**Třetí cvičení**

1. Sestavte program, který bude na výstupu převodníku MCP4822 střídavě s periodou 2 sec vysílat napětí 1V a 3V.
2. Sestavte program, který bude na výstupu převodníku MCP4822 vysílat pilu, která stoupá od 0V do maximálního napětí, pak opět „spadne“ do 0. Perioda pily je 10 sekund.

Jak vidíte , příklady jsem upravil tak, abyste nepotřebovali osciloskop. Nicméně k odevzdání:

Pro každý příklad si uděláte jiný zdrojový file. Zdrojový file bude mít název 17tPrijmeniJ\_CC\_PP.c

kde t je A pro 17A , D pro 17D : CC je dvoumístné číslo cvičení , PP je dvoumístné číslo příkladu. Takže výše uvedená pila bude ve file 17AKubalikT\_03\_02.c

Soubor pak uložíte do zadané složky v Microsoft Teams ( pan Machač to aktuálně připravuje, budete včas informováni). Oba příklady naprogramujete do své bastldesky, na výstup připojíte multimetr a předvedete, jak to funguje. Multimetr spolu s bastldeskou si natočíte na svůj mobilní telefon a soubor s videem uložíte taktéž do složky v Teems. Ke každému příkladu tedy odevzdáte dvě věci:

1. fungující zdrojový file
2. video s natočenou bastldeskou a voltmetrem, které dokazuje, že jste to udělali a že to funguje . Video nahrajete v jakémkoli formátu, který se dá přehrát pomocí VLC, a uložíte do Teams. Nebo ho uložíte na YouTube a do Teams dáte odkaz.

K řešení příkladu si přečtěte návod , který máte v <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/3A_2020.docx> , Druhé cvičení

ANO, OPRAVDU, i 3D čte návod pro 3A !!!!!!!!!!!!

Do přerušení od SPI si dejte příkaz, který neguje nějaký výstupní pin, a na něj si dejte LED. Pak poznáte, zda přerušení funguje.

**Čtvrté cvičení**

1. Sestavte program, který bude mít dva vstupní bity. Na těchto bitech budete mít kontakty proti zemi. Na výstupu převodníku MCP4822 bude program vysílat takové napětí ve voltech, které odpovídá číslu na vstupních bitech.
2. Sestavte program, který bude na výstupu převodníku MCP4822 vysílat pilu, která stoupá od 0V do maximálního napětí, pak opět „spadne“ do 0. Perioda pily je 10 sekund. Program má dále jeden vstupní bit – kontakt proti zemi. Pokud je kontakt rozepnut, výstupní napětí stoupá, jak je popsáno výše. Pokud kontakt sepneme, pila se zastaví a napětí zůstane stálé do doby, než kontakt opět rozepneme. Pak pokračuje ve stoupání.

Pokyny k odevzdání a řečení jsou stejné jako u cvičení třetího. Kontakty proti zemi uděláme drátkem. Nezapomeňte zapnout pull-up rezistory.

ad 1

Předpokládám, že zadání je naprosto jasné, ale pro jistotu ještě uvádím tabulku:

|  |  |
| --- | --- |
| Hodnota vstupních bitů | výstupní napětí na převodníku |
| 00 | 0 V |
| 01 | 1 V |
| 10 | 2 V |
| 11 | 3 V |

Na výstup převodníku dáte multimetr, batldesku i s multimetrem umístíte do záběru svého mobilního telefonu, natočíte video, kde bude vidět, jak přepínáte vstupní čísla a jaké je výstupní napětí.

ad 2

Příklad 2 z třetího cvičení doplníte jedním vstupním bitem, kterým budete zapínat a vypínat stoupání pily. Možností je celá řada:

1. zapnout – vypnout přerušení od čítače
2. v přerušení udělat if() u příkazu, kterým zvyšujeme napětí
3. a tak dále a tak dále

**Páté cvičení**

1. Sestavte program, který bude mít na vstupu stejnosměrné napětí v rozsahu 0 – 5V. V programu zprovozníte AD převodník procesoru. Na výstupu bude mít čtyři bity. Na těchto bitech se objeví čtyři nejvýznamnější bity z převodníku.

