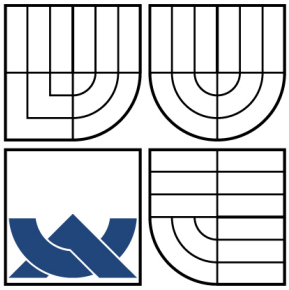
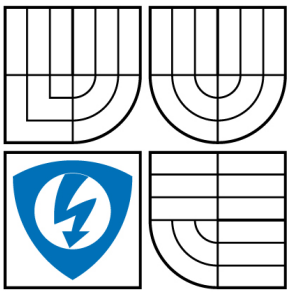


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

VYSÍLAČ MIDI POVELŮ PG

TRANSMITTER OF PG MIDI MESSAGES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

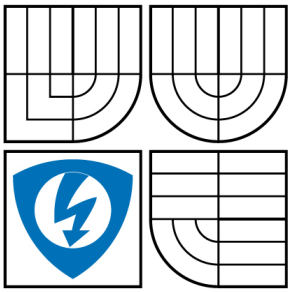
AUTOR PRÁCE — JOSEF ČEPL

AUTHOR

VEDOUCÍ PRÁCE — Ing. LADISLAV KÁŇA

SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Čepl Josef	***ID:***	78540
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Vysílač MIDI povelů PG

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte systém umožňující odesílání MIDI PG a CC zpráv do externího zvukového modulu. Zadávání zpráv řešte pomocí numerické USB klávesnice. Pro komunikaci volte dvouřádkový displej, k řízení použijte některý z mikrořadičů ATMEL. Data budou odesílána do modulu z portu MIDI OUT.

DOPORUČENÁ LITERATURA:

[1] MATOUŠEK, D.: USB prakticky s obvody FTDI, BEN, Praha, 2003.
[2] MATOUŠEK, D.: Práce s inteligentními displeji, BEN, Praha, 2005.
[3] FORRÓ, D.: MIDI komunikace v hudbě, Grada, Praha, 1996.

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Ladislav Káňa

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Josef Čepl
Bytem: Slatinice - Lípy 81, 78342, p. Slatinice
Narozen/a (datum a místo): 6.2.1986, Olomouc

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Vysílač MIDI povelů PG
Vedoucí/školitel VŠKP: Ing. Ladislav Káňa
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

ANOTACE V ČESKÉM JAZYCE

Bakalářská práce se zabývá problematikou MIDI komunikace v moderním hudebním průmyslu se zaměřením na ovládání externího zvukového modulu pomocí uživatelsky jednoduchého zařízení. K realizaci byly vybrány dostupné technické prostředky, jakými jsou mikroprocesor, klávesnice a LCD zobrazovač. Výrobek bude rozšiřovat možnost použití hudebních nástrojů, zejména klávesových, které kromě hraní nedisponují vlastností ovládat externí moduly kompatibilní s MIDI. Vysílač bude zapojen do řetězce MIDI systému mezi nástroj a uvažované koncové zařízení. Programové vybavení bude zaměřeno na jeden z uživatelsky nejrozšířenějších programovacích jazyků C, který právě podporuje vývojové prostředí freewaru WinAVR. Technologie AVR programátorů pak umožní implementaci programu do samotného mikrokontroléru. První částí práce je teoretický rozbor architektury MIDI rozhraní používaného v komerčně dostupných hudebních systémech od specifikace rozhraní až po používané typy zpráv. Druhá část práce obsahuje podrobné řešení zadaného úkolu s ohledem na jeho praktické služby případnému uživateli. Pozornost je věnována návrhu jednoduché komunikace s obsluhou, technickému zpracování zařízení a programovému vybavení. Součástí je shrnutí dosažených výsledků práce a případných dalších rozšíření a aktualizací výrobku.

Klíčová slova: MIDI, PG zpráva, CC zpráva, vysílač, mikrokontrolér, maticová klávesnice

ABSTRACT

Bachelor`s thesis consider in problems of MIDI communication in musical industry with a view to control external sound module assist in user-simply equipment. For the realisation were chosen facilities such as microprocessor, keyboard and LCD display. Product will extend possibilities of usage musical instruments which, except playing, doesn't dispose the ability to control external MIDI compatible modules. The transmitter will involve to MIDI system from between instrument and possible reflected end use device. Software will specialize on programming language C, which is one of the most wide-spread. There is development environment freeware WinAVR which supports even C language. Technology of AVR programmers will allow implementation of program to microcontroler. The first part of thesis is analyse of MIDI interface used in commercial reachable musical systems from specification of the interface to types of messages. The second part contains of detail solution with aspect on it's practical use to potentional user. The conclusion sumarizes of work and possible other updates and actualizations of manufacture.

Keywords: MIDI, PG message, CC message, transmitter, microcontroller, matrix keyboard

BIBLIOGRAFICKÁ CITACE MÉ PRÁCE

ČEPL, J. *Vysílač MIDI povelů PG*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 60 s. Vedoucí bakalářské práce Ing. Ladislav Káňa.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Vysílač MIDI povelů PG“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

PODĚKOVÁNÍ

Zde bych rád poděkoval vedoucímu bakalářské práce Ing. Ladislavu Káňovi za ochotu, vstřícnost a metodickou pomoc při studiu problematiky a technickou realizaci výsledného výrobku.

V Brně dne ....

podpis autora

Seznam použitých zkratek a symbolů

A	Ampér	- Jednotka elektrického proudu
AVR	Advanced Virtual RISC	- Zkratka označující typ jader mikrokontrolérů
CC	Controllers Change	- Změna kontroléru
DCB	Digital Communication Bus	- Digitální komunikační sběrnice
DIN	Deutsches Institut für Normung	- Německý ústav pro průmyslovou normalizaci a z něj vycházející zkratka typu konektoru pro MIDI rozhraní
DPS	Deska plošných spojů	- Zkratka pro desku s vodivými cestami určenou pro zapájení součástek
EOX	End Of System Exclusive	- Konec bloku zvláštních systémových dat
F	Farad	- Jednotka kapacity
JMSC	Japan MIDI Standard Committee	- Japonský výbor pro standardizaci MIDI
LSB	Least Significant Bit	- Bit s nejnižší vahou
MIDI	Musical Instruments Digital Interface	- Digitální rozhraní hudebních nástrojů
CV	Control Voltage	- Řídicí napětí
Trig	Trigger Voltage	- Spouštěcí napětí
MMA	MIDI Manufacturers Association	- Asociace výrobců MIDI
MSB	Most Significant Bit	- Bit s nejvyšší vahou
NAMM	National Association of Music Merchants	- Mezinárodní asociace obchodníků s hudebními nástroji (výstava)
PG	Program Change	- Změna programu
UART	Universal Asynchronous Receiver-Transmitter	- Univerzální asynchronní přijímač/vysílač (typ signálu)
USB	Universal Serial Bus	- Univerzální sériové rozhraní
USI	Universal Synthesizer Interface	- Univerzální hudební rozhraní
V	Volt	- Jednotka elektrického napětí
Hz	Hertz	- Jednotka frekvence
Ω	Ohm	- Jednotka elektrického odporu

Obsah

Seznam obrázků

Seznam tabulek

ÚVOD

V dnešní době proniká elektronika téměř do všech odvětví techniky, vědy a průmyslu. Její prudký rozvoj a zvyšování dostupnosti v posledních desetiletích má za následek masivní rozšiřování i mezi širokou veřejnost. Elektronická zařízení jsou cennými pomocníky pro každého z nás. Stejně tak, jako se málokterému odvětví vyhýbá nasazení elektronicky řízených obvodů, již od prvopočátku vývoje elektroniky se nevyhýbá ani hudebnímu průmyslu. Od samotných elektronických nástrojů začínaje až po řízení světelných a laserových show konče. Mezi výhody neoddiskutovatelně patří možnost propojení nástrojů s ostatními nástroji nebo hudebními moduly, snadný způsob nahrávání a následná přesná reprodukce a také možnosti dodatečných úprav pomocí dostupného softwaru. Do popředí vyčnívá také možnost automatizace, která omezuje počet lidských úkonů například při zvučení či osvětlování kapelního vystoupení. Pokud jsou všechna zařízení v řetězci kompatibilní (nemusí jít nutně jen o hudební zařízení) a schopná komunikace na základě standardizovaného systému, je možné vytvořit efektivní celek, který ve výsledku redukuje počet lidí provádějících obsluhu, zkvalitňuje a zjednodušuje ovládání.

Jak bylo psáno, vývojem došlo k potřebě spojování nástrojů nebo hudebních modulů. Logicky musel vzniknout jakýsi systém komunikace, který bude společný pro všechny výrobce hudebních zařízení a bude tak zajištěna vzájemná kompatibilita. Bylo tedy usilováno o systém s celosvětovým standardem. První pokusy ztroskotávaly právě na nekompatibilitě zařízení mezi sebou, až byl mezinárodní asociací výrobců hudebních nástrojů ustanoven systém řízení a komunikace hudebních nástrojů s přesně danými normami a pravidly – MIDI. I téměř po 25 letech existence se systém MIDI stále upravuje a rozvíjí a je nedílnou součástí velkého množství elektronických hudebních nástrojů. Pro výrobce je tento typ rozhraní stále velice důležitým atributem při návrhu jejich výrobků.

Hlavní myšlenkou práce je vytvořit zařízení, které bude schopno externě ovládat parametry zvuku hudebních nástrojů na základě výše zmiňovaného systému. Zařízení se může stát ceněnou pomůckou buď přímo zvukaře nebo interpreta na hudební nástroj. Bude zapojeno do MIDI řetězce tak, že speciálními typy zpráv, které používá MIDI systém, bude umožněno ovládat některé parametry na dálku. Výhodou bude možnost uživatelského naprogramování a pozdější rychlé interakce během hraní nebo například ozvučování. K realizaci výrobku byly zvoleny volně dostupné elektronické součástky a hojně budou využity možnosti dnešních inteligentních mikrokontrolérů. Popsána je logická výstavba systému MIDI pro pochopení základní architektury a v závěrečné části bude rozebrán návrh vysílače od jeho teoretické výstavby až po naprogramování. Výsledky práce a možnosti eventuelního rozšíření schopností výrobku jsou shrnuty v závěru.

1 MIDI ROZHRANÍ

Zkratka MIDI pochází z anglického názvu *Musical Instrument Digital Interface*, což volně přeloženo do českého jazyka znamená *digitální rozhraní hudebních nástrojů*. Jedná se o komunikační rozhraní především používané v hudebním průmyslu, které i přes svoji relativní zastaralost přežívá dodnes a naopak nabízí ještě i možnosti rozvoje a vylepšení do budoucna.

Podnětem pro vznik MIDI rozhraní a systémů byla potřeba umožnění vzájemné komunikace mezi hudebními nástroji. Snaha vedla ke znásobení paralelního množství propojených nástrojů, aby se tak docílilo vyššího množství současně znějících zvuků. Dále bylo potřebné propojení s počítačem, který by umožnil automatizovanou hru, převod na standardní notaci a také zadávání a editaci dat v reálném čase.

Data přenášená po digitálním rozhraní však nejsou audiosignálem, jak bývá zvykem u jiných systémů reálného času, ale v čase pouze řídicími daty, která vyvolává svoji hrou na například klávesový hudební nástroj jeho interpret. Řídicí data odpovídají tedy událostem, které vznikají v průběhu samotné hry. Jako událost si například můžeme představit stisk klávesy, modulace tónů, nastavování parametrů generovaného zvuku během hraní atd. Při propojení s libovolným nahrávacím zařízením, které je schopno zpracovávat MIDI data, a záznamu posloupností událostí, je možné následné přehrání zaznamenaných dat. Nahrané pasáže se dají následně upravovat, měnit tempo skladby, hrát si s barvami zvuků a je možná spousta dalších činností, které vedou ke zkvalitnění výsledné skladby. [1],[9],[10],[11]

2 HISTORIE MIDI SYSTÉMŮ

2.1 Pre-MIDI systémy

V dobách, kdy ještě nebylo rozhraní MIDI vynalezeno, i tak vznikaly systémy, které byly později označeny za předchůdce rozhraní MIDI. Prvotní myšlenky na propojení nebo jakousi automatizaci hudebních nástrojů vznikaly od počátku jejich vývoje. Zmínky se dochovaly z období antiky a dále pak ze 16. století, kdy vznikaly zvonkohry a hrací strojky řízené různými děrnými štítky, pásy nebo kolečkovým bubnem, který plnil funkci paměti.

