Osmibitový čítač, a výpočet polynomu

Petr Kozák, xkozakp1@fel.cvut.cz

Část I
SPSY – Požadavky na systém

1 Zadání

Navrhněte pro přípravek Explorer 2 s procesorem PIC18F87J11 program, který bude v pravidelných intervalech zvětšovat hodnotu proměnné, kterou zobrazí na portu procesoru, a po změně vypočte hodnotu polynomu. Čítač bude možné vynulovat, či pozastavit čítání a znovu jej spustit.

Čítač bude realizován jako osmibitový, který počítá stále dokola v rozsahu 0 – 255, a je zvětšen o jedničku každých 600 ms. Požadavek na interval 600 ms je striktní. Hodnota čítače bude zobrazena na portu PORTD procesoru, na příslušných LED diodách přípravku — D0 (nejnižší bit) až D7 (nejvyšší bit).

Pro ovládání čítače budou použita tlačítka, která jsou na přípravku, S1 (port RB0 procesoru), a S2 (port RA5 procesoru). Stisknutím tlačítka S2 dojde k vynulování čítače. Při stisknutí tlačítka S1 bude čítání pozastaveno a při opětovném stistnutí se čítání opět spustí, pozastaví se zvětšování hodnoty čítače. Tlačítka budou ošetřena na přechodové děje, a mě-li by reagovat do 50 ms.

Výpočet polynomu bude proveden vždy při změně hodnoty čítače, tj. každých 600 ms. Polynom je zadán vzorcem $y = 7x^2 + 10x + 9$, kde x je aktuální hodnota čítače.

Při resetu procesoru (spuštění napájení), bude čítač spuštěn a nastaven na nulu.

Část II
SPSW – Požadavky na software

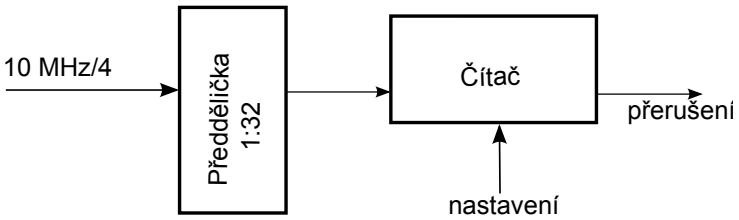
1 Úvod

Požadavky na program jsou rozděleny do několika částí. Jsou to osmibitový čítač, který musí být v pravidelném intervalu obsloužen, není-li ovládacími prvky nastaveno jinak. Dále výpočet polynomu z aktuální hodnoty čítače, kde výsledkem je 24 bitové číslo. Ošetření stavu tlačítek tak, aby nebylo ovlivněno chování programu během přechodového děje, který se vyskytuje na mechanických kontaktech. A nakonec realizace přepínače z tlačítka S1, pro zastavení a opětovné spuštění čítače.

1.1 Osmibitový čítač

Pro zajištění požadovaného intervalu, ve kterém se zvětšuje proměnná čítače o jedničku, bude použit jeden z časovačů procesoru PIC18F87J11. Pro tento účel je vybrán časovač procesoru (viz obr. 1), který umožňuje pomocí před-děličky a čítače nastavit dlouhé intervaly, pokud by to bylo v budoucnu potřeba.

Na časovač je kladen požadavek na dodržení přesnosti intervalu 600 ms. Na přípravku Explorer 2 je hodinový kmitočet pro procesor 10 MHz, ze kterého se odvozuje i kmitočet pro časovače v poměru 1:4.



Obrázek 1: Časovač procesoru

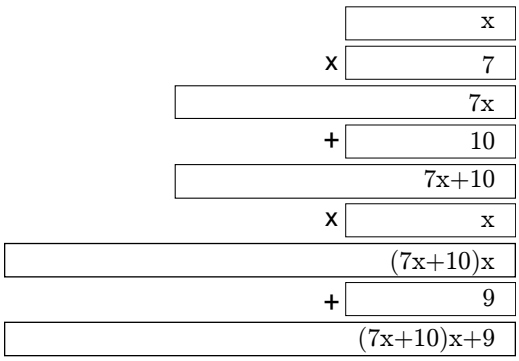
Bude-li použit časovač s před-děličkou 1:32, je umožněno nastavit hodnotu čítače děličky tak, aby výsledný interval pro přerušení byl přesně požadovaných 600 ms. Výpočtem získáme požadovanou hodnotu pro nastavení čítače $(10\,\mathrm{MHz} \cdot 600\,\mathrm{ms})/(4 \cdot 32) = 46875$. Přerušení od časovače bude zpracováno v rutině přerušení s vyšší prioritou. Obsluha časovače s nízkým přerušením nezajišťuje dodržení zadaného intervalu.