Je to nový prvek, proto máte jenom jeden příklad. Již jste si doufám prostudovali látku z teoretických hodin, bez ní to nepůjde. Naprogramujete převodník, vstup si dáte kamkoli se vám to hodí. Dále potřebujeme zdroj vzorkování. To si uděláme pomocí nějakého čítače a v přerušení od tohoto čítače „odstartujeme“ převod pomocí **bclr ADCON1, #SAMP**  . Tuto variantu doporučuji pro první pokusy, protože v simulátoru tak můžete snadno kontrolovat, zda jste opravdu odstartovali převod. Až vám tohle půjde, můžete experimentovat s jiným způsobem odstartování převodu, např. pomocí

100 = Timer5 compare ends sampling and starts conversion

010 = Timer3 compare ends sampling and starts conversion

ASAM=1 , SSRGC = 0

Dále si zprovozníme přerušení od DA převodníku. V něm přečteme registr s výslednou hodnotou a vyšleme na výstupní port. Na port si dáme čtyři LED, abychom viděli, jak nám to krásně funguje. Pokud si LED a port zvolíte šikovně, můžete výstupní data tupě okopírovat. Pokud to uděláte nešikovně, budete muset data posunout – k tomu si nastudujte <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/logicke/logicke.doc>

Na vstup si dáte potenciometr. Budete s ním točit, a na výstupních diodách bude vidět, jak se napětí mění. Tohle natočíte na video na svůj mobil a spolu se zdrojovým file uložíte do teams. V komentářích ve zdrojovém file bude uvedeno:

1. který vstupní pin jste použili a kde je to nastaveno
2. jaký kmitočet vzorkování jste použili a kde je to nastaveno
3. kde a jak převádíte výstupní čísla z převodníku na LED
4. jakým způsobem získáváte proměnné napětí na vstupu. To bude také samozřejmě vidět na videu

Pokud nemáte potenciometr, udělejte si odporový dělič. Pokud nemáte odpory, zkuste baterky v různém stádiu vybitosti. Pokud ani to nemáte, udělejte si Voltův článek. Pokud ve 3. ročníku elektrotechnické průmyslovky nejste schopni zajistit si zdroj proměnného stejnosměrného napětí, odejděte na gymnázium.

A zopakujte si, co dělá digitální signálový procesor. Několikrát nahlas, snad vám to dojde.

**Šesté cvičení**

1. Sestavte program, který bude fungovat jako drát ( dostali jsme se tedy k legendárnímu programu „drát z PICa“ ). Program bude mít na vstupu stejnosměrné napětí v rozsahu 0 – 5V, pomocí AD převodníku je převeďte na čísla, pomocí DA převodníku převeďte čísla opět na napětí. Vzorkovací kmitočet je 8kHz.

Na vstup si dáte potenciometr (nebo jiný zdroj napětí stejně jako ve cvičení 5  ). Na výstup převodníku dáte voltmetr. S potenciometrem budete točit, a na voltmetru bude vidět, jak se napětí mění. Tohle natočíte na video na svůj mobil a spolu se zdrojovým file uložíte do teams. V komentářích ve zdrojovém file bude uvedeno:

1. který vstupní pin jste použili a kde je to nastaveno
2. kde je a jak je nastaven vzorkovací kmitočet
3. kde a jak převádíte výstupní čísla z převodníku z AD převodníku na DA převodník
4. který výstup DA převodníku jste použili a kde je to nastaveno
5. jakým způsobem získáváte proměnné napětí na vstupu. To bude také samozřejmě vidět na videu

**Sedmé cvičení**

Sestavte program, který přepne kmitočet vnitřního oscilátoru ze standardní hodnoty na hodnotu, kterou máte zadanou v tabulce . Kmitočet dolaďte pomocí OSCTUNE na co nejpřesnější hodnotu. Program bude blikat s libovolnou LED frekvencí asi 1Hz. Po pěti sekundách přepne vnitřní oscilátor na vyšší kmitočet, to bude vidět na zrychleném blikání diody - nastavení čítačů tedy neměníme ! Odevzdáte následující věci:

1. programový kód s komentáři
2. v textovém souboru popsaný postup výpočtu jednotlivých koeficientů pro nastavení PLL
3. obrázek z datasheetu na straně <http://ozeas.sdb.cz/panska/mikroproc/33EV32/pdf/33EV32.pdf> na straně 124. Do obrázku doplníte vaše konkrétní hodnoty kmitočtů.
4. Video s natočenou blikající diodou.