V polovině 50-tých let 20. století vznikají první pokusy o propojení elektronických hudebních nástrojů mezi sebou a také s počítačem. Existovaly dva hlavní důvody, proč propojovat nástroje s počítači. Prvním byla samotná automatizovaná hra nebo také dálkové řízení nástrojů a druhým důvodem bylo nahrání do počítače a následná práce se soubory. Od roku 1963 existovala první generace analogových syntetizérů, která pracovala na principu napěťového řízení. V celém systému existovaly dvě skupiny signálů:

1. ***signály řídicího napětí*** - anglicky Control Voltage (zkratka CV)
2. ***spouštěcí signál*** - anglicky Trigger Voltage (zkratka Trig)

Podle názvu těchto signálů byla soustava také někdy označována jako CV/Trig. První signál udával velikost výsledného parametru a druhý zajišťoval spuštění dané akce – synchronizaci. Pro pochopení principu slouží **Obr. 1**.

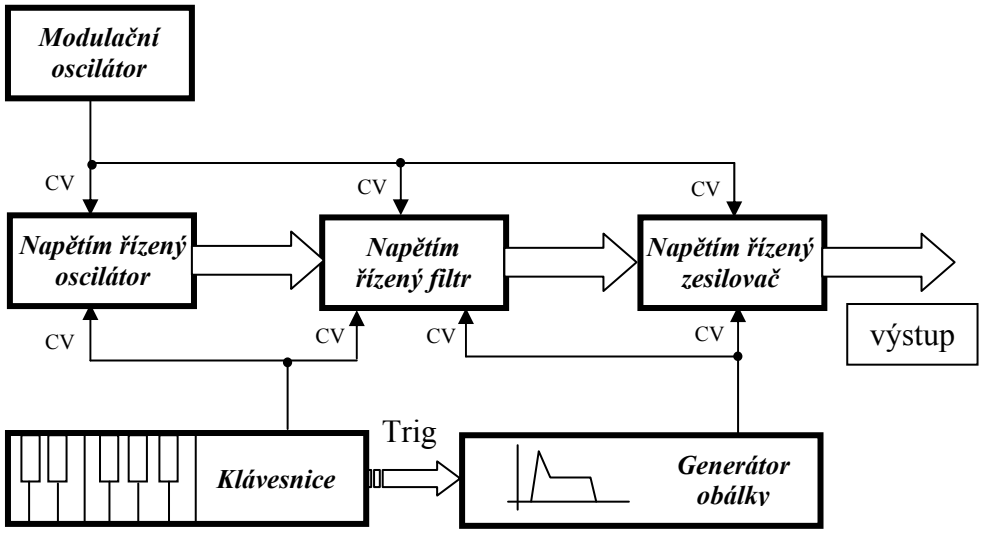
Signály označené na **Obr. 1** se primárně používají uvnitř syntezátoru, ale mohou být také vyvedeny a použity pro synchronizaci s ostatními synchronizátory. Tento systém se však neosvědčil zejména proto, že různí výrobci používali pro signály rozdílné typy a rozsahy. CV signály mohou být totiž:

periodické x neperiodické **schodovité x kontinuální**

Průběh signálů se používal:

a) *exponenciální* - používala například firma Roland. Zvýšení kmitočtu řízeného oscilátoru je úměrné vzrůstu napětí o 1V na jeho vstupu. Jednalo se o normu V/Oct.

b) *lineární* - implementovaly jej do svých nástrojů firmy Korg a Yamaha. Zvýšením napětí o 1V na vstupu se kmitočet oscilátoru změnil o konstantní hodnotu udávanou v Hz, tedy norma Hz/Oct.



Obr. 1: Struktura analogového syntezátoru řízeného CV/Trig signály

Signály Trig mohly být také periodické nebo neperiodické a navíc mohly používat kladnou nebo zápornou logiku. Pokud mluvíme o průběhu Trig signálu a reakce zařízení na něj, mohly být tyto buď spouštěcí nebo hradlové.

Za zmínku stojí i dálkové přepínání rejstříků (programů) nožním spínačem využívané firmami *KORG* a *ROLAND*.

Analogová propojení však byla poměrně primitivní a byla nevhodná díky malému praktickému využití pro uměleckou potřebu. Nevýhodou byla i vysoká mechaničnost ovládání. [1],[9],[11]

Vývojem vystřídalo čistě analogovou techniku digitální propojení. Ze začátku byly zaváděny provozy založené na paralelní komunikaci, ale nestaly se standardem. Na vině byl především drahý hardware (například až 24-vodičové propojovací kabely). Na počátku 80-tých let vznikly první hardwarově nenáročné sériové systémy. Většina výrobců se držela zavedených standardů pro sériová rozhraní typu *RS-232* nebo *IEEE-488*. Komerčně nejúspěšnější se s postupem času stal systém navržený a realizovaný firmou Roland nazvaný *DCB* (Digital Communication Bus) tedy digitální komunikační sběrnice. Rozhraní umělo přenášet informace o stisku kláves, přepnutí rejstříku a mělo i napěťové řízení. Protokol *DCB* obsahoval již rozdělení na datové a stavové byty (identifikátory). Pro stavové byty bylo vyhrazeno 15 kódů, z nichž se ale využívaly jen tři. První, Patch Code, zajišťoval přepnutí rejstříku a byl následován jedním datovým bytem. Druhý Key Code přenášel data o stisknutých klávesách a následoval za ním různý počet datových bytů podle počtu hlasových jednotek nástroje. Posledním stavovým bytem byl ukazatel konce databloku End Mark, který neměl pevně daný počet datových bytů. Výhodou tohoto systému také bylo to, že kromě samotných hudebních nástrojů se vyráběly také různé záznamníky DCB dat, což umožňovalo záznam a editaci hudby. Obecně jsou DCB systémy považovány za přímé předchůdce MIDI systémů. [1],[9],[11]

2.2 MIDI systém

Na červnové mezinárodní výstavě *NAMM* roku 1981 se sešli představitelé tehdejších významných hudebních firem, aby projednali a vypracovali návrhy na vznik univerzálního hudebního rozhraní. V říjnu téhož roku vzniklo hudební rozhraní označované zkratkou *USI*. O rok později proběhla opět schůzka představitelů a ke standardu *USI* byly přidány další úpravy a vzniká rozhraní MIDI. Konečná verze rozhraní MIDI, verze 1.0, byla stanovena na začátku srpna 1983. Stále však díky nejasnosti ve výkladu normy vznikaly problémy s kompatibilitou mezi nástroji jednotlivých výrobců. Vznikají tedy dva orgány, které zodpovídají za dodržování norem - celosvětová *MMA* a také *JMSC*. MIDI norma je stále pozměňována a jsou do ní přidávány nové dodatky a doplňky. [1],[9],[11]

2.3 Současnost MIDI systémů

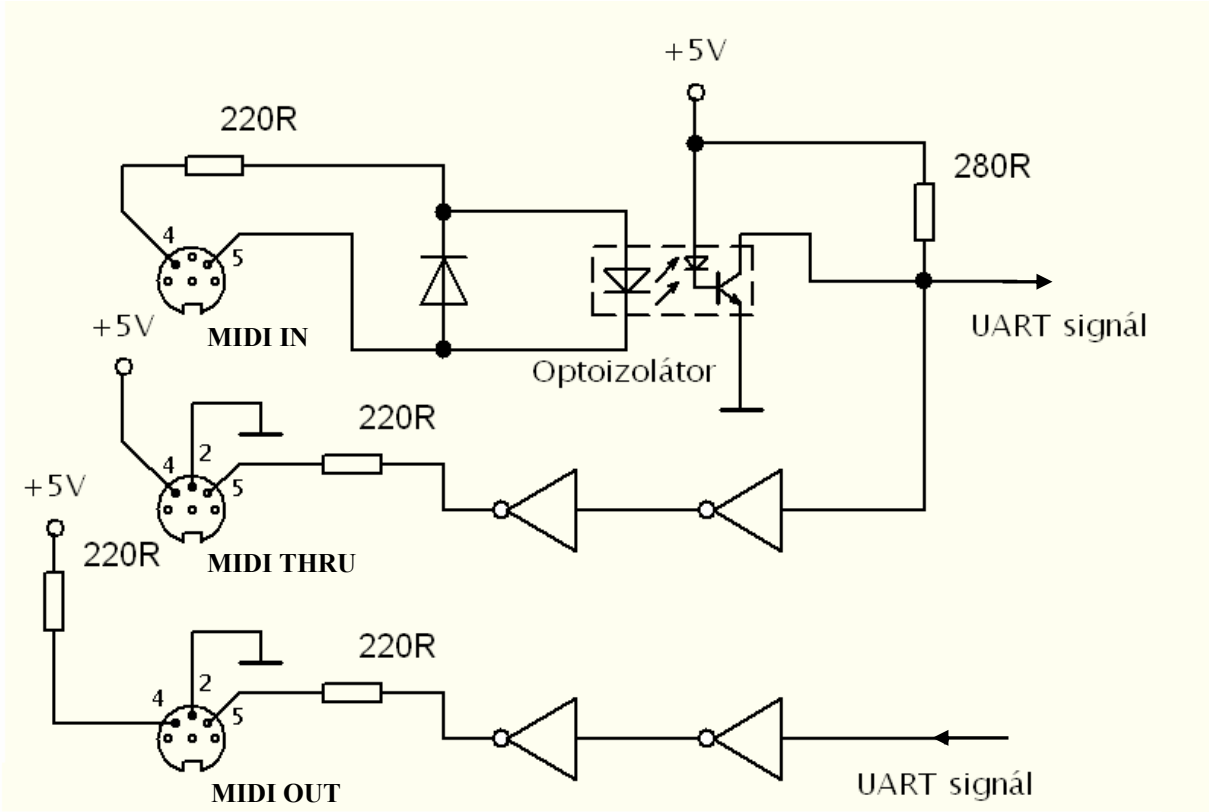
MIDI systém v dnešní době představuje výborné rozhraní s celosvětovým průmyslovým standardem a je součástí většiny elektronických nástrojů (obzvláště klávesových). Samozřejmě má určité limitace a nedokonalosti, které mu však nebrání, aby i tak pracoval velmi uspokojivě a, i po téměř 25 letech svého vývoje, stále ovlivňoval podobu elektronicky produkované hudby.

3 TECHNICKÉ ŘEŠENÍ ROZHRANÍ MIDI

MIDI rozhraní si lze v jednoduchosti představit jako proudovou smyčku se jmenovitým proudem 5mA. Úrovňová logika je vystavěna tak, že logické hodnotě „0„ odpovídá tekoucí proud a logické hodnotě „1„ odpovídá nulová hodnota proudu. Přenosová rychlost rozhraní je 31,25 kBaudů. Použit je asynchronní přenos.

Hlavní částí rozhraní je trojice 5 pinových DIN konektorů pojmenovaných:

1. ***In*** - vstupní konektor, který přijímá do zařízení signály z vnějšího prostředí
2. ***Out*** - výstupní konektor, ze kterého jsou odesílány signály do vnějšího prostředí
3. ***Thru*** - konektor propojený s konektorem In pro bufferovaný průchod dat, umožňuje odeslání nezměněných dat na vstupu například do dalšího zařízení a realizovat tak paralelní či jiné spojení nástrojů



Obr. 2: Zapojení konektorů MIDI rozhraní podle normy 1.0

Z důvodu zabránění zemních smyček mezi jednotlivými přístroji se nesmí země na konektorech připojovat k zemnicím svorkám přístroje. Optoizolátor v zapojení na **Obr. 2** slouží ke galvanickému oddělení vstupu od vlastního přístroje. Optoizolátor nebo také optočlen by neměl mít dobu reakce delší než 2μs. Právě díky zpoždění tohoto prvku v zapojení je omezen počet přístrojů zapojujících se do Thru řetězce na maximálně čtyři. U některých zařízení se vstupní signál slučuje s interně generovanými daty a vše je vysláno na konektor MIDI Thru - tomuto způsobu se říká Soft Thru.

