**První cvičení**

**všechny další příklady v prvním cvičení realizujte pomocí přerušení a čítačů.** Výstupní tóny jsou u tohoto cvičení „hranaté“ , tedy TTL

1. Sestavte program, který bude na nějakém pinu procesoru generovat obdélníkový kmitošet s frekvencí 1kHz.

až sem za 4

1. Sestavte program, který bude fungovat jako telegraf – na nějakém vstupním pinu budete mít kontakt proti zemi, na jiném pinu výstupní signál. Pokud je kontakt sepnut, signál zní.

až sem za 3

1. Sestavte program, který bude na nějakém pinu procesoru 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 2

1. Sestavte program, který bude na nějakém pinu procesoru hrát dokolečka stupnici C-dur (nebo nějakou jinou, vámi oblíbenou).

Až sem za 1

Ad 1 - tohle máme skoro hotové ze školy. Nastavíme čítač, zprovozníme přerušení. V přerušení pak NAPŘÍKLAD negujeme nějaký bit portu PORTBbits.RB10 = ! PORTBbits.RB10 ; A NEBUDEME opakovat oblíbenou chybu z 2. ročníku, kdy vám to generovalo poloviční kmitočet.

Ad2 – k příkladu 1 přidáme vstupní piny - nezapomeneme nastavit pull up rezistor. No a potom v přerušení ještě navíc testujeme, zda je pin v 0 nebo v 1 , a podle toho provedeme nebo neprovedeme inversi výstupního bitu – něco jako

if( PORTBbits.RB5 == 1 ) {

}

Samozřejmě, protože logická hodnota je vyjádřena číslem, bude fungovat i

if( PORTBbits.RB5 ) {

}

Dále doporučuji udělat if – else, a pokud telegraf nemá pípat, nastavte na výstupní bit logickou 0 , abyste zbytečně neproháněli reproduktorkem proud.

Ad 3

budeme mít dva čítače. První čítač generuje tón stejným způsobem, jako v bodu 1. 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 4

Budeme postupovat podobně jako v příkladu 3. To znamená, že druhý čítač bude generovat čacové intervaly, po kterých se má změnit tón prvního čítače. 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 [KONECPOE+1] = { 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 .

Samozřejmě pak potřebujete globální proměnnou, která říká, z kterého indexu pole máte číst data, tu inkrementujete v přerušení od druhého čítače. Až dojde na konec pole, nezapomeňte ji vrátit na 0 , jinak procesor vyhlásí „Address Error“ a reaguje RESETem .

**Druhé cvičení – SPI**

1. Sestavte program, který bude na D/A převodníku generovat pilovitý signál s maximální možnou amplitudou, tedy od 0V do 4.075V. Vzorkovací kmitočet zvolte 8 kHz.

až sem za 3

1. Sestavte program, který bude na nějakém výstupu D/A převodníku generovat ten průběh, který jste dostali zadán za domácí úkol na FFT v TS. Vzorkovací kmitočet ponechte 8kHz.

až sem za 2

1. Sestavte program, který bude generovat sinusový průběh pomocí DSP algoritmu. Kmitočet bude 400 Hz, vzorkovací kmitočet 8kHz, amplituda co možná největší.

až sem za 1

ad 1 .

Potřebujeme SPI a PPS. O PPS si přečtete na <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/PPS/> , pokud ho umíte z loňska, zajímá vás jenom to, jak se PPS ovládá v Cčku. K tomu tam je i vzorový příklad PPS.c . Dále nastavíme SPI stejně jako loni. Pilu uděláme tak, že necháme nějakou proměnnou inkrementovat od ... do .... . Nezapomeňte zamaskovat horní bity, které řídí chování převodníku. Vzorkovací kmitočet uděláme pomocí čítače a přerušení, v přerušení od čítače budeme pokaždé „startovat“ přenos dat po SPI. Nezapomeňte na ovládání CS převodníku ! Až pila dojde na maximum, vynulujte příslušnou proměnnou a začněte od 0 . Některé proměnné nesmí zapomenout svoji hodnotu i mimo přerušení – ty budou samozřejmě globální.