Tabulka s vašimi kmitočty je na URL <http://ozeas.sdb.cz/panska/mikroproc/33EV32/programovani/progr_a_vysv/oscilator_kmitocty.xlsx>

Zde není příliš co radit, postupujte podle materiálů z předchozích teoretických hodin.

ad 3: pokud nejste schopni upravovat pdf file, zkopírujte si obrázek jako obrázek a v nějakém obrázkovém editoru (např. populární Malování ve Windows) do něj doplňte vaše kmitočty . Tím myslím následující: v obrázku na straně 124 je například 120 MHz < FSYS(1) < 340 MHz Vy do tohoto místa napíšete 214,28 MHz , samozřejmě ten kmitočet, který používá váš program. V obrázku tak přepíšete celkem 4 slovy: čtyři hodnoty . Čtvrtá hodnota je kmitočet FIN , ten se samozřejmě nastavuje děličkou FRCDIV, která je před blokem PLL.

**Osmé cvičení – PWM**

Sestavte program, který bude na vámi zvoleném pinu generovat PWM. Pomocí PWM budeme ovládat jas připojené LED. Kmitočet PWM zvolte tak, aby blikání LED nebylo okem viditelné. Procesor má dále dva vstupní piny – kontakt proti zemi. Pomocí čísel na vstupních pinech se nastaví PWM následovně:

00 – Duty cycle 20 %

01 – Duty cycle 40 %

10 – Duty cycle 60 %

11 – Duty cycle 80 %

Jako obvykle uložíte do TEAMS zdrojový file a video, které dokládá funkčnost vašeho programu. Můžete použít LED na bastldesce, také můžete použít osciloskop, který je součástí PICKITu ( k tomu viz např. <http://ozeas.sdb.cz/panska/mikroproc/PICKIT2/PICKIT2/PICKIT2_pouziti.doc> , od strany 25 dále )

Při testování vstupních pinů si uložte do nějaké proměnné jejich minulý stav a porovnávejte se stavem stávajícím. Změny v registrech PWM pro změnu Duty cycle delejte jen v případě, že detekujete změnu na vstupních pinech.

**Deváté cvičení – PWM podruhé.**

Sestavte program, který bude na vámi zvoleném pinu generovat PWM. Pomocí PWM budeme ovládat jas připojené LED. Kmitočet PWM zvolte tak, aby blikání LED nebylo okem viditelné. Jas LED se bude postupně během 5 sekund zvyšovat od 0 až do maxima, během dalších 5 sec se bude snižovat od maxima až do 0 . Celý cyklus se neustále opakuje.

Cvičení navazuje na osmé cvičení. Stejně jako tam zprovozníme generátor PWM. Dále nastavíme nějaký čítač. V přerušení od tohoto čítače budeme postupně zadávat do řídícího registru PWM taková čísla, aby se Duty cycle měnil od 0 do 100 % . Kmitočet čítače musíte udělat tak, aby kmitočet změn Duty cycle byl podstatně menší než je kmitočet PWM ( předpokládám, že je jasné, že není možno změnit parametry PWM dříve, než se dokončí alespoň jedna, nebo lépe několik period PWM. )

**Desáté cvičení – derivační člen.**

Sestavte program, který bude sloužit jako derivační člen pro analogové signály. Na vstupní AD převodník budeme přivádět měnící se napětí, na výstupu DA převodníku budeme sledovat na voltmetru derivaci vstupního napětí. Pro demonstraci funkce použijte průběh napětí podle následujícího obrázku. Nedílnou součástí úkolu je i slovní teoretický rozbor, ve kterém vysvětlíte , jaké hodnoty očekáváte na výstupu a proč, a zdali jste dosáhli deklarovaných výsledků. Bez tohoto slovního rozboru bude úkol hodnocen známkou 5.



