Středoškolská technika 2009

Setkání a prezentace prací

středoškolských studentů na ČVUT

Miksísek – simulátor sériové komunikace

Jiří Blecha

SŘEDNÍ PRŮMYSLOVÁ ŠKOLA SDĚLOVACÍ TECHNIKY

Praha1, Panská 3

Anotace

Miksísek – simulátor sériové komunikace

Miksísek – simulátor sériové komunikace je zařízení, které slouží k výuce sériové komunikace. Je sestaveno z jednoho hardwarového modulu a softwaru pro PC.

Hardwarový modul je osazen jednočipovým mikroprocesorem PIC16F688, který je srdcem celého zařízení. Zařízení je možno přepnout do 5-ti různých pracovních módů. Módy byly zvoleny s ohledem na různorodost používaných periférií tak, aby usnadnily studentům pochopení sériové komunikace a eventuelně i používání a práci s periferiemi mikroprocesoru. Firmware pro mikroprocesor je napsán v programovacím jazyce C. Tento jazyk byl zvolen pro lepší porozumění zdrojového kódu, než by bylo dosaženo u jazyku Assembler.

Software pro PC je napsán v programovacím jazyce Visual Basic. Tento jazyk byl zvolen pro svojí jednoduchost, ačkoli se následně právě jednoduchost stala téměř nepřekonatelnou překážkou. Spousta předepsaných funkcí, které už stačí pouze použít je naprosto nepoužitelná, díky nemožnosti nastavení některých důležitých parametrů. I přes všechny překážky se program povedlo napsat a upravit do docela svůdné formy, která vyhovuje jak pro přehlednost, tak i pro použitelnost a porozumění.

Annotation

Miksísek – simulator of serial communication

Miksísek – simulator of serial communication is a device which serves to teaching of serial communication. It's made from one hardware module and software for PC.

Hardware modul is mounted by single-chip microcontroler PIC16F688 which is heart of the device. The device can be switched to one of 5 different working modes. Modes were chosen with regard to difference of used peripheries to simplify understanding of serial communication and possibly to simplify using and working with the pheripheries of the microcontroler. Firmware for microcontroler is written in programming language C. This language was chosen for better understanding of the source code than it could be achieved with programming language Assembler.

Software for PC is written in programming language Visual Basic. This language was chosen for its simplicity. In future the simplicity become almost impassable interruption. Many of specified functions are totaly unusable because of impossibility to set up some important arguments. Over all hurdles the program were written and edited to quiete good form which is OK for lucidity, usability and understanding.

Obsah

1. ÚVOD

Cílem této DMP bylo zrealizovat zařízení, které by usnadnilo studentům naučení se sériové komunikace, s pomocí jednoho hardwarového modulu a softwaru pro PC. Hardwarový modul měl obsahovat nějaký mikroprocesor se sériovým modulem. Požadavky byly také kladeny na velikost modulu, který se měl vejít do malé krabičky. Software pro PC neměl mít žádné speciální parametry. Funkce zařízení měla být vcelku jednoduchá. Pouze jsem musel vybrat několik názorných věcí, na kterých by bylo dobře vidět posílaná data. Realizace celé práce je uvedena níže.

2. POPIS ZAŘÍZENÍ A VÝVOJ

V této kapitole se zaměřím na průběh vývoje zařízení, což obsahuje návrh schéma, návrh dps, popis RS232 komunikace atd.

2.1 NÁVRH SCHÉMATU

Po rozmyšlení co by zhruba zařízení mělo dělat, přišlo na řadu navrhnout schéma. Pro návrh byl vybrán software Eagle od firmy Cadsoft. Obvyklý postup kreslení schématu je takový, že si nejdříve vybereme všechny součástky v odpovídajících velikostech a pouzdrech, které požadujeme a přidáme si je do návrhu. Následně přijde na řadu zamyšlení nad rozmístěním. Součástky se obvykle rozmísťují tak, aby bylo schéma přehledné, což znamená hlavní obvody, které mají mnoho vývodů se umisťují doprostřed, vstupní obvody se umisťují vlevo a výstupní obvody vpravo. Tyto všechny okolnosti jsou pouze orientační a pokud je nutnost to mezi sebou nějakým způsobem proházet, rozhodně to nevadí. Tyto situace obvykle nastávají, pokud máme obsáhlejší schéma nebo pokud chceme spořit místem. Pro přehlednost se může využít tzv. sběrnic (bus), kterými můžeme schéma krásně rozdělit do bloků. Je to o mnoho přehlednější, než kdybychom vše spojovali jednotlivými čarami. Napájecí vývody se obvykle nespojují, používá se napojení na schematické značky (Vdd,Vss,Gnd, ...). Viz schéma, které je uvedeno v přílohách. Návrh schématu byl vcelku jednoduchý, díky předchozím zkušenostem. Popis jednotlivých bloků schématu je uveden v další podkapitole.

2.2 POPIS SCHÉMATU

V této chvíli přichází na řadu rozebrání jednotlivých bloků ve schématu a jejich stručný popis funkce. Doporučuji si vzít schéma k ruce. Schéma bude popisováno zleva doprava. Úplně na kraji si můžeme všimnout 2 konektorů. Konektor CANN je 9-ti pinový konektor, sloužící k připojení zařízení k počítači. Konektor ICSP je 5-ti pinový konektor sloužící k možnosti naprogramování hlavního obvodu PIC16F688. Když se posuneme dále, můžeme si všimnout 2 diod, které přivádí kladné napětí do obvodu 78L05, který ho stabilizuje na 5V a umožňuje tak stabilní napájení dalším obvodům. Pro signalizaci zapnutého zařízení zde slouží červená LED dioda, před kterou musí být předřazen odpor, který slouží jako převodník napěťového zdroje na proudový. Po dalším posunu, vidíme obvod

MAX232CWE, potřebný pro převod RS232 logických stavů na TTL 5V. Toto je nutné pro umožnění obvodu PIC16F688 komunikovat vzájemně s PC. U obvodu jsou 4 kondenzátory, které jsou připojeny k nábojovým pumpám uvnitř obvodu. Zapojení je převzané z datasheetu. Další součástkou na řadě je již zmíněný mikroprocesor PIC16F688. Pro správnou funkci obvodu je nutno mít dobré napájení a stabilní oscilátor. Oscilátor je vyřešen pomocí krystalu 3,6864MHz. Možná se ptáte proč takováto frekvence ? Tato rychlost je ideální pro vydělení různých rychlostí USART modulu v mikroprocesoru. O tomto si více povíme v dalších kapitolách. Nad krystalem je tlačítko, sloužící k restartu zařízení do základního nastavení. Vpravo od mikroprocesoru je dvoubarevná LED dioda se společnou katodou. Opět jsou zde odpory, k převedení na proudový zdroj. Dioda je připojena k mikroprocesoru a slouží k ukázce jednoho z operačních módů zařízení, konkrétně PWM. Jako poslední je na schématu fototranzistor. Tato součástka není tak obvyklá, tak si k ní něco řekneme. V podstatě funguje jako klasický bipolární tranzistor, pouze má mimo pevně připojené báze prostor pro dopad světla. Fototranzistor je tedy ovládán světlem. Zvolený tranzistor reaguje spíše na IR spektrum světla, než na klasické viditelné. Při zkouškách se celkem osvědčila klasická žárovka. V tomto zapojení funguje tak, že při osvětlení se otevírá a začíná protékat proud, který tvoří odpovídající úbytek napětí na rezistoru. Toto napětí je následně snímané ADC převodníkem v mikrokontroléru. Obsluhuje operační mód ADC.

2.3 NÁVRH DPS

Po úspěšném nakreslení schématu a několikanásobném zkontrolování se můžeme vrhnout na návrh DPS (desky plošného spoje). Opět je použit software Eagle od firmy Cadsoft. V prvních okamžicích navrhování je zmatenost normální pocit. Na monitoru máme spoustu součástek spojených tenkými čarami a je v tom doslova „zmatek“. Nejdříve se musíme zaměřit na základní rozmístění součástek. Je nutno volit na jakém místě se součástka vyskytuje, i na jaké straně DPS bude. Ideální je si otevřít schéma a koukat co s čím je spojené, to dosti ulehčí následné routování. Nečekejte, že se to povede hned na poprvé. Je potřeba mít pevné nervy a hodně síly. Pro rozmísťování je vhodné použít autorouter a občas nechat součástky automaticky proroutovat, abyste zjistili, jak by to mohlo být lepší. Nedoporučuji používat autorouter na routování. Je to sice jednoduché řešení, ale přece jenom je to program a v Eaglu ještě ne moc inteligentní. Má obvykle ve zvyku dělat na DPS krásné antény, které následně slouží jako přijímač pro různé rušení a to může způsobit ne jeden problém ve funkčnosti zařízení. Nejlepší je si routy udělat sám. Můj návrh proběhl bez problémů. Je samozřejmé, že prošel několika úpravami, jak estetickými, tak i vzhle-

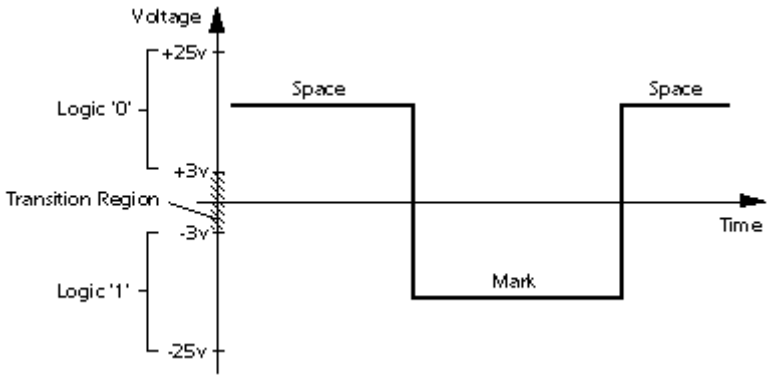
dem ke složitosti výroby. DPS je oboustranné, protože vzhledem k požadavkům, aby se to vešlo do hodně malé krabičky, nebylo jiné východisko. Viz DPS, které je uvedeno v přílohách.

