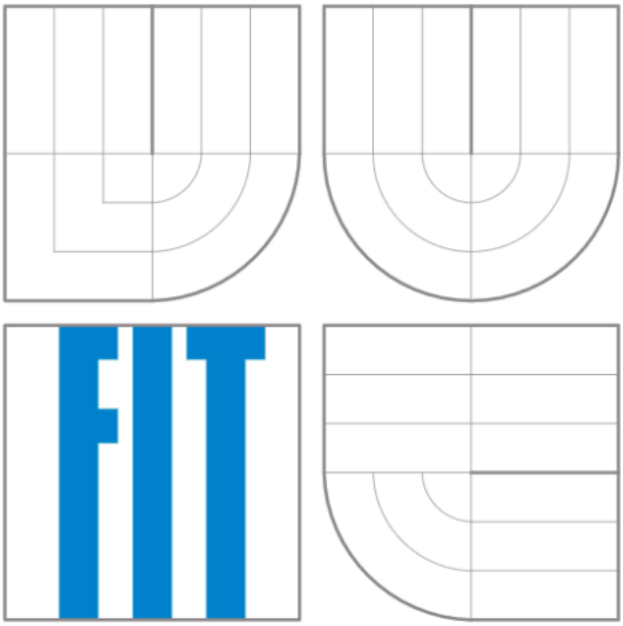
Vysoké učení technické v Brně
Fakulta informačních technologií



Dokumentace k projektu do předmětu IMP

Simulace v CW: Světelné noviny

15. prosince 2016

Autoři: Jakub Vitásek, xvitas02@stud.fit.vutbr.cz
Fakulta informačních technologií
Vysoké učení technické v Brně

Obsah

1 Úvod

V této práci je řešena implementace simulace světelných novin ve vývojovém prostředí Freescale CodeWarrior. Cílem projektu je vypisovat login na displej složený z LED uspořádaných do matice o 8 řádcích a 32 sloupcích, přičemž uživatel pomocí nástroje Visualization Tool ovládá chování rotace loginu na displeji.

Uživatel má k dispozici inicializační tlačítko (tlačítko `init`), které umístí login na výchozí pozici. Pro spuštění animace lze vybírat mezi horizontální (tlačítko `horiz`) a vertikální (tlačítko `vert`) rotací, jejíž rychlost může uživatel určovat pomocí táhla rychlosti.

2 Implementace

Samotnou implementace projektu lze rozdělit do několika logických celků. Každý z nich je detailně popsaný níže.

2.1 Konfigurace

Zdrojem projektu je soubor obsahující konfiguraci rozložení prvků pro nástroj Visualization Tool. Obsahuje specifikaci komponent a jejich umístění na pracovní ploše. Pro účely projektu je nicméně nutné komponenty namapovat na zvolené porty. Vzhledem k počtu LED na displeji je vhodné zvolit různé bitové hladiny pro jednotlivé řádky. Jako dedikované porty pro LED jsem určil 0xDx a 0xEx, kde x jsou indexy jednotlivých diod na každé hladině.

Tlačítka jsou umístěny na portech 0xF1 – 0xF3, táhlo rychlosti na 0xF0.

2.2 Reprezentace loginu

Každý sloupec diod lze vyjádřit jako jeden bajt, v kterém každý bit znamená jeden řádek. Tento princip jsem aplikoval převodem každého sloupce z binárního čísla na dekadické a následné uložení do paměti v rámci struktury pole login.

2.3 Inicializace

Použití tlačítka `init` způsobí navrácení do výchozího stavu programu, kdy jsou na displeji pouze 4 znaky loginu počítané od jeho začátku. Tento výchozí stav lze reaktivovat kdykoliv během chodu programu – ukončí veškerou rotaci a umístí login zpět na prvotní pozici.

Toho je docíleno pomocí dvojitého nekonečného cyklu ve funkci `main`, kde se v druhé vrstvě provádí rotace a v případě aktivního tlačítka `init` se druhý cyklus přeruší a vrátí řízení první vrstvě, v které je se pozice nastaví na výchozí hodnoty, dokud není aktivováno jiné tlačítko.