Poznámky k realizaci:

sestavíme si „drát z PICa “ podle šestého cvičení. Mezi data, která odcházejí ze vstupního AD převodníku, a data, která cpeme do výstupního DA převodníku, vložíme algoritmus pro výpočet derivace ( viz tux.panska.cz:/home/kubalik/prace/ukol04 ). Prostě vezmete kód z ukol04 a nacpete ho do kódu pro šesté cvičení mezi výstup AD a vstup DA převodníku. Samozřejmě, musíme vzít správnou část kódu.

Další nepříjemnost je, že derivace bude vycházet kladná i záporná. To vyřešíme tak, že nulu na výstupu posuneme doprostřed rozsahu ( prostě k číslům, která dodá algoritmus pro derivační člen, přičtete vhodně velké číslo. Tím přidáte stejnosměrnou složku a výstupní signál posunete ).

Vstupní signál dostaneme tak, že si na vstup připojíme potenciometr a budeme s ním kroutit. Zkuste kroutit rychleji a pomaleji a sledujte, co to dělá s výstupní derivací. Vzorkovací kmitočet si zvolte tak, aby vám program dobře počítal ( tady si prostě zvolíte nějaký vzorkovací kmitočet, podle zadaného průběhu napětí si spočtete několik hodnot, „ručně“ spočítáte několik výstupních vzorků. kouknete na to a řeknete si: „ta čísla jsou moc malá / velká , takže u vzorkovacího kmitočtu musím přidat/ubrat“ )

Prostě to udělejte tak, abyste na voltmetru viděli použitelné hodnoty. Přidáte jako obvykle video, na kterém bude vidět vstupní potenciometr a výstupní voltmetr .

**Jedenácté cvičení – UART poprvé.**

Sestavte program, který bude sloužit jako vysílač UARTu . Bude vysílat pořád dokolečka velkou abecedu ABDC....XYZ rychlostí 2400 Bd. Výstupní piny zvolte tak, abyste mohli vysílaná data přijímat pomocí PICKITu – UART TOOL . Vysílání udělejte co možná nejrychlejší, tedy po odvysílání jednoho písmena hned začněte vysílat písmeno další.

Doporučuji následující postup:

1. naprogramujeme vysílací a přijímací piny pomocí PPS tak, aby souhlasily s piny na PICKITu. Jde to, musíte se podívat na schéma, kam co vede.
2. Dále si naprogramujeme čítač s rozumně dlouhou periodou. V přerušení od tohoto čítače budeme dokolečka vysílat jediné písmenko, třeba H.
3. Dále program upravíme tak, aby postupně pomalu vysílal celou abecedu. Dělali jsme v prvním pololetí.
4. Teď už zbývá jenom splnit podmínku „Vysílání udělejte co možná nejrychlejší“ . Naprogramujeme si přerušení od vysílače UARTu, a do něj přestěhujeme kód, který vyrábí abecedu. V jednom zavolání přerušení samozřejmě vyšleme jen jedno písmeno ! UART funguje tak, že když nemá co vysílat, volá vysílací přerušení. Možná budete muset v hlavním programu odstartovat vysílání prvního písmenka tím, že nastavíte požadavkový bit přerušení do 1, a tím procesor donutíte zavolat vysílací přerušení.

**Dvanácté cvičení – UART podruhé.**

1. Sestavte program, který bude sloužit jako voltmetr. Výsledky bude vysílat na rozhraní UARTu v ASCII kódech, s periodou 1 sec. To znamená, že každou sekundu dodá voltmetr na výstup aktuální napětí na vámi zvoleném vstupu.
2. Sestavte program, který bude sloužit jako smyčka. Písmeno, které mu pošlete po UARTu, hned vyšle po UARTu.
3. Sestavte program, který bude pomocí UARTu ovládat jas vámi zvolené LED pomocí PWM. Na vstup UARTu posíláme číslice 0-9 v ASCII kódu, procesor reaguje tak, že nastaví duty cycle u PWM na hodnoty 0 – 90 procent ( čáslice x 10 = duty cycle v procentech ) . Pokud pošleme jíný znak než šíslici, procesor odpoví ERROR
4. Sestavte program, který bude sloužit jalo měnitelná atenuátor. Útlum atenuátoru v dB budete zadávat pomocí UARTu. Oddlělovací znak je mezera, pokud zadáte nesmzslené znaky, procesor odpoví ERROR , opět po UARTu.

