozici pro panáka na tácku, přičemž senzor bude hlídat, aby bylo nalito přesné množství alkoholu, které je odvážené předem. Na tyto senzory je ještě možné zařadit výjimku, že by došlo k otočení tácku na další pozici v případě, je-li váha nulová. Další váhový senzor je zařazen u samotné láhve s alkoholem, který hlídá, aby nám alkohol nedošel. Pokud nám váha obsahu láhve klesne pod předem stanovenou hranici, dojde k přerušení a k hlášce na LCD displeji, aby byl alkohol doplněn.

3.2.2.2 obsluha digitálních výstupů

V tomto příkladu máme digitální výstupy ve své podstatě pouze za účelem zobrazování. Je zde zařazen LCD displej, který nám zobrazuje číslo panáka a jeho specifikace, tzn. velikost panáku. Dále se na displeji budou zobrazovat možnosti nastavení při vytváření nové objednávky. Zobrazen bude vždy obsah

lahve, který bude procentuálně určen poměrem váhy obsahu lahve a spodní hranice, kdy je láhev téměř prázdná. Ještě je možné zařadit diodu, která by v případě překročení hranice obsahu láhve například dvacet procent, začala červeně blikat jako signalizace docházejícího alkoholu.

3.2.2.3 měření analogových veličin

Jak jsem již výše zmínil, budou se v tomto případě využívat váhové senzory, které budou samozřejmě neustále měřit váhu buď panáka, nebo lahve. U panáka máme pouze 3 hodnoty, bude možné tedy využívat volání podprogramu, při splnění podmínky. Jednotlivé hodnoty jsou tedy velký panák, malý panák a žádný panák. U lahve je to ovšem trochu jiné. Máme zde samozřejmě dvě hodnoty, kdy dojde k volání příslušného podprogramu. Těmi hodnotami jsou nízká hladina obsahu alkoholu v láhvi a druhou hodnotou je kritická hladina obsahu alkoholu v láhvi, nutné doplnění. U lahve musíme ovšem zobrazovat na LCD displeji nepřetržitě aktualizovaný procentuální obsah, abychom měli přehled o tom, kdy přibližně bude nutné doplnění. Samozřejmě z tohoto důvodu je zde zavedena signalizační dioda, nicméně ta se aktivuje až po překročení určité hranice. Tímto zobrazením získáme celkový přehled.

3.2.2.4 Podprogramy

V tomto příkladu komplexního řešení je možná celá řada podprogramů, záleží pouze na tom, jak propracované řešení hodláme vytvořit.

První skupinu tvoří podprogramy hlídající zadávání objednávky. Po spuštění „panákovače“ bude na LCD displeji zobrazena výzva k zadání objednávky a rovnou dojde k volání podprogramu pro objednávání. Budeme požádáni, abychom zadali počet panáků, které chceme naplnit a jejich velikost (malý, nebo velký panák). V tomto řešení nebude možné mít na jedné objednávce zvolené rozdílné velikosti panáků, budeme-li chtít například čtyři velké a dva malé panáky, musíme vytvořit objednávku na velké a na malé panáky odděleně.

Další podprogram bude nalévání panáků, neboli vyřízení objednávky. Do tohoto podprogramu je nutné zařadit hned několik ochranných prvků. Nejprve je nutné na otočný tácek nandat panáky zvolené velikosti. Pokud by byl přidán malý panák, přestože je objednávka na panáky velké, dojde k přerušení a varovné hlášce na LCD displeji „Špatný panák!“. Jakmile bude připraven správný panák, dojde k otočení tácku, aby mohl být připraven další. Pokud bude počet velký nebo budou využity všechny pozice na tácku (kterých bude pravděpodobně 8), budou se postupně nalévat panáky, které se dostanou pod trysku. Pokud tedy budeme požadovat všech 8 panáků, bude po připravení osmého panáka na další pozici (tedy první) stát již připravený, nalitý panák. Během nalévání bude probíhat nepřetržitá kontrola obsahu alkoholu v láhvi, pokud by došlo k tomu, že během objednávky dojde k poklesu obsahu pod kritickou hranici, bude vyvoláno přerušení s prosbou o doplnění láhve.

Posledním podprogramem bude rozdávání vyřízené objednávky. Pokud bude odebrán plný panák, dojde automaticky k otočení na dalšího panáka v pořadí. V případě, že vlastník aktuálního panáka k odběru není přítomen, jsou zde zařazena tlačítka na ruční ovládání tácku s posunem vpřed, případně zpět, kdyby se majitel přeskočeného panáka vrátil.

3.2.2.5 Pole

Pro kontrolu obsahu lahve je zde využito pole, které má 19 pozic. Každá pozice reprezentuje 5 % a její hodnotou je váha lahve včetně obsahu. Pokud váha klesá, samozřejmě klesá i obsah lahve a tím se mění pozice v poli. Hodnota pozice pole se tedy bude vždy zobrazovat na LCD displeji. Pozic je pouze 19, protože poslední hodnotou nebude nula procent, ale 5%, aby byl alkohol včas doplněn.

3.2.2.6 Přerušení

Budeme zde využívat hned několik typů přerušení. Nejdůležitějším přerušením je přerušení při kontrole velikosti panáka, aby nedošlo k přetečení.

V případě, že si vybereme velkého panáka a vložíme malého, nebudeme vyzváni k vložení dalšího prázdného panáka, ale na LCD displeji vyskočí varovná hláška chybného vložení panáka. Přerušení zastaví program do té doby, dokud nevložíme správného panáka. Dalším neméně důležitým přerušením je přerušení při kontrole obsahu alkoholu v lahvi. Jakmile se obsah lahve dostane na hranici pěti procent obsahu lahve, vyvolá se přerušení, které zastaví veškerou činnost programu, dokud nedojde k doplnění lahve. Při tomto přerušení bude na LCD displeji zobrazena hláška vyžadující doplnění. Posledním přerušením běhu programu je přerušení při využití ručního ovládání otočného tácku. Při tomto přerušení se zablokuje automatické otáčení tácku. Po stisku tlačítka pro otočení o jednoho panáka požadovaným směrem, se přerušení ukončí a opět pokračuje automatické otáčení.

3.2.3 Vyvrtávačka

Řízení souřadnicové vyvrtávačky

3.2.3.1 Obsluha digitálních vstupů

Mezi digitální vstupy bych v první řadě zařadil optické závory, které jsou dvojího typu. Prvním typem je kontrola zda v prostoru, kam máme souřadnicemi zvolená místa k vrtání, se nachází nějaký předmět, aby nebylo vrtáno „naprázdno“. Bylo by tedy možné navrhnout optické závory, které budou synchronizovány do kříže a v případě, že na zvolených souřadnicích bude něco v cestě, bude provedeno vrtání. V opačném případě by byla do počítače odeslána chybová sběrnice s hláškou „nenalezen materiál k vyvrtání“. Tato kontrola by probíhala vždy před začátkem vrtání. Druhým typem je kontrola prostoru kolem samotné vrtačky, aby nedošlo k úrazu. Jakmile je zvednuto víko, kterým je vrtačka zakrytá, automaticky dojde k přerušení programu a vypnutí vrtačky, protože optická závora procházející víkem nezaznamená nic v cestě. Dalším ochranným prvkem by mohl být tlakový senzor, který bude kontrolovat tlak vznikající na vrtáku.