1.2 Výpočet polynomu

K výpočtu polynomu budou použity standardní instrukce procesoru pro násobení, sčítání a sčítání s přenosem dvou osmibitových čísel. Pro uložení výsledku bude potřeba 24 bitů

— násobí se tři osmibitová čísla u nejvyšší mocniny polynomu a další sčítání již nemá na celkový počet bitů výsledku vliv.

Pro výpočet polynomu bude z důvodu optimalizace počtu operací vhodné použít Hornerovo schéma (algoritmus). Polynom se převede na výpočet $y = (7x + 10)x + 9$, kde jsou pouze dvě násobení a dva součty. Ukázka výpočtu je na obr. 2.



Obrázek 2: Výpočet polynomu

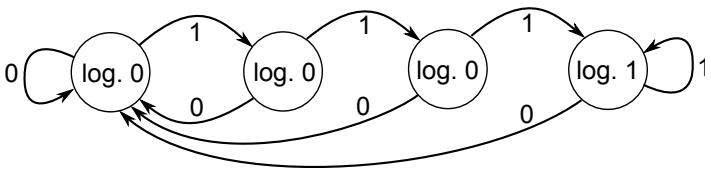
Výpočet polynomu bude proveden v rutině přerušení společně s čítačem, kde je jeho hodnota zvětšována.

1.3 Čtení stavu tlačítek

Tlačítka na přípravku Explorer 2 mají obrácenou logiku, tj. není-li tlačítko stisknuto je na vstupu portu procesoru log. 1, v opačném případě log. 0. Není nutné ji inverovat do pozitivní logiky, schéma použití bude ošetřeno v programu, a negativní logika bude zachována. Je ale potřeba ošetřit tzv. *debouncing*.

Tlačítka jsou mechanické spínače, která při stisknutí resp. uvolnění generují sekvenci nul a jedniček než dojde k ustálení — přechodový děj. Přípravek Explorer 2 nemá k dispozici hardwarové prostředky pro ošetředí tohoto děje, a tento jev bude muset být ošetřen programovými prostředky. Je to nutné, aby nedocházelo k nesprávnému vyhodnocení stavu tlačítka a tím i chování programu.

Bylo zjištěno, že přechodový děj na tlačítku odezní po cca 10 ms, po této době je hodnota ustálená a platná. Teoreticky by stačilo zjistit hodnotu tlačítka po ustálení přechodového děje, tj. cca po 10 ms od jeho změny stavu, to ale není správná metoda, jelikož nevíme kdy přesně začal. Proto bude využito vzorkování hodnoty tlačítka v intervalu cca 10 ms, a budou-li tři po sobě jdoucí hodnoty stejné stav se vyhodnotí jako platná hodnota (ustálený stav). Budou-li se vyhodnocovat tři hodnoty v 10 ms bude zajištěno odečtení stavu tlačítka do požadovaných 50 ms. To vede na stavový automat, detekování tří po sobě stejných hodnot log. 1, resp. log. 0. Na obrázku 3 je stavový automat pro detekci pro uvolnění tlačítka.

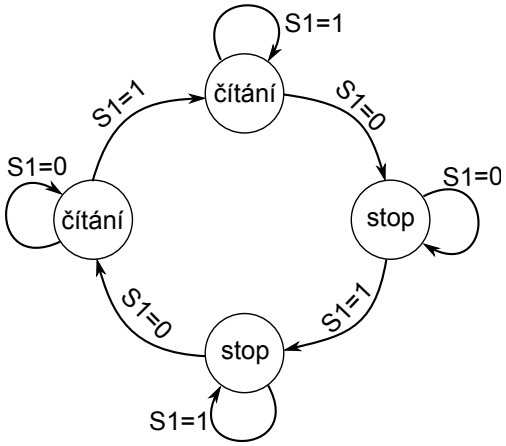


Obrázek 3: Detekce tří po sobě následujících jedniček

Čtení hodnoty tlačítka a přechodu v automatu bude realizováno pomocí časovače, který bude nastaven na interval cca 10 ms, kdy bude vyvoláno přerušení. Toto přerušení může mít nízkou prioritu.