ad 1

Je to poměrně jednoduché. Zprovozníte převodník, doporučuji formát čísel unsigned integer. Naprogramujete čítač, který bude volat přerušení s frekvencí 1Hy. V přerušení od čítače odstartujete převod převodníku. V přerušení od převodníku přečteme číslo, které odpovídá vstupnímu napětí. Číslo musíme upravit, protože maximálnímu napětí, které je ... V, odpovídá číslo ..... . Dále je nutno číslo převést na string. Doporučuji includnout string.h a zkusit použít funkci sprintf. Pokud to nepůjde, použijeme Hornerovo schéma .

SEM:

Zjistíme zbytek po celočíselném vydělení deseti. Tento zbytek jsou jednotky čísla. Uložíme do nějaké proměnné, já samozřejmě doporučuji do pole, aby se algoritmus dal zacyklit. Číslo celočíselně vydělíme deseti. GOTO SEM

Zbytky jsou postupně jednotky, desítky, tisíce, desetitisíce ...... z původního čísla.

Nyní musíme číslo převést na jeho reprezentaci v ASCII kódu . K tomu účelu si napíšeme pod sebe do sloupce čísla 0 – 9 , a vedle nich do sloupce ASCII kódy číslic 0 – 9, v desítkovém tvaru. Budeme se na to dívat tak dlouho, až nám dojde, co se s tím musí udělat. ( a pokud vám to nedojde, tak zkuste třeba něco přičíst – odečíst, vynásobit-vydělit, umocnit – odmocnit ........ )

No a potom už to jenom nacpeme do RXREG. Samozřejmě, musíme počkat, až se číslice odvysílá ! K tomu si udělejte přerušení od vysílače UARTu, a další znak vysílejte tehdy, až se přerušení zavolá. Číslo doplňte nějakým hezkým řetězcem, například „Napeti je “ a „V“ . Jo – odřádkování má také ASCII kódy, je rozdíl mezi Linuxovým a DOSovským koncem řádku.

ad 2

Zprovozníme UART. Dále uděláme přerušení od přijímače UARTu . V přerušení to, co přišlo ( RXREG ), hned odvysíláme (RXREG)

ad 3

To je možná jednodušší než 1. Zprovozníme PWM, dále UART. Uděláme přerušení od přijímaše UARTu. V přerušení budeme testovat, co přišlo, a podle hodnoty přišedšího znaku nastavíme registr, který ovládá Duty cycle.

ad 4

Zprovozníme „drát z PICa“ . Dále zprovozníme UART a přerušení od přijímače UARTu. Podle toho, co přijde, dám mezi číslo z AD převodníku a číslo, které odchází do DA převodníku, útlum. úUtlum 6dB znamená, že se s napětím udělá ....... , útlum 12cdB znamená, že se s napětím udělá .......

Pro první pokusy doporučuji posílat jednu číslici a útlum nějak nastavovat podle ní. Dále se pustíme do doopravdické interpretace útlumu. To znamená, že přijmete několik číslic , za kterými následuje mezera. Přicházející číslice si budete postupně ukládat do pole. Až přijde mezera, která má ASCII kód ..... , zahájíme akci. String převedeme na číslo pomocí funkce ( viz první pololetí ). Pokud se to nepovede, napíšete ERROR nebo něco podobného na vysílač. Pokud je číslo rozumně velké, převedete dB na to, co máte s napětím udělat, a uděláte to. Pokud v math.h nenajdete mocninu, pak prostě číslo zlogaritmujete, vynásobíte exponentem, a odlogaritmujete . Připomínám, že string v Cčku končí ........ . Tohle číslo musíte vy do toho pole uložit , „vono samo“ se to fakt neudělá.