**První cvičení**

1. Sestavte program, který bude neustále kopírovat data z pinu 3 na pin 10 procesoru a z pinu 26 na pin 2. U vstupních pinů zapněte pull-up rezistory, abyste mohli použít kontakt proti zemi.

až sem za 4

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

1. Sestavte program, který bude na některém pinu procesoru vysílat obdélníkový signál o kmitočtu 1kHz . Zobrazte na osciloskopu.

až sem za 3

1. Sestavte program, který na čtyřech různých pinech procesoru bude generovat čtyři různé signály o kmitočtu 150kHz, 1kHz, s periodou 100ms , s periodou 2s. SAMOZŘEJMĚ NAJEDNOU ! Signály postupně zobrazte na osciloskopu. Na osciloskopu také zobrazte signál 150kHz, když je synchronizace od kmitočtu 1kHz. Popište pozorovaný děj.

Až sem za 2

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 1

ad 1

nastavíte vstupy a výstupy portů, a potom už jenom dokolečka čtete a zapisujete na příslušný bit portu. V header file je nadefinována struktura pro jednotlivé bity portu, takže existuje něco jako PORTAbits.RA0 PORTBbits.RB15 a podobně. Na <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/asm_progr_a_vysv/porty/porty.doc> si přečtěte, jak se nastavují pull-up rezistorz a co to je. Tento manuál jste si ostatně měli přečíst již dávno.

ad 2

. Přerušení a práci s ním máte popsánu na <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/preruseni/> a <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/asm_progr_a_vysv/preruseni/>

blikání diodou jsme dělali ve škole. Teď tedy máte školní příklad „přestěhovat“ do přerušení.

ad 3

kmitočet 150kHz j úmyslně dost na hraně. Proč, to zjistíte velmi lehce – dejte si v simulátoru breakpoint na začátek přerušení, které ho generuje, do VARIABLES di dejte příslušný čítač TMR , dívejte se na hodnoty v TMR a mačkejte F7 . Pokud by vám to opakovaně nefungovalo, můžete kmitočet trochu snížit. Oproti příkladu 2 musíme samozřejmě nastavit několik čítačů a několik přerušení.

ad 4

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.

**Druhé cvičení**

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

Až sem za 4

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

Až sem za 3

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

Až sem za 3

1. Sestavte program, který bude fungovat jako piano. Podle stisku některého z osmi vstupních kontaktů bude na zvoleném výstupním pinu hrát tóny v rozsahu jedné oktávy.

Až sem za 1

ad 1

vyjdeme z příkladu 4 z prvního cvičení. 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 registru 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 ho najděte a použijte.

ad 2

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 3

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 )

ad 4

Na nějaké vstupní piny (doporučuji port B, ale v zásadě je to jedno) si připravíme kontakty proti zemi ( budeme realizovat prostým drátkem, kterým se dotkneme vstupu, druhý konec drátku je uzemněn. Nezapomeňte zapnout PULL-UP rezistory. ) Dokolečka testujeme vstupní piny, pokud je na pinu hodnota, odpovídající sepnutému stavu, uložíme do registru PR číslo, které odpovídá kmitočtu příslušného tónu. Vlastní tón generujeme čítačem a přerušením. Pokud není sepnut žádný kontakt, čítač vypneme.

**Třetí cvičení**

1. Sestavte program, který bude na výstupu SDO modulu SPI dokolečka vysílat číslo 0x9fa5. Kmitočet vysílání zvolte 8kHz. Kmitočet hodin 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 střídavě generovat napětí 1V a 2V s periodou 2sec ( tedy 2 sec bude na výstupu 1V, další 2sec 2V, stále dokolečka ) až sem za 2
2. 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 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: 8kHz 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. Datasheet od obvodu MCP4822 je na http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/asm\_progr\_a\_vysv/SPI/MCP4822.pdf 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/>

ad 3

To je hračka. V přerušení od čítače voláme funkci, která posílá data do MCP, a jednou do ní uložíme číslo, které znamená 1V, podruhé číslo, které znamená 2V.

ad 4

je také hračka. Do převodníku postupně vysíláme zvyšující se čísla, a je to .

**Čtvrté cvičení**

1. Sestavte program, který bude na výstupu převodníku generovat sinusovku o kmitočtu 100 Hz. Vzorkovací kmitočet je 8kHz. Amplituda signálu je maximální možná. Sinusovku počítejte pomocí funkce sin .

Až sem za 4

1. Program podle bodu 1 doplňte dvěma vstupními bity. Pokud je na těchto bitech číslo 0 , program funguje stejně jako v bodě 1 . Pokud na vstupních pinech je číslo 1, bude výstupní napětí o 6dB nižší než v příkladu 1. Pokud na vstupních pinech je číslo 2, bude výstupní napětí o 12dB nižší než v příkladu 1. Pokud na vstupních pinech je číslo 3, bude výstupní napětí o 18dB nižší než v příkladu 1.

Až sem za 3

Ad 1 .

Z třetího cvičení již umíme ovládat DA převodník. Nyní už jenom vypočteme vzorek sinusovky a jeho hodnotu pošlete do převodníku. Nezapomeňte, že rozsah hodnot funkce sin je od … do… , rozsah čísel, která posíláme do převodníku, je od …. do …. . Takže čísla musíte vhodným způsobem transformovat. Dále může být problém s délkou výpočtu funkce sin. To zjistíme pomocí stopwatch. Pokud by délka výpočtu byla delší, než odpovídá našemu vzorkovacímu kmitočtu, zvolte si jiný vhodný vzorkovací kmitočet

Ad 2

Čteme hodnoty ze vstupních pinů, podle jejich hodnoty změňte výstupní napětí tak, aby bylo o příslušný počet decibelů nižší.