1.4 Realizace přepínače

Tlačítko S1 (port RB0 procesoru) má sloužit jako přepínač pro spuštění resp. zastavení čítání. Převod tlačítka na přepínač bude vyřešeno statovým automatem, který je řízen změnou hodnoty tlačítka. Stavový diagram je zachycen na obr. 4, kde S1=0 odpovídá stisknutému tlačítku a S1=1 opačně.



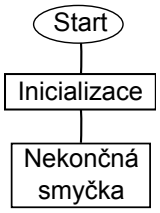
Obrázek 4: Stavový automat pro realizaci přepínače

Přechody nastanou při změně stavu tlačítka, obsluha proto bude umístěna do časovače, který zjišťuje hodnotu a stav tlačítek.

Část III
SASW – specifikace architektury

1 Základní algoritmus

Pro správnou funkci programu v procesoru PIC18F87J11 je nutné nejprve nastavit přerušovací systém, funkce jednotlivých časovačů pro obsluhu čítače, čtení stavu tlačítek a nastavení režimu portů procesoru. Logika programu je umístěna do rutin pro obsluhu přerušení. Vývojový diagram je na obrázku 5.



Obrázek 5: Vývojový diagram programu

2 Inicializace

2.1 Časovačů Timer0, Timer1

Časovač Timer0 procesoru je použit pro obsluhu čítače. Jeho inicializace je provedena pomocí podprogramu `Timer0Init`, kde jsou nastaveny registry procesoru pomocí konstant. Časovač je nastaven do režimu 16bitového čítače s předděličkou 1:32 tak, aby přerušení přišlo každých 600 ms.

Pro obsluhu tlačítek je nastaven Timer1 pomocí konstant v podprogramu `Timer1Init` na 10 ms. Používá se jako 16 bitový s předděličkou 1:8.
Timer0: `CONST_TIMER0_T0CON`, `CONST_TIMER0_TMR0H` a `CONST_TIMER0_TMR0L`.
Timer1: `CONST_TIMER1_T1CON`, `CONST_TIMER1_TMR1H` a `CONST_TIMER1_TMR1L`.

2.2 Přerušení

Přerušení je nastaveno v podpogramu `InterruptInit`. Přerušovací systém je nakonfigurován jako dvouúrovňový tak, aby vyvolal přerušení od Timer0, resp. Timer1 při přetečení. Časovač Timer0 má vyšší prioritu přerušení, která je obloužena podprogramem `Timer0Interrupt`. Časovač Timer1 je obsluhován podprogramem `Timer1Interrupt`, s nižší prioritou.

Oba časovače jsou nyní zastaveny, spuštěny budou podprogramem `TimerStart`, po dokončení všech inicializací.

2.3 Čítačů

Hodnota čítače `VARIABLE_X` se nastavuje na nulu v `SolveInit`. Výpočet polynomu pro danou hodnotu se provádí podprogramem `SolveResult`, výsledek je uložen do proměnné `VARIABLE_RESULT`. Výstupní hodnota čítače se zobrazuje na příslušných LED diodách voláním podprogramu `OutputVariableX`.

2.4 Tlačítek a portů pro LED

Porty procesoru je nutné nastavit nejprve do požadovaného režimu. Dle požadavků je port PORTD procesoru nastaven jako výstupní. Porty pro tlačítka RB0 (S1) a RA5 (S2) jsou nastaveny jako digitální vstupy. Je zajištěno v podprogramu `PortInit`.

Podprogram `ButtonInit` nastavuje hodnoty tlačítek. Nejsou nastaveny aktuální hodnoty což by bylo ideální, ale takové, které odpovídají předpokladu přípravku Explorer 2, tj. tlačítka nejsou stisknuta. Hodnota stavu tlačítek je v proměnné `BUTTON_STATE`, kde každé tlačítko má jeden bit dle konstant.
S1 (RB0): `CONST_BUTTON_BIT_RB0_S1`.
S2 (RA5): `CONST_BUTTON_BIT_RA5_S2`.

2.5 Přepínačů

Pomocí podprogramu `SwitcherInit` je nastaven stavový automat pro realizaci přepínače z tlačítka S1 (port RB0).

3 Hlavní program

Celá logika programu je v rutinách přerušení pro Timer0, který obsluhuje podprogram `Timer0Interrupt` a Timer1, který obsluhuje podprogram `Timer1Interrupt`.

