**První cvičení**

**všechny další příklady v prvním cvičení realizujte pomocí přerušení a čítačů.**

1. Sestavte program, který bude na pinu RB2 generovat dva střídající se obdélníkové signály s intervalem čistá kvarta. Periodu opakování udělejte tak, aby střídání signálů bylo pohodlně slyšet, výstup zapojte na reproduktor, nezapomeňte na rezistor.

Až sem za 4

1. Sestavte program, který bude dokolečka hrát kvintakord C-dur c – e – g - c

Až sem za 3

1. Sestavte program, který bude dokolečka hrát stupnici C-dur

Až sem za 2

1. Sestavte program, který bude hrát nějakou píseň, alespoň v rozsahu písně Ovčáci, čtveráci

Až sem za 1

ad 1

budeme mít dva čítače. První čítač generuje tón stejným způsobem, jako v bodu 2. Druhý čítač mění kmitočet prvního tím, že změní obsah registru PR u prvního čítače. Kmitočet druhého čítače je samozřejmě takový, abyste změnu pohodlně slyšeli. Doporučuji zpívat si kvartu, v okamžiku změny tleskat. Perioda tlesknutí je pak periodou druhého čítače. U druhého čítače pak potřebujeme v okamžiku zavolání jeho přerušení vědět, zda máme nastavit první tón nebo druhý tón ( např: a – dis). K tomu účelu si určíme nějakou globální proměnnou, v přerušení od druhého čítače tuto proměnnou negujeme ( ! ). A podle stavu této proměnné nastavíme do registru PR prvního čítače správné číslo.

ad 2

vyjdeme z příkladu 1. Opět budeme mít dva čítače, jeden bude odměřovat délku tónů, druhý bude generovat vlastní tón. Dále potřebujeme počítátko, které bude určovat, který tón z kvintakordu právě hrajeme. A potom už je to jednoduché – podle hodnoty počítátka nastavíme do registu PR čítače, který generuje tón, tu správnou hodnotu. Lze to udělat pomocí soustavy if-ů, lepší je samozřejmě příkaz switch, ale ten jsme z důvodu časoví tísně neprobrali. Klidně si honajděte a pouřžijte.

ad 3

Samozřejmě můžeme postupovat stejně jako v příkladu 2 , ale stupnice už má docela dost tónů. Doporučuji proto uložit hodnoty pro registr PR do pole a postupně je číst. Pole v paměti EEPROM uděláte příkazem

const int pole [50] = { 4,8,25,45,13,28} ; v závorkách {} jsou hodnoty, které se při překladu uloží do pole . const znamená konstantní, tedy neměnné, a překladač to uloží do EEPROM a zařídí čtení pomocí PSV .

ad 4

no a pokud máte hotový příklad 3, tak je písnička hračkou. Delší tón udělejte třeba tak, že dáte za sebe dvě stejné hodnoty. Na konec pole dejte nějaké vhodné číslo, třeba 0, a tím poznáte, že jste na konci. Pomlku uděláte třeba tak, že do pole dáte jiné vhodné číslo – třeba 0xffff, a při jeho přečtení vypnete čítač, který vyrábí tón ( TxCONbits.TON ). Nezapomeňte jej potom zapnout ( třeba takto: čítač je vypnut, a v poli je číslo jiné než 0xffff , tak čítač zapnout )

**Druhé cvičení**

1. Sestavte program, který bude na výstupu SDO modulu SPI dokolečka vysílat číslo 0x9fa5. Kmitočet vysílání zvolte podle standardní PCM (bude se nám to dál velmi hodit). Kmitočet SPI zvolte co možná nejmenší, ale takový, aby vysílání čísla zabralo maximálně jednu desetinu doby mezi vysílanými čísly ( abychom potom stíhali dělat během vysílání i další věci )

až sem za 4

1. Pomocí PPS upravte příklad 1 tak, abychom číslo vysílali do obvodu MCP4822. Zprovozněte obsluhu signálu CS a číslo upravte tak, abyste do obvodu vysílali data, která nastavují výstupní napětí na 1.5 V.

až sem za 3

1. Sestavte program, který bude na výstupu převodníku MCP4822 generovat pilu, která vzrůstá od 0V do maxima a pak opět „spadne“ na 0V .

až sem za 2

1. Sestavte program, který bude na výstupu obvodu MCP4822 generovat sinusovku.

až sem za 1

ad 1

Naprogramujeme SPI, na výstup si dáme osciloskop. Nějaký čítač naprogramujeme na frekvenci PCM, v přerušení od tohoto čítače odstartujeme vysílání čísla po SPI. to se dělá tak, že zapíšeme ................................. Úvaha ohledně kmitočtu SPI je jednoduchá: PCM má kmitočet ....., to odpovídá periodě .... . Pokud má vysílání po SPI zabrat maximálně jednu desetinu této periody, musí být dlouhé ......... . Vysíláme celkem .... bitů, takže na jeden bit připadá čas ....... . No a z toho je jasný kmitočet hodin, který je ..... . Nastavíme tedy primární děličku na .... a sekundární děličku na .....

ad 2

Doporučuji udělat si na vysílání funkci. Tu pak někde zavoláte a ono samo to zajistí odvysílání čísla do MCP4822. Na začátku funkce musíte dát signál CS pro MCP4822 do 0 . CS je připojen na pin ...... , takže je to hračka. Číslo, které dáme do naší funkce jako formální parametr, upravíme. Převodník je ..... bitový, takže maximální velikost je ..... . Maskování uděláme nejlépe pomocí AND, tedy & ( jenom JEDNO & , dvě && je logický AND, nikoli bitový. Ostatně si o tom přečtěte na <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/logicke/> ). Dále nastavte horní 4 bity tak, abyste posílali data do správného převodníku, správně nastavíme GAIN, převodník nebude vypnut. Na konci vysílání čísla potřebujeme vrátit CS u MCP na hodnotu 1. To uděláme v přerušení od SPI. Poslední věc je, že musíme pomocí PPS nastavit SDO a SCK tak, aby vedly na ty piny, které jsou spojeny s obvodem MCP. SPI se nastavuje úplně na začátku, ještě před tím, než vůbec začneme nastavovat modul SPI. PPS máte v <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/PPS/>

Datasheet převodníku máte na <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/asm_progr_a_vysv/SPI/MCP4822.pdf>

zde strana 6 a strana 17 , samozřejmě si to přečtěte celé

ad 3

To je hračka. V přerušení od čítače voláme funkci, která posílá data do MCP, a postupně do ní ukládáme zvyšující se čísla.

ad 4

je také hračka. Nezapomeňte, že vámi vygenerovaný sinus má hodnoty od .... do ...... , zatímco čísla, která posíláte do převodníku, jsou od ..... do .......... . Takže musíte sinus posunout a vynásobit. Dále se v simulátoru podívejte, jak dlouho trvá výpočet sinu, pokud by byl delší, než je čas mezi dvěma vysílanými vzorky , musíme buď sníčit vzorkovací kmitočet nebo zvýšit kmitočet vnitžního oscilátoru procesoru ( zapnout PLL, viz <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/oscilator/> )