**První cvičení**

1. Zprovozněte demonstrační program z <http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/progr_a_vysv/zaklady/> - druhy\_03.asm, predvadeni.docx . Na PICKITu předveďte, že vysílá data v sériovém tvaru, na osciloskopu předveďte PWM, připojte diody, které mají blikat.

Až sem za 4

1. Sestavte program, který bude periodicky kopírovat číslo z portu A na port C.

Až sem za 3

1. Sestavte program, který bude periodicky kopírovat číslo z portu B na port C. Vstupní hodnoty budeme zadávat pomocí kontaktů proti zemi (realizujte pomocí drátků, které se dotknou)

Až sem za 2

1. Na portu C máme dvě čtyřbitová čísla. Sestavte program, který tato dvě čísla sečte a výsledek zapíše na port A.

Až sem za 1

Ad 1:

Na začátku programu musíme nastavit všechny potřebné registry – TRIS, ANSEL . To uděláme v celém programu jen jednou, na začátku. Potom dokolečka čteme číslo z portu A a zapisujeme na port C. Cyklus uděláme pomocí goto a návěští . Určitě si tento program vyzkoušejte v simulátoru **a zejména se podívejte na nastavení registrů TRIS.**  Pokud se vám povede připojit výstupní port na vypínače, na kterých je „natvrdo“ +5V nebo 0V, port vám shoří. Vstupní hodnoty na portu zadáváme pomocí Stimulus v simulátoru <http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/MPLABX/simulator_podruhe.doc> )

Ad 2:

Oproti 1 potřebujeme dále zapnout Pull-Up rezistory. Nezapomeňte na bit WPUEN v registru OPTION\_REG ( instrukce bsf , bcf )

 Ad 3

Porty nastavíme jako vstupní a výstupní podle potřeby. Přečteme číslo z portu C. To musíme dále rozdělit na dvě čísla. Takže si ho uložíme do dvou buněk. Dále musíme z osmi bitů dostat dvě čtyřbitová čísla. To uděláme tím, že číslo zamaskujeme ( and ) . Jedno z čísel je ale na horních bitech ( 7 - 4), a my ho potřebujeme dostat na dolní bity ( 3 -0 ) To uděláme pomocí posuvu - LSRF. Instrukce máte na <http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/prirucky/instrukce.doc> . Takto získaná dvě čtyřbitová čísla pak prostě sečteme a zapíšeme na port A. No a tyto akce děláme pořád dokolečka – goto navesti. Vše si nejprve odzkoušejte v simulátoru !

**Druhé cvičení**

1. sestavte program, který bude na pinu RA2 generovat obdélníkový signál o kmitočtu 10kHz.

až sem za 4

1. Sestavte program, který bude na pinu RB5 generovat obdélníkový signál o kmitočtu 100Hz

až sem za 3

1. Program podle bodu 2 doplňte jedním tlačítkem – kontakt proti zemi. Pokud je kontakt rozepnut, je kmitočet signálu 100 Hz. pokud je sepnut, je kmitočet signálu 400Hz.

až sem za 2

1. Sestavte program, který bude na pinu RC1 generovat obdélníkový signál s periodou 1sec. Střída impulsů bude 1:3 . Signál kontrolujte pomocí LED diody.

až sem za 1

ad 1

Je to velmi jednoduché. Pin nastavíte do 1, chvíli počkáte, pak ho nastavíte do 0, chvíli počkáte, a tak pořád dokolečka ( bcf bsf ). Nezapomeňte uložit do registru OSCCON číslo 0x6a , abyste měli kmitočet instrukcí 1MHz. Do kódu vložte hlavičku – ostatně se podívejte na vzorový příklad http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/progr\_a\_vysv/zaklady/druhy\_03.asm

 \_\_CONFIG \_CONFIG1, \_FOSC\_INTOSC & \_WDTE\_OFF & \_PWRTE\_ON & \_MCLRE\_OFF & \_CP\_OFF & \_BOREN\_OFF & \_IESO\_OFF & \_FCMEN\_OFF & \_CLKOUTEN\_OFF