ad 2

Data pro váš průběh jste si už dávno udělali ve vašem domácím úkolu. Teď je musíme trochu upravit – ve vašem domácím úkolu bylo největší číslo ........ , maximální napětí pro převodník je číslo ...... . To znamená, že vaše čísla musíte vynásobit / vydělit číslem ...... ( je to oblíbená přímá úměra, která je i na gymnasiu považována za samozřejmost). Nevím, zda někdo z vás měl napětí v domácím úkolu záporná – pokud ano, tak to prostě posuňte nahoru tak, aby všechna čísla byla kladná.

Hodnoty vzorků si uložíme do pole

const int pole [KONECPOE+1] = { 4,8,

25,45,

13,28

2069 , 4011

5,

6,

100,

2} ;

Potom postupně čteme jednotlivé vzorky z pole a vysíláme do převodníku pomocí SPI. Pro informaci, na kterém indexu pole se právě nacházíme, potřebujeme samozřejmě ukazovátko. To bude globální proměnná, nesmí zapomenout svoji hodnotu mimo přerušení. A zase – až dojdete na konec pole, dáte ukazovátko na 0 a začnete od začátku.

Mnohokrát jsem vám připomínal, že odřádkování, mezera, tabulátor nemají v C význam. Pokusil jsem se to předvést i na příkladu definice pole výše. Znamená to také, že potřebná čísla, která budete ukládat do pole, si můžete nechat vyblít z Excelu nebo jiného programu, včetně těch čárek za číslem. A dodělat tam jenom otvírací a zavírací závorku.

ad 3

Generování sinusovky jsme dělali. Tak prostě vezmete ten algoritmus, který vám funguje v obecném Cčku, a nacpete ho do programu pro PICa . Je tam nějaké počáteční nastavení proměnných – to bude v hlavním programu, a potom části, které dokolečka počítají další a další hodnotu sinu. Ty budou v přerušení od čítače, na jedno zavolání přerušení se spočte jedna další hodnota sinu. A vyšle se na D/A převodník. Opět upozorňuji, že hodnoty proměnných, které počítají sinus, se nesmí zapomenout mezi voláním přerušení, takže tyto proměnné budou globální.

Další věc je, že DSP algoritmus počítá hodnoty sinu v intervalu čísel od ...... do ....... .Ale náš převodník vyžaduje čísla v intervalu od ... do ... . Takže to, co vybleje DSP algoritmus, musíte příslušně zvětšit a posunout.

Poslední důležité upozornění: budete tam používat proměnné float. Výpočty pomocí nich trvají dost dlouho. Nastavte si v simulátoru správný kmitočet oscilátoru a podívejte se, jak dlouho vám výpočty trvají. Pokud by to celé trvalo déle, než odpovídá vzorkovacímu kmitočtu, tak si dejte vzorkovací kmitočet příslušně nižší a také generovaný kmitočet udělejte ve stejném poměru nižší. Opravdu nevím, co to udělá, a je to na vás – s tím si musí programátor poradit. SAMOZŘEJMĚ , ti šikovnější z vás si prostě zvýší kmitočet oscilátoru a je to.

A asi po dvacáté: v tom algoritmu se NIKDE nepoužívá Cčková funkce sin a cos. Jsou tam samozřejmě dvě čísla, která se pomocí těchto funkcí vypočetla. Ta si ale spočítáte na kalkulačce a do programu dosadíte přímo tato čísla.