2.4 RS232 KOMUNIKACE

Pro spojení modulu s PC byla zvolena sériová komunikace RS232. Pokud bychom jí chtěli přirovnat k RM OSI, tak zjistíme, že je nadefinována pouze pro fyzickou vrstvu. Veškeré další protokoly musí být napsány výhradně programově. RS232 komunikace je sériová komunikace s použitím 2 logických stavů – 1 a 0. Log. 1 je někdy označována jako marking state nebo také klidový stav, Log. 0 se přezdívá space state. Log. 1 je indikována zápornou úrovní, zatímco Log. 0 je přenášena kladnou úrovní výstupních vodičů. Povolené napěťové úrovně jsou uvedeny v tabulce. Nejběžněji se pro generování napětí používá napěťový zdvojovač z 5 V a invertor. Logické úrovně jsou potom přenášeny napětím +10 V pro log. 0 a –10 V pro log. 1.

Datové signály		
Úroveň	Vysílač	Přijímač
Log. 0	+5 V to +15 V	+3 V to +25 V
Log. 1	-5 V to -15 V	-3 V to -25 V
Nedefinovaný	-3 V to +3 V	-3 V to +3 V

Řídící signály		
Signál	Driver	Terminátor
"Off"	-5 V to -15 V	-3 V to -25 V
"On"	5 V to 15 V	3 V to 25 V



Standard RS 232 uvádí jako maximální možnou délku vodičů 15 metrů, nebo délku vodiče o kapacitě 2500 pF. To znamená, že při použití kvalitních vodičů lze dodržet standard a při zachování jmenovité kapacity prodloužit vzdálenost až na cca 50 metrů. Kabel lze také prodlužovat při snížení přenosové rychlosti, protože potom bude přenos odolnější vůči velké kapacitě vedení. Uvedené parametry počítají s přenosovou rychlostí 19200 Bd. Texas Instruments uvádí jako výsledek pokusných měření následující délky vodičů v závislosti na přenosové rychlosti. Vzhledem k „laboratorním“ podmínkám tohoto měření je třeba brát tyto údaje pouze jako orientační. V praxi je třeba počítat s rušením atd.

Baud rate [Bd]	**Max length [ft]**	**Max length [m]**
19200	50	15
9600	500	150
4800	1000	300
2400	3000	900

RS232 Používá asynchronní přenos informací. Každý přenesený byte konstantní rychlostí je proto třeba synchronizovat. K synchronizaci se používá sestupná hrana tzv. Start bitu. Za ní již následují posílaná data. Po ukončení posílání dat je odeslán tzv. Stop bit, který zajistí oddělení prvního znaku od druhého. Další znak je opět uvozen Start bitem.

2.5 RS232 – TTL

Používáte-li v zařízení TTL nebo CMOS obvody, budete muset jejich logickou RS232 linku napěťově upravit před připojením do PC, protože napěťové úrovně RS-232 nejsou přímo slučitelné z žádnou logikou. Pro toto upravení se standardně používaly obvody 1488 a 1489, které ale potřebovaly + 12 V a –12 V pro vytvoření výstupních úrovní. To bylo mimochodem jedním z důvodů, proč je v klasickém PC ze zdroje vyvedeno i –12 V a –5 V (dalším důvodem byla potřeba většího rozdílového napětí u historických dynamických pamětí pro zvýšení jejich rychlosti). Průlom v tomto směru udělala firma MAXIM svým obvodem MAX232. Využila totiž svých znalostí ve vývoji spínaných nábojových měničů napětí a vyvinula obvod, který vystačil s +5 V a potřebné napětí si samostatně vyrobil pomocí 4 externích kondenzátorů. MAX232 se stal neuvěřitelným šlágrem a dnes jeho obdobu najdete téměř ve všech komerčních zařízeních připojovaných k RS 232.

2.6 MAX232

Jedná se o převodník TTL na RS232. Obsahuje dvě dvojice oddělovačů konvertujících napěťové úrovně. Napětí pro RS 232 se získává pomocí nábojové pumpy, a výstupní napětí proto značně závisí na kvalitě použitých kondenzátorů, která u elektrolytických kondenzátorů časem značně klesá. Napětí je možno získat na pinech 2 a 6 a použít pro další obvody. Obvod funguje vždy na první zapojení. Maxim vyrábí i verze s minimální externí kapacitou – (MAX 232A – 0,1 μF) nebo verze pracující v rozsahu 7,5 – 13 V (určeno pro bateriové aplikace) – MAX 201 a MAX 231. Specialitou firmy MAXIM jsou obvody MAX203 a MAX 233, které dokáží pracovat úplně bez potřeby vnějších kondenzátorů. Pro více informací viz. datasheet.

2.7 PIC16F688

PIC16F688 je jednočipový 8-mi bitový mikroprocesor od firmy Microchip. Jak již jeho označení ukazuje je zařazen do řady 16F. Firma Microchip vyrábí mnoho různých typů. Můžeme se setkat s řadami 10F,12F,16F,18F atd. Každá z těchto řad obsahuje odlišné typy mikroprocesorů, ty se mohou lišit např. výrobní technologií, periferiemi, architek-

turou, či možností uplatnění. Díky předchozím zkušenostem jsem chtěl využít nabídky řady 18F, ale bohužel jsem nenašel žádný vhodný mikroprocesor. Požadavky byly – co nejmenší, 5V napájení, musí obsahovat USART, A/D převodník a nějaký timer. Nakonec byl vybrán obvod PIC16F688. Jak už bylo zmíněno, obvod je z řady 16F. Tato řada vyznačuje celkem velikou nekompatibilitou mezi jednotlivými mikroprocesory, což znamená, že program napsaný pro jeden typ, nefunguje na druhém. Vybraný mikroprocesor je poměrně nový. Byl vyroben v roce 2007. Je postaven na 8-mi bitové RISC architektuře. Celkem obsahuje 35 instrukcí. Maximální rychlost krystalu nebo přiváděné frekvence je 20MHz, což znamená, že maximální provozní rychlost obvodu je 5MIPS. Obvod má 14-vývodů a dělá se v několika různých pouzdrech. Pro vývoj se obvykle používá pouzdro PDIP a na již hotových zařízeních se používají smd pouzdra SOIC, TSSOP, či QFN. Ne všechna smd pouzdra je možné použít v domácích (školních) podmínkách. Rozumně se nechá použít pouzdro SOIC. Podrobnému popsání všech periferií není účelem této dokumentace. Spíše se zaměřím na hrubý popis používaných periferií v zařízení. Tyto periferie jsou TIMER0,TIMER1,ADC,USART,EEPROM,PORTA a PORTC.

TIMER0 je 8-mi bitový timer. Pro možnost běhu na různých rychlostech mu byla přidělena 8-mi bitová předdělička a vstup externích hodin. Tyto věci se nastavují programově. Timer může generovat přerušení při přetečení, takže se krásně nechá použít na dělání časových intervalů nebo i pro jiné funkce. V mém programu slouží ku příkladu ke měnění střídy signálu, čili PWM.

TIMER1 je 16-ti bitový timer. Opět je k němu připojena předdělička, tentokrát ovšem pouze 3-bitová. Pro zvolení různých frekvencí máme opět možnost přivést externí signál nebo zapojit LP oscilátor přímo k timeru. Timer může operovat synchronně nebo asynchronně. Opět je možnost generování přerušení při přetečení. Vše je nastavitelné programově. Využití toho timeru je podobné jako u TIMER0, ovšem je postavený tak, že dokáže přesněji generovat časové intervaly, takže se obvykle používá na nějaký odpočet času. V mém programu je použit na generování časových intervalů 0,1s.

ADC je 10-ti bitový A/D převodník. Jeho funkce je obdobná, jako u jiných A/D převodníků. Výhoda implementace v zařízení je, že po vzorkování, kvantování a kódování nám vyhodí číslo, s kterým můžeme hned programově pracovat, nemusíme složitěji přijímat data od externího ADC. Referenční napětí pro A/D převod můžeme zvolit buď jako interní napájecí napětí a nebo můžeme přivést nějaké jiné na vývody obvodu. Externí referenční napětí musí být menší, než je napájecí napětí obvodu! Je umožněno programově

nastavit vzorkovací frekvenci. Podle Nyquistova pravidla víme, že vzorkovací frekvence signálu musí být minimálně 2x větší než nejvyšší frekvence obsažená ve vzorkovaném signálu. Toto pravidlo je nutné používat pro dobrou reprodukci signálu, pokud pravidlo nedodržíme, začne docházet k aliasingu. V mém programu je ADC použit na zjištění úbytku napětí na rezistoru, který se mění v závislosti na osvícení fototranzistoru.