 \_\_CONFIG \_CONFIG2, \_WRT\_OFF & \_LVP\_OFF & \_PPS1WAY\_ON & \_ZCDDIS\_ON & \_PLLEN\_OFF & \_STVREN\_ON & \_LPBOR\_OFF & \_BORV\_HI & \_LPBOR\_OFF

**!!!!!!! Hlavička v druhy\_03.asm je trochu jiná, má tam \_CLKOUTEN\_ON – to znamená, že na pinu 3 procesoru vychází kmitočet vnitřního oscilátoru. Pokud chcete tento kmitočet vypnout, musíte do konfigurace napsat \_CLKOUTEN\_OFF , stejně jako je to v tomto file ( ten, který právě čtete)**

Celé si to zkontrolujte pomocí STOPWATCH. Návod máte na <http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/MPLABX/simulator_stopwatch.docx>

ad 2

Totéž jako 1 , jenom potřebujete delší časy na čekání. Nezbude vám nic jiného, než si pro čekání udělat vhodný cyklus.

ad 3

3 je příklad 2, který doplníme tlačítkem. Čteme hodnotu portu, a podle její hodnoty nastavíme čísla do buněk, které v cyklech ovládají dobu čekání. Podle stavu pacičky procesoru lze přímo provést skok, např.

BTFSS PORTA,2

BTFSC PORTC,5

a podobně

ad 4

Pokud nevíte, co je střída impulzů, tak zpátky na stromy a doučit. Časy pro čekání jsou výrazně delší, musíte tedy použít dva cykly v sobě ( nebo možná 3, nepočítal jsem to )

**Třetí cvičení**

1. Sestavte podprogram C50MIKROS , který zajistí čekání po dobu 50 mikrosekund. S pomocí tohoto podprogramu sestavte program, který bude na pinu RA2 generovat obdélníkový signál o kmitočtu 10kHz

Až sem za 4

1. Sestavte podprogram C1MILI, který bude čekat po dobu 1 milisekunda. S pomocí tohoto podprogramu sestavte program, který bude na pinu RA1 generovat obdélníkový signál o kmitočtu 250Hz, střída impulsů bude 1:3.

Až sem za 3

1. Sestavte dva podprogramy, C100MILI, který bude čekat po dobu 100 milisekund, a CEK1S, který bude čekat po dobu 1 sec. S pomocí podprogramu CEK1S sestavte program, který bude blikat s diodou na nějakém pinu s periodou 1 sekunda.

Až sem za 2

1. Sestavte podprogram WMIKROS, který počká po dobu tolika stovek mikrosekund, jako je obsah registru W v okamžiku jeho volání. S pomocí tohoto podprogramu sestavíme měnitelný generátor kmitočtu. Kmitočet se bude ovládat číslem na portu C, výstupní pin si zvolte libovolně.

Až sem za 1

ad 1

V podprogramu uděláme několik NOP a v cyklu je budeme opakovat tak, abychom dostali požadované zpoždění. Jednoduché ! Generování obdélníkového průběhu se pak děje stejně jako u druhého cvičení, tedy nastavit pin, do 1, počkat, nastavit pin do 0, počkat, a dokolečka zacyklit

ad 2

V C1MILI  podprogramu prostě v cyklu několikrát zavoláme C50mikros, a je to.

ad 3

dále pokračujeme úplně stejně - několikrát zopakujeme v cyklu C1MILI, a tím dostaneme C100MILI, a tak dále až k té sekundě.

ad 4

Obsah registru W si na začátku podprogramu uložíme do buňky, která slouží jako počítátko průchodů cyklem, v cyklu pak vhodnou dobu počkáme. V hlavním programu pak čteme port C, uložíme do W, voláme WMIKROS , samozřejmě nezapomeneme nastavovat výstupní pin do 0 a do 1, tím vyrábíme obdélník.