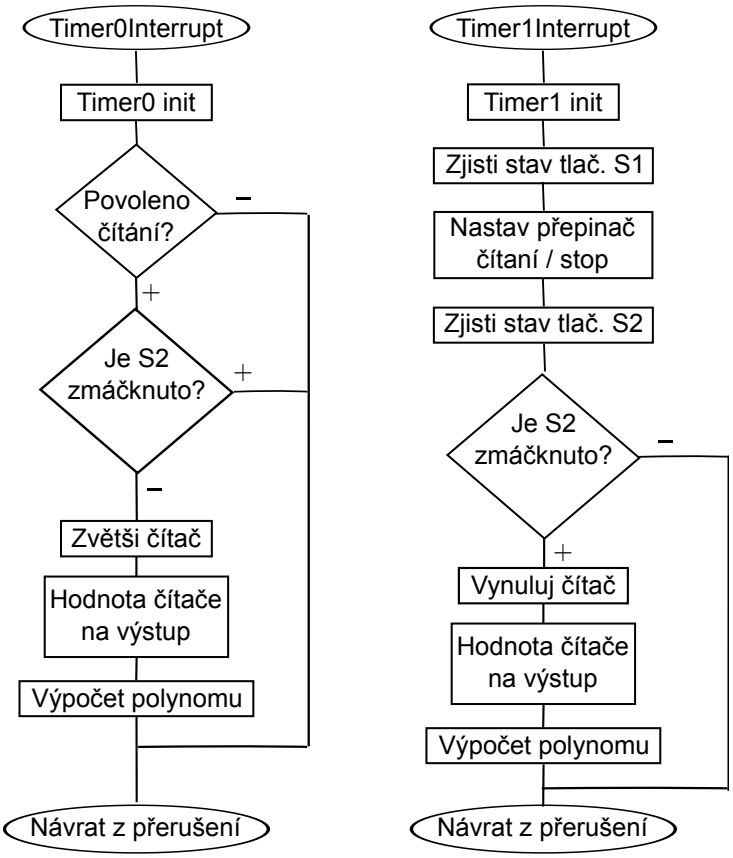
3.1 Obsluha pro Timer0 a Timer1

Obsluha rutiny přerušení pro Timer0 je na adrese `0x8` (vysoká priorita přerušení), kde je odskok na podprogram `Timer0Interrupt`. Pro Timer1 je obsluha na adrese `0x18` (nízká priorita přerušení), kde je odskok na podprogram `Timer1Interrupt`. Vývojový diagram pro zpracování přerušení od obou časovačů je na obr. 6.

4 Základní algoritmy

4.1 Vyhodnocení stavu tlačítek

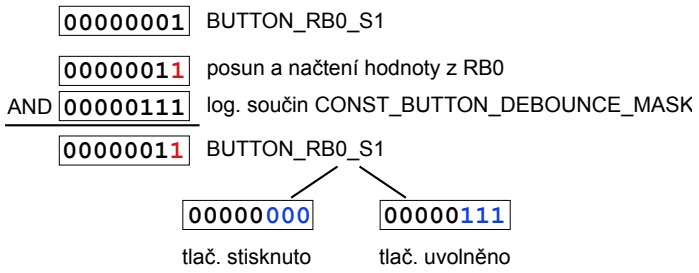
Tlačítka jsou obsluhována podprogramy `CheckButtonS1` a `CheckButtonS2`, přičemž výsledný stav daného tlačítka je uložen v proměnné `BUTTON_STATE`, kde každé tlačítko má



Obrázek 6: Přerušení pro Timer0 a Timer1

svůj příslušný bit dle konstant `CONST_BUTTON_BIT_RB0_S1` a `CONST_BUTTON_BIT_RA5_S2`. Je-li bit nastaven, není tlačítko stisknuto, a opačně. Podprogram je volán z přerušení od časovače Timer1. Musí být vyhodnocena sekvence tří po sobě jdoucích stejných hodnot (nul nebo jedniček) než je změněn stav tlačítka — ošetření zakmitů.

