První cvičení

1. Sestavte program, který bude neustále kopírovat data z portu A na port B. Na polovině bitů portu A zapněte pull-up rezistory, na druhé polovině pull-down rezistory. Předveďte, že program správně funguje.
2. Sestavte program, který vytvoří na portu B 16-ti bitový čítač, který bude čítat nahoru s libovolně vysokou frekvencí. Na portu A je dále jeden vstupní bit. Pokud je na bitu log. 1, čítač čítá. Pokud je na bitu log. 0, čítač stojí. Funkci čítače předveďte na osciloskopu.
3. Sestavte program, který přečte z portu B dvě čtyřbitová čísla. Na port A vyšle jejich součet.
4. Sestavte program, který na libovolném bitu portu B bude vysílat obdélníkový signál o kmitočtu asi 1kHz. Na portu A je dále jeden vstupní bit – spínač proti zemi. Pokud je spínač sepnut, je kmitočet dvojnásobný.

Ad 2: není udána rychlost, takže prostě uděláme nějakou proměnnou, u ní ++ , a budeme kopírovat na port B. Dále testujeme vstupní bit, pokud je v 0 , tak prostě pomocí if () zvyšování proměnné vyřadíme.

Ad 3: Tady přečteme celý port B. Pomocí AND zamaskujeme tu část, kterou nepotřebujeme. Pomocí posuvů si „šoupneme“ bity do té polohy, kterou potřebujeme pro sčítání. Seščteme a zapíšeme na port A. Pro logické funkce a posuvy si nastudujte <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/logicke/> . ten program si musíte přečíst až do konce. Prozradím vám, že při posuvech se proměnné chovají jinak, pokud jsou signed a unsigned. Jeden posuv dělá ASR, druhý posuv dělá LSR ( jo, pokud jste zapomněli, tak to jsou instrukce assembleru, zopakujte si )

ad 4 . Zatím moc neumíme dělat zpoždění, tak si prostě v cyklu zopakujte několikrát asm( „nop“ ); (Word odmítá dát uvozovky nahoru) . Protože překladač vygeneruje množství kódu pro obsluhu cyklu for, tak vám výpočty nebudou vycházet ani náhodou. Pustíme si simulátor, stopwatch, a tím zjistíme skutčnou délku cyklu, upravíme ho. Kmitočet samozřejmě vyrábíme tak, že nastavíme bit do 1, chvíli počkáme, nastavíme do 0 atd. adt. atd.

Druhé cvičení

1. Sestavte program, který bude na pinu RB5 vzsílat obdélnékový signál s kmitočtem 1kHz. Použijte čítač a přerušení.
2. Sestavte program, který bude na pinu RB5 vysílat čistou kvartu c2 –f2 . Použijte čítače a přerušení.
3. Sestavte program, který bude podle stisknutí 8 tlačítek – kontakt proti zemi – vydávat na vámi zvoleném pinu tóny v rozsahu jedné oktávy
4. Sestavte program, který bude na vámi zvoleném pinu hrát písničku alespoň v rozsahu písně „Ovčáci - čtveráci“

ad 1. Zprovozníme čítač, dále naprogramujeme přerušení, nezapomeneme vrátit požadavkový bit pro přerušení. V přerušení pak negujeme bit RB5. O registr W5 – stack pointer – se nestaráme, to zařídí gcc .

ad 2. Naprogramujte dva čítače, samozřejmě dvě přerušení. Jeden z čítačů ( rychlejší ) vyrábí tón, druhý čítač přepíná frekvenci prvního čítače. Při vstupu do přerušení od druhého čítače potřebujeme vědět, zda první čítač právě vysílá c2 nebo f2. K tomu si vytvoříme globální logickou proměnnou. Podle jejího stavu poznáme, co právě vysílá první čítač, a proměnnou znegujeme. Samozřekjmě, jde to i jinak, např můžete testovat číslo v registru PR atd. atd. atd.

ad 3. Hračka ! Jeden čítač vyrábí tón, v hlavním programu testujeme vstupní bity a podle toho, který je sepnut, změníme kmitočet prvního čítače.

ad 4. Data pro písničku si uložíme do paměti EEPROM jako pole. Opět máme dva čítače, jeden vyrábí tón, druhý slouží jako časová základna pro změnu tónů. Při každém vstupu do přerušení od druhého čítače se inkrementuje index pole s daty, podle čísla v poli nastavíme kmitočet prvního čítače. Pokuste se implementovat i pomlky . Jak se ukládají data do EEPROM ( mechanismus PSV ) si přečtěte na <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/data/>