**Čtvrté cvičení – přerušení**

1. Pomocí čítače TMR0 a přerušení sestavte program, který bude na nějakém portu procesoru vyrábět obdélníkový kmitočet 1kHz.

až sem za 4

1. Program podle bodu 1 doplňte kontaktem proti zemi a vytvořte tak telegraf – kontakt sepnut – tón zní, kontakt rozepnut – je ticho.

až sem za 3

1. Pomocí čítače TMR0 a přerušení sestavte program, který bude na nějakém pinu procesoru blikat s diodou s periodou 1 sec ( 0,5 sec svítí, 0.5 sec zhasnutá)

až sem za 2

1. Příklad podle bodu 3 doplňte třemi vstupními bity. Číslo na těchto bitech znamená délku periody blikání diody, 0 – zhasnuto.

až sem za 1

ad 1

Doporučuji vaší pozornostti file prerus03.asm, tam je uvedeno, jak se zkrátí cyklus čítače TMT0. Možná budete potřebovat i dělišku, to už nechám vaší laskavé úvaze. Nezapomeňte nastavit registr OSCCON. Dále může dělat trochu problém střídání 0 a 1 na pinech procesoru – zde si prostudujte instrukci COMF . port procesoru je registr. Nevadí, když se bude měnit celý. Kmitočet si zaveďte do reproduktorku na pultíku, nezapomeňte na vhodný rezistor. Reproduktorek nejdříve zkuste, ne všechny fungují.

ad 2

Příklad 1 doplníme nějakým vstupním pinem s PULL-UP rezistorem. Pomocí tohoto pinu budeme buď zapínat a vypínat čítač TMR0, nebo – to je možná jednodušší – povolovat a zakazovat přerušení od čítače.

Ad 3

Zde si prostudujte file prerus04, tam vidíte, jak se dá prodloužit doba , kterou generuje čítač.

ad 4

Příklad 3 doplníme třemi vstupními bity. Podle stavu na tšchto bitech měníme pořítátko v přerušení u příkladu 3. Nebo – zase možná lepší varianta – si zavedeme další počítátko, které dále vznásobí periodu 1 sec z příkladu 3, a toto počítátko budeme nastavovat pomocí těchto tří bitů.

**Páté cvičení – přerušení a překlad tabulkou**

1. Pomocí čítače TMR2 a přerušení sestavte program, který bude na nějakém portu procesoru vyrábět obdélníkový kmitočet 1kHz.

až sem za 4

1. Program podle bodu 1 upravte tak, aby vysílal hasičskou kvartu – „ho-ří“. Použijte čítače TMR2/4/6

až sem za 3

1. Sestavte program, který bude sloužit jako dekodér sedmisegmentového displeje-jeden znak. Na čtyřech vstupních bitech budete zadávat číslo, a to se objeví na znakovce. Použijte „překlad tabulkou“.

až sem za 2

1. Pomocí příkladu 3 zkonstruujte jednoduchý čítač, který bude inkrementovat s periodou 1 sec. a bude čítat od 0 do 9, pořád dokola. Číslo bude zobrazeno pomocí sedmisegmentové znakovky.

až sem za 1

ad 1

To je hračka. Naprogramujeme čítač, nezapomeneme povolit přerušení ve všech potřebných registrech, v přerušení budeme invertovat nějaký port.

ad 2

Příklad 1 doplníme dalším čítačem. Tímto druhým čítačem budeme každou sekundu měnit parametry prvního čítače tak, aby měnil kmitočet generovaného tónu. Jeden z tónů kvarty nechť je komorní a, ten druhý tón si dopočítejte.

ad 3

To je vzorový program <http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/progr_a_vysv/preklad_tabulkou/> , doplněný o čtení z portu a výstup na jiný port. Nezapomeňte na maskování přečteného čísla.

ad 4

To je příklad 3, doplněný nějakým čítačem, který bude generovat čas 1 sec. V přerušení pak budeme inkrementovat vhodnou buňku, číslo, které v ní bude, pak překódujeme na ovládací signály pro LED znakovku.