USART je modul v procesoru, který slouží k sériové komunikaci s ostatními zařízeními. Funguje podle standartu RS232, který je popsán výše. Modul je možno nastavit k synchronní, či asynchronní komunikaci. Obvykle se spíše používá asynchronní. Jak by se dalo očekávat, že bude modul obsahovat paritní ochranu posílaných a přijímaných dat, tak tomu tak není. Pokud chceme používat paritu, musíme si jí programově obsloužit. K tomuto je zde aspoň přidán 9-tý bit. V mém programu je USART nastaven na asynchronní komunikaci s měnitelnou rychlostí. Parita byla dopsána programově a funguje vcelku obstojně.

EEPROM je modul obsahující elektronicky mazatelnou statickou paměť, která hlavně slouží pro uložení nějakých dat,které nechceme ztratit ani po vypnutí napájení. Paměť obsahuje 256 8-mi bitových buněk. V mém programu je použita pro uložení nastavení USART.

PORTA a PORTC jsou dva datové porty, které obsahuje procesor. Každý pin může být nastaven buď jako vstupní nebo jako výstupní. Vstupní piny se obvykle používají pro připojení tlačítek nebo nějakých čidel s TTL logikou na výstupu. Výstupní piny mohou mít mnoho aplikací. Např. s nimi můžeme signalizovat různé stavy pomocí LED diod, s použitím relé, či tranzistoru můžeme spínat různá silnoproudá zařízení atd. Pro důkladnější informace o procesoru viz. datasheet.

3. POPIS FIRMWARE

Firmware je program napsaný pro hardwarový modul, konkrétně pro PIC16F688. Pro přehlednost je napsaný v programovacím jazyce C a přeložen pomocí překladače HI-TECH PICC. Zdrojový kód je uveden v přílohách. Program stručně popíšu.

Celý program se nechá rozdělit na dvě nejdůležitější funkce. První funkce se jmenuje MCUInit(). Tato funkce nastaví mikroprocesor do použitelné konfigurace. Nastaví požadované periférie a celkově umožní správnou funkčnost. Druhá funkce je interrupt isr(). Tato funkce obsluhuje přerušení. V této funkci je skryta celková funkce zařízení. Nejdůležitější periférií, kterou musíme obsloužit je USART.

```
if(RCIF && RCIE)    //prijem znaku po USART
{
        RX9Dtemp = RX9D;
        RxTemp = RCREG;
        ...
}
```

Obsluha této periférie začíná hned na začátku funkce. Je zde obsaženo dekódování parity (pokud je zapnuta) a příjem jednotlivých znaků, které jsou následně vyhodnoceny a zpracovány. Další periférie obsluhovaná ve funkci interrupt isr() je TMR0.

```
if(T0IF && T0IE)     //PWM
{
        T0IF = 0;
        ...
}
```

TMR0 zde v podstatě funguje jako softwarové PWM. Pro bližší studium viz. zdrojový kód. TMR1 je zde také obsluhován.

```
if(TMR1IF && TMR1IE)      //preruseni po 0,1s + tlacitko
{
        TMR1L = 0;
        TMR1H = 76;
        TMR1IF = 0;
        ...
}
```

TMR1 je nastaven na generování časových intervalů 0,1s. Tento časový interval je využit na odesílaní dat z ADC, na ochranu proti zákmitům u restartovacího tlačítka, či na přenastavení USART nebo na uložení nových nastavení do EEPROM. Další obsluhovanou periférií je ADC.

```
if(ADIF && ADIE)    //snimani svetla
{
        ADIF = 0;
        ...
}
```

ADC je využito na zjištění stavu osvětlení fototranzistoru. Aby byly výsledky přesnější, provádí se odečtení 64 hodnot a jejich následné zprůměrování a převod na procenta.

Dále můžeme v programu najít spoustu dalších funkcí. Tyto funkce jsou obvykle pomocné. Alespoň orientačně každou popíšu.

delay()	-	zpožďovací funkce, pouze smyčka, je použita v jiných funkcích
carka()	-	s pomocí funkce delay() je nastaven časový interval na čárku
tecka()	-	s pomocí funkce delay() je nastaven časový interval na tečku
MorseCode()	-	tato funkce vybliká pomocí funkcí tecka(), carka() dany znak
UsartConf()	-	tato funkce nastaví USART podle požadavku

PWMConf() - tato funkce nastaví střídu PWM

ParityCheck() - v této funkci se kontroluje správnost parity

SendString() - tato funkce pošle řetězec znaků po USART

SendChar() - tato funkce odešle znak po USART

EERead() - přečte data z určité adresy EEPROM

EEWrite() - zapíše data na určitou adresu EEPROM

4. POPIS SOFTWARE

Software je program pro PC. Je napsaný v programovacím jazyce Visual Basic. Tento jazyk je vcelku jednoduchý. Využívá objektového programování. Zdrojový kód programu je uveden v přílohách. Program stručně popíšu.

Celý program se krásně nechá rozdělit na funkce, které se spouští při nějaké události. Například funkce Miksisek_Load() se spouští, jakmile spustíme program a form Miksísek vytvoří událost Load. Popíšu funkci po funkci.

Miksisek_Load() - tato funkce se spouští při zapnutí programu, nalezne dostupné COM porty a přidá je do menu, dále kontroluje, jestli je COM port otevřený, pokud ano, zavře ho

btnConnect_Click() - tato funkce se spouští při kliknutí na tlačítko Connect (Disconnect), podle názvu tlačítka se provede daná akce, pokud je nápis Connect, tak se otevře COM port podle zvolených nastavení, pokud je nápis Disconnect, tak se COM port uzavře a vše se nastaví na defaultní nastavení

btnSettings_Click() - tato funkce se spouští při kliknutí na tlčítko Module settings (OK), podle názvu tlačítka se provede daná akce, pokud je nápis Module settings, tak se umožní změnit nějaká nastavení v modulu, pokud je nápis OK, tak se potvrdí nastavení

SerialPort_DataReceived() - tato funkce se spouští, když přijdou data po otevřeném COM portu, odkazuje na funkci ReadSerial(), která zpracuje přijatá data

ReadSerial() - tato funkce přijme data z COM portu a zpracuje je podle spuštěného módu

tbcMod_SelectedIndexChanged() - tato funkce se spouští, pokud je překliknuto na jinou záložku, podle zvolené záložky (modu) se odešlou nastavovací data po COM portu do modulu

btnEcho_Click() - tato funkce se spouští, pokud je zvolena záložka Echo a je kliknuto na tlačítko Send, odešle po COM portu text z textového pole

btnMorse_Click() - tato funkce se spouští, pokud je zvolena záložka Morse a je kliknuto na tlačítko Send, odešle po COM portu text z textového pole

btnVtip_Click() - tato funkce se spouští, pokud je zvolena záložka Vtip a je kliknuto na tlačítko Vtip, odešle po COM portu znak

Timer1_Tick() - tato funkce se spouští, pokud dojde k přetečení Timer1 a pokud je zvolena záložka PWM, po 0,1s odesílá data pro PWM

Timer2_Tick() - tato funkce se spouští, pokud dojde k přetečení Timer2, generuje zpoždění 0,05s pro zapnutí modulu a odešle data pro nastavení probíhajícího módu

5. NÁVOD K VÝROBĚ

Výroba modulu není zas tak složitá, ale vyžaduje pěknou dávku trpělivosti, nějakou tu zručnost a asi nejdůležitější jsou zkušenosti. Jedná se o výrobu oboustranného DPS a pájení smd součástek o velikosti 0805.

5.1 VÝROBA DPS

DPS je navrhnuto na výrobu fotocestou. Jedná se postup výroby, který je velice přesný a proveditelný i doma na koleni. Budeme potřebovat oboustrannou DPS s fotocitlivou vrstvou, či klasickou oboustrannou DPS + fotocitlivý Pozitiv ve spreji, Vývojku (1,5% roztok NaOH), leptací roztok ($FeCl_3$), průhlednou fólii na tisk laserovou tiskárnou, osvětlovací přístroj (stačí nějaká výbojka UV světla), špendlíky, líh, kalafunový lak (kalafuna rozpuštěná v lihu) a mikrovrtačku.