Třetí cvičení

1. Sestavte program, který bude na D/A převodníku MCP4822 střídavě vysílat napětí 1V a 3V. Interval mezi změnou napětí udělejte asi 5sec, abyste mohli napětí pohodlně změřit. Časový interval generujte pomocí čítače(ů).
2. Na převodníku vyšlete pilovitý signál s kmitočtem 200Hz , pila lineárně vzrůstá od 0V do 4.095V a pak skočí do 0.
3. Příklad 2 doplňte dvěma vstupními bity. Binární číslo na těchto bitech udává kmitočet pily ve stovkách Hz, pokud je na vstupu 0, je na výstupu převodníku napětí 2.048V.
4. Sestavte si svůj vlastní bizarní signál. Uložte si ho jako vzorky do pole, signál bude mít alespoň 1000 vzorků. Signál pak nechte generovat procesorem.

Ad 4: Odvažte se ! Udělejte třeba postupný nárůst, pak polokruh, pak část sinusovky, pak exponenciálu, pak schody, pak …….. . Na generování vzorků si udělejte program v jazyce C. Program vybleje přesně to, co potom nacpete do svého programu, nejlépe jako include file – viz opět <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/data/>

A-čková část si už pomaličku studuje <http://ozeas.sdb.cz/panska/3ac_stare/wav/> , neboť nás čeká záznam vzorků pomocí A-zákona

Čtvrté cvičení

1. Sestavte program, který bude na D/A převodníku MCP4822 vysílat sinusovku o kmitočtu 200Hz. Sinusovku generujte pomocí DSP algoritmu. V simulátoru si ujasněte, jak dlouho trvá výpočet jednoho vzorku - budeme to potřebovat pro další příklady. Výstup si pustíme na reproduktor, nebo si přineste nějaký vhodný zesilovač.
2. Sestavte program, který bude podle stisknutí osmi kontaktů hrát pomocí převodníku stupnici C-dur.
3. Sestavte program, který pomocí D/A převodníku zahraje nějakou oblíbenou píseň, asi tak v rozsahu „We don't need no education“.
4. Sestavte program, který zahraje několik tónů houslí ( nebo jiného vámi zvoleného nástroje).

ad 1. DSP algoritmus známe, připomínám, že nikde v programu se neobjeví funkce sin. Počáteční konstanty se podle ní sice počítají, ale to jsou jenom čísla, která si dopředu vypočtete a zadáte je do programu. Dále nezapomeňte, že DSP algoritmus vyrábí čísla typu ...... v mezích od .... do ...., a převodník potřebuje čísla typu ...... v mezích od ...... do . Takže to, co nám vyjde, musíte posunout, zvětšit, přetypovat. Nezapomeneme číslo, které pouštíme do převodníku, vhodně upravit (nejvyšší 4 bity). Ale na to jste si už určitě udělali v předchozím příkladu vhodnou funkci. Vzorkovací kmitočet si uděláme pomocí nějakého čítače, v přerušení pak počítáme hodnotu dalšího vzorku a vyšleme na převodník pomocí SPI. No, možná by se hodil i další trik – ono to časem totiž bude už časově nestíhatelné. Takže nejdříve vyšleme vzorek na SPI , a potom počítáme další vzorek. ten si uložíme do nějaké globální proměnné, a až podruhé zavoláme přerušení, tak vyšleme tento uložený vzorek. To samozřejmě znamená, že úplně napoprvé vyšleme nesmysl, ale tím se nezneklidňujte. Toto uspořádání vám umožní vysílat vzorky v přesném čase – vyšlete hodnotu na SPI, a „vono samo“ to pomaličku vysílá bity do D/A převodníku. A paralelně s tím počítáte další vzorek.

ad 2. Dokolečka testujeme vstupní piny, když zjistíme změnu, vynulujeme všechny proměnné, které počítají sinusovku, a začneme od začátku s novými hodnotami koeficientů. Koeficienty si dopředu spočítáme a uložíme do pole.

ad 3. Tóny písně si uložíme do pole. Dále si naprogramujeme další čítač, který bude krokovat inkrementovat index pole s písničkou, a vždy podle uložené noty znovu odstartuje DSP algoritmus s jinými koeficienty

ad 4. Housle ( nebo jakákoli jiný nástroj) dodávají kromě základního tónu i vyšší harmonické, a tím se liší od jiných nástrojů. Popis máte např. v práci