3.2.3.2 Obsluha digitálních výstupů

Digitálních výstupů je v tomto programu celá řada. Nejprve jsou zde optické závory, které jsem musel již uvádět u digitálních vstupů, protože je komunikace ve své podstatě obousměrná, respektive výstupem je samotná závora a vstupem přijímač, který vyhodnocuje chybovost. Dalším digitálním výstupem jsou krokové motory, které jsou opět dvojího typu. Prvním typem krokových motorů, jsou motory, které ovládají pozicování vrtačky podle souřadnic. Nejprve bude nutné určit, s jak podrobným měřítkem souřadnicového systému chceme, aby vyvrtávačka fungovala. Podle toho zvolíme vhodné krokové motory. Spočítáme, kolik kroků odpovídá jedné jednotce na souřadnicové mapě a do pole uložíme pod jednotlivé pozice odpovídající hodnoty posunu krokového motoru. Na stejném principu bude pracovat i druhý typ krokových motorů, který bude ovládat optické závory pro kontrolu, zda obsluha vložila materiál připravený k vyvrtání. Posledním krokovým motorem bude motor ovládající posun vrtáku směrem dolu k materiálu.

3.2.3.3 RS232

Sběrnice RS232 v tomto případě bude sloužit pro obousměrnou komunikaci s počítačem. Ve výstupním směru z počítače bude sloužit pro zadávání souřadnic jednotlivých míst k vyvrtání. Po spuštění programu, bude obsluha vyzvána k zadání nejprve souřadnice na ose x s potvrzením enterem a následně k zadání souřadnice na ose y s potvrzením enterem. Posledním údajem je hloubka vrtu. Zadaná hodnota se opět bude potvrzovat enterem. Pokud obsluha nevyplní žádnou hodnotu nebo zadá nulu, bude programem zadání vyhodnoceno jako „vrtání skrz“. Ve vstupním směru do počítače bude sloužit pro zobrazování chybových hlášek během přerušení.

3.2.3.4 Podprogramy

V hlavním programu se budou zadávat proměnné neboli souřadnice místa k vyvrtání, které bude probíhat na počítači prostřednictvím sběrnice RS232, jak

jsem výše zmínil. Po zadání souřadnic se spustí podprogram kontroly, zda byl vložen materiál k vyvrtávání. Z pole je podle souřadnic získán posun krokového motoru v ose x a v ose y. Optické závory následně podle nastavení zjistí, zda mají či nemají materiál k vrtání v cestě. V případě, že nebyl vložen, tak dojde k přerušení a odeslání chybové hlášky na obrazovku počítače. V případě, že vložen byl, je vyvolán podprogram vrtání. Nejprve se opět z pole získá posun krokových motorů, které přesunou vyvrtávačku na potřebnou pozici. Následně dojde ke spuštění vrtačky. Zde přijde řada na tlakový senzor, který nejprve zaznamenává nízký tlak (vrtačka se přibližuje k materiálu), následuje zvýšení tlaku na vrták (vrták začal vrtat) a poslední změnou bude opět pokles tlaku na vrták (vrták prošel skrz materiál). Jakmile je materiál provrtaný skrz, udělá krokový motor ovládající posun vrtáku ještě jeden krok a vrátí se do nulové pozice. Po návratu do nulové pozice vypne otáčení vrtáku. V případě vrtání pevně dané hloubky se použije pevně spočítaný počet kroků krokového motoru. Tyto kroky se ovšem budou počítat okamžitě, jak dojde ke změně tlaku na tlakovém senzoru, aby bylo zabezpečeno, že bude vrt přesný pro materiály různé tloušťky. Jakmile je díra vyvrtána, tak program vyskočí z podprogramu vrtání a vrátí se do hlavního programu, kde bude čekat na zadání nových souřadnic.

3.2.3.5 Pole

V tomto programu budou vytvořena dvě pole. První pole bude využíváno pro výpočet posunu čtyř krokových motorů. Každý člen pole bude obsahovat počet kroků krokového motoru vyvrtávačky. Pokud tedy zadáme hodnotu souřadnice na ose x, bude z pole vyvolána hodnota na pozici odpovídající zadané souřadnici. Získaná hodnota určí posun po ose x. Stejný postup bude probíhat i v případě osy y a tentýž způsobem bude probíhat i posun krokových motorů u optických závor, které kontrolují, zda je vložený na zvolených souřadnicích nějaký materiál. Druhé pole bude sloužit pro ovládání krokového motoru, sloužícího k vertikálnímu posunu vrtáku. Jednotlivá pozice bude

znamenat jeden milimetr a hodnota pozice pole bude odpovídat počtu kroků krokového motoru.

3.2.3.6 Přerušení

Přerušení se v tomto programu budou využívat hlavně pro ochranu. Prvním přerušením bude zastavení programu při chybějícím materiálu k vyvrtání. Pokud nevložíme žádný materiál, tak optické závory testující zvolené souřadnice zjistí, že jim nic nestojí v cestě a vyvolají přerušení. Toto přerušení zastaví program do té doby, dokud nebude vložen materiál a vypíše na počítači chybovou hlášku a zároveň výzvu, abychom vložili materiál k vyvrtání. Jakmile optické závory zaznamenají nějaký objekt v cestě, dojde ke spuštění programu a to pouze za podmínky, že je zavřené víko vyvrtávačky. Tím se dostávám k dalšímu přerušení, které okamžitě zastaví program, jakmile dojde k otevření víka vyvrtávačky. Program pak je možné opět spustit enterem, ovšem jen tehdy pokud je víko zavřené. Poslední přerušení bude vyvoláno v případě, že zadáme souřadnice, které překračují povolený rozsah. V tomto případě se program zastaví a odešle do počítače chybovou hlášku, že obsluha zadala chybné souřadnice. Tlakový senzor na vyvrtávačce hlídající tlak na vrtáku by vyvolával 2 druhy přerušení. V případě překročení horní hranice tlaku na vrták, dojde k přerušení, které vrátí vrták do nulové pozice a spustí program od začátku, aby nedošlo k poškození vrtáku. Druhé přerušení bude vyvoláno, pokud dojde k poklesu tlaku na vrták, programu bude předpokládat, že došlo k provrtání materiálu skrz. Krokový motor ovládající vertikální posun vrtáku, udělá ještě jeden krok a poté se vrátí do nulové pozice.

3.3 Vzorová řešení úloh

3.3.1 Úloha č. 1

Programování jednočipů Atmel AVR
Oblast: Digitální výstupy
Úloha č. 1 Téma: Blikající řada diod
Programovací jazyk: Bascom

Cíl:

Naučit se ovládat digitální výstupy. Při řešení je nutné respektovat proudovou zatížitelnost portu procesorů. Pro čipy AVR platí následující omezení:

jeden pin max. 20 mA, celý port max. 100 mA, celý čip max. 200 mA

Konfigurace čipu v programu, časové prodlevy. Naučit se ovládat programově digitální výstupy individuálně po jednotlivých pinech a celý jako celý port.

Použitý HW:

Procesorový modul

Modul LED

Technický rozbor úlohy:

U mikrokontrolérů AVR se pro ovládání I/O brány používá skupina registrů (kde n definuje název portů A, B, C a D):

`PORTn` – je datový registr, který odpovídá hodnotě, kterou zapisujeme na příslušný pin brány.

`PINn` – (Pins Input) tento registr je určen pro čtení a odpovídá logické úrovni signálu přečtené z příslušného pinu.

Do…Loop - je nekonečný cyklus, který jde přerušit podprogramem nebo přerušením.

Waitms - udává zpoždění, které se počítá podle frekvence nastavené na jednočipu.