**Třetí cvičení – ADC**

1. Sestavte program, který bude pomocí A/D převodníku měřit napětí na některém pinu. Převedené číslo si zobrazte na nějakém portu, na LED si dejte nejvyšší bity převedeného čísla a předveďte, že převodník funguje. Vzorkovací kmitočet zvolte 8 kHz.

až sem za 3

1. Sestavte program, který „ udělá z PICa drát “ . To znamená, že zprovozníte A/D pževodník, dále D/A převodník, na vstup přivedete nějaké měnící se napětí a na výstupu budeme pozorovat totéž napětí. Dále předveďte aliasing. Vzorkovací kmitočet ponechte 8kHz. !!!! Předvedení aliasingu je nutnou podmínkou k uznání tohoto bodu !!!!

až sem za 1

Ad1: To je základní úloha. Nastavíte A/D převodník, dále si uděláme přerušení od čítače s kmitočtem 8kHz. V tomto přerušení vždy „odstartujeme“ převod. Dále uděláme přerušení od A/D převodníku, v něm přečteme ADCBUF a hodnotu uložíme na vhodný port. Na správné bity dáme LED a díváme se, co převodník dělá. Vstupní proměnné napětí získejte např. pomocí potenciometru. Pokud někdo řekne, že neví, jak se to dělá, tak rovnou jděte k panu řediteli a odhlaste se ze školy.

Ad2: A/D převodník máme z prvního příkladu, D/A převodník nám funguje z druhého cvičení. Tak to prostě spojíme. Nezapomeňte zamaskovat bity 15 – 12 , které posíláte do D/A převodníku. Na vstup si přivedeme nějaké napětí, nejlépe sinusovku, a díváme se, co vzchází z D/A převodníku. Použijeme k tomu samozřejmě osciloskop, dva kanály.

Aliasing si samostatně nastudujete, možná ho s vámi již probíral Ing. Šerých nebo Ing. Novotný. Nastavíte vhodné parametry signálu, na osciloskopu ukážete: „Tohle je aliasing“ , vyfotíte si to a nadosmrti si to zapamatujete.

**Čtvrté cvičení**

1. Sestavte program, který bude fungovat jako derivační člen. Jeho funkci předveďte na osciloskopu. Vzorkovací kmitočet zvolte 8 kHz.

až sem za 3

1. Sestavte program, který bude fungovat jako analogová zpožďovací linka s co největším zpožděním. Vzorkovací kmitočet ponechte 8kHz. Funkci předveďte na osciloskopu. až sem až sem za 1

Ad 1: Derivační člen je hračka, podívejte se do sešitu z teorie. Nezapomeňte, že na výstupu budou kladná i záporná čísla ( pokud vstupní napětí stoupá, pak kladná, pokud vstupná napětí klesá, záporná). Do D/A převodníku ovšem pouštíte pouze kladná čísla ! Musíte tedy výstup z derivačního členu před odesláním do D/A převodníku stejnosměrně posunout, to znamená, že s čísly uděláte ...... Výstupní čísla vám možná vyjdou moc velká nebo moc malá . V tom případě je vznásobte vhodným číslem tak, abzste dostali použitelný výsledek.

Ad 2: opět hračka. Uděláme si pole o maximální možné velikosti. Z pole přečteme číslo a vyšleme do D/A. Následně do stejného místa uložíme číslo, které jsme přečetli z A/D. Inkrementujeme index pole. Až index dojde na konec, nastavíme ho na začátek pole, tedy do něj uložíme číslo ........

Předpokládám, že mnohým se bude zdát divné, že máme nejdříve přečíst číslo z pole a potom do stejného místa jiné uložit. Jak si jistě pamatujete, program se píše odprostředka. To znamená, že tohle všecho už dávno běží a běží, takže v poli máte nějaká čísla z minulých cyklů. Na začátku programu jste samozřejmě do pole uložili nějaká vhodná čísla, takže i v prvním průchodu D/A něco vysílá. Pokud jste tak neučinili, jsou v poli čísla ......... , takže převodník v prvním průchodu cyklem vysílá ..................... . To vadí/nevadí (nehodící se deletujte)

**Páté cvičení**

1. Sestavte program, který přepne kmitočet vnitřního oscilátoru na 140 MH. Přepnutí vhodným způsobem dokažte.

až sem za 4

1. Sestavte program, který bude fungovat buď jako pásmová zádrž, nebo jako pásmová propusť. Kmitočty si zvolte libovolně, ale tak, aby se funkce dala dobře ukázat na osciloskopu. Použijte filtr FIR, tak, aby měl řekněme 20-25 koeficientů b. Vzorkovací kmitočet ponechte 8 kHz. Funkci předveďte na osciloskopu.