2.4 Horizontální rotace

Zadání specifikuje, že se horizontální rotace v případě loginu končícího na jiné dvoučíslí než 00 bude provádět zleva doprava. Rotaci lze tedy implementovat cyklem procházejícím od nejvyšší hodnoty šírky displeje až po nulu, v každém průchodu dekrementující iterátor a vyhledávání hodnoty pro danou diodu v poli loginu. Modulo délky loginu zajistí, že program nebude sahat mimo paměť pole.

2.5 Vertikální rotace

Dle zadání bylo nutné implementovat také vertikální rotaci, která v případě loginu končícího na jiné dvoučíslí než 00/01 probíhá zdola nahoru. Základem funkce je stejný cyklus jako v horizontální rotaci, s rozdílem, že se řízení dále zanořuje do sekundárního cyklu, v kterém je pro každý řádek původní vzorek posunut operací left shift. Po vynoření z obou cyklů je opět zařazeno zpoždění v podobě busy wait.

2.6 Opoždění

Funkce táhla ovlivňujícího rychlost rotace lze implementovat metodou busy wait, kdy na základě hodnoty táhla určíme, jak dlouho bude probíhat zpožďovací cyklus. V cyklu provádíme operace bez výstupu, které adekvátně zpomalují provedení jedné iterace v rámci funkce rotace.

3 Dekompozice

Po definici potřebných konstant a globálních proměnných program začíná ve funkci `main`, kde inicializuje horizontální/vertikální indexy používané napříč oběma cykly v těle funkce `main`. Zároveň nastavuje tlačítka na nulový stav, tedy vypnuto, a rychlost rotace na výchozí hodnotu (určil jsem 1200).

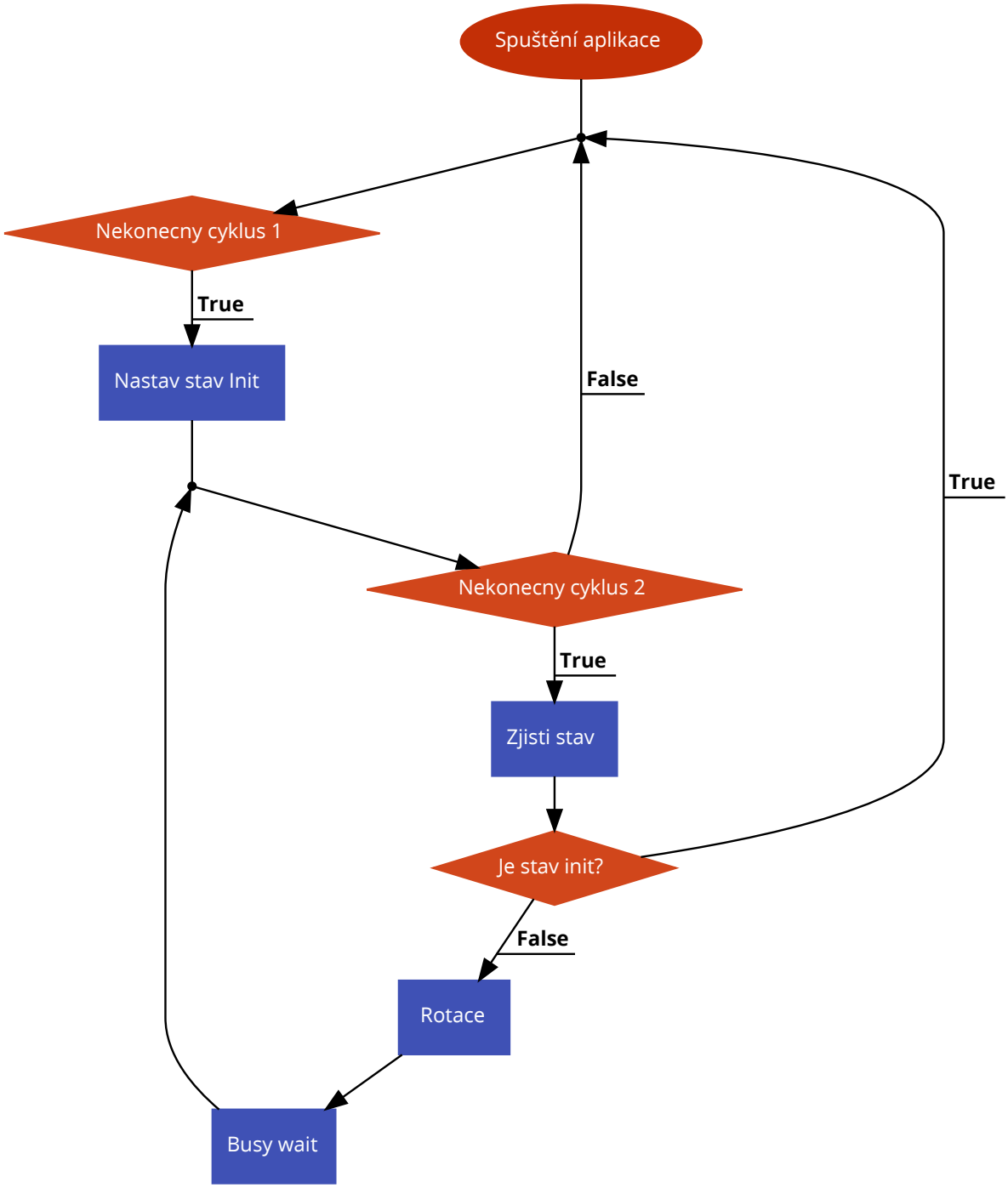
První cyklus má na starost vykonávání inicializační funkce a zároveň i zanoření do dalšího cyklu, v kterém je prováděna detekce stavu a případně rotace – pouze pokud je tlačítko `horiz` nebo `vert` v aktivní poloze.