4 SOFTWAROVÉ ŘEŠENÍ ROZHRANÍ MIDI

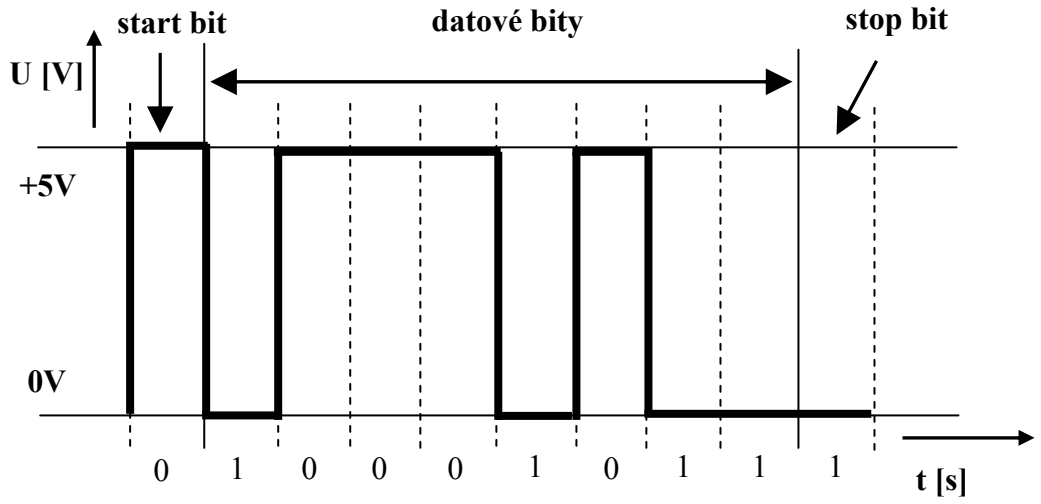
4.1 Struktura MIDI protokolu

Po MIDI rozhraní se přenášejí jednotlivá slova, která mají přesně daný tvar, který je naznačen na Obr. 3. MIDI komunikace je ale založena na tzv. zprávách (anglicky Message). Ty sestávají z několika bytů a místo pojmu Message se též někdy používá pojem MIDI událost nebo anglicky MIDI event. MIDI data se dělí na dvě velké kategorie:

1. *Kanálová data*
2. *Systémová data*

Stejně tak se dají rozdělit byty na:

1. *Stavové byty*
2. *Datové byty*



Obr. 3: MIDI slovo

4.1.1 Kanálová data

Podle svého názvu se jedná o data, která se vztahují pouze k určitému kanálu, jehož číslo je zakódováno v dolním nibblu stavového bytu. Jedním nibblem se rozumí posloupnost 4 bitů. Čtyřmi bity lze vyjádřit 16 čísel, což znamená, že je možno adresovat 16 MIDI kanálů. Data tedy ovládají pouze ten kanál, pro která jsou určena. Kanálová data můžeme dále rozdělit na hlasová a režimová. Funkce vyplývá již z jejich názvu. Hlasová data slouží pro řízení jednotlivých hlasů nástroje a režimová data definují reakci nástroje na přijatá hlasová data. Ve stavovém bytu po adresování kanálu zbydou už jen tři bity, které slouží pro identifikaci kanálových dat. Těmito bity se dá adresovat celkem 7 událostí. Osmá kombinace je totiž určena pro systémová data. **Tab. 1** uvádí kanálová hlasová data. [1],[9],[11]

Kanálová hlasová data				
Tvar stavového bytu	**Hexadecimálně**	**Počet následných datových bytů**	**MIDI zpráva**	**Informace**
1000nnnn	8N	2	Note Off	Nota vypnuta
1001nnnn	9N	2	Note On	Nota zapnuta
1010nnnn	AN	2	Polyphonic Key Pressure	Individuální tlaková citlivost
1011nnnn	BN	2	Controllers Change	Změna kontroleru
1100nnnn	CN	1	Program Change	Změna programu
1101nnnn	DN	1	Channel Pressure	Společná tlaková citlivost
1110nnnn	EN	2	Pitch Bend Change	Ohýbání tónu

Tab. 1: Kanálová MIDI data

Poznámky k **Tab. 1**:

nnnn – binární vyjádření uvažovaného kanálu
N – fyzické číslo kanálu, tedy binárně 0000 znamená 1. kanál

4.1.2 Systémová data

Systémová data nepřenáší informaci o virtuálním MIDI kanále, protože jsou společná pro všechny kanály. Obecně rozdělujeme systémová data na tři skupiny:

1. *společná data* - zamýšlena pro všechny jednotky nacházející se v systému. Jako příklad těchto dat můžeme uvést například lokátor (ukazuje na pozici ve skladbě), resetování a naladění systému.

2. *povely reálného času* - také jsou zamýšleny pro všechny jednotky nacházející se v systému. Obsahují pouze stavové byty. Jejich zvláštností je také to, že mohou být vysílány kdykoliv během přenosu a to i pokud jsou mezi byty zpráv, které mají jiný stavový byte - v tomto případě je příkaz buď ignorován nebo se provede až po návratu z předchozího stavového bytu. Primárně data slouží pro synchronizaci přístrojů, které pracují s reálným časem.

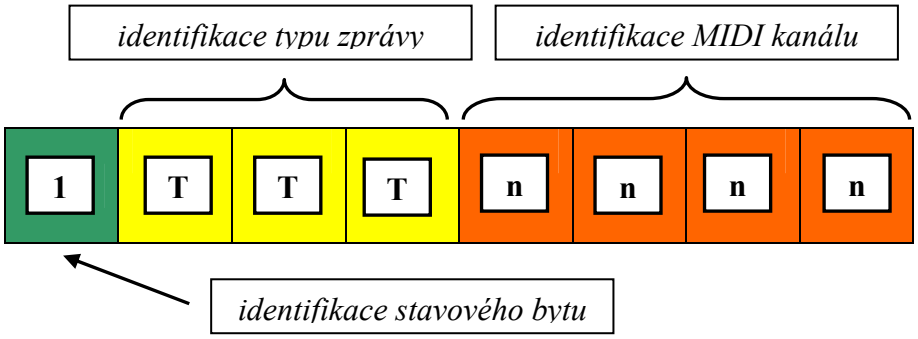
3. *zvláštní systémová data* - obsahují libovolný počet datových bytů a zpravidla bývají zakončena bytem se zkratkou EOX, což znamená konec systémových dat. K jejich ukončení může také nastat automaticky při přijetí libovolného jiného stavového bytu. Tyto informace obsahují identifikační kód výrobce a pokud jim přijímač nerozumí, měl by je ignorovat. [1],[9],[11]

4.1.3 Stavové byty

Stavové byty chápeme jako osmibitová čísla. Ve stavovém bytu je určen typ MIDI zprávy a jednoznačně je po přijetí stavového bytu určeno, co následuje v databytu za ním. Efektivní je rozdělit byte na dvě skupiny po čtyřech bitech – horní skupina bitů a dolní skupina bitů. Těmto skupinám se také říká *nibbly*. V horní skupině bitů je MSB bit u stavového bytu vždy roven 1. Zbývající 3 bity vyjadřují v hexadecimálním tvaru hodnoty 8 – F , které říkají, o jaký typ MIDI zprávy se jedná. V dolním nibblu je zakódováno číslo kanálu, pro který příkaz platí. Principiální obrázek tvaru stavového bytu je na Obr. 4.

Průchozí stavový byte – byl definován za účelem zvýšení průchodnosti MIDI sběrnice a dokáže výrazně snížit provoz na ní. Většinou se totiž na sběrnici přenášejí data se stejným stavovým bytem. Ten je tedy možno přijmout pouze jednou a poté už jen přijímat databyty do doby, než je potřeba změnit typ zprávy. Pokud tedy přijímač obdrží stavový byte, zůstane nastaven na přijímání zpráv zakódovaných v tomto bytu

až do doby, než obdrží jiný byte s jiným typem zprávy. Velice užitečné je ošetření MIDI zpráv Nota zapnuta a Nota vypnuta pomocí tohoto průchozího stavového bytu. Přijímač obdrží informaci o stlačení klávesy s určitou rychlostí a pokud je klávesa puštěna, neobdrží přijímač zprávu Nota vypnuta, ale obdrží MIDI zprávu Nota zapnuta s rychlostí 0. Tím je tedy ušetřen jeden stavový byte na jeden stisk a puštění klávesy. Průchozí stavový byte je ukončen pouze jiným typem stavového bytu. Například data reálného času jej pouze přerušují, ale to vyplývá z kapitoly **5 Uspořádání a priorita dat** v MIDI systémech.



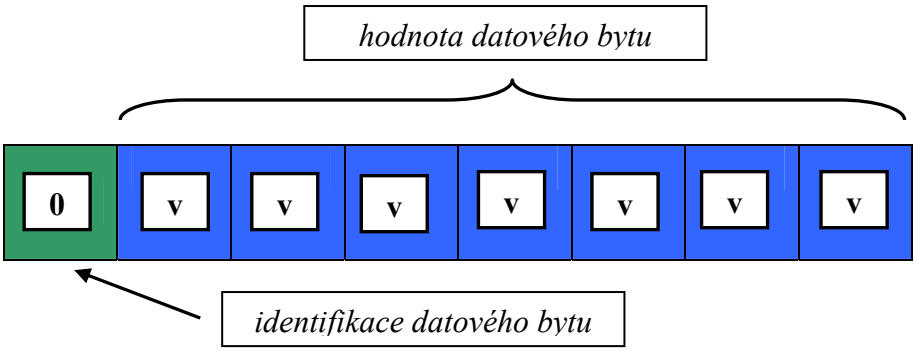
Obr. 4: Tvar stavového bytu

Ignorovaný stavový byte – jedná se o stavový byte, který přijímač neumí rozpoznat. Takové by měly být ignorovány i s daty, která následují za nimi.

Nedefinovaný stavový byte – tento typ bytu by neměl být vůbec vysílan a měla by též být ošetřena situace, kdy při zapnutí nebo vypnutí nástroje by mohl být nedefinovaný byte odeslán. Pokud i přesto dojde k přijmutí takového bytu, měl by jej přijímač ignorovat i s případnými dalšími datovými byty, které by následovaly. [1],[9],[11]

4.1.4 Datové byty

Jedná se také o osmibitová čísla nesoucí obsah zprávy. Následují za každým stavovým bytem s výjimkou dat reálného času. Databyte může následovat jeden nebo také dva, v závislosti na typu MIDI zprávy. Jednoznačně je databyte přijímačem rozeznán podle svého MSB bitu, který má vždy hodnotu „0„. Za identifikačním bitem následuje 7 bitů, které udávají výslednou hodnotu databytu. Můžeme tedy přenášet čísla od 0 do 127. Pro lepší pochopení syntaxe datového bytu slouží Obr. 5. [1],[9],[11]



Obr. 5: Tvar datového bytu

5 USPOŘÁDÁNÍ A PRIORITA DAT V MIDI SYSTÉMECH

Když byly vysvětleny jednotlivé typy dat v MIDI systému, je nutné dále také uvést jejich prioritu. V MIDI systému, jako ve většině podobných systémů, existují totiž upřednostňované typy zpráv. Nejdůležitější data a tedy ta, která mají nejvyšší prioritu, jsou data nesoucí informaci o resetování systému. Za nimi následují data reálného času, která slouží například pro synchronizaci systému. Jako třetí v pořadí jsou zvláštní systémová data následovaná společnými systémovými daty. Data s nejmenší prioritou jsou paradoxně ta, kterých je v MIDI přenosu nejvíce – kanálová data.