<https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=151624>

od strany 50 jsou tabulky s číselnými hodnotami

Náš program bude tedy pomocí DSP algoritmu generovat několik harmonických s různými amplitudami. V citované práci je popsán jejich poměr v dB, ale to doufám pro vás nebude problém. Amplitudu harmonických zvolíme tak, aby se jejich součet i v té nejhorší možné variantě vešel do rozsahu převodníku. No a potom už jenom odstartujeme algoritmus, počítáme, sčítáme a vysíláme do převodníku. Tady už možná procesor nebude stíhat počítat, tak mu zvedneme kmitočet pomocí PLL. Parametry jednotlivých harmonických si zase uložíme do pole včetně jejich amplitud. Zkusíme zahrát několik tónů – např. autorem práce popisované prázdné struny houslí g d a e. Volbu tónu uděláme podobně jako v příkladu 2 – pomocí kontaktů.

**Páté cvičení**

1. Sestavte program, který bude měřit velikost napětí na vámi zvoleném pinu. Čtyři nejvyšší bity výsledku zobrazte na čtyřech diodách.

až sem za 3-

1. Sestavte program, který „udělá z PICa drát“, tedy: v procesoru naprogramujeme A/D převodník, k procesoru připojíme D/A převodník. Stejnosměrné napětí, které budeme přivádět na vstupní pin procesoru, se bude kopírovat na výstupní pin D/A převodníku. Zařízení má pracovat v telefonním pásmu. U zařízení předveďte, co je aliasing (nastavte takové vstupní podmínky, aby aliasing byl viditelný na osciloskopu)

až sem za 2

Bod 2 je „conditio sine qua non“

1. Sestavte program, který bude sloužit jako zpožďovací linka s maximálním možným zpožděním.

Až sem za 1

ad 1

Prostě nastavíme A/D převodník, pomocí čítače si zajistíme vhodnou frekvenci vzorkování. V přerušení od čítače vždy odstartujeme převod převodníku, v přerušení od převodníku pak čteme převedené číslo a jeho nejvyšší bity zapíšeme na port.

ad 2

Číslo, které dodal A/D převodník, vyšleme pomocí SPI do D/A převodníku. Nezapomeneme na maskování horních čtyř bitů - řídící slovo pro D/A převodník

ad 3

Mezi příjem a vysílání vložíme zpožďovací linku. Ta bude vytvořena pomocí pole maximální velikosti. Data, která dodal převodník, ukládáme do pole a zároveň čteme data pro D/A převodník. Udělejte to následovně: v přerušení od A/D převodníku NEJDŘÍVE přečteme data z jednoho prvku pole a vyšleme je na SPI, a POTOM do téhož prvku pole uložíme data z A/D převodníku. Index pole pak zvětšíme o 1. Až index pole doleze na konec pole, nastavíme mu opět počáteční hodnotu. V tomto systému samozřejmě při prvním průchodu cyklem bude D/A převodník vysílat nesmysly. To ale nevadí, stejně to vůbec nijak nezaznamenáte.

**Šesté cvičení**

1. Sestavte program, který bude fungovat jako derivační člen.

Ve zbytku času si můžete opravit 5 z minulých hodin. Připomínám, že programy budou v určených adresářích.

ad 1

Je to úplně jednoduché. Sestavíme „ drát z PICa “ a mezi vstup a výstup vložíme DSP algoritmus pro derivační člen. Nezapomeňte, že výstup z algoritmu je kladný i záporný, takže si s tím musíte nějak poradit. Předvedete, že derivační člen dělá to, co má dělat. Pokud by výstupní napětí bylo příliš velké nebo malé, vynásobte výstup vhodným číslem.

Na vstupu potřebujeme oddělit stejnosměrnou složku a udělat „virtuální zem“, tedy ss napětí 2.5 V. Na bastldesce na to máte připravené R4 R3 C9 , popřípadně R5 R2 C8

**Sedmé cvičení**

1. Sestavte program, který bude fungovat jako měnitelný atenuátor. Kromě analogového vstupu a výstupu má procesor dva vstupní bity. Podle čísla na těchto bitech vkládá program do cesty signálu útlum 0dB, 6dB, 12dB, 18dB

až sem za 4

1. Sestavte program, který se bude chovat jako Schmittův klopný obvod s rozhodovacími úrovněmi 1V a 3V. Výstupní napětí obvodu bude 0.5V a 3.5V. ( pokud teď začnete brečet, že nevíte, co to je, tak skočte za svými vyučujícími z EO. Anebo místo toho zkuste „ask Google“. Prostě si poraďte ) .