Samotná funkce rotace v inicializační fázi provede definici limitu pro zpožďovací cyklus. Limit je spočítán násobením kvantifikátoru rychlosti s hodnotou táhla a odečten od maximální hodnoty limitu. Rotaci provádí první cyklus v těle funkce.

```
for(x = DISPLAY_HORIZONTAL_SIZE-1; x >= 0; --x) {
     port_led[x] = login[(horizontal-x)%LOGIN_SIZE];
}
```

Na konci cyklu je v indexu x proměnné `port_led` hodnota určující, které indexy LED aktivovat. Modulo zajišťuje udržení hodnot v rámci paměti pole. Následně se provádí busy wait dle limitu specifikovaného výše a vrací se řízení cyklu ve funkci `main`.

4 Vývojový diagram



5 Paměťové nároky

Definovaných uživatelských konstant v paměti programu je 9:

LOGIN_SIZE	64	Počet hodnot pole s loginem
DISPLAY_HORIZONTAL_SIZE	32	Počet sloupců s LED
DISPLAY_VERTICAL_SIZE	8	Počet řádků s LED
DEFAULT_SPEED	100	Výchozí rychlost rotace
MAX_SPEED	1200	Maximální rychlost rotace
SPEED_QUANTIFIER	4	Násobitel hodnoty táhla
STATE_INIT	0	Stav po aktivaci tlačítka init
STATE_HORIZONTAL_ROTATION	1	Stav po aktivaci tlačítka horiz
STATE_VERTICAL_ROTATION	2	Stav po aktivaci tlačítka vert

Globálních proměnných je v kódu použito 7:

port_btn_init	unsigned char *	Identifikátor portu tlačítka init
port_btn_horizontal	unsigned char *	Identifikátor portu tlačítka horiz
port_btn_vertical	unsigned char *	Identifikátor portu tlačítka vert
port_speed	unsigned char *	Identifikátor portu táhla rychlosti
port_led	unsigned char *	Identifikátor portu první diody
login	unsigned char[64]	Pole s daty, které LED zapnout
state	unsigned char	Aktuální stav

Obecné metriky kódu:

LOC	162
Počet lokálních proměnných	13
Počet globálních proměnných	7
Počet konstant	9