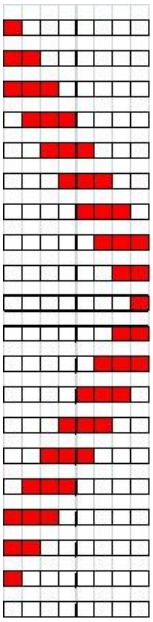
Select case - je instrukce, která se využívá na větvení podle proměnné. Použití je následující:

```
Select case proměnná
     Case proměnná
     Case proměnná + 1
     Case proměnná + 2
     Proměnná = proměnná + 1
End select
```

Zadání úlohy:

Máme řadu osmi diod. Vytvoříme úlohu tak, že budou diody cyklicky běhat z levé strany na druhou a zpět, přičemž maximálně mohou svítit pouze 3 najednou (tzv. Knight rider).

Ukázka simulace:



Postup řešení:

Nejprve je v programu nutné nadefinovat jednočip, který bude srdcem modulu a nastavit jeho pracovní frekvenci. V našem případě jsem využil čip m8535 s pracovní frekvencí 1Mhz.

Dále musíme vědět, jaký port máme k dispozici pro náš program. V tomto případě si můžeme vybrat ze všech 4 (A – D), protože program bude plnit pouze jednu funkci. Zvolil jsem tedy port A, který jsem pro náš případ nastavil jako výstupní, protože výstupem programu bude blikající led panel. Jakmile máme nastavený jednočip, jeho frekvenci a porty, přichází řada na definici proměnných. V tomto případě mi stačila pouze jedna proměnná (pocitadlo), podle její hodnoty se vybírají jednotlivé části programu. Po provedení definice proměnné jsem ještě provedl její nulování, aby nedošlo k nějakým chybám v programu. Jelikož mám zadáno, že celý program má být realizován v nekonečném cyklu, použil jsem tedy cyklus Do…Loop. V těle cyklu nejprve musím nastavit zvyšování pocitadla o jednu jednotku v každém průběhu cyklu. Následuje již samotná selekce. Selekce vybírá tedy podle hodnoty proměnné pocitadlo. V každé části jsem nastavil port a na požadovanou hodnotu. V části jedna na hodnotu nula, aby celý LED panel byl zhasnutý. V druhé části jsem port a nastavil na hodnotu 128 (tato hodnota odpovídá 2^7, tzn. jednička na pozici 7, bude tedy svítit poslední dioda LED panelu, tj. první zleva). V další části musím spočítat hodnotu pro rozsvícení první a druhé diody zleva. Následuje ještě jedna část, kde nastavím na port a hodnotu odpovídající rozsvíceným třem diodám z leva. V další části musím spočítat, kolikrát se musí tři rozsvícené diody posunout po panelu směrem doprava tak, aby zůstala svíti první dioda zprava. V dalších částech provedu to samé, ale z pravé strany. Nejprve zhasnu celý LED panel a postupně rozsvítím první tři diody na pravé straně. Následně opět budu posouvat hodnoty na portu a směrem doleva. Zde ovšem nastává změna, posun bude stejný jako v případě doprava, ale je nutné zařadit ještě jednu část s posunem doprava. V oné poslední části dojde k nulování proměnné pocitadlo, aby se program choval stejně jako po spuštění. Teprve tehdy ukončím selekci podle proměnné pocitadlo a nastavím zpoždění o 150ms. Úmyslně je zpoždění až po selekci, protože tímto krokem musí program projít pokaždé. Tím docílím úspory zdrojového kódu a ušetřím paměť programu. Nakonec ukončím nekonečný cyklus.

Zdrojový kód:

```
$crystal = 1000000                  ´nastavení frekvence čipu
$regfile = "m8535.dat"              ´výběr čipu použitého v úloze
$sim                                ´podpora simulace

Config Porta = Output               ´nastavení portu a jako
                                    ´výstupní
Dim Pocitadlo As Byte               ´definice proměnné pocitadlo
Pocitadlo = 0                       ´nastavení proměnné pocitadlo
                                    ´na hodnotu nula

Do                                  ´začátek cyklu

Pocitadlo = Pocitadlo + 1           ´inkrementace proměnné
Select Case Pocitadlo               ´selekce podle proměnné
                                    ´pocitadlo
    Case 1                          ´první pozice selekce
        Porta = 0                   ´nastaven portu a na nulu
    Case 2                          ´druhá pozice selekce
        Porta = 128                 ´nastavení na binární hodnotu
    Case 3                          ´odpovídající svícení pouze
        Porta = 192                 ´poslední diodě
    Case 4                          ´stejný postup s nastavením
        Porta = 224                 ´dle předlohy
```

```
      Case 5 To 11                  ´Od páté do jedenácté pozice
                                    ´selekce posouvej hodnoty na
         Shift Porta , Right        ´portu a směrem doprava
      Case 12                       ´dvanáctá pozice
         Porta = 0                  ´selekce, zhasnutí portu
      Case 13                       ´následně rozsvícení prvních
         Porta = 1                  ´tří diod, podle předlohy, opět
      Case 14                       ´pomocí binárního kódu
         Porta = 3
      Case 15
         Porta = 7
      Case 16 To 21                 ´od šestnácté do jednadvacáté
                                    ´pozice selekce posouvej port
         Shift Porta , Left         ´A směrem doleva
      Case 22                       ´dvaadvacátá pozice selekce
         Shift Porta , Left         ´posun portu a doprava
         Pocitadlo = 0              ´a zároveň nulování pocitadla
      End Select                    ´ukončení selekce
Waitms 150                          ´zpoždění mimo selekci
Loop                                ´ukončení cyklu
```

Závěr:

Nejprve jsem uvedený zdrojový kód testoval na simulátoru, kde bez problémů fungoval. Následně jsem ho testoval také na modulu LED panelu tak, že jsem přes programátor nahrál program do jednočipu a propojil s modulem. Program nejprve nefungoval, proto jsem hledal chybu nejprve v zapojení, když jsem chybu neodhalil, začal jsem kontrolovat zdrojový kód. Chybu jsem zjistil hned ve třetím řádku, zapomněl jsem vymazat instrukci pro simulátor a z toho důvodu si jednočip nedokázal poradit s programem. Jakmile jsem tuto instrukci odstranil a opět nahrál program do jednočipu, fungovalo vše podle předpokladů.

3.3.2 Úloha č. 3

Programování jednočipů Atmel AVR
Oblast: Digitální výstupy
Úloha č. 3Téma: Sedmisegmentovka
Programovací jazyk: Bascom

Cíl:

Naučit se ovládat digitální výstupy.

Použitý HW:

Procesorový modul

Modul sedmisegmentovky

Nově použité direktivy a programové struktury:

`If` - je instrukce, která větví program podle podmínky. Tvar je následující:

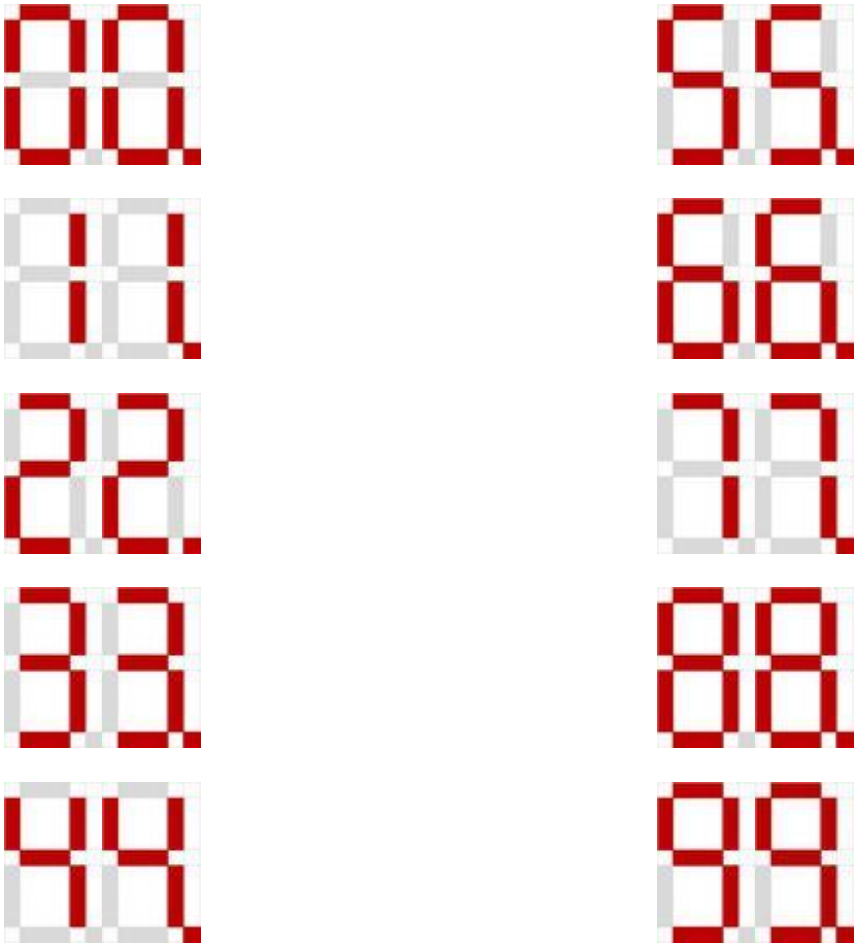
```
If podmínka then
…
Else
…
End if
```

`Toggle` - tato instrukce změní hodnotu pinu/bitové proměnné na opačnou. (A=1, po provedení této instrukce je A=0).

Zadání úlohy:

Máme sedmisegmentový display, který naprogramujeme tak, že nám postupně zobrazí všechna čísla (0-9). Změna bude po jedné sekundě, přičemž desetinná tečka bude mít 2 stavy (rozsvícená a zhasnutá) mezi těmito stavy bude rozestup pouze 0,5. Bez využití instrukce Select Case.

Ukázka simulace:



Postup řešení:

Nejprve je nutné v programu nadefinovat jednočip, který bude srdcem modulu a nastavit jeho pracovní frekvenci. V našem případě jsem využil čip m8535 s pracovní frekvencí 1Mhz. Pro zobrazování číslic jsem použil port A, který jsem nastavil jako výstupní. Protože budu ovládat poslední pin portu A samostatně, nadefinoval jsem si ho také jako výstupní. Dále přišla na řadu definice proměnných. Použil jsem proměnnou pocitadlo, podle které se bude určovat, které číslo bude zobrazeno na sedmisegmentovém displeji. Následně jsem proměnnou vynuloval a vynuloval jsem také hodnotu na pina.7, aby nesvítila desetinná tečka, jelikož je v zadání, že se čísla mají měnit v cyklu a není určen počet cyklů, musím využít nekonečný cyklus Do…Loop. Na začátku těla cyklu jsem vytvořil podmínku, že dojde k nulování proměnné pocitadlo, přesáhne-li její hodnota číslo devět. V zadání je uvedeno, že nemám využívat instrukci select case, proto musím využít podmínky IF. Využití této podmínky je velice jednoduché na logiku programu, nicméně trochu zvětšuje rozsah zdrojového kódu, ale zadání dodržím. Zařadil jsem tedy podmínku, že v případě kdy je pocitadlo rovno nule, tak na

port A přiřaď binární hodnotu odpovídající zobrazení čísla nula. Následuje podmínka, která přiřadí portu A binární hodnotu odpovídající zobrazení čísla jedna, pokud je pocitadlo rovno číslu jedna. Následují stejné podmínky pro všechny čísla (2 až 9), které se liší pouze binární hodnotou, která je přiřazována na port A, aby bylo vždy zobrazeno správné číslo. Jakmile mám připraveny všechny podmínky pro zobrazování jednotlivých číslic, zařadím inkrementaci proměnné pocitadlo, aby při každém průchodu cyklem bylo změněno zobrazované číslo. Jelikož má ještě podle zadání blikat desetinná tečka, zařadil jsem za inkrementaci proměnné obrácení hodnoty na pina.7, čímž dojde k rozsvícení desetinné tečky. Následně jsem zařadil zpoždění 500ms, což odpovídá přibližně 0,5s, jak bylo uvedeno v zadání. Následně musím diodu opět zhasnout a na to jsem použil opět stejný příkaz, který obrátí hodnotu na pina.7. Nakonec jsem použil opět zpoždění 500ms, aby byla prodleva mezi změnami hodnot na segmentu desetinné tečky odpovídající při obou změnách. Nakonec jsem ukončil nekonečný cyklus.

Zdrojový kód:

```
$crystal = 1000000                  ´nastavení frekvence čipu
$regfile = "m8535.dat"              ´výběr jednočipu

Config Porta = Output               ´nastavení portu A aPina.7
Config Pina.7 = Output              ´jako výstupní

Dim Pocitadlo As Byte               ´definice proměnné pocitadlo
Pocitadlo = 0                       ´nulování proměnné pocitadlo
Pina.7 = 0                          ´nulování pina.7

Do                                  ´začátek cyklu
      If Pocitadlo> 9 Then          ´podmínka, je-li pocitadlo
            Pocitadlo = 0           ´větší než 9, vynuluj
      End If
```

```
      If Pocitadlo = 0 Then          ´podmínka, je-li pocitadlo
            Porta = &B00111111       ´rovno nule, na port A nastav
                                     ´hodnotu
      End If
      If Pocitadlo = 1 Then
                                     ´podmínka, je-li pocitadlo
            Porta = &B00000110       ´rovno jedné, na port A nastav
      End If                         ´hodnotu
      If Pocitadlo = 2 Then          ´podmínka, je-li pocitadlo
                                     ´rovno dvěma, na port A  nastav
            Porta = &B01011011       ´hodnotu
      End If
                                     ´podmínka, je-li pocitadlo
      If Pocitadlo = 3 Then          ´rovno třem, na port A nastav
            Porta = &B01001111       ´hodnotu
      End If
      If Pocitadlo= 4 Then           ´podmínka, je-li pocitadlo
                                     ´rovno čtyřem, na port A nastav
            Porta = &B01100110       ´hodnotu
      End If
                                     ´podmínka, je-li pocitadlo
      If Pocitadlo = 5 Then          ´rovno pěti, na port A nastav
            Porta = &B01101101       ´hodnotu
      End If
      If Pocitadlo = 6 Then          ´podmínka, je-li pocitadlo
            Porta = &B01111101       ´rovno šesti, na port A nastav
      End If                         ´hodnotu
      If Pocitadlo = 7 Then
                                     ´podmínka, je-li pocitadlo
            Porta = &B00000111       ´rovno sedmi, na port A nastav
      End If                         ´hodnotu
      If Pocitadlo = 8 Then          ´podmínka, je-li pocitadlo
            Porta = &B01111111       ´rovno osmi, na port A nastav
                                     ´hodnotu
End If
```

```
        If Pocitadlo = 9 Then          ´podmínka, je-li pocitadlo rovno
              Porta = &B01101111       ´devíti, na port A nastav hodnotu
        End If
        Pocitadlo = Pocitadlo + 1      ´inkrementace proměnné
        TogglePina.7                   ´obrácení hodnoty na pina.7
        Waitms 500                     ´zpoždění 500ms
        TogglePina.7                   ´obrácení hodnoty na pina.7
        Waitms 500                     ´zpoždění 500ms
Loop                                   ´konec cyklu
```

Závěr:

Vzniklý program jsem testoval na simulátoru prostředí BASCOM, jenž fungoval přesně dle předpokladů. Následně jsem program testoval také na modulu sedmisegmentového displeje, kde se čísla bohužel nezobrazovala korektně. Chyba byla ve špatném zadávání binárních hodnot, které byly posílány na port A k zobrazení. Na modulu se segment rozsvěcí nulou, zatímco v simulátoru jedničkou, a proto jsem změnil hodnoty tak, aby odpovídaly správnému zobrazení číslic. Program teprve potom pracoval správně.