Nejdříve si vytiskneme předlohy DPS na průhlednou fólii. Jsou uvedeny v přílohách. Teď přichází na řadu osvětlování. Předlohy jsou opatřeny synchronizačními dírami, aby byly díry přesně nad sebou. Po úspěšném vytisknutí si vystřihneme obě předlohy. Pomocí synchronizačních značek si je spojíme špendlíky, aby díry byly krásně nad sebou. Pokud se nám vše povedlo, můžeme se vrhnout na vystřihnutí DPS. DPS musí být dostatečně velké, aby se na něj vešel celý návrh DPS, ale zase dostatečně malé, aby nevadil synchronizačním dírám (špendlíkům). Po úspěšném vystřihnutí nasadíme předlohu na DPS a z obou stran jí přilepíme. Vytáhneme špendlíky a necháme desku osvětlit. Pokud osvětlujeme v nějakém profesionálním přístroji, tak mohou být časy různé. Doporučuji si přečíst manuál. Pokud osvětlujeme nějakým zbastleným přístrojem, tak si nejdřív musíme vyzkoušet dobu osvitu. Nejlépe to vyzkoušíme na nějakém kousku DPS a s jednoduchou předlohou - pár čar. Když se na desku poté podíváme, měla by být neosvětlená plocha výrazně tmavší než osvětlená. Vložením do vývojky si zkontrolujeme, zda je deska správně osvětlená. Pokud se krásně odmyje osvětlená část a neosvětlená zůstane netknutá, tak máme správný čas. Chce to zkoušet. Po zjištění správného času můžeme nechat osvětlit naší práci. Osvětlíme jí z obou stran. Po osvícení odlepíme fólii a dáme DPS do vývojky. Osvětlené části se začnou odmývat, až tam zbyde pouze neosvětlená fotocitlivá vrstva. Vyndáme DPS z vývojky a omyjeme ho vodou. Ještě mokrý ho můžeme nechat odleptat v leptacím roztoku. Leptání můžeme urychlit ohřátím roztoku např. pomocí teplé vody

(nevlévat do roztoku!!!). Leptání probíhá podle kvality leptacího roztoku. Až se odleptá veškerá měď, opláchneme DPS vodou a necháme oschnout. Při schnutí si připravíme mikrovrtačku na vrtání. Vyvrtáme veškeré díry a natřeme DPS kalafunovým lakem. Necháme zaschnout, popřípadě ještě upravíme rozměry DPS, aby se vešel do krabičky. Nyní máme připraveno na další fázi.

5.2 OSAZOVÁNÍ

Na osazování budeme potřebovat mikropáječku s nějakým slušným hrotem, pájku, svěráček a koupené součástky(viz příloha). Osazujeme podle obrázku osazení v přílohách. Pokud osazujeme zařízení jak se smd součástkami, tak i s klasickými, nejdříve pájíme smd. Na pořadí nijak nezáleží, ale doporučuji nejdříve pájet vícevývodové součástky v našem případě PIC16F688 a MAX232CWE. Po napájení smd se vrhneme na klasické součástky, doporučuji nejdříve propojky mezi vrstvami, poté tlačítko, diody s fototranzistorem a následně konektory. Při pájení smd součástek je dobré si DPS chytit do svěráčku, aby se nehýbalo. Po úspěšném osazení je na řadě naprogramování mikroprocesoru.

5.3 NAPROGRAMOVÁNÍ

K naprogramování mikroprocesoru budeme potřebovat nějaký programátor (Pickit2, ICD2, Presto, ...) a k němu vhodnou utilitu, přes kterou provedeme programování. Program je přiložen na CD. K naprogramování slouží na zařízení 5-ti pinový ICSP konektor, takže stačí pouze zapojit, naprogramovat a máme hotovo. Po naprogramování již můžeme zařízení vyzkoušet.

6. NÁVOD K POUŽITÍ

Zařízení je velice snadno ovladatelné. Napájení je řešeno přes COM port, takže stačí pouze zapojit do počítače, pro lepší manipulovatelnost je dobré použít prodlužovací kabel. Po připojení si na počítači spustíme dodaný program. Program je celkem přehledný. Vlevo si zvolíme potřebná nastavení (defaultně je zařízení nastaveno na rychlost 9600 a parita žádná) a klikneme na tlačítko Connect. Poté by se nám mělo zaktivovat dolní políčko Mod. Vidíme tam 5 různých záložek. Každá z nich reprezentuje 1 unikátní mód, kterým zařízení disponuje. Můžeme vyzkoušet všechny. Pokud chceme nastavit zařízení na jinou rychlost, stačí pouze kliknout na tlačítko Module settings, které nám zaktivuje nastavení rychlosti a parity. Po zvolení parametrů je nutné potvrdit tlačítkem OK. Chvíli počkáme a můžeme opět používat. Když zařízení přestane reagovat nebo prostě nebude reagovat už od začátku stačí stisknout tlačítko a držet ho po dobu 5-ti sekund. Po rozsvícení červené LED je zařízení resetováno do defaultních parametrů. Pro odpojení a vypnutí zařízení stačí pouze kliknout na tlačítko Disconnect.

7. ZÁVĚR

Celkově nebyla práce nijak náročná. Byla spíše zaměřena na širší spektrum znalostí. Téměř veškeré práce na zařízení se obešly bez větších problémů. K menšímu záseku pouze došlu při programování software pro PC a programovacímu jazyku Visual Basic. Nedoporučuji tento jazyk používat v důležitějších projektech, mohli byste narazit. Při testování nebylo s úspěchem nic zničeno a bylo dosaženo dobrých výsledků. Vše fungovalo jak má, pouze se objevily nějaké chyby na vyšších přenosových rychlostech (57600,115200), kdy se zařízení občas sekne. Toto je způsobeno malým výpočetním výkonem a chybami u USART v určitých revizích PIC16F688, které bohužel posílají jako vzorky.

8. ZDROJE INFORMACÍ

- www.gme.cz
- www.hw.cz
- www.microchip.com
- www.wikipedia.org

a mnoho dalších webových stránek

9. SEZNAM POUŽITÉHO SOFTWARE

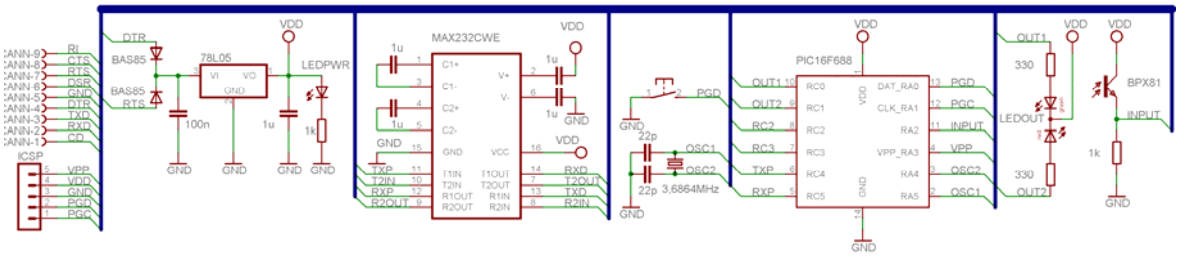
- Visual Studio 2005
- MPLAB v8.15a
- HI-TECH PICC Lite v9.60
- Microsoft office 2003
- Eagle 4.16r2

10. PŘÍLOHY

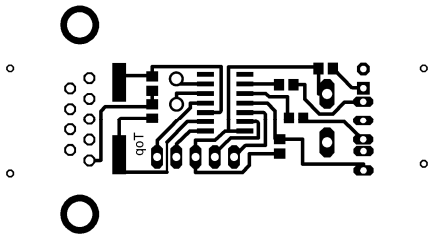
10.1 SEZNAM SOUČÁSTEK

Diody	-	BAS85 smd	-	2x
C	-	100nF, smd 0805	-	1x
	-	1uF, smd 0805	-	5x
	-	22pF, smd 0805	-	2x
R	-	1k, smd 0805	-	2x
	-	330R, smd 0805	-	2x
LED	-	5mm, červená, 2mA	-	1x
	-	5mm, dvoubarevná(červená,zelená) společná katoda	-	1x
Fototranzistor	-	BPX81, smd	-	1x
IO	-	MAX232CWE, smd	-	1x
	-	PIC16F688, SOIC14	-	1x
Stabilizátor	-	78L05F, smd	-	1x
Krystal	-	3,6864MHz, smd 2 vývody	-	2x
Patice	-	SIL10PZ	-	1x
DPS	-	s pozitivem, minimálně 60x30	-	1x
Konektor	-	CAN 9 Z 90, bez plastu	-	1x

10.2 SCHÉMA

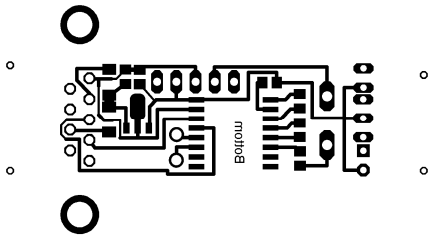

ICSP
VPP
VDD
GND
PGD
PGC
RI
CTS
RTS
DSR
GND
DTR
TXD
RXD
CD
DTR
BAS85
BAS85
RTS
78L05
VI
VO
GND
100n
1u
LEDPWR
1k
GND
VDD
MAX232CWE
C1+
C1-
C2+
C2-
V+
V-
VCC
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
TXP
RXP
RXD
TXD
PGD
PGC
22p
OSC1
OSC2
3,6864MHz
PIC16F688
OUT1
OUT2
RC0
RC1
RC2
RC3
RC4
RC5
DAT_RA0
CLK_RA1
RA2
VPP_RA3
RA4
RA5
INPUT
VPP
330
LEDOUT
BPX81