Z výše popsaného tedy plyne, že data na nižší pozici žebříčku priority mohou být přerušována daty s vyšší prioritou. MIDI přijímač přijímá data a v případě, že uvnitř posloupnosti databytů přijme například byte s logickou „1„ na začátku (příznak stavového bytu) a jedná se o zprávu reálného času, zpracuje nejdřív tuto zprávu s vyšší prioritou a poté pokračuje dále v přijímání kanálových dat. [1]

6 POPIS ZPRÁV

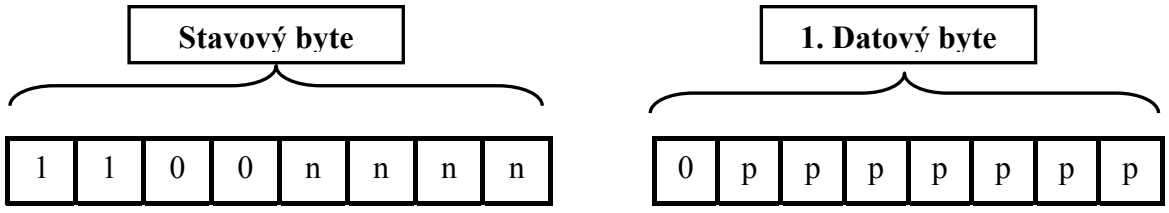
Jak bylo uvedeno, MIDI systém obsahuje různé typy dat. Nyní bych se zabýval dvěma typy zpráv, které jsou hlavní náplní práce. Podrobnější popis ostatních typů zpráv není potřebný a přesahoval by rámec práce. Stačilo základní rozdělení v kapitole 4.1. V následujícím pokračování kapitoly budou řešeny dva typy zpráv, jejichž data jsou velice užitečná pro praktické využití MIDI povelů. Bude se jednat o zprávy obsahující změny programů a o zprávy obsahující změny kontrolérů. Popíši syntaxi MIDI zpráv a jejich rozdělení. Dále se budu zabývat využitím zpráv a problémy, které mohou v celém systému vzniknout při použití daných povelů.

6.1 MIDI zpráva Program Change (PG zpráva)

Volba programu má stavový byte ve tvaru **Cn H**, po němž následuje jediný databyte s hodnotami v rozsahu 0 - 127. Z toho vyplývá, že lze tedy alternovat mezi 128 programy, které se budou nacházet v paměti daného hudebního zařízení. Pro lepší pochopení struktury této zprávy slouží **Obr. 6**. Význam jednotlivých písmen místo bitů je následující:

Bity n - udávají výsledné číslo zamýšleného kanálu, dohromady 16 kombinací -> 16 MIDI kanálů

Bity p - udávají číslo programu, dohromady 128 kombinací



Obr. 6: Syntaxe MIDI zprávy Program Change

128 hodnot pro výběr programu je však ve většině případu málo. Programy bývají tedy organizovány do bank, které jsou přepínány dalším typem MIDI zprávy *Bank Select*. Při praktickém využití tohoto typu kanálových dat se setkáme s různými způsoby organizace a označování programů. Například se setkáme s číslováním desítkovým nebo také existuje výše zmíněné osmičkové dělení na banky po osmi programech.

U starších hudebních nástrojů vzniká také problém s vlastním přepnutím programu. Při přepínání programu během hry na nástroj může vzniknout několik nepříjemných situací. Většinou se stává, že přepnutí způsobí useknutí doznívajících tónů. Jindy je případ ošetřen tak, že tón ihned po přepnutí začne znít s novou barvou. Časová obálka tedy proběhne celá znovu nebo pouze od části s názvem sustain (dozvuk tónu). Výčet negativních vlastností přepínání programu nutí k přepínání programů pouze při doznění všech tónů ale také s dostatečným předstihem před zahráním dalšího tónu. Každý hudební nástroj potřebuje na přepnutí určitou dobu a pokud by tón přišel těsně po MIDI zprávě na změnu programu, mohlo by dojít ke zpoždění nástupu tohoto tónu. Nejlepší volbou je přepínat programy pouze v pauzách. Moderní nástroje však tento problém částečně řeší tím, že „počkají„ na stav, kdy nebude stisknuta ani jedna klávesa a také bude vypnutý kontrolér pro změnu doznívání tónu. [1]

6.2 MIDI zpráva Controllers Change (CC zpráva)

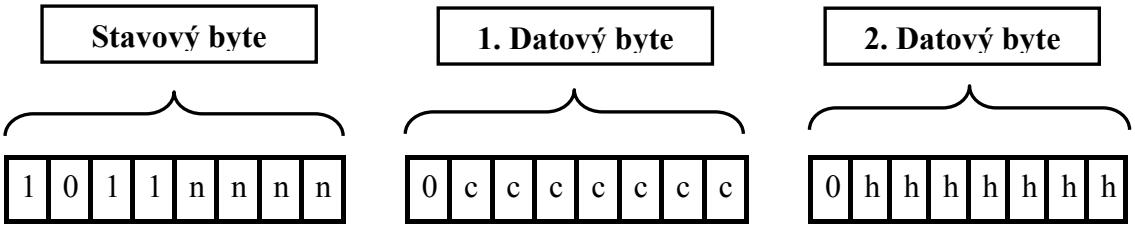
6.2.1 Syntaxe

Tento typ zprávy obsahuje stavový byte **Bn H** a je následován 2 databyty. První z nich označuje číslo zamýšleného kontroléru a druhý udává jeho hodnotu. Syntaxe zprávy je uvedena pro lepší názornost na **Obr. 7**. Význam jednotlivých písmen místo bitů je následující:

Bity n – udávají výsledné číslo zamýšleného kanálu, dohromady 16 kombinací -> 16 MIDI kanálů

Bity c – udávají číslo kontroléru, jehož hodnota se bude upravovat, celkově 128 kombinací

Bity h – udávají hodnotu kontroléru, který je určen bity c, možnost určení 128 hodnot (poloh) kontroléru.



Obr. 7: Syntaxe MIDI zprávy Controllers Change

6.2.1 Rozdělení

MIDI kontroléry se dělí na různé skupiny podle funkce, kterou mají plnit. Pokud nás zajímá oblast funkce MIDI kontrolérů, můžeme je rozdělit na následující skupiny:

a) ***Průběžné kontroléry*** - říká se jim též spojité kontroléry. Ve druhém databytu mohou nabývat hodnot od 0 až po 127.
b) ***Spínače*** - existují pro ně pouze dva stavy - „zapnuto„ a „vypnuto„. Stavu „zapnuto„ odpovídá hodnota 2. databytu v rozmezí 0 - 63. Stav „vypnuto„ je indikován rozsahem hodnot od 64 do 127 ve 2. databytu.
c) ***Inkrementační a Dekrementační kontroléry*** - při přijetí tohoto kontroléru nezáleží na jeho hodnotě. Provede se pouze přičtení nebo odečtení hodnoty 1 od aktuálního nastavení kontroléru. Druhý databyte bývá většinou nastaven na hodnotu 127.
d) ***Povely*** - opět kontrolér, u kterého nezáleží na přijaté hodnotě (s výjimkou kontroléru č. 122 – *Local Control* – hodnota „0„ u něj znamená stav „vypnuto„ a hodnota 127 stav „zapnuto„). Funkce se vykoná pouhým přijetím čísla kontroléru.

Čísla kontrolérů se dělí do následujících skupin:

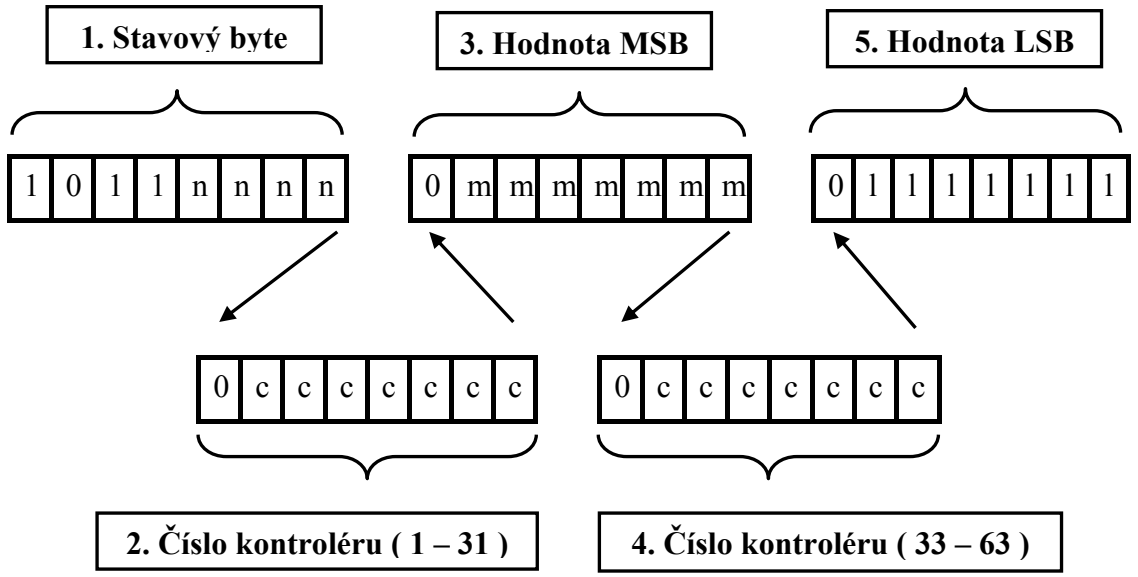
Kontroléry s čísly 0 – 31	– obsahují MSB kontinuálních kontrolérů
Kontroléry s čísly 32 – 63	– obsahují LSB kontinuálních kontrolérů
Kontroléry s čísly 64 – 96	– další jednobytové kontroléry
Kontroléry s čísly 96 – 101	– kontroléry inkrementace/ dekrementace a čísel parametrů
Kontroléry s čísly 102 – 119	– kontroléry, které dosud nebyly definovány
Kontroléry s čísly 120 – 127	– obsahují povely

V následujícím popisu budou rozebrány některé skupiny čísel kontrolérů a uvedu k nim některé představitele a jejich praktický význam. [1],[9],[11]

6.2.1.1 Kontinuální (průběžné) kontroléry

Můžeme si je představit například jako potenciometry, které se nacházejí na režijním pracovišti v nahrávacím studiu. I jejich význam je stejný a sice díky nim je možná plynulá změna například hlasitosti a mnoha dalších parametrů signálu. Při návrhu však bylo usouzeno, že 128 hodnot, které dokážeme definovat ve druhém datovém bytu zprávy, je málo. Vznikly tedy tyto dvoubytové kontrolery s 14 bitovým rozlišením a schopností definovat 16384 hodnot. Princip přenosu spočívá v přenesení MSB a následně LSB bytu. Po přijetí MSB bytu některého z kontrolérů 1–31 (Kontroler 1 *Bank Select* bude zvláště popsán později, protože používá rozdílnou systematiku) nastaví přijímač LSB na hodnotu „0„ a čeká na příchod

odpovídajícího LSB bytu pro daný kontrolér (33 – 63). V případě potřeby pouze 128 úrovní je tedy možné vysílat pouze MSB byte, protože LSB byte se automaticky nastaví na 0. Naopak v případě, že přijímač přijme LSB byte, ponechá hodnotu MSB bytu na stávající hodnotě, z čehož plyne, že pro malé změny některého z kontrolérů není třeba zbytečně vysílat MSB byte, ale stačí vyslání LSB bytu. Na **Obr. 8** je tvar zprávy klasického kontinuálního kontroléru s MSB i LSB.