**Šesté cvičení – nepřímé adresování**

1. Sestavte program, který na připojené LED znakovce postupně odvysílá alespoň dvacet cifer Ludolfova čísla. Číslo uložte do EEPROM a čtěte ho pomocí nepřímého adresování, číslo vysílejte stále dokolečka.

až sem za 3

1. Sestavte program, který zahraje na zvoleném pinu píseň „Ovčáci, čtveráci“ nebo jinou srovnatelně obtížnou. Data opět uložte do EEPROM a čtěte pomocí nepřímého adresování.

až sem za 1

ad 1

Naprogramujeme si čítač, který bude volat přerušení řekněme dvakrát za sekundu. V přerušení pak vždy čteme pomocí nepřímého adresování z paměti EEPROM data, která znamenají jednotlivé cifry Ludolfova čísla. Implementujte i desetinnou tečku a tmu – zhasnuté všechny segmenty. V přerušení samozřejmě také inkrementujeme registr FSR, když dojdeme na konec čísla, nastavíme ho opět do počátečního stavu. Nepříjemné pro toto použití je „automatic conext saving“., tedy pokud změníte hodnotu FSR v přerušení, nepřenese se tato změna do hlavního programu. To odstraníme tak, že obsah FSR v přerušení prostě uložíme do „shadow registers“ v bance 31.

ad 2

Naprogramujeme si dva čítače. Jeden čítač bude odměřovat délku not, druhý bude vyrábět kmitočet pro jednotlivé tóny (tento bude TMR2/4/6). Čísla, podle kterých se bude nastavovat výška tónu, tedy kmitočet tónu, si uložíme do EEPROM. Pomocí nepřímého adresování je pak postupně čteme s periodou danou prvním čítačem, a podle nich nastavujeme druhý čítač tak, aby vyráběl potřebný kmitočet. Notu dvojnásobné délky vytvoříme tak, že dáme za sebe dvě noty základní délky. Pro pomlku si zvolíme nějaké vhodné číslo (třeba 0), když toto číslo přečteme, vypneme druhý čítač. Bylo by vhodné zároveň nastavit vysílací piník tak, aby při pomlce do reproduktoru nedodával žádný proud.

**Sedmé cvičení – modul CCP a PWM**

1. Sestavte program, který bude na jednom pinu generovat PWM o kmitočtu 2kHz a střídě impulzů 1:4 a současně na druhém výstupu PWM o kmitočtu 7kHz a střídě impulzů 1:8

až sem za 4

1. Sestavte program, který bude na vhodném pinu generovat PWM o kmitočtu 10kHz. Program má dále čtyři vstupní piny. Pomocí čísla, přivedeného na tyto piny, je možno nastavit „Duty cycle“ od 0 do 100%.
2. k procesoru připojte LED diodu přes vhodný odpor. Posuďte, jak se mění jas dipody pro různý “Duty cycle” a do sešitu si poznamenejte jeho hodnoty pro maximální jas, ¾ jas, poloviční jas, ¼ jas.
3. k procesoru připojte dolní propust, která odstraní vyšší harmonické a propustí první harmonickou. Do sešitu si napište, co se děje s amplitudou prví harmonické, když měníme “Duty cycle”, a co se děje s jejím kmitočtem. Vše samozřejmě předvedete vyučujícímu!
4. k procesoru připojte dolní propust, která spolehlivě odstraní i první harmonickou a ponechá jen stejnosměrnou složku. Do sešitu si napište, co se děje na výstupu takovéhoto obvodu při změnách “Duty cycle”. Předveďte vyučujícímu.

až sem za 3

1. Na základě znalostí z bodu 2 sestavte program, který bude postupně rozsvěcovat LED od úplné tmy až do maximálního jasu, pokud možno lineárně (optický dojem bude lineární), pořád dokolečka. Periodu rozsvěcení udělejte asi 2 sec.

až sem za 2

1. Na základě znalostí z bodu 2 sestavte jednoduchý převodník, na jehož výstupu bude pilovitý signál s amplitudou 5V a kmitočtem asi 100 Hz

až sem za 1

ad 1

Trivialita. Naprogramujeme dva CCP moduly , pro PWM si vybereme dva různé čítače TMR2/4/6. Každý modul má svůj vlastní výstup, není co řešit.