10.3 DPS TOP



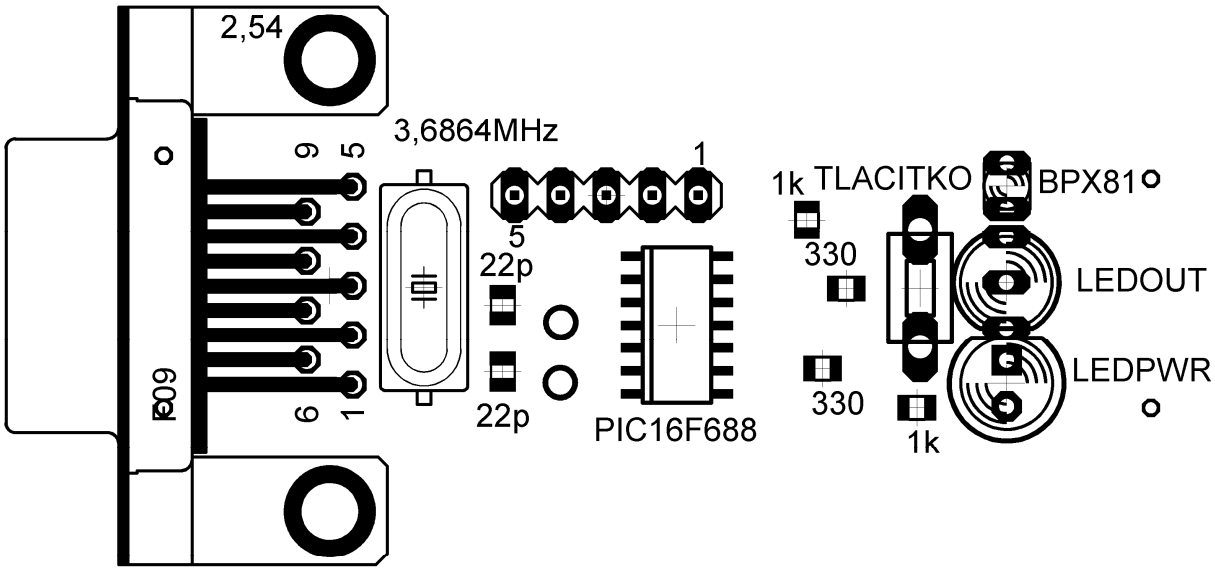
56mm

10.4 DPS BOTTOM

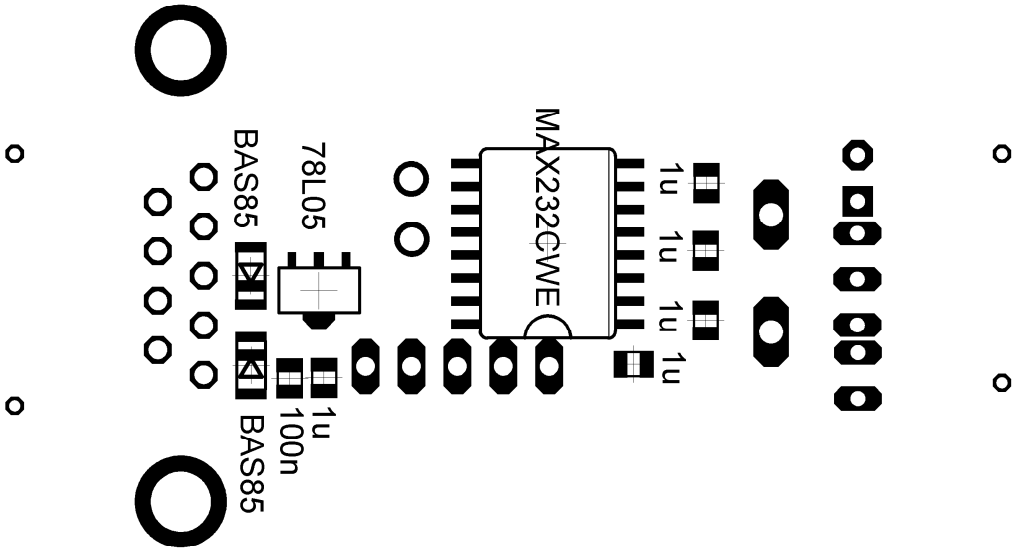


56mm

10.5 OSAZENÍ TOP



10.6 OSAZENÍ BOTTOM



10.7 FIRMWARE

```
/*********************************************************************
 *
 *                                       Miksisek
 *
 *********************************************************************
 * FileName:        main.c
 * Dependencies:    none
 * Processor:       PIC16F688
 * Compiler:        HI-TECH PICC Lite Version 9.60
 *
 *********************************************************************/

#include <htc.h>
__CONFIG(FCMDIS & IESODIS & BORDIS & UNPROTECT & MCLRDIS & PWRTDIS &
WDTDIS & HS);

//*******************************************************************
// Makra
#define SETBIT(ADRESS,BIT) (ADRESS|=(1<<BIT))
#define CLEARBIT(ADRESS,BIT) (ADRESS&=~(1<<BIT))
#define CHECKBIT(ADRESS,BIT) ((ADRESS&(1<<BIT))>>BIT)
//*******************************************************************
//EEPROM adresy
#define EEUSRTCFG 0

// Pinout

#define TLAC            RA0         //PGD
#define PGC             RA1
#define INPUT           RA2         //ADC
#define VPP             RA3
#define OSC2            RA4         //OSCILATOR
#define OSC1            RA5
#define LED1            RC0         //LED
#define LED2            RC1
#define RC2             RC2
#define RC3             RC3
#define TXP             RC4         //UART
#define RXP             RC5


#define RESETT          3           //reset time v s

#define chmezera  delay(60)

#define TXSTAS          0b00100100
#define RCSTAS          0b10010000
#define BAUDCTLS  0b00001000
#define SPBRGHS         0
#define SPBRGS          95

//*******************************************************************
```

```
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);

__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);

__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);

__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);
__EEPROM_DATA(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);

//*******************************************************************
// Promenne

unsigned char UsartConfig = 0;
unsigned char OpMode = 0;
unsigned char PWMperiod = 0;
unsigned char ADCstatus = 0;
unsigned int ADCtemp = 0;
unsigned char reset = 0;
unsigned char RCStatus = 0;
unsigned char RxTemp = 0;
unsigned char RxTemp1 = 0;
unsigned char Parity = 0;
unsigned char ParityCount = 0;
unsigned char ADCdata = 0;
unsigned char z = 0;

bank1 unsigned char Temp[5];

bank2 unsigned char buffer[80];

bit RX9Dtemp = 0;
bit ReceiveCommand = 0;
bit Stav = 0;
bit SetUsart = 0;
```

```
bit SaveUsart = 0;
bit PWMstatus = 0;
bit RAInterrupt = 0;
bit TLACS = 0;

//*********************************************************************
// Deklarace fci
void MCUInit(void);
unsigned char EERead(unsigned char);
void EEWrite(unsigned char, unsigned char);
void SendString(const unsigned char*);
void SendChar(unsigned char);
void UsartConf(unsigned char);
void PWMConf(unsigned char);
void MorseCode(unsigned char);
void delay(unsigned int);
void carka(void);
void tecka(void);
unsigned char ParityCheck(unsigned char);

//*********************************************************************
void main(void)
{
      MCUInit();
      while(1);
}

//*********************************************************************

void MCUInit(void)      //inicializace veskerych periferii
{
      TRISA = 0b11111111;
      TRISC = 0b00000000;

      PORTA = 0b00000000;
      PORTC = 0b00000011;


      ANSEL = 0b00000100;           //RA2 analog
      SCS = 0;          //external clock source
      WPUA0 = 1;        //pull-up tlacitko
      CMCON0 = 7;       //comparators off, digital IO

      //set TMR0
      OPTION = 0b00000100;    //900Hz
      TMR0 = 0;

      //set TMR1
      T1CON = 0b00010001;
      TMR1L = 0;
      TMR1H = 76;

      //set UART
      TXSTA = TXSTAS;
      RCSTA = RCSTAS;
      BAUDCTL = BAUDCTLS;

      SPBRGH = SPBRGHS;
      SPBRG = SPBRGS;  // 9600 b/s

      //set ADC
```

```
	ADCON0 = 0b10001000;
	ADCON1 = 0b01010000;


	//set interrupts
	IOCA0 = 1;  //tlacitko, preruseni PORTA, reset
	TMR1IF = 0;
	TMR1IE = 1;
	RCIE = 1;
	T0IE = 1;
	ADIF = 0;
	ADIE = 1;
	PEIE = 1;
	GIE = 1;    //globalni povoleni preruseni

	if(EERead(EEUSRTCFG) != 0xFF)
	{
		UsartConfig = EERead(EEUSRTCFG);    //config USART
		SetUsart = 1;     //nastav Usart
	}

}

//*********************************************************************

static void interrupt isr(void)
{
	if(RCIF && RCIE)  //prijem znaku po USART
	{
		RX9Dtemp = RX9D;
		RxTemp = RCREG;

		if(Parity)
		{
			if(ParityCheck(RxTemp)) //lichy pocet jedicek
			{
				if(RX9Dtemp) RX9Dtemp = 0;    //celkove je sudy
poce jednicek
				else RX9Dtemp = 1;      //celkove je lichy pocet
jednicek
			}
			else  //sudy pocet jednicek
			{
				if(RX9Dtemp) RX9Dtemp = 1;    //celkove je lichy
pocet jednicek
				else RX9Dtemp = 0;      //celkove je sudy pocet
jednicek
			}
		}

		if(Parity==0 || (Parity==1 && RX9Dtemp) || (Parity==2 &&
!RX9Dtemp))
		{
			if(RxTemp==144) ReceiveCommand = 1; //prijde 0x90,
ocekavame prikaz
			else if(ReceiveCommand) //dekodovani prikazu
			{
				RCStatus = 0;

				Temp[0] = RxTemp;
				for(z=1;z<3;z++)
```

```
                        {
                              while(!RCIF);
                              Temp[z] = RCREG;
                              if(Temp[0]!=Temp[z]) RCStatus++;
                        }

                        if(!RCStatus || RCStatus==1) RxTemp = Temp[0];
                        else if(RCStatus==2 && Temp[1]==Temp[2]) RxTemp =
Temp[1];
                        else
                        {
                              ReceiveCommand = 0;
                              //SendChar('N');
                        }

                        if(ReceiveCommand)      //nastaveni jednotlivych
modu
                        {
                              ReceiveCommand = 0;
                              //SendChar('Y');

                              RxTemp1 = (RxTemp & 0xF0) >> 4;
                              RxTemp = RxTemp & 0x0F;

                              switch(RxTemp1)
                              {
                                    case 1:     //echo
                                          OpMode = 1;
                                          if(!RxTemp) Stav = 0;
                                          else if(RxTemp==1) Stav = 1;
                                          break;

                                    case 2: //pwm
                                          OpMode = 2;
                                          if(!RxTemp)
                                          {
                                                Stav = 0;
                                                LED1 = 1;
                                                LED2 = 1;
                                          }
                                          else if(RxTemp==1)
                                          {
                                                Stav = 1;
                                                LED1 = 0;
                                                LED2 = 1;
                                          }
                                          break;

                                    case 3: //adc
                                          OpMode = 3;
                                          if(!RxTemp)
                                          {
                                                Stav = 0;
                                                ADON = 0;
                                          }
                                          else if(RxTemp==1)
                                          {
                                                Stav = 1;
                                                ADON = 1;
                                                GODONE = 1;