až sem za 3

1. Sestavte program, který se bude chovat jako omezovací člen. Program má dva vstupní bity. Podle velikosti čísla na těchto bitech omezí výstupní napětí na 1V, 2V, 3V, 4V . Jinak se chová jako „drát z PICa“

až sem za 2

1. Sestavte program, který se chová jako integrační člen.

až sem za 1

ad všechno: na bastldesce máte dva odpory, které tvoří virtuální zem ( 2.5V ). K nim máte kondenzátor pro oddělení stejnosměrné složky. Tento obvod použijte na vstupu, abyste do zařízení mohli pouštět střídavé napětí kladné i záporné, přes kondenzátor. Napěťový dělič pak připojíte na vstup převodníku. Tím, posunete střídavé napětí o 2.5V

ad 4

Bohužel se to bude chovat jako integrační člen s ideálním operačním zesilovačem, to mimo jiné znamená, že výstup bude „ujíždět“ někam do saturace. První věc: na vstupu potřebujeme kladná i záporná čísla. Použijeme tedy u D/A převodníku „signed integer“ a on nám vybleje správné číslo. Nebo od „unsigned integer“ odečtete číslo, které odpovídá polovině rozsahu. To ale také nebude stačit. Předpokládám, že použijete virtuální zem. Ta samozřejmě nebude mít 2.5V, ale trochu jiné napětí. Zkuste si třeba dát na nějaký digitální pin tlačítko, to tlačítko stisknete, když na vstupu nebude žádné střídavé napětí, a uložíme si hodnotu, kterou převodník dodá při stisku tlačítka. Tuto hodnotu pak použijeme jako referenční - prostě ji odečteme od čísel, která bude dodávat převodník. Tím dost omezíme „ujíždění“ integračního členu do saturace, ale ne absolutně. Ale alespoň na nějakou chvíli by to mělo stačit.

**Osmé cvičení**

1. Sestavte program, který bude na výstupu UARTu v sériovém tvaru postupně vysílat písmena abecedy ABCDEF…..XYZ . Rychlost nastavte co možná nejvyšší. Vstup a výstup UARTu nastavte na procesoru tak, abyste mohli přímo použít UART tool u programovátka PICKIT2. Interval mezi vysláním znaku udělejte asi 1 sec, ke generování časových intervalů použijte čítač a přerušení.

Až sem za 4

1. Sestavte program, který se bude měřit napětí na některém pinu procesoru a bude ho vysílat „human readable “ na UARTu. Periodu vzorků volte asi 1 sec.

Až sem za 3

1. Sestavte program, který bude měřit VA charakteristiky jednoduchých součástek. Předveďte na diodě.

Až sem za 1

Ad 2

Je prostým prodloužením příkladu 1. UART má FIFO, takže do něj můžeme „vychrlit“ 5 znaků, podle toho si zvolte, jak bude vypadat formát čísla. Nezapomeňte na odřádkování. Možná bude maličko problém s převodem na ASCII, ale je to snadné. Číslo, které přečteme z převodníku, upravíme tak, aby 5V odpovídalo číslu int 5000. Pak je to jednoduché: číslo budeme celočíselně dělit 10 , a zbytek jsou tisícíny. Pak ho vydělíme normálně 10. To, co vyjde, budeme celočíselně dělit 10 , a zbytek jsou setiny . Pak ho normálně vydělíme 10 . To, co vyjde, budeme celočíselně dělit 10 , a zbytek jsou desetiny . Pak ho normálně vydělíme 10 . atd. atd. atd. Tento postup se nazývá Hornerovo schéma, zeptejte se vašich matematikářů. Zbytky si uložíme nejlépe do pole. Poslední věc je, jak převést číslici na odpovídající ASCII kód. Napište si vedle sebe číslice a jejich ASCII kódy , chvíli se na to dívejte a dojde vám to. Nějak takhle:

1. ……. ASCII kód 0
2. ……. ASCII kód 1
3. ……. ASCII kód 2
4. ……. ASCII kód 4

Atd. atd. atd. Připomínám, že Cčko umí i věci jako

b = c + ‘X’ ;

Ad 3

Náš D/A převodník nedá příliš velký proud. Podívejte se do datasheetu a podle toho zvolíme diodu, kterou budeme měřit. Můžete si samozřejmě dát za převodník zesilovač, ale pro naše pokusy bude stačit D/A převodník sám. Abychom mohli měřit proud, dáme do série s diodou ( nebo jiným prvkem ) rezistor. Tohle celé zapojíme na výstup D/A převodníku. Proud pak vypočteme z napětí a odporu. Dále potřebujeme napětí na diodě. K tomu musíme zapojit ještě jeden vstup A/D převodníku. Napětí pak vypočtete jako rozdíl napětí mezi vstupy převodníku. Pak je to jednoduché: zvyšujeme napětí na D/A převodníku a měříme si napětí na A/D, z toho vypočteme proud diodou a napětí na diodě. Pomocí UARTu to pak vyšleme, nejlépe ve tvaru

0.002 0.456

0.003 0,567

0.004 0,899

Atd. atd., tak, abyste to pak mohli importnout do Excelu a udělat graf. UART Tool umí zapsat přijatá data do file.