3.3.3 Úloha č. 5

Programování jednočipů Atmel AVR
Oblast: Digitální výstupy
Úloha č. 5 Téma: Sedmisegmentovka – rozšíření 1
Programovací jazyk: Bascom

Cíl:

Naučit se ovládat digitální výstupy.

Použitý HW:

Procesorový modul

Modul 4x Sedmisegmentovka

Technický rozbor úlohy:

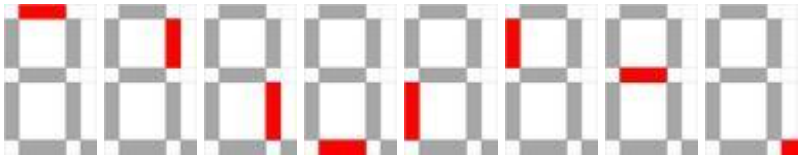
`Set` – nastaví daný port/pin na 1

`Reset` – nastaví daný port/pin na 0

Zadání úlohy:

Na panelu 4 sedmisegmentovek rozsviťte pouze na prvním z displejů postupně všechny segmenty ve směru hodinových ručiček s tím, že svítit smí pouze jeden segment. Následně to samé na druhém, třetím a čtvrtém displeji a to vše opakovat v cyklu. Každý z displejů se rozsvítí přivedením 0 na příslušný spínací tranzistor, který je připojený na jeden konkrétní pin portu (např. portc.3), avšak aktivní může být pouze jeden.

Ukázka simulace:



Postup řešení:

Nejprve je v programu nutné nadefinovat jednočip, který bude srdcem modulu a nastavit jeho pracovní frekvenci. V našem případě jsem využil čip m8535 s pracovní frekvencí 1Mhz. Pro ovládání zobrazování na sedmisegmentovém panelu jsem v tomto programu použil opět port A, který jsem nastavil jako výstupní, protože je používán pro zobrazovací účely. Pro ovládání spínacích tranzistorů (které určují, přivedením nuly na příslušný spínací tranzistor, jaký ze sedmisegmentových displejů bude svítit) jsem využil port C. Pro každý jednotlivý spínací tranzistor, jsem použil jeden pin a to konkrétně piny nula až tři. Všechny jednotlivé používané piny portu C jsem nastavil jako výstupní, protože slouží pouze ke spínání tranzistorů. Dále jsem provedl definici jednotlivých proměnných. První jsem zařadil proměnnou selekce, která mi určí, jaký spínací tranzistor bude aktivní. Druhou proměnnou je pocitadlo, které mi určí, jaký ze segmentů se aktuálně rozsvítí. Třetí proměnnou je cislo, které je využívané jako zástupný symbol pro nastavení hodnoty odpovídající rozsvícení jednoho ze segmentů displeje. Poslední proměnnou je I, která slouží jako zástupný symbol v cyklu FOR. Jedná se o cyklus s pevně daným počtem průběhů. Následně jsem standardně všechny proměnné vynuloval. Všechny používané piny portu C jsem nastavil na jedničku, pomocí instrukce SET. Dále již začíná hlavní nekonečný cyklus Do…Loop. Na začátku cyklu dojde k nulování proměnné selekce, za podmínky, že její hodnota je vyšší než tři. Stejně tak dojde k nulování proměnné pocitadlo, za podmínky, že její hodnota je vyšší než sedm. Těmito podmínkami jsem si ošetřil to, aby proměnné selekce vždy vybrala jeden ze čtyř displejů (0-3) a proměnná pocitadlo vždy zvolila jednu hodnotu, která se zobrazí na displeji (0-7). Jakmile jsem vyřešil hranice, pustil jsem se do samotné selekce displejů. Celá selekce probíhá pomocí proměnné selekce a v každé ze čtyř částí selekce je jeden port pomocí instrukce RESET nastaven na nulu a zároveň jemu předcházející pin nastaven pomocí instrukce SET na jedničku, aby se nestalo, že budou funkční dva displeje sedmisegmentovky. Po ukončení volby spínacích tranzistorů jsem provedl inkrementaci proměnné selekce, aby byl v příštím průběhu cyklu aktivní displej následující. V tomto bodě programu přichází na řadu cyklus FOR, který využije proměnnou i a počet průběhů jsem nastavil na hodnotu osm. Do těla tohoto cyklu jsem zařadil nejprve selekci podle proměnné pocitadlo a je zde vybíráno, který ze segmentů

bude svítit. Vždy jsem hodnotu zadával binárně a zapsal do proměnné cislo. Po ukončení této selekce jsem vytvořil jednotlivé podmínky pro rozsvícení jednotlivých displejů sedmisegmentovek. V případě kdy je některý z pinů portu C roven nule, přivede se na port a hodnota proměnné cislo. Ještě před ukončením cyklu FOR jsem provedl inkrementaci proměnné pocitadlo a zařadil zpoždění 150ms. Zpoždění jsem zařadil pouze jedno a úmyslně až na toto místo, aby jím prošel program v každém průběhu cyklu, a tím jsem ušetřil spoustu zdrojového kódu. Po ukončení cyklu FOR jsem ukončil také nekonečný cyklus Do…Loop. Tím, že jsem vložil cyklus FOR do nekonečného cyklu Do…Loop, jsem zajistil, že obsah cyklu FOR proběhne podle nastaveného počtu průběhů a tyto všechny průběhy budou odpovídat jednomu průběhu cyklem Do…Loop. Proto vždy oběhnou všechny segmenty na displeji a až potom se rozsvítí následující displej.