!!!!! Filtr budete mít navržený již při příchodu do laboratoře a dostanete za něj další známku. Každý žák bude mít svůj filtr, i když pak dvojice realizuje jenom jeden.

až sem až sem za 1

ad 1: je to hračka, nastavíme si PLL a je to. Dále si vyvedeme vydělený kmitočet na vhodný pin procesoru ( použijeme vhodný čítač ). No a pokud vám bude výstupní kmitočet souhlasit s výpočty, je asi vše v pořádku.

ad 2:

navrhneme si filtr FIR. Můžete zkusit např. naši oblíbenou <http://t-filter.engineerjs.com/> . S návrhem si musíte trochu pohrát. 20 koeficientů jsem zvolil proto, aby procesor byl schopen „upočítat“ další hodnotu v čase mezi jednotlivými vzorky. Nemůžete udělat příliš strmé boky té propusti. Až budete mít sestavený program pro výpočet dalšího vzorku , podívejte se v simulátoru, jak to dlouho trvá. Nastavte si kmitočet 140MHz. Při 20-ti koeficientech b by čas na výpočet měl být menší než čas mezi dvěma vzorky. Program udělejte nejlépe následovně:

čítačem generujte vzorkovací kmitočet

v přerušení od čítače přečtěte hodnotu z AD převodníku

odstartujte další AD převod

vypočtěte další hodnotu výstupního vzorku pro filtr FIR

vzorek vyšlete na DA převodník

tím dosáhneme toho, že AD převod a počítání běží paralelně. Dále běží vysílání na DA převodník, zase „samo“ , nezávisle na výpočtech. Na konci vysílání do DA převodníku vracíme v přerušení CS zpátky do 1. Je možné, že se tohle bude dít až v době, kdy přijde další přerušení od čítače. **Přerušení od SPI proto dejte vyšší prioritu než přerušení od čítače.**

Dále nezapomeňte na to, že musíte přepočítat hodiny pro AD převodník a hodiny pro SPI, protože pracujete s kmitočtem 140 MHz na procesoru.

Doporučuji nastavit AD převodník do režimu „signed integer“ , dále se podívejte na amplitudovou charakteristiku vaší propusti – je možné, že filtr bude zesilovat. V tom případě vstupní čísla vhodně zmenšete. Na výstupu nezapomeňte čísla upravit tak, aby byla kladná, stejně jako u derivačního členu.

Při předvádění pustíte do bastldesky z generátoru nějaký kmitočet a podíváme se, co vzchází na výstupu. Podíváme se na charakteristiku filtru a zhodnotíme, zda to tak má nebo nemá být. To uděláme pro více kmitočtů, abychom tak zdokumentovali celou přenosovou charakteristiku našeho filtru FIR .

Na vaší bastldesce <http://ozeas.sdb.cz/panska/mikroproc/33EV32/bastldeska/33EV32_TRTUSEK_2019/cad/Schema.pdf>

máte obvod R4 R5 C10 a obvod R2 R3 C9 , který lze použít pro posuv vstupního střídavého napětí .

**Šesté cvičení**

1. Sestavte filtr FIR, který bude realizovat pásmovou propust nebo zádrž s alespoň čtyřmi pásmy ( tedy propust bude mít čtyři pásme , ve kterých propouští, a pět pásem, ve kterých nepropouští , zádrž analogicky ). NAprogramujte do PICa pomocí DSP-engine. Činnost filtru předveďte na osciloskopu, vzorkovací kmitočet je 8 kHz.

až sem za 1

Poznámky: AD převodník nastavíme tak, aby dodával čísla signed fractional. Pro vlastní výpočet použijeme postup podle <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/fixed_point_simplex/fixed_point_simplex.docx> . A dál už není co řešit !

Při provádění výpočtů vás asi nebude omezovat čas, samozřejmě si přetaktujeme procesor na 140 MHz. Omezení bude z hlediska paměti, navrhněte si takový filtr, aby se vám koeficienty vešly do paměti.