                                          }
```

```
                                    break;

                              case 4: //morse
                                    OpMode = 4;
                                    if(!RxTemp) Stav = 0;
                                    else if(RxTemp==1) Stav = 1;
                                    break;

                              case 5: //citat
                                    OpMode = 5;
                                    if(!RxTemp) Stav = 0;
                                    else if(RxTemp==1) Stav = 1;
                                    break;

                              case 6: //reset
                                    OpMode = 6;

                                    UsartConfig = 0b10000011;
//default nastaveni

                                    SaveUsart = 1;    //pozadavek na
zapis do eeprom
                                    SetUsart = 1;     //nastavit
nove hodnoty
                                    break;

                              case 0x08:  //usart config
                              case 0x0A:
                              case 0x0B:
                                    UsartConfig = (RxTemp1 << 4) +
RxTemp;
                                    SaveUsart = 1;    //pozadavek na
zapis do eeprom
                                    SetUsart = 1;     //nastavit
nove hodnoty
                                    break;
                        }
                  }

            }
            else  //data pro mody
            {
                  if(Stav)    //pokud je mod zaply, zpracovavame
data
                  {
                        switch(OpMode)
                        {
                              case 1:     //echo
                                    buffer[0] = RxTemp;
                                    for(z=1;RxTemp!=144;z++)
                                    {
                                          while(!RCIF);
                                          RX9Dtemp = RX9D;
                                          RxTemp = RCREG;
                                          if(Parity)
                                          {

      if(ParityCheck(RxTemp)) //lichy pocet jednicek
                                                {
                                                      if(RX9Dtemp)
RX9Dtemp = 0;     //celkove je sudy poce jednicek
```

```
                                                      else RX9Dtemp
= 1;  //celkove je lichy pocet jednicek
                                                }
                                                else  //sudy pocet
jednicek
                                                {
                                                      if(RX9Dtemp)
RX9Dtemp = 1;     //celkove je lichy pocet jednicek
                                                      else RX9Dtemp
= 0;  //celkove je sudy pocet jednicek
                                                }
                                          }

                                          if(Parity==0 ||
(Parity==1 && RX9Dtemp) || (Parity==2 && !RX9Dtemp))
                                          {
                                                buffer[z] = RxTemp;
                                          }
                                          else
                                          {
                                                z--;
                                                RxTemp = 0;
                                          }
                                    }
                                    buffer[--z]=0;

                                    SendString(buffer);

                                    break;

                              case 2:     //pwm
                                    PWMConf(RxTemp);
                                    break;

                              case 4:     //morse
                                    buffer[0] = RxTemp;
                                    for(z=1;RxTemp!=144;z++)
                                    {
                                          while(!RCIF);
                                          RX9Dtemp = RX9D;
                                          RxTemp = RCREG;
                                          if(Parity)
                                          {

      if(ParityCheck(RxTemp)) //lichy pocet jednicek
                                                {
                                                      if(RX9Dtemp)
RX9Dtemp = 0;     //celkove je sudy pocet jednicek
                                                      else RX9Dtemp
= 1;  //celkove je lichy pocet jednicek
                                                }
                                                else  //sudy pocet
jednicek
                                                {
                                                      if(RX9Dtemp)
RX9Dtemp = 1;     //celkove je lichy pocet jednicek
                                                      else RX9Dtemp
= 0;  //celkove je sudy pocet jednicek
                                                }
                                          }
```

```
                                                      if(Parity==0 ||
(Parity==1 && RX9Dtemp) || (Parity==2 && !RX9Dtemp))
                                                      {
                                                            buffer[z] = RxTemp;
                                                      }
                                                      else
                                                      {
                                                            z--;
                                                            RxTemp = 0;
                                                      }
                                                }
                                                buffer[z]=144;
                                                for(z=0;buffer[z]!=144;z++)
                                                {
                                                      if(RCIF && RCREG==0x40)
                                                      {
                                                            Stav = 0;
                                                            break;
                                                      }

                                                      MorseCode(buffer[z]);
      //vyblikani zpravy
                                                }

                                                SendChar('D');    //posle D,
vyblikano
                                                break;


                                          case 5:     //citat
                                                if(RxTemp=='G')
SendChar(TMR0/26);      //pokud prijde G, odesilame nahodne cislo v
intervalu 0 - 9
                                                break;

                                    }
                              }
                        }
                  }
      }


      if(T0IF && T0IE)  //PWM
      {
            T0IF = 0;

            if(Stav)
            {
                  if(OpMode==2)     //PWM
                  {
                        LED1 = !LED1;
                        LED2 = !LED2;

                        if(PWMstatus) TMR0 = 255 - PWMperiod;
                        else TMR0 = PWMperiod;

                        PWMstatus = !PWMstatus;
                  }
            }
      }
```

```
	if(TMR1IF && TMR1IE)    //preruseni po 0,1s + tlacitko
	{
		TMR1L = 0;
		TMR1H = 76;
		TMR1IF = 0;

		reset++;
		if(reset==(RESETT*20) && LED2==0) LED2 = 1;     //vypnuti
indikace resetu

		if(Stav && OpMode==3) SendChar(ADCdata); //odeslani dat z
ADC, po 0,1s

		if(RAIF && !RAInterrupt)      //defaultni nastaveni
		{
			if(!TLAC)
			{
				RAInterrupt = 1;
				reset = 0;
			}
			RAIF = 0;
		}