Zdrojový kód:

```
$crystal = 1000000                  ´nastavení frekvence čipu
$regfile = "m8535.dat"              ´výběr jednočipu


Config Porta = Output               ´nastavení portu a jako
Config Portc.0 = Output             ´výstupní
Config Portc.1 = Output             ´nastavení jednotlivých
Config Portc.2 = Output             ´používaných pinů portu C jako
Config Portc.3 = Output             ´výstupní


Dim Selekce As Byte                 ´definice proměnné selekce
Dim Pocitadlo As Byte               ´definice proměnné pocitadlo
Dim Cislo As Byte                   ´definice proměnné cislo
Dim i As Byte                       ´definice proměnné I
I = 0                               ´nulování proměnných
Pocitadlo = 0
Selekce = 0
Cislo = 0
```

```
Set Portc.0                          ´nastavení 1 na portc.0
Set Portc.1                          ´nastavení 1 na portc.1
Set Portc.2                          ´nastavení 1 na portc.2
Set Portc.3                          ´nastavení 1 na portc.3


Do                                   ´začátek cyklu
  If Selekce > 3 Then                ´pokud selekce přesáhne hodnotu
    Selekce = 0                      ´tři, proveď nulování
  End If
    If Pocitadlo > 7 Then
    Pocitadlo = 0                    ´pokud pocitadlo přesáhne
  End If                             ´hodnotu sedm, proveď nulování
  Select Case Selekce                ´začátek selekce
    Case 0                           ´první displej
      Reset Portc.0                  ´nula na portc.0
      Set Portc.3                    ´jednička na portc.3
    Case 1                           ´druhý displej
      Reset Portc.1
      Set Portc.0
    Case 2                           ´třetí displej
      Reset Portc.2
      Set Portc.1
    Case 3                           ´čtvrtý displej
      Reset Portc.3
      Set Portc.2
  End Select                         ´konec selekce
  Incr Selekce                       ´inkrementace proměnné
  For i = 1 To 8                     ´začátek FOR s 8 průběhy
    Select Case Pocitadlo            ´začátek selekce
      Case 0                         ´první segment
        Cislo = &B11111110           ´binární hodnota segmentu
      Case 1                         ´druhý segment
        Cislo = &B11111101
```

```
      Case 2                          ´třetí segment
        Cislo = &B11111011
      Case 3                          ´čtvrtý segment
        Cislo = &B11110111
      Case 4                          ´pátý segment
        Cislo = &B11101111
      Case 5                          ´šestý segment
        Cislo = &B11011111
      Case 6                          ´sedmý segment
        Cislo = &B10111111
      Case 7                          ´osmý segment
        Cislo = &B01111111
    End Select                        ´konec selekce
    If Portc.0 = 0 Then               ´pokud je na portc.0
      Porta = Cislo                   ´nula, zobraz cislo - segment
    End If
    If Portc.1 = 0 Then               ´pokud je na portc.1
      Porta = Cislo                   ´nula, zobraz cislo - segment
    End If
    If Portc.2 = 0 Then               ´pokud je na portc.2
      Porta = Cislo                   ´nula, zobraz cislo - segment
    End If
    If Portc.3 = 0 Then               ´pokud je na portc.3
      Porta = Cislo                   ´nula, zobraz cislo - segment
    End If
    Incr Pocitadlo                    ´inkrementace proměnné
    Waitms 150                        ´zpoždění 150ms
  Next                                ´konec cyklu FOR
Loop                                  ´konec nekonečného cyklu
```

Závěr:

Tento program jsem netestoval na simulátoru, protože v simulátoru není zobrazení displeje sedmisegmentovky. Testování probíhalo tedy přímo na modulu sedmisegmentovek. V prvním testování jsem měl opačně nastavené binární hodnoty posílané na displej. Svítili mi všechny segmenty, kromě jednoho, který se posouval ve směru hodinových ručiček, tedy přesně opačně než jsem potřeboval, tak jsem musel v programu přepsat binární hodnoty posílané na port A. V tomto případě se totiž rozsvítil segment přivedením nuly na příslušnou pozici a přivedením jedničky byl segment zhasnutý. Po této opravě již fungovalo vše dle předpokladů.

3.3.4 Úloha č. 10

Programování jednočipů Atmel AVR
Oblast: Digitální výstupy
Úloha č. 10Téma: LCD panel – 1. rozšíření
Programovací jazyk: Bascom

Cíl:

Naučit se ovládat digitální výstupy.

Použitý HW:

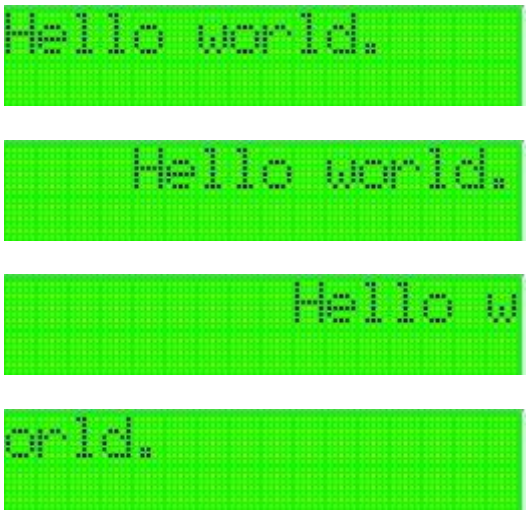
Procesorový modul

Modul LCD

Zadání úlohy:

Vytvořte úlohu tak, aby se na displeji zobrazil řetězec textu „Hello World.“. Tento řetězec se bude posouvat směrem doprava a vysune se na levé straně a to vše v cyklu. Proveďte úpravu posunu tak, aby nebylo zpoždění mezi zmizením textu z pravé strany a objevením se textu na levé straně.

Ukázka Simulace:



Postup řešení:

Nejprve je v programu nutné nadefinovat jednočip, který bude srdcem modulu a nastavit jeho pracovní frekvenci. V našem případě jsem využil čip m8535 s pracovní frekvencí 1Mhz. Následně jsem si vybral z možností, jaký rozměr LCD displeje budu používat. Zvolil jsem si displej o dvou řádcích o délce šestnácti znaků. Konfigurace jednotlivých pinů musí vždy probíhat podle zapojení LCD displeje na modulu. V simulátoru není zapojení úplně podstatné. U zapojení jednotlivých pinů jsem musel v mém případě počítat se čtyřmi datovými piny, jedním pinem pro Enable (E) a druhým pro reset (RS). Dále jsem si nadefinoval proměnné. V tomto programu využívám pouze jednu proměnnou, a to proměnnou I, která slouží jako zástupný symbol v cyklu s pevně daným počtem průběhů, tedy v cyklu FOR. Následně jsem proměnnou vynuloval, aby nedošlo k chybám. Na dalším řádku programu jsem vymazal obsah zobrazovaný na LCD displeji, opět kvůli prevenci chyb. Dále jsem na LCD displej poslal text, který chci rotovat („Hello World.“). Následuje již samotné posouvání textu směrem doprava. Řešení tohoto problému nebylo jednoduché, protože při běžném posouvání, vznikala prodleva mezi zmizením textu na pravé straně a objevením se textu na levé straně. Tuto chybu způsoboval buffer. Musel jsem tedy zjistit, jaké je délka bufferu. Postupně jsem tedy přidával znaky a bez úpravy posunu je rotoval směrem doprava. Jakmile mi znaky rotovaly v přesném kruhu, zjistil jsem přesnou délku bufferu. V mém případě čtyřicet. Problém v úpravě posunu byl ještě jeden, protože při spuštění textu nezačíná text na levé straně, ale uprostřed displeje. Po zobrazení textu na LCD displeji jsem tedy vytvořil cyklus FOR se šestnácti průběhy, protože přesně tolikrát je třeba posunout text „Hello World“ po LCD displeji směrem doprava, než zmizí na pravé straně i poslední znak. V těle cyklu je tedy pouze posun obsahu LCD displeje doprava a zpoždění v tomto případě 300ms. Za tímto cyklem následuje další cyklus FOR s délkou šestnáct až dvacet šest, který pokračuje v posouvání textu směrem doprava, ovšem bez zpoždění bude přeskočena nepotřebná část bufferu. Tímto mám vyřešený problém s rotováním textu po spuštění programu. Následuje nekonečný cyklus Do…Loop. Na jeho začátku jsem musel vynulovat proměnnou I, která se bude využívat v následujících cyklech. V jeho těle se nacházejí opět dva cykly FOR. První cyklus, který posouvá text po LCD displeji. Na levé straně se text objeví a na pravé mizí. Opět jsem do tohoto cyklu zařadil zpoždění 300ms. Počet průběhů tohoto cyklu odpovídá hodnotě dvacet osm. Druhý cyklus FOR se stará o přeskočení nepotřebné délky bufferu. V tomto případě je hodnota průběhů v rozmezí dvacet osm až třicet devět (což odpovídá délce

bufferu, neboť proměnná I se počítá od nuly). Po ukončení tohoto druhého cyklu FOR, jsem ukončil také nekonečný cyklus Do…Loop.