Obr. 8: Syntaxe MIDI zprávy kontinuálního kontroléru s použitím 14bitového rozlišení hodnot

Zvláštním případem je kontinuální kontrolér *Volba banku* (*Bank Select*). V kapitole **6.1** MIDI zpráva Program Change (PG zpráva) bylo řečeno, že je možné adresovat 128 programů. Pokud je však tento počet nedostačující, existuje způsob, kterým je možné mnohonásobně rozšířit počet programů. Děje se tak právě použitím kontroléru 0/32 *Bank Select*. Programy jsou řazeny do jednotlivých bank, které se přepínají právě tímto kontrolérem. Pro kontrolér neplatí pravidla jako pro zbytek případů ze skupiny kontinuálních kontrolérů. Při přijetí Bank Select kontroléru (kontrolér 0) přijímač čeká na přijetí párového bytu (kontrolér 32) a poté je očekávána zpráva o změně programu (PG zpráva). Teprve po přijetí všech popsaných bytů je provedena změna požadovaného programu.

V **Tab. 2** uvedu všechny využívané i nedefinované kontroléry. Je třeba však mít na paměti, že ne každý hudební nástroj nebo systém s MIDI komunikací musí zákonitě respektovat a dodržovat přiřazení funkcí jednotlivým číslům kontrolérů. V dnešní době by nemělo docházet k problémům s kompatibilitou mezi MIDI nástroji, ale můžeme se setkat se staršími nástroji, u kterých si jejich výrobci vyložili MIDI normy podle svého nebo je

dokonce záměrně porušili. Právě problémy s kompatibilitou způsobené různými výrobci patřily od počátku k velkým problémům MIDI systémů. [1],[9],[11]

Číslo kontroléru	***Anglický název***	***Český název - význam***
0	*Bank Select*	Volba banky programů
1	*Modulation Wheel*	Modulační kolečko k ohýbání tónů, někde použito pouze jako přepínač nebo vícepolohový spínač
2	*Breath Controller*	Dechový ovladač
3	*Undf*	Nedefinováno (dříve YAMAHA definovala jako kontrolér pro společnou tlakovou citlivost, pro porušení normy však vyřazen z provozu)
4	*Foot Controller*	Nožní ovladač (pro nejrůznější účely)
5	*Portamento Time*	Čas (rychlost) portamenta
6	*Data Entry MSB*	Zadávání dat (některé firmy je vysílají se dvěma databyty)
7	*Channel Volume*	Ovládání kanálové hlasitosti
8	*Balance*	Vyvážení
9	*Undf*	Nedefinováno (dříve individuální tlaková citlivost)
10	*Pan*	Směrování, většinou použité jako přepínač
11	*Expression Controller*	Ovladač akcentace hlasitosti kolem nastavené základní hladiny hlasitosti nástroje
12	*Effect Control 1*	Řízení efektu 1
13	*Effect Control 2*	Řízení efektu 2
14-15	*Undf*	Nedefinováno
16-19	*General Purpose Controllers*	Ovladače pro obecné použití (u některých firem například pro souřadnice joysticku atd.)
20-31	*Undf*	Nedefinováno
32-63	*LSB bytes*	LSB byty kontrolérů 0 - 31

Tab. 2: Kontinuální kontroléry

6.2.1.2 Spínače

Kontinuální kontroléry typu spínač mohou nabývat pouze dvou stavů a to „zapnuto„ a „vypnuto„. Používané Spínací kontroléry jsou zaznačeny v **Tab. 3**. Spínače respektují tedy dvě základní polohy dané hodnotou 0 a 127 v datovém bytu. Jejich databyte však může obsahovat kteroukoliv hodnotu mezi těmito mezními. Přijímač se však zachová tak, že

přijme-li hodnoty od 0 do 63 odpovídá to stavu vypnuto a pokud je v databytu hodnota 64 a vyšší bere to přijímač jako vypnutí kontroléru. [1],[9],[11]

Číslo kontroléru	***Anglický název***	***Český název - význam***
64	*Sustain*	Provede podržení všech zahraných tónů
65	*Portamento*	Provede zapnutí efektu glissando
66	*Sostenuto*	Podrží pouze ty tóny, které znějí v okamžiku stisku pedálu, další tóny budou hrány normálně
67	*Softpedal*	Skokové zeslabení hlasitosti nástroje o předem nastavenou hodnotu
68	*Legato*	Spojení tónů v čase
69	*Hold2*	Drží znějící tóny, ale pouze v jiné části obálky než je část Sustain

Tab. 3: Spínací kontroléry

6.2.1.3 Zvukové kontroléry

Slouží ke změnám vlastností generovaného zvuku hudebními nástroji jako jsou barva zvuku a další parametry, kterými lze docílit libozvučných tónů. Některé kontroléry též ovládají výsledný tvar časové obálky po spuštění tónu a tím lze docílit například pozvolnějšího nástupu tónu a také dobu doznívání. [1],[9],[11]

Číslo kontroléru	***Anglický název***	***Český název - význam***
70	Sound Variation	Zvuková variace - volba variací zvuku
71	Timber/Harmonic Intensity	Mění obsažení harmonických složek
72	Release Time	Doba návratu - určuje, jak dlouho má doznívat tón po uvolnění klávesy
73	Attack Time	Určuje, jak dlouho (s jakou strmostí) naběhne tón po stisku klávesy
74	Brightness	Mění zvukovou ostrost

Tab. 4: Zvukové kontroléry

6.2.1.4 Kontroléry pro speciální povely

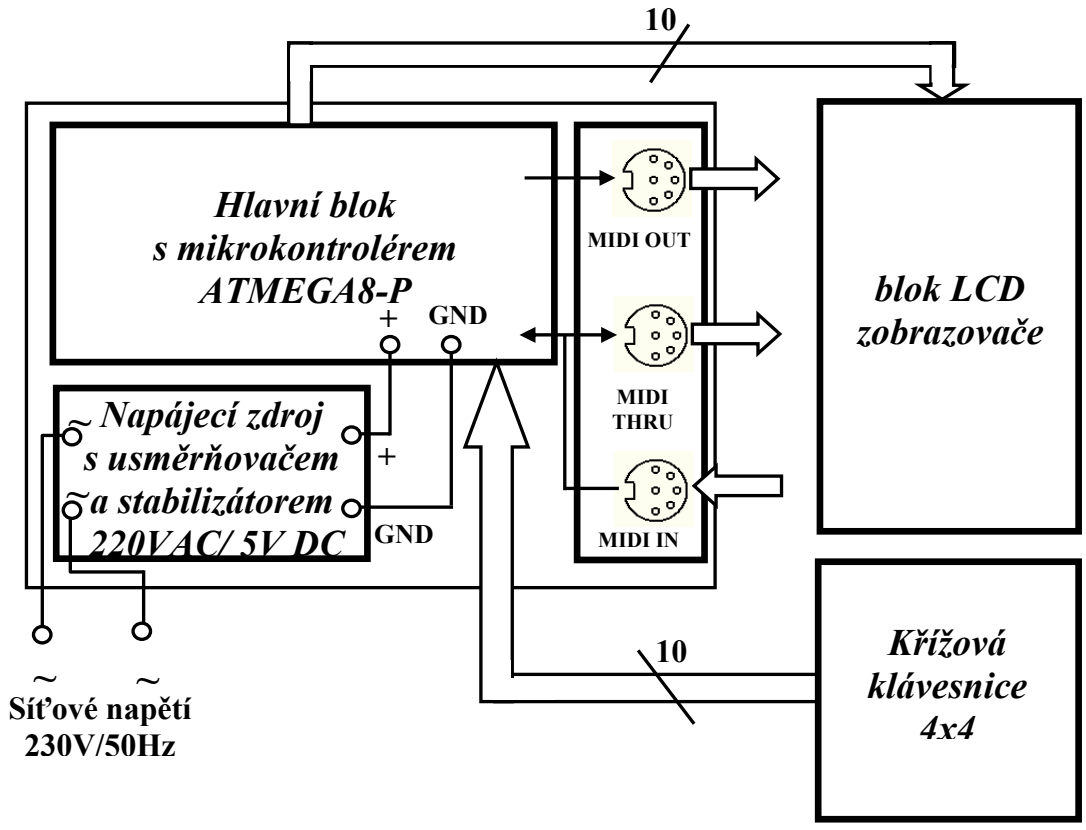
Patří mezi ně užitečné povely jako například resetování všech kontrolérů nebo vypnutí všech zvuků, popřípadě not. To může posloužit při odpojení zařízení během hraní, kdy může zůstat v systému jeden nebo několik znějících tónů. Někteří výrobci používají tyto povely při každém zapnutí a vypnutí přístroje. [1],[9],[11]

Číslo kontroléru	***Anglický název***	***Český název - význam***
120	*All Sound Off*	Provede vypnutí všech zvuků
121	*Reset All Controllers*	Resetování všech kontrolérů a jejich nastavení do výchozích hodnot
122	*Local Control*	Zapíná lokální řízení
123	*All Notes Off*	Příkaz provede vypnutí všech not v systému
124	*Omni Off*	Vypíná režim Omni
125	*Omni On*	Zapíná režim Omni
126	*Mono On*	Zapíná režim Mono
127	*Poly On*	Zapíná režim Poly

Tab. 5: Kontroléry pro speciální povely

7 REALIZACE VYSÍLAČE MIDI ZPRÁV

Na **Obr. 9** je uvedeno blokové schéma zapojení výrobku. Fyzicky jde o tři bloky. Prvním z nich je základ vysílače s MIDI rozhraním, mikrokontrolérem a napájecím zdrojem. Další celek na zvláštní desce je zobrazovací část s LCD displejem a potřebným zapojením pro jeho doprovodné funkce jako podsvětlení displeje a nastavení kontrastu. Poslední částí je ovládací maticová klávesnice. Jednotlivé bloky jsou mezi sebou propojeny odnímatelným paralelním kabelem, v obou případech desetižilovým. Jednoduché oddělení částí by mohlo mít v budoucnu význam pro připojení eventuelních jiných periférií než zobrazovače LCD a křížové klávesnice. Ve stádiu vývoje bylo také toto řešení výhodné z hlediska lepšího odladění a přehlednosti při oživování. Na **Obr. 10** je vyfocený již sestavený vysílač.

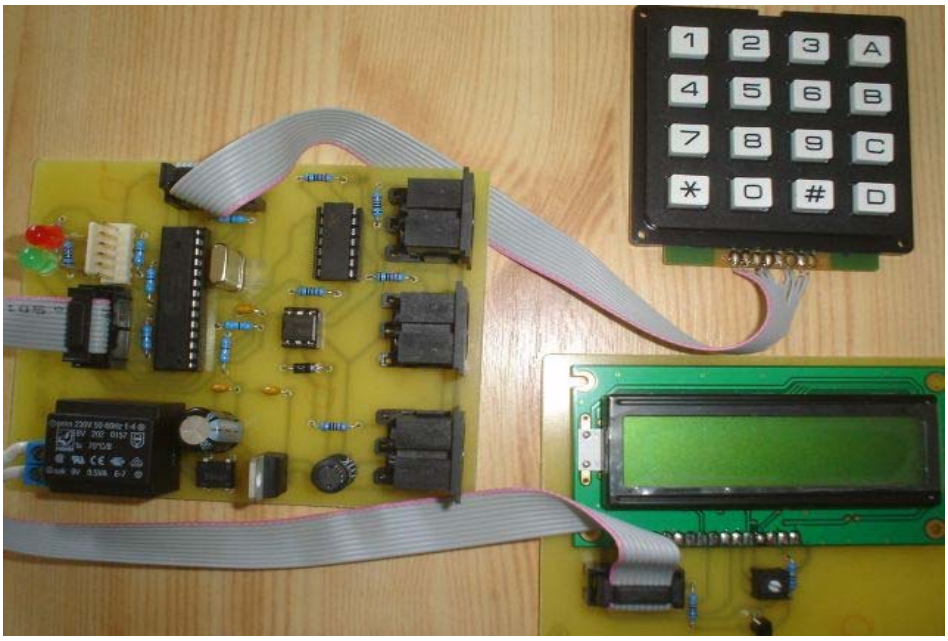


Obr. 9 Blokové schéma vysílače MIDI zpráv

Vysílač lze rozdělit na několik stěžejních funkčních celků, jde o následujích pět:

1. *Napájecí blok*
2. *Blok řízení s mikrokontrolérem*
3. *Rozhraní MIDI*
4. *Zobrazovací modul*
5. *Ovládací maticová klávesnice*

V následujících kapitolách bude rozebrán úkol jednotlivých bloků a význam součástek v nich obsažených. To by mělo sloužit k pochopení činnosti zařízení. Seznam součástek a také schémata zapojení i obrázek cest na desce plošných spojů je uveden v příloze.