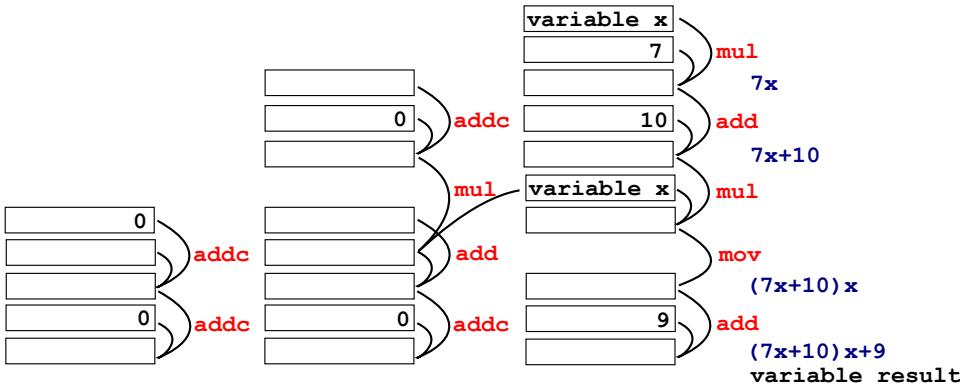
Algoritmus (obr. 7) pro vyhodnocení stavu tlačítka ošetřuje přechodový děj. Stavový automat (obr. 3) pro sledování sekvence tří po sobě následujících stejných hodnot (nul nebo jedniček), je realizován pomocí posuvného registru `BUTTON_RB0_S1` a `BUTTON_RA5_S2`, kde je uchována sekvence posledních hodnot čtených z portu daného tlačítka. Logickým součinem s maskou `CONST_BUTTON_DEBOUNCE_MASK` je vyhodnoceno, zda následuje sekcence nul, pak je výsledek součinu nula, či zda je vyhodnocena sekcence jedniček, a výsledek tak odpovídá masce a stav tlačítka je uložen v proměnné `BUTTON_STATE`. V jiném případě se stav tlačítka nemění.



Obrázek 7: Ošetření zákmitů tlačítka S1

4.2 Výpočet polynomu

Výpočet polynomu je v podprogramu `SolveResult`, který probíhá dle algoritmu jak je naznačeno na obr. 8. Výpočet je proveden ze vstupní promenné `VARIABLE_X` a výstup výpočtu je uložen do `VARIABLE_RESULT`. Aby se zabránilo race condition jsou výpočty prováděny v pomocných proměnných `VARIABLE_X_TMP` a `VARIABLE_RESULT_TMP`.



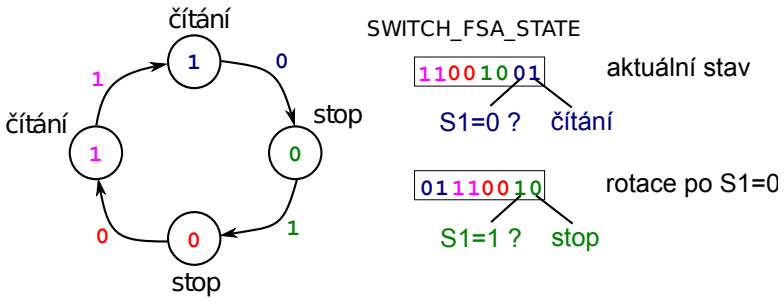
Obrázek 8: Výpočet polynomu

4.3 Stavový automat přepínače

Stavový automat realizující přepínač pro tlačítko S1 (RB0), který je na obr. 4 je implementován formou jedné osmibitobé proměnné.

Dvojice bitů pro daný stav automatu reprezentuje — nižší bit zda je povoleno čítání, vyšší bit jaká hodnota tlačítka způsobí přechod do dalšího stavu. Přechod je proveden rotací v pravo o dva bity. Automat má celkem čtyři stavy (obr. 9) a zakódování celého automatu je v osmibitech — proměnná `SWITCH_FSA_STATE`. Nejnižší bit určený konstantou `CONST_SWITCH_FSA_BIT_COUNT`, reprezentuje logickou hodnotu, a určuje zda je povoleno čítání či nikoliv.

Inicializační zakódování je uloženo v konstantě `CONST_SWITCH_FSA_INIT`. Inicializace automatu je v podprogramu `SwitcherInit`, a přechody dle stavu tlačítka provádí podprogram `SwitcherFSA`.



Obrázek 9: Zakódování automatu přepínače

Část IV
STSW – specifikace testování

Testování navržených algoritmů, bloků a celého programu je jednak v simulátoru prostředí MPLAB a pak na přípravku Explorer 2. Pro potřebu testování v prostředí MPLAB je program doplněn o podmíněný překlad řízený definováním proměnné `DEBUG`, kde nejsou čteny hodnoty portů tlačítek, ale proměnné `PORTA_DBG` a `PORTB_DBG`, které lze v prostředí libovolně měnit.

1 Simulátor MPLAB

Pro testování v simulátoru je nutné, aby byl program přeložen s definovanou proměnnou `DEBUG`. Překlad musí být bez chyb. Jako simulátor je zvolen MPLAB SIM. Každá sekce níže je prováděna po restartu simulace.