Zdrojový kód:

```
$crystal = 1000000                    ´nastavení frekvence jednočipu
$regfile = "m8535.dat"                ´výběr jednočipu


Config Lcd = 16 * 2                   ´nastavení rozměrů LCD
Config Lcdpin = Pin , Db4 =           ´nastavení jednotlivých pinů
Porta.0 , Db5 = Porta.1 , Db6 =       ´LCD displeje podle zapojení na
Porta.2 , Db7 = Porta.3 , E =         ´použitém modulu
Porta.4 , Rs = Porta.5
Dim I As Byte                         ´definice proměnné I
I = 0                                 ´nulování proměnné I
Cls                                   ´vymazání LCD displeje
Lcd "Hello world."                    ´zobrazení textu na LCD
For I = 1 To 16                       ´první cyklus for, posouvající
      Shiftlcd Right                  ´text od středu do prava, dokud
                                      ´nezmizí všechny symboly
      Waitms 300                      ´zpoždění 300 ms
Next                                  ´konec cyklu for
For I = 16 To 26                      ´druhý cyklus for, posouvající
      Shiftlcd Right                  ´text bez zpoždění zbytkem
                                      ´bufferu
Next                                  ´konec druhého cyklu
Do                                    ´začátek nekonečného cyklu
      I = 0                           ´nulování proměnné I
      For I = 1 To 28                 ´první cyklus FOR, ve kterém se
            Shiftlcd Right            ´text objeví na levé straně a
                                      ´zmizí na pravé
            Waitms 300                ´zpoždění 300 ms
      Next                            ´konec cyklu
```

```
        For I = 28 To 39                    ´druhý cyklus for, posouvající
                                            ´text bez zpoždění zbytkem
              Shiftlcd Right                ´bufferu

        Next                                ´konec cyklu FOR

Loop                                        ´konec nekonečného cyklu
```

Závěr:

Tento program jsem se pokoušel testovat v simulátoru, nicméně simulátor zobrazuje text na začátku LCD displeje a nepodporuje buffer, proto toto testování bylo poněkud zbytečné. Jakmile jsem přešel k testování na LCD modulu, musel jsem odladit délku bufferu, jak zmiňuji výše. Poté, co jsem vypočítal přesné rozsahy cyklů FOR, fungoval program přesně dle předpokladů.

3.3.5 Úloha č. 11

Programování jednočipů Atmel AVR
Oblast: Digitální výstupy
Úloha č. 11 Téma: Blikající řada diod – volba módů
Programovací jazyk: Bascom

Cíl:

Naučit se ovládat digitální výstupy.

Použitý HW:

Procesorový modul

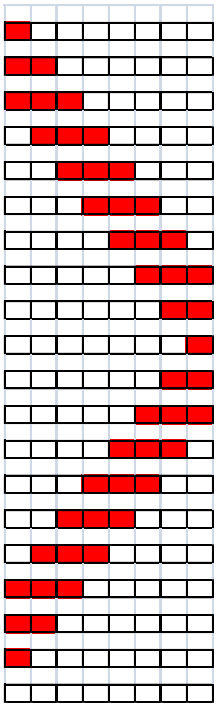
Modul LED

Zadání úlohy:

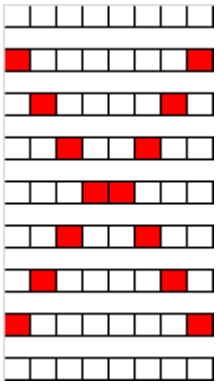
Vytvoříme úlohu tak, abychom přepínači 1 a 2 (pinb.5 a pinb.6) zvolili příslušnou kombinaci, jak se budou diody rozsvěcet. V první pozici, budou diody cyklicky běhat z levé strany na druhou a zpět, přičemž maximálně můžou svítit pouze 3 najednou (tzv. Knight rider). V druhé pozici rozsvítíme jednu diodu na obou stranách a následně je v cyklu posouváme směrem do středu, kde se „srazí“ a otočí se zpátky. Třetí pozicí bude jednoduché blikání celého led panelu. Poslední čtvrtá pozice pouze rozsvítí celý LED panel.

Ukázka simulace:

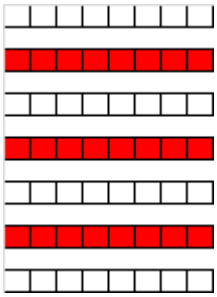
1.mód



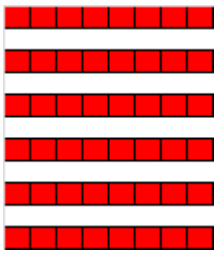
2.mód



3.mód



4.mód



Postup řešení:

Nejprve je v programu nutné nadefinovat jednočip, který bude srdcem modulu a nastavit jeho pracovní frekvenci. V našem případě jsem využil čip m8535 s pracovní frekvencí 1Mhz. V zadání mám přesně určeno, že dva piny portu B budou vstupní. Nastavím tedy nejprve piny B 5 a 6 jako vstupní a celý port a jako výstupní. Na portu a bude připojený modul s LED panelem. Dále musím nadefinovat proměnné, kterých v tomto programu budu potřebovat hned několik. Využil jsem proměnnou I, podle které se bude určovat mód, ve kterém bude LED panel blikat. Další proměnnou je proměnná pocitadlo, která bude sloužit u výběru částí programu při běhu prvního módu. Další proměnné budou P1 a P2, které jsou využívány ve druhém módu programu. Následuje nulování všech proměnných kromě proměnné I. Ze zadání plyne, že musím použít nekonečný cyklus, tedy cyklus Do…Loop. V samém začátku těla cyklu musí být proveden výpočet, ze kterého zjistím, jaký mód bude spuštěn. Podle zadání volbu módu provádím kombinací na tlačítkách, které jsou připojeny na pinb.6 a pinb.5. z tohoto důvodu musím zařadit nulování proměnné až nyní na začátek výpočtu. K proměnné i nejprve přičtu hodnotu na pinb.6, následně vynásobím obsah proměnné i dvěma, abych se při stisku toho tlačítka dostal do pozice dvě. Teprve potom k hodnotě proměnné i přičítám hodnotu proměnné pinb.5. Mechanismus znázorňuje následující tabulka:

Pinb.6	Pinb.5	I
0	0	0
0	1	1
1	0	2
1	1	3

Jakmile mám vyřešeno, jakým způsobem budou vybírány jednotlivé módy, zbývá mi už jenom napsat samotnou selekci podle proměnné I. Pro první (tedy nultý) mód jsem použil zdrojový kód z úlohy č. 1. Pro druhý mód jsem již musel vymyslet algoritmus. Zde jsem do portu a zapsal hodnoty proměnný P1 a P2, přičemž jsem využil instrukce Or, aby došlo k využití obou proměnných zároveň. Dále jsem zařadil podmínku, pokud bude P1 rovno nule, tak do P1 zapiš binární hodnotu pro rozsvícení první diody vlevo a do P2 zapiš binární hodnotu pro rozsvícení první diody vpravo. Pokud nebude podmínka splněna (což po prvním průchodu nebude) posuň P1 doprava a P2 doleva. Nyní je na řadě třetí mód, tedy blikání led

panelu. Toto jsem provedl velice jednoduše. Nastavil jsem nejprve port a na hodnotu 255 (svítí celý port), dále zařadil zpoždění 150ms a poté nastavil port a na nulu (celý port zhasne). V posledním módu, kdy má celý LED panel pouze svítit, jsem na port a nastavil hodnotu 255. Tímto mám hotové všechny módy a končí celková selekce, nyní teprve zařadím zpoždění 150ms, protože tímto bodem musí projít každý průběh cyklu a ušetřím tím spoustu zdrojového kódu a tím pádem i místo v paměti programu. Nakonec ukončím nekonečný cyklus.