Obr. 10: MIDI vysílač povelů PG

7.1 Napájecí blok

Jeho úkolem je napájení obvodů celého přípravku včetně použitého zobrazovače a také napájení konektoru MIDI Thru. Přes transformátor do DPS *TR1* je převáděno síťové napětí 230V/50Hz na střídavé napětí o jmenovité hodnotě 9V při nezatíženém výstupu. Transformátor je dimenzován na maximální jalový výkon 0,5 VA a použitelný je až do teploty 70°C, což by mělo být vzhledem k odběru a použitelným podmínkám zařízení

dostačující. Dále je transformované napětí přivedeno na diodový usměrňovací Graetzův můstek *B1*. Za můstkem následuje filtrační elektrolytický kondenzátor *C1* s hodnotou 1mF, jehož hlavním úkolem je udržet vstupní napětí na potřebné úrovni pro tu část periody, kdy je napájecí střídavá amplituda nižší. Následují součástky potřebné pro stabilizaci napětí. Hlavní součástí je obvod stabilizátoru 7805 s označením ve schematu *IC2*. Úkolem kondenzátorů *C1* a *C3* je zlepšovat stabilitu a odezvu na případnou skokovou změnu zátěže a kondenzátor *C2* zlepšuje potlačení zvlnění na výstupu (udává se hodnota asi 15dB, tedy zhruba pětinásobně). Tlumivka *L1* zajišťuje potlačení proudových impulzů do obvodu za ní. Výsledným stabilizovaným a vyhlazeným napětím jsou napájeny integrované obvody *IC1*, *IC3*, *IO3*, obě LED diody *D2* a *D3* a napájení je přes svorkovnici *SV2* dále distribuováno do bloku LCD zobrazovače. Je jím napájen i pin číslo 5 svorkovnice *J1* použitého pro programátor mikrokontroléru.

7.2 Blok řízení s mikrokontrolérem

Srdcem celého bloku a také přípravku se stal mikrokontrolér firmy *ATMEL* s označením ATMEGA8-16PU. Jak je vidět ze schématu, jedná se o 28-pinový integrovaný obvod, který je pro svůj počet použitelných portů a zabudovanou paměť dnes používán v mnoha aplikacích. Jedná se o 8bitový mikroprocesor, který je vybavený standardními funkcemi pro svoji třídu procesorů jako například: RISC, Flash paměť, A/D převodníky, seriové sběrnice SPI, I2C, sériové rozhraní USART a v neposlední řadě je velkou výhodou tohoto obvodu možnost volby, zda použít port jako vstupní nebo jako výstupní. Za zmínku stojí i dobrá dostupnost vývojového prostředí pro programování procesoru v jazyce C – freewarová verze WinAVR nebo komerční verze CodeVision AVR softwaru. Podrobnější popis mikrokontroléru nalezneme v datasheetu na oficiálních stránkách výrobce.

K mikrokontroléru náleží dále krystal *X1* o hodnotě 4MHz s SMD kondenzátory *C4* a *C5*. Krystal udává taktovací frekvenci, na které běží řídící procesor. Svorkovnice *SV1* je desetipinová a slouží pro připojení klávesnice pomocí desetižilového paralelního kabelu, přičemž použito je pouze 9 pinů svorkovnice (9 žil kabelu), protože se jedná o maticovou klávesnici s 16 tlačítky. Svorkovnice *SV2* je též desetipinová a slouží pro napájení a odesílání dat do zobrazovacího modulu. Svorkovnice *J1* slouží k připojení programátoru na mikrokontrolér. Pomocí této svorkovnice se bude do jeho paměti ukládat obslužný program pro funkci vysílače. Velkou výhodou tak bude možnost pružné změny hlavního programu, z čehož tedy plyne, že výrobek bude moci být libovolně aktualizován, popřípadě bude možno přidávat nové části zdrojového kódu programu pro vylepšení bez nutnosti vyjmutí mikrokontroléru a jeho samostatnému programování v programátoru. Poslední částí jsou dvě LED diody *D2*, *D3* a jejich ochranné odpory *R6*, *R7*, které slouží pro indikaci stavu, ve kterém se vysílač nachází. Červená LED dioda indikuje programovací režim, ve kterém se

volí a ukládají hodnoty do paměti mikrokontroléru. Rozsvícená zelená dioda indikuje aktivní stav, který znamená, že stlačením libovolné klávesy 1 – 9 se do konektoru MIDI OUT vyšle zpráva naprogramovaná pod daným tlačítkem. Pro přehlednost uvedu seznam pinů mikrokontroléru a jejich význam v **Tab. 6**.

7.3 Rozhraní MIDI

Použito je standardní rozhraní MIDI jako ve většině zapojení. Rozhraní bylo teoreticky popsáno již v dřívější části práce. Hlavní součástí jsou tři zásuvkové konektory a odpory *R1 – R5*. Dále je zde integrovaný obvod 4069 (*IC3*), který plní funkce negátorů. Jako optočlen pro galvanické oddělení vstupu vysílače a převodu proudová smyčka/logická napěťová úroveň na straně vstupu je použit integrovaný obvod Sharp PC900V (IO3), na jehož vstupu je navíc zapojena ochranná dioda *D1*. Pokud používáme delší propojovací kabely, tak se na MIDI vstupu mohou objevit parazitní zákmity s opačnou polaritou a právě tento problém řeší zapojení diody. Při absenci diody *D1* by mohlo dojít k nenávratnému poškození LED diody optočlenu, která má povolené napětí v závěrném směru jen asi 5V.

7.4 Zobrazovací modul

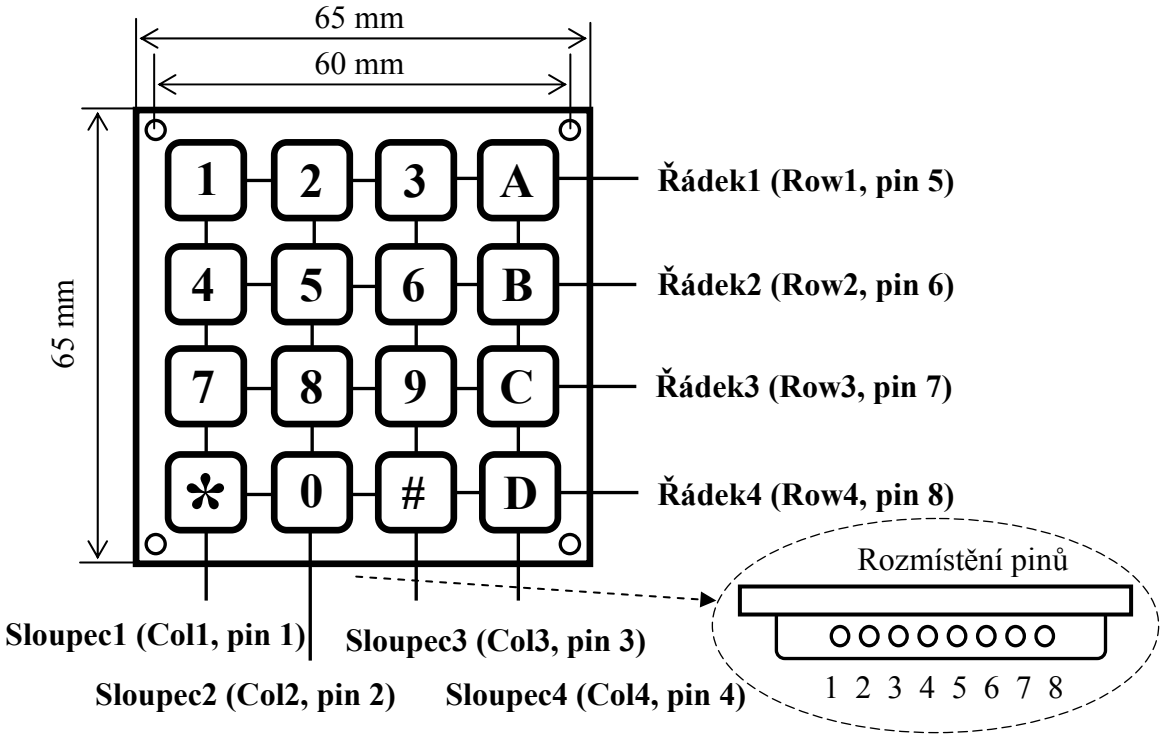
Hlavní částí je dvouřádkový LCD displej AMT1602B. Jeho zapojení a technické parametry je možné si prohlédnout v datasheetu. Tranzistor *T1* spolu s odpory *R1* a *R2* slouží pro ošetření podsvícení displeje. Pro logickou „0„ je podsvícení aktivní a pro logickou „1„ je podsvícení vypnuto. *R1* omezuje proud do báze tranzistoru a rezistor *R2* umožňuje změnit intenzitu podsvětlení displeje. Nejvyšší intenzitu bychom dosáhli použitím odporu s nulovou hodnotou. Trimr *P1* slouží k nastavení kontrastu displeje, který se dá měnit jeho přeladěním na vyšší nebo nižší hodnotu odporu. U většiny displejů by stačilo připájet odpor o hodnotě 1kΩ.

Číslo pinu	*Originální označení podle datasheetu*	*Vstup/ výstup, význam a funkce v zapojení*
1	PC6 (Reset)	Vstup, použit jako resetovací vstup pro programování
2	PD0 (RXD)	Vstup, přijímá MIDI data z konektoru MIDI IN
3	PD1 (TXD)	Výstup, odesílá UART data do konektoru MIDI OUT
4	PD2 (INT0)	Výstup, ovládání podsvětlení displeje
5	PD3 (INT1)	Nezapojen
6	PD4 (XCK/T0)	Vstup/výstup, pin 1 na svorkovnici SV1 (klávesnice)
7	VCC	Pin pro napájení obvodu
8	GND	Uzemňovací pin
9	PB6 (XTAL1/OSC1)	Vstup, zapojení krystalu 4MHz
10	PB7 (XTAL2/OSC2)	Vstup, zapojení krystalu 4MHz
11	PD5 (T1)	Vstup/výstup, pin 2 na svorkovnici SV1 (klávesnice)
12	PD6 (AIN0)	Vstup/výstup, pin 3 na svorkovnici SV1 (klávesnice)
13	PD7 (AIN1)	Vstup/výstup, pin 4 na svorkovnici SV1 (klávesnice)
14	PB0 (ICP1)	Vstup/výstup, pin 5 na svorkovnici SV1 (klávesnice)
15	PB1 (OC1A)	Vstup/výstup, pin 6 na svorkovnici SV1 (klávesnice)
16	PB2 (SS/OC1B)	Vstup/výstup, pin 7 na svorkovnici SV1 (klávesnice)
17	PB3 (MOSI/OC2)	Vstup/výstup, pin 8 na svorkovnici SV1 (klávesnice) Vstup, pin 3 použitý pro programátor
18	PB4 (MISO)	Zapojena zelená LED dioda (indikace aktivní režimu) Vstup, pin 6 použitý pro programátor
19	PB5 (SCK)	Zapojena červená LED dioda (indikace programovacího režimu), Vstup, pin 4 použitý pro programátor
20	AVCC	Nezapojen
21	AREF	Nezapojen
22	GND	Nezapojen
23	PC0 (ADC0)	Výstup, datový bit č.4 pro odesílání dat do LCD displeje
24	PC1 (ADC1)	Výstup, datový bit č.5 pro odesílání dat do LCD displeje
25	PC2 (ADC2)	Výstup, datový bit č.6 pro odesílání dat do LCD displeje
26	PC3 (ADC3)	Výstup, datový bit č.7 pro odesílání dat do LCD displeje
27	PC4 (ADC4/SDA)	Výstup, bit RS pro určení charakteru dat (příkaz/data)
28	PC5 (ADC5/SCL)	Výstup, bit E povolující zápis