Ještě další problém bude s vysíláním znaků. Máme jich na řádek poměrně hodně, takže již nevystačíme s 4-úrovňovou FIFO v UARTu . Uděláme si proto globální pole char. Do tohoto pole budeme postupně zapisovat znaky. Udělejte si na to nejlépe funkci zapisznak( char znak ) . Součástí funkce je i globální ukazovátko, které ukazuje na poslední obsazený prvek pole. Dále naprogramujeme vysílací přerušení od UARTu a naučíme ho, že pokud máme v bufferu nějaký znak, tak vysíláme. Pokud nemáme znak, zakážeme vysílací přerušení. Hodnoty v bufferu se samozřejmě musí posouvat , když odvysíláte prvek. To ale už nechám na vás. Je doufám jasné, že po zprovoznění tohoto mechanismu máte vystaráno, prostě pomocí vysliznak nacpete do bufferu znaky a „vono samo“ je to odvysílá .

**Deváté cvičení**

1. Sestavte program, který bude pomocí UARTu ovládat osm LED diod na portu B. Bude fungovat např. takto: na UART vyšleme LED1ON a dioda se rozsvítí, LED1OFF a dioda zhasne. V případě chyby program odpoví ERROR

Až sem za 3

1. Sestavte program, který bude přijímat data pomocí UARTu.. Do UARTu vždy pošleme hodnotu napětí, kteá se má objevit na výstupu převodníku, a program toto napětí nastaví. Například: 1.458V 2.654V 0.125V atd. atd. V případě zadání nesprávné hodnoty program odpoví ERROR a vzsvětlí, co je špatně ( desetinná tečka je čárka, napětí je moc velké, napětí je záporné, naprosto zmatený příkaz atd. atd. )

Až sem za 1

Ad 1

Znaky si postupně ukládejte do pole, až do okamžiku, kdy přijde konec řádky. Includněte si string.h a přišedší řetězec porovnejte s požadovaným, no a podle výsledku udělejte akci. Substring ve stringu jsme už hledali.

AD 2

Doporičuji vaší pozornosti funkce jako atoi , atof atd. atd. atd.

**Desáté cvičení**

1. Sestavte program, který bude vysílat na některém pinu PWM s proměnnou střídou. Střídu určíme pomocí UARTu. Na UART budeme vysílat číslice, a tyto číslice budou znamenat střídu PWM v desítkách procent. Například: přijde číslice 2, PWM se nastaví na 20 procent atd. atd. V případě chyby program odpoví ERROR

až sem za 3

1. Sestavte program, který bude ovládat tři různé PWM na třech výstupech. Střída impulzů se bude opět ovládat pomocí UARTu, tentokrát nějak takto:

PWM1 28 nastaví na první PWM střídu 28 procent

PWM3 59 nastaví na třetí PWM střídu 59 procent atd. atd. V případě chybného zadání procesor odpoví ERROR.

až sem za 1

**Jedenácté cvičení**

1. Sestavte program, který bude pomocí DSP engine generovat sinusovku. Udělejte asi tak 20 vzorků na periodu, nastavte co možná nejvyšší kmitočet oscilátoru pomocí PLL a nechte vysílat na DA převodníku sinusovku o co možná nejvyšším kmitočtu.
2. Pomocí DSP engine sestavte filtr FIR s co možná nejvyšším vzorkovacím kmitočtem.

ad 1

Čteme si návod v adresáři ./fixed\_point/engine01. Pro odstartování se hodí něco jako podprogram uvod a uvod02 – nacpat do registrů počáteční hodnoty pro sinusovku, nastavit DSP engine na signed fractional, nastavit počáteční hodnoty. No a potom v přerušení od čítače voláme něco jako treti – počítáme hodnotu sinusovky pomocí DSP algoritmu a blejeme ji nven nebo ji hned cpeme do SPI – číslo nutno pošoupnout (ASR) a přičíst k němu polovinu rozsahu DA převodníku, zamaskovat

ad 2

přečteme si v návodu o modulo addressing a použijeme to . AD převodník můžeme nastavit do módu „signed fractional“, takže nám bude rovnou dodávat čísla ve správném formátu.