Zdrojový kód:

```
$regfile = "m8535.dat"                    ´výběr jednočipu
$crystal = 1000000                        ´nastavení frekvence jednočipu
$sim                                      ´instrukce pro podporu simulace
Config Porta = Output                     ´nastavení portu a jako
                                          ´výstupní
Config Pinb.6 = Input                     ´nastavení pinb.6 a pinb.5
Config Pinb.5 = Input                     ´jako vstupní
Dim i As Byte                             ´definice proměnné I
Dim Pocitadlo As Byte                     ´definice proměnné pocitadlo
Dim P1 As Byte                            ´definice proměnných P1 a P2
Dim P2 As Byte
P1 = 0                                    ´nulování proměnných P1, P2
P2 = 0                                    ´a pocitadlo
Pocitadlo = 0


Do                                        ´začátek cyklu
  I = 0                                   ´nulování proměnné I
  I = i + Pinb.6                          ´výpočet volby podle hodnot na
  I = i * 2                               ´vstupních tlačítkách
  I = i + Pinb.5
```

```
Select Case I                          ´začátek hlavní selekce
  Case 0                               ´první mód
  Pocitadlo = Pocitadlo + 1            ´inkrementace proměnné
  Select Case Pocitadlo                ´start selekce podle proměnné
                                       ´pocitadlo
   Case 1                              ´rozsvícení první diody
     Porta = 128
   Case 2                              ´rozsvícení druhé diody
     Porta = 192
   Case 3                              ´rozsvícení třetí diody
     Porta = 224
   Case 4 To 10                        ´následujících šest
     Shift Porta , Right               ´částí, posun celého portu
   Case 11                             ´a vpravo
     Porta = 0                         ´zhasnutí celého LED ´panelu
   Case 12
     Porta = 1                         ´rozsvícení první diody
   Case 13
     Porta = 3                         ´rozsvícení druhé diody
   Case 14
     Porta = 7                         ´rozsvícení třetí diody
   Case 15 To 21                       ´následujících šest
     Shift Porta , Left                ´částí, posun celého portu
                                       ´a vlevo
   Case 22                             ´poslední posun vlevo a zároveň
     Shift Porta , Left                ´nulování pocitadla
     Pocitadlo = 0
  End Select                           ´konec selekce 1. Módu
```

```
    Case 1                          ´druhý mód
     Porta = P1 Or P2               ´nastavení P1 nebo P2 na port A
     If P1 = 0 Then                 ´pokud je P1 rovno nule, nastav
       P1 = &B10000000              ´na P1 a P2 následující
       P2 = &B00000001              ´hodnoty
     Else
       Shift P1 , Right             ´pokud není P1 rovno
       Shift P2 , Left              ´nule, posuň hodnotu P1 doprava
     End If                         ´a hodnotu P2 doleva
    Case 2                          ´konec podmínky
     Porta = 255                    ´třetí mód
     Waitms 150                     ´rozsvícení portu A
     Porta = 0                      ´nastavení zpoždění
    Case 3                          ´zhasnutí portu A
    Porta = 255                     ´čtvrtý mód
    End Select                      ´rozsvícení portu a
  Waitms 150                        ´konec hlavní selekce
Loop                                ´hlavní zpoždění
                                    ´konec cyklu
```

Závěr:

Nejprve jsem uvedený zdrojový kód testoval na simulátoru, kde bez problémů fungoval. Následně jsem ho testoval také na modulu LED panelu tak, že jsem přes programátor nahrál program do jednočipu a propojil s modulem LED panelu a s modulem s tlačítky. Program fungoval, ale nebylo možné přepínat mezi jednotlivými módy, proto jsem hledal chybu nejprve v zapojení. Chybu jsem udělal já sám, protože jsem využíval špatná tlačítka. Neprošel jsem si pořádně způsob zapojení na modulu s tlačítky, a proto jsem měl zapojená jiná tlačítka, než jsem očekával. Jakmile jsem pro volbu módu využíval připojená tlačítka, vše fungovalo dle předpokladů.

4 Závěr

Tato bakalářská práce se zabývá programováním jednočipových mikropočítačů. V teoretické části je nastíněna historie vzniku jednočipových mikropočítačů a také jejich vlastnosti. Dále je v této části popsán programovací jazyk BASCOM, jeho vývojové prostředí a výhody oproti jiným vyšším programovacím jazykům.

Cílem praktické části bylo navrhnout soubor úloh, které budou využívány převážně pro výuku programování jednočipových mikropočítačů. Proto jsou úlohy situovány tak, aby byla postupně zvyšována náročnost. Celkem jsem připravil 25 úloh, jejich pracovní protokoly a vzorová řešení.

5 Reference

[1] *Mikrokontroléry PIC* [online]. 2012 [cit. 2013-09-23]. Dostupné z: http://mikrokontrolery-pic.cz/zaciname/mikroprocesor-mikropocitac-mikrokontroler/

[2] *TECH1: Von Neumannovské a Harvardské schéma počítače, popis, funkce.* [online]. [cit. 2013-12-27]. Dostupné z: http://ai-fim-uhk.wikispaces.com/TECH1

[3] NOVOTNÝ, J. a J. GERBICH. *Mikropočítače* [online]. 1987 [cit. 2013-10-05]. Dostupné z: http://www.ics.muni.cz/25let/galerie/clanky/mikropocitace/skripta-24.html

[4] *SM-lomos.cz: Bascom* [online]. 2008 [cit. 2013-12-27]. Dostupné z: http://www.sm-lomos.cz/sluzby/bascom/

[5] *Stránky o elektronice pro elektrotechniky: Bascom* [online]. 2008 [cit. 2013-12-27]. Dostupné z: http://www.bretakral.eu/pirkl/hl_bascom.html

[6] *Tutorial - 1. část (co je assembler)* [online]. 2004 [cit. 2013-12-27]. Dostupné z: http://assembler.unas.cz/tutor.htm

[7] *NarrowFreeweb: CodeVision AVR - programování atmel v C* [online]. 2011 [cit. 2013-12-27]. Dostupné z: http://narrow.vzap.eu/elektrotechnika/atmely/44-codevision-avr-programovani-atmel-v-c

[8] *MCS electronic* [online]. 2013 [cit. 2013-12-27]. Dostupné z: http://www.mcselec.com/index.php?option=com_content&task=view&id=14&Itemid=41

[9] *Mikrokontroléry Atmel AVR: Programování v jazyce BASCOM*. Praha: Technická Literatura BEN, 2004. ISBN 80-7300-115-2.

[10] *Programingmicrocontrollers: Co je to programátor?* [online]. 2013 [cit. 2013-12-27]. Dostupné z: http://www.mcontrollers.com/programator.html

[11] *HW.cz: AVR STK500 Starterkit - jednoduchý začátek s AVR* [online]. 2012 [cit. 2013-12-27]. Dostupné z: http://www.hw.cz/teorie-a-praxe/uplne-neprakticke-veci/avr-stk500-starter-kit-jednoduchy-zacatek-s-avr.html

[12] *OnPa: Žádný začátek ještě nebyl tak lehký* [online]. 2010 [cit. 2013-12-27]. Dostupné z: http://shop.onpa.cz/?kit-evb-4.3,27

[13] *Czechduino.cz: Co je Arduino* [online]. 2012 [cit. 2013-12-27]. Dostupné z: http://www.czechduino.cz/?co-je-to-arduino,29

[14] *MLAB online: o projektu MLAB* [online]. 2013 [cit. 2013-12-27]. Dostupné z: http://www.mlab.cz/Web/About.cs.html

[15] *PaJa-trb: Automatický panákovač* [online]. 2012 [cit. 2013-12-28]. Dostupné z: http://paja-trb.cz/konstrukce/panakovac.html