ad 2

Budeme periodicky číst vstupní port. Když zjistíme změnu na portu, tak podle přečtené hodnoty změníme obsah registru, který ovládá „Duty cycle“. Změnu na portu zjistíme tak, že jeho minulou hodnotu si uložíme do nějaké buňky. Pak porovnáme minulou hodnotu s hodnotou „teď“. Pokud se hodnoty liší, změníme „Duty cycle“ u PWM. Porovnání minulé a současné hodnoty uděláme tak, že obě hodnoty odečteme. Pokud vyjde ..., byly stejné. Pokud zjistíme změnu, pak samozřejmě uložíme hodnotu „teď“ do buňky „minulá hodnota“, abychom měli s čím porovnávat. Tuto strukturu si určitě naprogramujte, bez ní nebude tento bod uznán. A bude se vám hodit v budoucnu. Nezapomeňte na maskování, čtete vždy 8 bitů.

ad 3

nezapomeňte na vhodný odpor. LED je vaše, doopravdická, NE ta na pultíku.

ad 4

Z bodu 2c vidíme, co se děje s napětím, když měníme střídu signálu u PWM. Takže budeme střídu postupně měnit tak, abychom dostali vzrůstající napětí od 0 do 5V. !!!!! Pozor !!! kmitočty musíte zvolit vhodně, jinak to nepůjde. Také musíte vhodně zvolit parametry dolní propusti , musí být nastavena tak, aby **nevyhladila**  kmitočet pily 100Hz, ale **vyhladila** kmitočet PWM. Úvaha je úplně stejná jako u demodulátoru AM – volba kondenzátoru, tak si to zopakujte.

**Osmé cvičení – modul PPS a PWM**

1. sestavte jednoduchý převodník, na jehož výstupu bude pilovitý signál s amplitudou 5V a kmitočtem asi 100 Hz. Použijte k tomu modul PWM (samostatně si nastudujte), výstupní pin procesoru bude pin 15.

až sem za 3-

1. Sestavte obvod, který bude pomocí PWM zhášet a rozsvěcet velkou žárovku pomocí spínacího tranzistoru. Žárovka se po dobu 2 sec pomalu rozsvítí asi na ½ maxima, pak zůstane na této úrovni 3 sec svítit, pak se za dobu 4 sec pomalu rozsvítí až na maximum, tam zůstane 2 sec svítit, pak za dobu 2 sec zhasne, zůstane 4 sec zhasnuta, a poté se cyklus opakuje. Pro vytvoření PWM použijte modul PWM4 a čítač TMR6. Hodnoty pro „duty cycle“ uložte do paměti EEPROM a čtěte je pomocí nepřímého adresování ( to je nutnost pro uznání tohoto příkladu )

až sem za 2-

1. sestavte jednoduchý převodník, na jehož výstupu bude sinusový signál s amplitudou asi 2.5V a kmitočtem asi 200 Hz. Použijte k tomu modul PWM (samostatně si nastudujte), výstupní pin procesoru bude pin 14.

až sem za 1

ad 1

Modul PWM si nastudujete v <http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/progr_a_vysv/PWM/> . Jeho ovládání je téměř shodné s ovládáním PWM u modulu CCP, takže není co řešit. Další postup je shodný s sedmé cvičení – bod 4. Samozřejmě pomocí PPS nastavíte požadovaný pin.

ad 2

Celý cyklus rozsvěcení a zhášení si rozdělíme alespoň na 200, lépe 400 nebo 800 vzorků. Do paměti EEPROM si uložíme postupně hodnoty „Duty cycle“. Dalším čítačem budeme generovat časy mezi jednotlivými vzorky, budeme číst vzorky z paměti EEPROM a ukládat do registru, který ovládá „Duty cycle“ u PWM. No a PWM sama zajistí rozsvěcení a zhášení žárovky. Na pacičku procesoru připojíme tranzistor, kterým budeme spínat žárovku. Žárovku si přineste tak velkou, aby ji tranzistor byl schopen sepnout a neshořel, napětí tak 12V nebo 24V, větší zdroje nemáme a nebylo by to bezpečné. !!!!!! Není to lineární, takže poloviční svítivosti NEODPOVÍDÁ 50% Duty cycle. Zkuste si to, postačí optický dojem, jak moc žárovka svítí.