		else if(RAInterrupt && reset==(RESETT*10))
		{
			if(!TLAC)
			{
				LED2 = 0;   //indikace resetu

				UsartConfig = 0b10000011;     //default nastaveni
				SaveUsart = 1;    //pozadavek na zapis do eeprom
				SetUsart = 1;     //nastaveni novych parametru

			}

			RAInterrupt = 0;
		}


		if(SetUsart)
		{
			SetUsart = 0;
			UsartConf(UsartConfig);
		}

		if(SaveUsart)
		{
			SaveUsart = 0;
			EEWrite(EEUSRTCFG,UsartConfig);
		}
	}

	if(ADIF && ADIE)  //snimani svetla
	{
		ADIF = 0;

		if(ADCstatus<64)
		{
			ADCtemp += ADRESH << 8;
			ADCtemp += ADRESL;
			ADCstatus++;
```

```
            }
            else
            {
                  ADCdata = ((ADCtemp/ADCstatus) * 100) >> 10;    //prevod
na procenta

                  ADCstatus = 0;
                  ADCtemp = 0;
            }

            GODONE = 1;
      }

}

//*********************************************************************
void delay(unsigned int time)
{
      unsigned int t10ms,wait,wait2,cas;
      wait = 695;
      for(t10ms=0;t10ms<time;t10ms++)
      {
            for(cas=0;cas<wait;cas++) NOP();
      }
}

void carka(void)
{
      LED1 = 0;
      delay(90);
      LED1 = 1;
      delay(30);
}

void tecka(void)
{
      LED1 = 0;
      delay(30);
      LED1 = 1;
      delay(30);
}

void MorseCode(unsigned char data)
{
      switch(data)
      {
            case 'A':   //akát
            case 'a':
                  tecka();
                  carka();
                  chmezera;
                  break;

            case 'B':   //blýskavice
            case 'b':
                  carka();
                  tecka();
                  tecka();
                  tecka();
                  chmezera;
                  break;
```

```
case 'C':   //cílovníci
case 'c':
      carka();
      tecka();
      carka();
      tecka();
      chmezera;
      break;

case 'D':   //dálnice
case 'd':
      carka();
      tecka();
      tecka();
      chmezera;
      break;

case 'E':   //erb
case 'e':
      tecka();
      chmezera;
      break;

case 'F':   //Filipíny
case 'f':
      tecka();
      tecka();
      carka();
      tecka();
      chmezera;
      break;

case 'G':   //Grónská zem
case 'g':
      carka();
      carka();
      tecka();
      chmezera;
      break;

case 'H':   //holubice
case 'h':
      tecka();
      tecka();
      tecka();
      tecka();
      chmezera;
      break;

case 'I':   //ivan
case 'i':
      tecka();
      tecka();
      chmezera;
      break;

case 'J':   //junácká hůl
case 'j':
      tecka();
      carka();
```

```
            carka();
            carka();
            chmezera;
            break;

        case 'K':   //království
        case 'k':
            carka();
            tecka();
            carka();
            chmezera;
            break;

        case 'L':   //lední hokej
        case 'l':
            tecka();
            carka();
            tecka();
            tecka();
            chmezera;
            break;

        case 'M':   //mládí
        case 'm':
            carka();
            carka();
            chmezera;
            break;

        case 'N':   //nástup
        case 'n':
            carka();
            tecka();
            chmezera;
            break;

        case 'O':   //ó náš háj
        case 'o':
            carka();
            carka();
            carka();
            chmezera;
            break;

        case 'P':   //papírníci
        case 'p':
            tecka();
            carka();
            carka();
            tecka();
            chmezera;
            break;

        case 'Q':   //kvílí orkán
        case 'q':
            carka();
            carka();
            tecka();
            carka();
            chmezera;
            break;
```

```
case 'R':   //rarášek
case 'r':
      tecka();
      carka();
      tecka();
      chmezera;
      break;

case 'S':   //světluška
case 's':
      tecka();
      tecka();
      tecka();
      chmezera;
      break;

case 'T':   //tůň
case 't':
      carka();
      chmezera;
      break;

case 'U':   //uličník
case 'u':
      tecka();
      tecka();
      carka();
      chmezera;
      break;

case 'V':   //vyvolený
case 'v':
      tecka();
      tecka();
      tecka();
      carka();
      chmezera;
      break;

case 'W':   //wagón klád
case 'w':
      tecka();
      carka();
      carka();
      chmezera;
      break;

case 'X':   //Xénie má
case 'x':
      carka();
      tecka();
      tecka();
      carka();
      chmezera;
      break;

case 'Y':   //Ýgar mává
case 'y':
      carka();
      tecka();
```

```
                carka();
                carka();
                chmezera;
                break;

        case 'Z':   //zrádná žena
        case 'z':
                carka();
                carka();
                tecka();
                tecka();
                chmezera;
                break;
    }
}

void UsartConf(unsigned char data)
{
    if(CHECKBIT(data,5))    //parita
    {
        RX9 = 1;
        TX9 = 1;
        if(CHECKBIT(data,4)) Parity = 1;    //licha parita
        else Parity = 2; //suda parita
    }
    else
    {
        RX9 = 0;
        TX9 = 0;
        Parity = 0;
    }

    switch(data&0b00000111) //rychlost
    {
        case 0:     //300
            SPBRGH = 0x0B;
            SPBRG = 0xFF;
            break;

        case 1: //1200
            SPBRGH = 0x02;
            SPBRG = 0xFF;
            break;

        case 2: //2400
            SPBRGH = 0x01;
            SPBRG = 0x7F;
            break;

        case 3: //9600
            SPBRGH = 0;
            SPBRG = 0x5F;
            break;

        case 4: //19200
            SPBRGH = 0;
            SPBRG = 0x2F;
            break;

        case 5: //57600
            SPBRGH = 0;
```

```
                  SPBRG = 0x0F;
                  break;

            case 6: //115200
                  SPBRGH = 0;
                  SPBRG = 0x07;
                  break;
      }
}

void PWMConf(unsigned char data)
{
      PWMperiod = (data*256)/100;
}

unsigned char ParityCheck(unsigned char data)
{
      ParityCount = 0;

      if(data%2) ParityCount++;     //pocita jednicky v bajtu
      data >>= 1;
      if(data%2) ParityCount++;
      data >>= 1;
      if(data%2) ParityCount++;
      data >>= 1;
      if(data%2) ParityCount++;
      data >>= 1;
      if(data%2) ParityCount++;
      data >>= 1;
      if(data%2) ParityCount++;
      data >>= 1;
      if(data%2) ParityCount++;
      data >>= 1;
      if(data%2) ParityCount++;

      if(ParityCount%2) return 1;   //pokud lichy pocet, vrat 1
      else return 0;    //pokud sudy pocet, vrat 2
}

void SendString(const unsigned char* String)    //tato fce posila string
dat po hardware UART
{
      unsigned char TxStringIndex = 0;

      while(String[TxStringIndex]) SendChar(String[TxStringIndex++]);
}

void SendChar(unsigned char data)   //posila znak po hardware UART
{
      while(!TRMT);     //ceka dokud nebude znak odeslan

      if(Parity)  //je parita povolena ?
      {
            if(Parity==1)     //licha parita
            {
                  if(ParityCheck(data)) TX9D = 0;
                  else TX9D = 1;
            }
            else if(Parity==2)      //suda parita
            {
                  if(ParityCheck(data)) TX9D = 1;
```

```
                else TX9D = 0;
            }
        }

        TXREG = data;
}

unsigned char EERead(unsigned char adress)      //cte danou adresu v
pameti
{
        EEADR = adress;
        RD = 1;
        return EEDATA;    //vraci prectena data
}

void EEWrite(unsigned char adress, unsigned char data)      //zapisuje
dana data na danou adresu v pameti
{
        GIE = 0;    //nutne zakazat preruseni pro spravny zapis
        EEADR = adress;
        EEDATA = data;

        EEPGD = 0;
        WREN = 1;

        EECON2 = 0x55;    //inicializace zapisu
        EECON2 = 0xAA;
        WR = 1;
        while(WR);
        WREN = 0;
        GIE = 1;
}
```