Tab. 6: Zapojení pinů mikrokontroléru ATMEL MEGA8-16PU

7.5 Ovládací maticová klávesnice

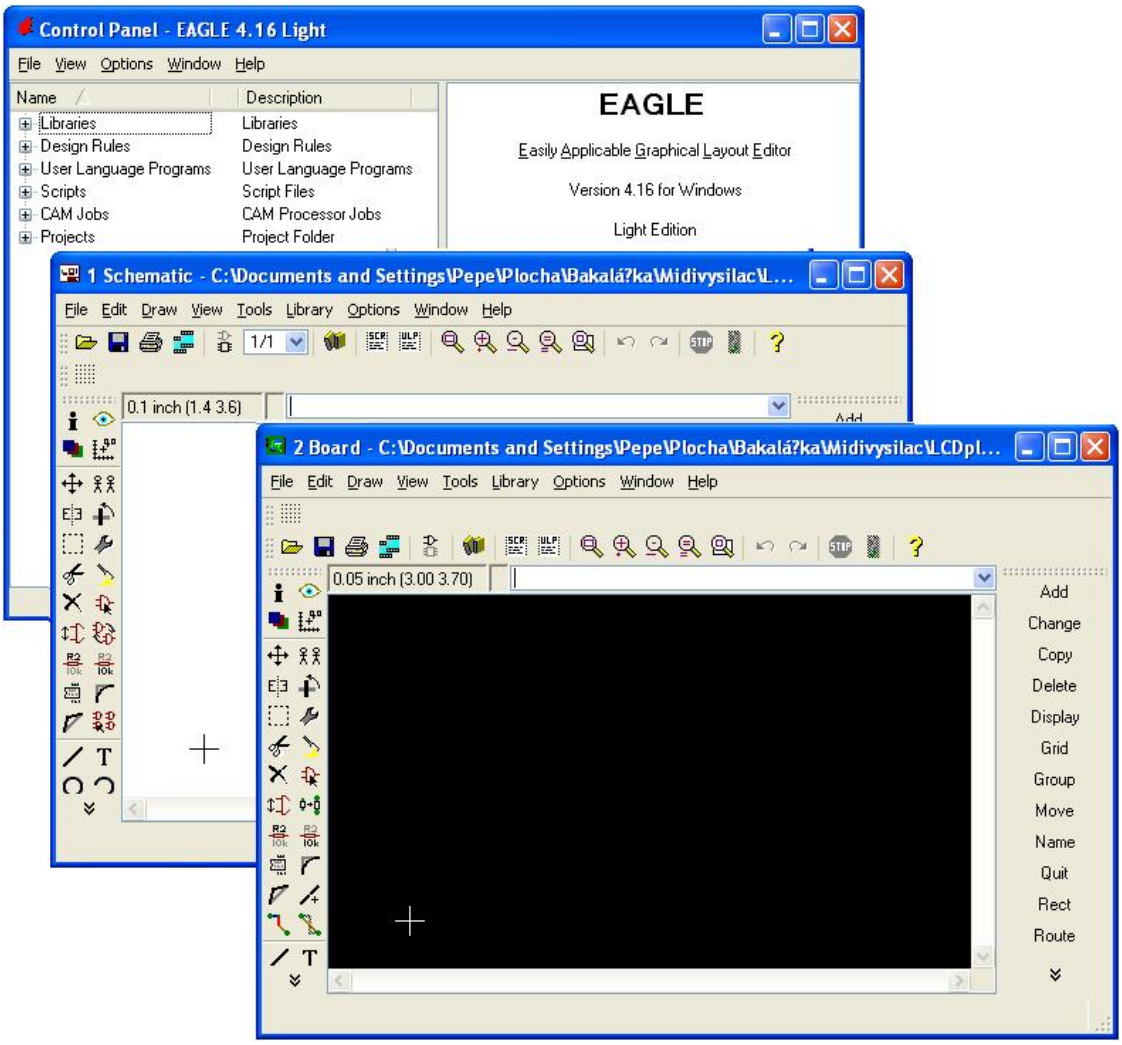
Místo USB klávesnice byla po dohodě s vedoucím práce zvolena klasická maticová klávesnice s 16 tlačítky. Stisk jednotlivých tlačítek na klávesnici lze mikrokontrolérem zjišťovat na základě 8 pinů, které klávesnice obsahuje. Zjištění, které tlačítko bylo stištěno, se provádí detekcí aktivního řádku a sloupce a podle toho jde jednoznačně určit, o jaké tlačítko se jedná. Z důvodů ochrany systému by se klávesnice měla používat spolu se zapájenými ochrannými odpory nebo diodami, které řeší situaci stištění více kláves najednou. Ke klávesnici neexistuje datasheet, proto je uvedeno schema klávesnice s rozmístěním pinů a tlačítek na **Obr. 11** .



Obr. 11: Schéma maticové klávesnice 4x4

7.6 Návrh a vývojové prostředí

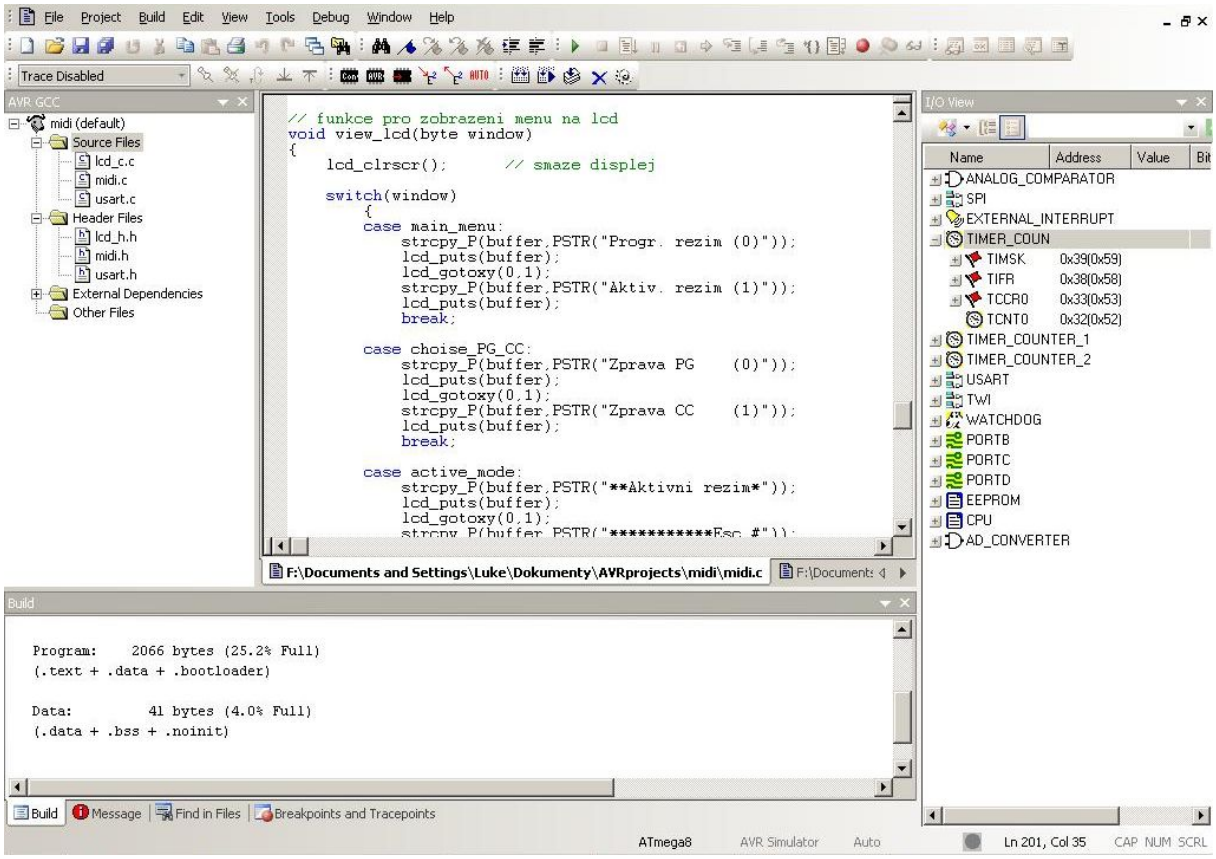
Pro sestrojení schématu výrobku a následného návrhu desky plošného spoje byla použita freewarová verze populárního programu Eagle Layout Editor 4.16 Light. Pro ilustraci jsou na **Obr. 12** screenshoty hlavních oken při vývoji výrobku (Hlavní panel, Panel pro vytváření schematu a panel pro vytváření desky plošných spojů).



Obr. 12: Program Eagle Layout Editor 4.16 Light

Z výsledného návrhu byly později i vyleptány desky plošných spojů pro jednotlivé bloky výrobku.

K sepsání a nahrání programu sloužil software AVR Studio ve verzi 4.13. Ukázku vývojového prostředí si můžeme prohlédnout na **Obr. 13**. Popis zacházení s jednotlivými programy by byl obsáhlý a tedy nad rámec této práce.