ad 3

V bodě 1 jsme se naučili postavit převodník. V tomto bodě jsme postupně zvyšovali hodnotu „Duty cycle“ a tím jsme vykreslili krásnou pilu. Nyní je třeba obdobným způsobem vykreslit sinusovku. Spočítáme si tedy hodnoty „Duty cycle“ alespoň pro 100 vzorků sinusovky (periodu sinusovky si rozdělíme na 100 dílků). Hodnoty vzorků si uložíme do paměti EEPROM a budeme je odtud číst pomocí nepřímého adresování – FLASH MEMORY. Použití požadovaného pinu je nutností, cvičíme PPS.

**Deváté cvičení – UART**

Pro celé toto cvičení si nastavíte piny TX a RX UARTu na takové pacičky procesoru, abyste nemuseli po jeho naprogramování měnit drátky, které vedou do PIKITu. K monitorování chování obvodu použijte UART TOOL.

1. Sestavte program, který bude na výstupní pin UARTu postupně vysílat malou abecedu a,b,c,.....,z s frekvencí 2 písmena za sekundu.

až sem za 4

1. Sestavte program, který bude na výstupní pin dokolečka vysílat nějaký delší nápis – například „Chci se naucit programovat mikroprocesory a udelam pro to vsechno“. Po odvysílání nápisu program 2 sec počká, a potom zase začne vysílat.

až sem za 3

1. Sestavte program, který bude sloužit jako smyčka pro UART – tedy to, co vyšleme po UARTu z PICKITu nám procesor hned pošle nazpět.

až sem za 2-

1. Sestavte program, který bude pomocí UART ovládat svícení LED diody. Ovládání udělejte např. pomocí číslic 0 – 9, ovládání jasu LED pomocí PWM.

až sem za 1

ad 1

Naprogramujeme si nějaký čítač, který bude generovat čas 0.5 sec. V přerušení od čítače vždy zapíšeme do TXREG číslo, které chceme vyslat. Čísla se postupně zvyšují, až dojdeme k písmenku z, opět nastavíme kód písmenka a. ASCII tabulku předpokládám znáte.

ad 2

Do paměti EEPROM si uložíme nápis. V přerušení od vysílacího registru UARTu vždy přečteme znak z paměti a zapíšeme do TXREG (přerušení znamená, že znak je odvysílán). Použijte nepřímé adresování, mód FLASH MEMORY. Nezapomeňte na „automatic context saving “ při použití FSR. Konec nápisu si označte nějakým kódem, např. 0 (tak ostatně končí řetězec v jazyce C). Jakmile dojdete na tuto hodnotu, prostě zakážete přerušení od UARTu a tím zlikvidujete další vysílání. (no, možná bude stačit prostě nevyslat další písmeno – nezapsat do TXREG). Samozřejmě si také nastavíte FSR na začátek řetězce. Odstartování vysílání dalšího nápisu provedeme pomocí nějakého čítače, který jednou za 2 sec povolí přerušení od UARTu , zapíše první vysílaný znak do TXREG a tím opět zahájí vysílání. Doporučuji vaší ctěné pozornosti soubor <http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/progr_a_vysv/zaklady/druhy_03.asm>

ad 3

Trivialita. Uděláme si přerušení od přijímací části UARTu (TXIF) , v přerušení přečteme RXREG a hned ho zapíšeme do TXREG.

ad 4

Podle přišedšího čísla prostě nastavíme „Duty cycle“ pro PWM a je to. Nezapomeňte, že když přijmete písmenko „0“ , přijali jste ve skutečnosti číslo ....... . Přijatá čísla na velikost „Duty cycle“ převedeme pomocí překladu tabulkou. Přijatá „0“ nebo „1“ nebo „9“ je fakt opravdu písmeno, v programátorské terminologii „character“ . Jo, pokud nevíte, co to je, tak to máte dost blbý.

měřič kmitočtu – vzsílat pomocí RS232

dvojitý měřič kmitočtu – a pomocí něho si hrát s PLL – zobrazit reakce na změnu – na jednotkový skok

reakce pro různé hodnotz CR ve smzčkách