10.8 SOFTWARE

```
Public Class Miksisek

    Dim RxTemp As Byte    'buffer pro prichozi data
    Dim RxTemp1 As String
    Dim citat As String
    Dim value As Byte
    Dim buffer() As Byte = {&H90, &H11, &H11, &H11}
    Dim rmod As String
    Public Delegate Sub myDelegate()    'delegat pro prijem dat
(multithreading)


   Private Sub Miksisek_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        For i As Integer = 0 To My.Computer.Ports.SerialPortNames.Count -
1 'nalezeni dostupnych portu a zobrazeni
            cbbPort.Items.Add(My.Computer.Ports.SerialPortNames(i))
        Next

        If cbbPort.Items.Count() <> 0 Then  'pokud byl nalezen alespon
jeden port, vyber prvni
            cbbPort.Text = cbbPort.Items.Item(0)
        Else
            MsgBox("Nebyl nalezen žádný seriový port!",
MsgBoxStyle.Exclamation)    'pokud nebyl, dej zpravu a zavri se
            Me.Close()
        End If

        If SerialPort.IsOpen Then   'pokud je port otevreny, zavri ho
            SerialPort.Close()
        End If

    End Sub


    Private Sub btnConnect_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnConnect.Click

        If btnConnect.Text = "Connect" Then  'pokud je tlacitko ve funkci
Connect

            If SerialPort.IsOpen Then   'pokud je port otevreny, zavri ho
                SerialPort.Close()
            End If

            If cbbPort.SelectedItem <> "" Then
                Try
                    With SerialPort
                        .PortName = cbbPort.SelectedItem    'nazev portu
                        .BaudRate = cbbRychlost.SelectedItem    'rychlost
                        Select Case cbbParita.SelectedItem  'parita
                            Case "žádná"
```

```
                                .Parity = IO.Ports.Parity.None  'bez
parity
                            Case "lichá"
                                .Parity = IO.Ports.Parity.Even  'licha
parita
                            Case "sudá"
                                .Parity = IO.Ports.Parity.Odd   'suda
parita
                        End Select

                        Select Case cbbStopbity.SelectedItem    'pocet
stop bitu
                            Case "1"
                                .StopBits = IO.Ports.StopBits.One
'jeden stop bit
                            Case "2"
                                .StopBits = IO.Ports.StopBits.Two   'dva
stop bity
                        End Select

                    End With

                    SerialPort.Open()   'otevri seriovy port

                    If SerialPort.IsOpen Then   'pokud je port otevreny

                        cbbPort.Enabled = False
                        cbbRychlost.Enabled = False
                        cbbParita.Enabled = False
                        cbbStopbity.Enabled = False
                        lblPort.Enabled = False
                        lblRychlost.Enabled = False
                        lblParita.Enabled = False
                        lblStopbity.Enabled = False

                        With SerialPort
                            .DtrEnable = True
                            .RtsEnable = True
                        End With

                        gpbMod.Enabled = True

                        btnConnect.Text = "Disconnect"   'zmena funkce
tlacitka
                        btnSettings.Enabled = True

                        Timer2.Enabled = True

                    End If

                Catch ex As Exception
                    MsgBox(ex.ToString)
                End Try

            End If


        ElseIf btnConnect.Text = "Disconnect" Then 'pokud je tlacitko ve
funkci Open
            Try
                SerialPort.Close()  'zavri seriovy port
```

```
                If SerialPort.IsOpen = False Then   'pokud je port
zavreny
                    cbbPort.Enabled = True
                    cbbRychlost.Enabled = True
                    cbbParita.Enabled = True
                    cbbStopbity.Enabled = True
                    lblPort.Enabled = True
                    lblRychlost.Enabled = True
                    lblParita.Enabled = True
                    lblStopbity.Enabled = True

                    With SerialPort
                        .DtrEnable = False
                        .RtsEnable = False
                    End With

                    gpbMod.Enabled = False

                    btnConnect.Text = "Connect"   'zmena funkce tlacitka
                    btnSettings.Enabled = False

                    Select Case tbcMod.SelectedTab.Text 'nastaveni
default parametru
                        Case "Echo"
                            rtbEcho.Clear()
                        Case "Pwm"
                            trkbPwm1.Value = "0"
                        Case "Adc"
                            pgrbAdc.Value = "0"
                        Case "Morse"
                            txbMorseStatus.Text = "Waiting"
                            tbcMod.Enabled = True
                        Case "Citat"

                    End Select

                End If

            Catch ex As Exception
                MsgBox(ex.ToString)
            End Try

        End If

    End Sub


    Private Sub btnSettings_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnSettings.Click
        If btnSettings.Text = "Module settings" Then
            cbbRychlost.Enabled = True
            cbbParita.Enabled = True
            btnSettings.Text = "OK"

        ElseIf btnSettings.Text = "OK" Then
            Try
                Select Case cbbParita.SelectedItem  'parita
                    Case "žádná"
                        buffer(1) = &H80
                    Case "lichá"
```

```
                    buffer(1) = &HA0
                Case "sudá"
                    buffer(1) = &HB0
            End Select

            Select Case cbbRychlost.SelectedItem
                Case "300"
                    buffer(1) = buffer(1) + &H0
                Case "1200"
                    buffer(1) = buffer(1) + &H1
                Case "2400"
                    buffer(1) = buffer(1) + &H2
                Case "9600"
                    buffer(1) = buffer(1) + &H3
                Case "19200"
                    buffer(1) = buffer(1) + &H4
                Case "57600"
                    buffer(1) = buffer(1) + &H5
                Case "115200"
                    buffer(1) = buffer(1) + &H6
            End Select
            buffer(2) = buffer(1)
            buffer(3) = buffer(1)

            SerialPort.Write(buffer, 0, 4)

            SerialPort.Close()

            With SerialPort
                .BaudRate = cbbRychlost.SelectedItem    'rychlost
                Select Case cbbParita.SelectedItem  'parita
                    Case "žádná"
                        .Parity = IO.Ports.Parity.None  'bez parity
                    Case "lichá"
                        .Parity = IO.Ports.Parity.Even  'licha parita
                    Case "sudá"
                        .Parity = IO.Ports.Parity.Odd   'suda parita
                End Select

            End With

            SerialPort.Open()

            If SerialPort.IsOpen Then
                cbbRychlost.Enabled = False
                cbbParita.Enabled = False
                btnSettings.Text = "Module settings"
            End If


        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try

    End If


End Sub
```

```
    Private Sub SerialPort_DataReceived(ByVal sender As Object, ByVal e
As System.IO.Ports.SerialDataReceivedEventArgs) Handles
SerialPort.DataReceived
        If SerialPort.BytesToRead <> 0 Then 'prijimani dat po seriovem
portu, pokud prisla data, zavolej delegata a precti data z bufferu
            Invoke(New myDelegate(AddressOf ReadSerial), New Object() {})
        End If
    End Sub

    Public Sub ReadSerial() 'cteni dat ze serioveho portu

        Select Case tbcMod.SelectedTab.Text
            Case "Echo" 'prijata data zobrazime v textovem poli
                RxTemp1 = SerialPort.ReadExisting    'precteni stringu
dat z bufferu
                With rtbEcho
                    .Font = New Font("Garamond", 10.0!, FontStyle.Bold)
                    .SelectionColor = Color.Red
                    .AppendText(RxTemp1 & vbCrLf)
                    .ScrollToCaret()
                End With

            Case "Adc"  'data pro ADC, pokud je cislo vetsi nez 100, data
jsou spatna
                RxTemp = SerialPort.ReadByte
                If RxTemp < 101 Then
                    pgrbAdc.Value = Str(RxTemp)
                End If

            Case "Morse"    'pokud prijde D, vysilani je hotove
                RxTemp1 = SerialPort.ReadExisting
                If RxTemp1 = "D" Then
                    txbMorseStatus.Text = "Waiting"
                    tbcMod.Enabled = True
                End If

            Case "Citat" 'zobrazeni vtipu podle prijateho cisla
                RxTemp = SerialPort.ReadByte()

                Select Case RxTemp
                    Case 0
                        citat = "Kdo nic neví, musí všemu věřit."
                    Case 1
                        citat = "Zlo samo sebe ničí."
                    Case 2
                        citat = "Odborníků na pravdu je víc, než znalců
vína - tomu se musí rozumět."
                    Case 3
                        citat = "Nejstarší a největší láska je láska k
životu."
                    Case 4
                        citat = "Hledat v ženě přítelkyni je o mnoho
těžší, než najít u ní přítele."
                    Case 5
                        citat = "Loweryho princip: Když to nejde po
dobrém, dělej to násilím. Jestli se to ulomí, šlo o vadnou součástku."
                    Case 6
                        citat = "Kdo mnoho začíná, máloco dokončí."
                    Case 7
                        citat = "Ideál je v tobě samém. Překážky k jeho
dosažení jsou také v sobě."
```

```
                    Case 8
                        citat = "Ze štěstí všech se raduj, neštěstím buď
dojat."
                    Case 9
                        citat = "Trp a doufej, svět se změní, bez bolesti
život není.."
                End Select

                MsgBox(citat)

        End Select

    End Sub


    Private Sub tbcMod_SelectedIndexChanged(ByVal sender As Object, ByVal
e As System.EventArgs) Handles tbcMod.SelectedIndexChanged

        Select Case rmod 'spusteni oznaceneho modu
            Case "Echo"
                buffer(1) = &H10
                rtbEcho.Clear()
            Case "Pwm"
                buffer(1) = &H20
            Case "Adc"
                buffer(1) = &H30
            Case "Morse"
                buffer(1) = &H40
            Case "Citat"
                buffer(1) = &H50
        End Select
        buffer(2) = buffer(1)
        buffer(3) = buffer(1)
        SerialPort.Write(buffer, 0, 4)

        Select Case tbcMod.SelectedTab.Text 'spusteni oznaceneho modu
            Case "Echo"
                buffer(1) = &H11
                rtbEcho.Clear()
            Case "Pwm"
                buffer(1) = &H21
                trkbPwm1.Value = "0"
            Case "Adc"
                buffer(1) = &H31
                pgrbAdc.Value = "0"
            Case "Morse"
                buffer(1) = &H41
            Case "Citat"
                buffer(1) = &H51
        End Select
        buffer(2) = buffer(1)
        buffer(3) = buffer(1)
        rmod = tbcMod.SelectedTab.Text
        SerialPort.Write(buffer, 0, 4)

    End Sub


    Private Sub btnEcho_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEcho.Click
        If txbEcho.Text <> "" Then
```

```
            Try
                SerialPort.Write(txbEcho.Text)  'posle text
                SerialPort.Write(buffer, 0, 1)  'ukonceni stringu
                With rtbEcho
                    .SelectionColor = Color.Black
                    .AppendText(txbEcho.Text & vbCrLf)
                    .ScrollToCaret()
                End With
                txbEcho.Text = String.Empty

            Catch ex As Exception
                MsgBox(ex.ToString)
            End Try
        End If
    End Sub


    Private Sub btnMorse_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMorse.Click
        If txbMorseSend.Text <> "" Then 'odesila text na odblikani
            Try
                SerialPort.Write(txbMorseSend.Text)  'posle text
                SerialPort.Write(buffer, 0, 1)  'ukonceni stringu

                txbMorseSend.Text = String.Empty

                tbcMod.Enabled = False
                txbMorseStatus.Text = "Transmitting"

            Catch ex As Exception
                MsgBox(ex.ToString)
            End Try
        End If

    End Sub


   Private Sub btnVtip_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCitat.Click
        SerialPort.Write("G")    'posle pozadavek na nahodne cislo
    End Sub


    Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        If SerialPort.IsOpen And tbcMod.SelectedTab.Text = "Pwm" Then
'odesilani PWM dat
            If Val(trkbPwm1.Value) = 100 Then
                buffer(0) = Val(trkbPwm1.Value) - 1
            Else
                buffer(0) = Val(trkbPwm1.Value)
            End If
            SerialPort.Write(buffer, 0, 1)
            buffer(0) = &H90
        End If

    End Sub


    Private Sub Timer2_Tick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Timer2.Tick
```

```
        Timer2.Enabled = False

        Select Case tbcMod.SelectedTab.Text 'spusteni oznaceneho modu
            Case "Echo"
                buffer(1) = &H11
                rtbEcho.Clear()
            Case "Pwm"
                buffer(1) = &H21
            Case "Adc"
                buffer(1) = &H31
            Case "Morse"
                buffer(1) = &H41
            Case "Citat"
                buffer(1) = &H51
        End Select
        buffer(2) = buffer(1)
        buffer(3) = buffer(1)
        rmod = tbcMod.SelectedTab.Text
        SerialPort.Write(buffer, 0, 4)

    End Sub

End Class
```