1.1 Časovač Timer0

Ověříme, že časovač Timer0 je nastaven správně tak, že zkontrolujeme, že se proměnná `VARIABLE_X` pravidelně zvětšuje. Zároveň se bude měnit i proměnná `VARIABLE_RESULT`, jako výsledek výpočtu polynomu. Ověříme si na výstupu logického analyzátoru pro PORTD, že půl-perioda u nejnižšího bitu je cca 1501541, to odpovídá přibližně 600 ms ($1501541 \cdot 4/10\,\text{MHz} \approx 600\,\text{ms}$).

1.2 Časovač Timer1

Umístíme breakpoint do podprogramu Timer0Interrupt a spustíme program. Po přerušení programu na breakpointu musí být hodnota proměnné `VARIABLE_TIMER1_DBG` rovna hodnotě okolo 60. To odpovídá nastavení časovače Timer1 na hodnotu okolo 10 ms ($600\text{ms}/60 = 10\,\text{ms}$).

1.3 Nulování čítače

Dále ověříme chování nulování čítače. Necháme program chvíli běžet a po změně hodnoty `VARIABLE_X` jej pozastavíme. Umístíme breakpoint do podprogramů `Timer1Interrupt` a `Timer0Interrupt`. V místě kde se testuje stav tlačítka S2 (`BUTTON_STATE`), přenastavíme proměnnou tak, aby druhý nejnižší bit byl nula. Provedeme další krok. Pro `Timer1Interrupt` se musí proměnná `VARIABLE_X` vynulovat a vypočítat nová hodnota do `VARIABLE_RESULT`, v logickém analyzátoru bude na portu PORTD nula. Pro `Timer0Interrupt` se přeskočí vykonávání příslušného kódu, tj. nebude změněna hodnota proměnných `VARIABLE_X`, `VARIABLE_RESULT`.

1.4 Ošetření stavu tlačítek

Zde by se měla ověřit funkčnost algoritmu pro ošetření zákmitů tlačítek. Algoritmy jsou stejné pro obě tlačítka, a proto zde budeme ověřovat obě najednou.

Umístíme breakpoint do funkce `CheckButtonS1` a `CheckButtonS2`. Po každém průchodu nastavíme proměnou `PORTB_DBG` a `PORTA_DBG` na hodnoty — 1,1,1,0,1,1,0,1,0,0,1,0,0,0. Na konci této sekvence bude příznak pro tlačítka S1 a S2 v `BUTTON_STATE` nastaven jako stisknuté (dva nejnižší bity log. 0). Sekvenci hodnot nyní zadáme v opačnem pořadí a na konci bude stav tlačítek indikován jako změněn (dva nejnižší bity log. 1). Během sekvence se nesmí stav tlačítka v `BUTTON_STATE` změnit.

1.5 Stavový automat přepínače

Stavový automat je realizován v proměnné `SWITCH_FSA_STATE`. Umístíme breakpoint do podprogramu `Timer0Interrupt`. Pro každý průchod nastavíme proměnnou `PORTB_DBG` na hodnoty stídavě 0,1,0,1,0,1,0,1 na konci podprogramu. Lichá pozice pro nulu musí zastavit čítač, nejnižší bit v proměnné `SWITCH_FSA_STATE` bude nula. Sudá naopak spustit.

2 Přípravek Explorer 2

Po ověření funkčnosti v simulátoru je možné přejít k testovaní na přípravku Explorer 2. Je nutné, aby byl program přeložen s nedefinovanou proměnnou `DEBUG`. Překlad musí být bez chyb. Po nahrání programu do procesoru a případném resetu, se bude na LED diodách zobrazovat podoba binárního čísla čítače s periodou 600 ms, které se bude zvětšovat. Každá sekce níže je prováděna po resetu procesoru.

2.1 Ověření funkce tlačítka S2

Při stisknutí tlačítka S2 na přípravku musí LED diody zhasnout, a musí zůstat zhasnuté pokud je tlačítko delší dobu než je 600 ms.

2.2 Ověření funkce tlačítka S1

Při stisknutí tlačítka se musí čítaní zastavit, což bude indikováno zastavením změny LED diod. Po opětovném stisknutí se čítač opět spustí, LED diody budou pokračovat v zobrazování hodnoty čítače. Délka držení tlačítek nesmí mít vliv na režim čítání, pozastavení čítání.