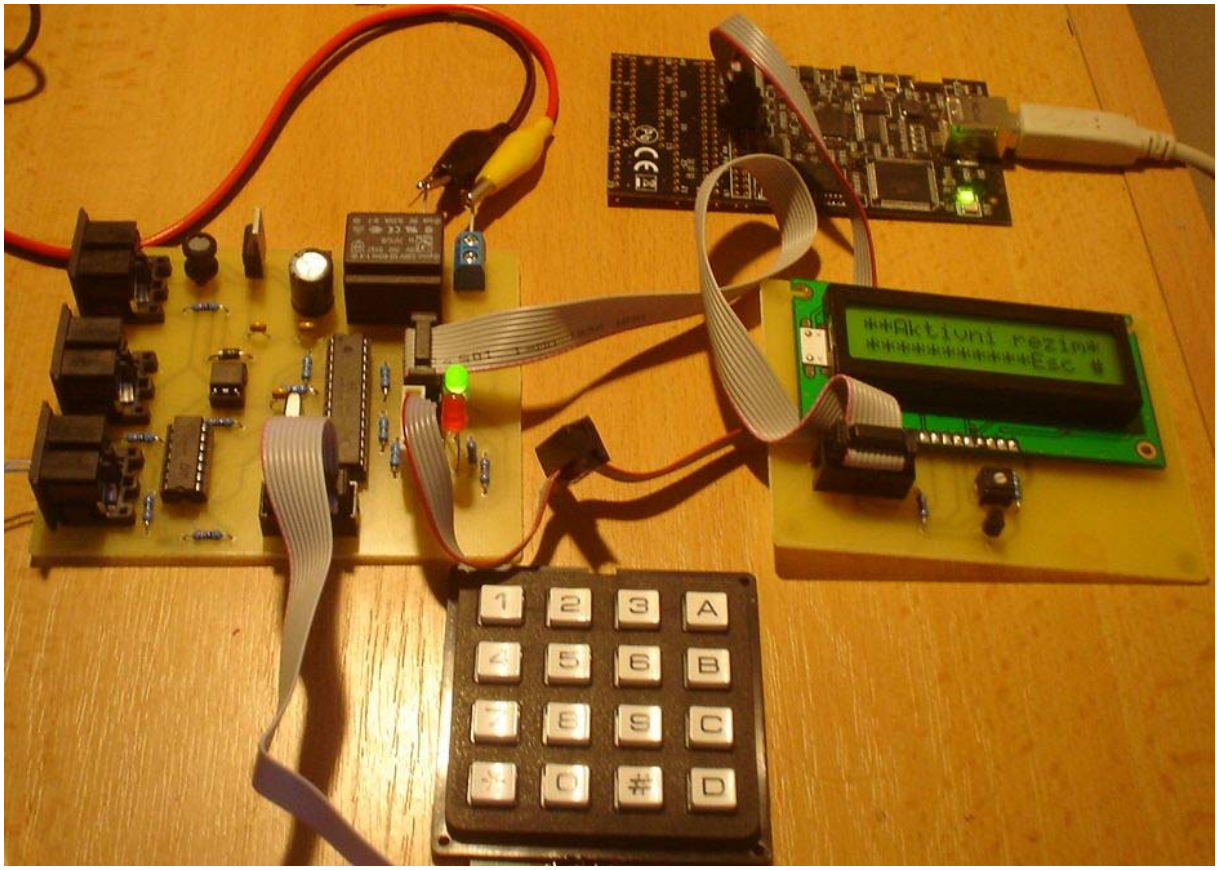


Obr. 13: Ukázka vývojového prostředí programu WinAVR v. 4.13

8 OSAZOVÁNÍ A OŽIVOVÁNÍ VÝROBKU

Po vyleptání desek bylo nutné odvrtat díry pro nožičky součástek. Nejprve bylo provedeno osazení bloku zdroje. Zapájený transformátor jsem proměřil, protože v případě poškození právě této součástky mohlo dojít ke zničení většiny součástek v zapojení vlivem vysokého síťového napětí, na které nebyla většina součástek za napájecím blokem dimenzována. Výstup transformátoru vykazoval střídavé napětí 15V, což je vzhledem k nezatíženému výstupu hodnota očekávaná. Dalo se tedy předpokládat, že transformátor je plně funkční a nehrozí spálení přípravku. Stabilizátor udržoval napájecí napětí na výstupu bloku na hodnotě nižší nebo rovné 5,02V. Následovalo osazení a zapájení ostatních součástek. Nejprve byly zapájeny odpory, kondenzátory, konektory, patice pro integrované obvody a další součástky, které nejsou citlivé na poškození například vysokou teplotou při pájení. Nakonec byly osazeny aktivní součástky jako tranzistory nebo například LCD displej. Za pomocí akustického „pípáku„ na voltmetru následovala kontrola kontaktů, které mělo za následek první zjištění, zda v obvodu nejsou slité cesty nebo studené spoje. Dalším bodem oživování byla kontrola napájecích napětí na příslušných místech výrobku. Až poté bylo možno do patic instalovat integrované obvody a za pomocí kabelů a konektorů připojit dvě periferie – zobrazovací modul a klávesnici. Základní funkčnost displeje jsem zjistil pouhým zapojením a nastavením kontrastu pomocí trimru *P1*, kdy se měla rozsvítit celá horní řada segmentů. Výrobek bylo možno považovat za kompletně osazený a oživený. Zbývalo už jen nahrát do mikrokontroléru obslužný program.

Program WinAVR studio umožňuje spoustu užitečných úkonů při tvorbě, odladění a nahrávání programu do mikrokontrolérů. Kromě samotného sepsání programu, včetně hlavičkových souborů, jde zejména o simulaci činnosti programu a sledování změn stavu proměnných a portů v přehledném prostředí. Po sepsání programu je tento zkompilován a program zahlásí případné chyby nebo varování, které je poté nutno odladit. Během programování je možné průběžně kontrolovat činnost programu nahráním do paměti mikrokontroléru, což bylo velice jednoduché díky použité vidlici pro připojení programátoru na samotném vysílači. Při programování tedy byl výrobek stále připojen na napájení a na programátor. Průběžně byla testována funkčnost programu. Hlavní vytvořený program si můžeme prohlédnout v **příloze 6** a pro ilustraci je na **Obr. 14** vyfoceno zapojení programátoru a vysílače**.** Užitečná funkce programu je i hlášení o stavu volné paměti mikrokontroléru v procentech. V případě programování MIDI vysílače byla vnitřní paměť mikroprocesoru zaplněna konečnou verzí programu zhruba 33%.



Obr. 14: Programování přípravku za pomocí programátoru

9 UŽIVATELSKÝ MANUÁL

Obecné pokyny

Vysílač je určen pro napájení síťovým napětím 220V/50Hz. Je tedy nutné jej připojovat pouze na zmíněné napětí. Jakýkoliv jiný zdroj napětí není dovolen a bude mít za následek nefunkčnost nebo úplné zničení výrobku. Uživatel komunikuje s přípravkem výhradně maticovou klávesnicí. Není tedy nutné zasahovat do zapojení ani jej dodatečně upravovat. V případě potřeby může být jediným nastavitelným prvkem odporový trimr, který nastavuje kontrast displeje. Nemělo by to však být potřebné, protože trimr je již nastaven na potřebnou hodnotu.

Zapojení

Vysílač se do MIDI řetězce dá zapojit dvojím způsobem. Pouze jako ,,ovladač,, , kdy připojíme uvažované zařízení přes konektor MIDI In na konektor MIDI Out vysílače. Druhý způsobem je zapojení do řetězce například mezi klávesy a hudební syntezátor. V tomto případě propojíme klávesy z konektoru MIDI Out do konektoru MIDI In na vysílači. Dále propojíme konektor MIDI Out z vysílače do konektoru MIDI In právě například do hudebního syntezátoru. Konektor MIDI Thru slouží v případě potřeby paralelního zapojení více nástrojů.

Ovládání

Po zasunutí vidlice do napájecí sítě se rozsvítí obě LED diody a provede se inicializace displeje, na kterém se zobrazí menu podle **Obr. 15**.

Obr. 15: Hlavní menu

Výběr programování zpráv provedeme tlačítkem ,,0,,. V opačném případě zvolíme pomocí tlačítka ,,1,, aktivní režim. Na displeji se zobrazí nápis podle **Obr. 16** a rozsvítí se zelená dioda indikující tento režim. Nyní je umožněno vysílání zpráv do připojeného modulu

prostým stiskem jednoho z desíti tlačítek (0-9). Návrat z aktivního režimu provedeme stiskem tlačítka „#„ a opět se dostáváme do hlavního menu.

Obr. 16: Aktivní režim

Po zvolení 1. řádku v hlavním menu se zobrazí nabídka podle **Obr. 17**, kde opět volíme za pomocí tlačítek „0„ nebo „1„ , zda chceme programovat zprávu PG nebo zprávu CC.

Obr. 17: Volba typu zprávy

Stiskem nuly se dostaneme do režimu programování zprávy PG. Nejprve je uživatel vyzván k zadání čísla kanálu, pro který bude zadaná zpráva myšlena (**Obr. 18**). Při zadání je nutné zvolit čísla od 0 do 15 a po volbě stiskem „*„ toto potvrdit. Pokud zadáme jakékoliv číslo mimo rozsah, zobrazí se chybové hlášení „neplatné číslo„ a volbu musíme opakovat.

Obr. 18: Zadání čísla kanálu

Následně je uživatel vyzván k zadání čísla uvažovaného programu podle Obr. 19, tentokrát s možností volby od 0 do 127. Po stisku potvrzovací klávesy „*„ je uživatel dotázán na číslo klávesy, pod kterou chce celou zprávu uložit (Obr. 20). Volbou klávesy od „0„ po „9„ a stiskem „*„ se objeví dialogová zpráva podle Obr. 21, volba se uloží do paměti a vysílač se vrátí do hlavního menu.

Obr. 19: Číslo programu

Obr. 20: Volba klávesy

Obr. 21: Ukládání do paměti

V případě volby programování zprávy CC provede uživatel stisk klávesy „1„ a je standardně dotázán na číslo zamýšleného kanálu. Postup je stejný jako u programování zprávy PG. Následuje zadání čísla kontroléru od 0 do 127 a potvrzení „*„. Dále uživatel zadává hodnotu uvažovaného kontroléru opět od 0 do 127 a potvrzuje „*„. Další postup je identický s programováním zprávy PG. Po úspěšném provedení se vysílač vrací do hlavního menu. Ve kterékoliv části uživatelského módu způsobí stisk klávesy „#„ návrat do hlavního menu.

ZÁVĚR

Úkolem bylo vytvořit samostaný výrobek, který bude schopen odesílat zprávy dvojího typu (PG a CC) do externího modulu a bude celkově zapojitelný do řetězce MIDI. Po dohodě s vedoucím bakalářské práce jsem použil místo USB klávesnice klasickou maticovou klávesnici. Bylo na ní nutno ošetřit případy, které by na USB klávesnici nevznikaly (zákmity, možnost stisku více tlačítek navzájem a tedy potenciální zničení portu mikroprocesoru), ale i tak to bylo výhodnější. USB klávesnice by vyžadovala vytvoření uživatelského prostředí v PC a pravděpodobně i propojení vysílače s osobním počítačem. S použitím maticové klávesnice bude možné zařízení používat bez osobního počítače, který je potřeba pouze při aktualizaci nebo změnách v hlavním programu a to ve spojení s programátorem.

Zařízení jsem teoreticky navrhl v prostředí programu EAGLE Layout Editor 4.16 a to včetně návrhu desky plošných spojů. Následně byly na základě návrhu vyleptány a odvrtány desky plošných spojů. Celkově se zařízení sestává ze tří desek – vysílač, klávesnice a zobrazovací LCD modul. Po osazení přišel na řadu první test funkčnosti a sice odběr zařízení a případná kontrola zkratů. Při nákupu součástek jsem zvolil špatný optočlen, který měl jiné rozmístění země a napájecího pinu a docházelo tedy ke zkratu v zapojení. Po výměně za správný optočlen však systém vykazoval správnou činnost. Napájecí napětí bylo udržováno na konstantní hodnotě a rozvedeno na všechny napájecí piny. LCD modul plně funkční i s podsvícením. Další dílčí zkouškou funkčnosti bylo propojení přípravku napájeného ze sítě s programátorem mikrokontroléru a odzkoušení jednoduchým programem. Komunikace se zdařila napoprvé a do mikrokontroléru byl nahrán jednoduchý program na odzkoušení komunikace klávesnice, vysílače a zobrazovače. Po odladění problémů s maticovou klávesnicí (ošetření zákmitů atd.) systém obstál i při další zkoušce. Odladění problému se však sestávalo z dočasného připájení odporů z vnějšku ke klávesnici a přivedením napájecího napětí na jednu z volných žil paralelního kabelu za pomocí drátové propojky na straně spojů. Pro lepší vzhled výsledku by tedy bylo nutné ještě udělat desku plošného spoje i pro samostatnou klávesnici. Poslední částí bylo programování a nahrání konečného hlavního programu. Postupné odladění zdrojového kódu, odzkoušení výstupů zařízení a uživatelské komunikace ukázalo, že zařízení je opravdu funkční i po této stránce. Z časových důvodů jsem bohužel nestihl „ostrou zkoušku„ zařízení propojením do MIDI řetězce například mezi hudební nástroj a syntetizér.

Výsledkem práce je tedy funkční zařízení na odesílání MIDI zpráv do externího modulu. Vysílač vyniká jednoduchostí zapojení a možností změny obslužného programu v případě potřeby nebo aktualizace. Tato změna je umožněna použitím programovacího portu, který se používá při napájené desce vysílače. Další výhodou by mohla být možnost připojení jiných periferií než pouze klávesnice, což by ale vyžadovalo samozřejmě změnu hlavního

programu. Mírnou nevýhodou je uživatelsky nedokonalé prostředí, které by však šlo za použití složitějšího hlavního programu odstranit stejně jako prozatím nevzhledný tvar vysílače jako jednotlivých desek plošných spojů propojených paralelními kabely. Celkově by šlo zařízení vylepšit a to pracnější změnou programu, rozšířit jeho možnosti, například využívané typy MIDI zpráv. Z tohoto hlediska je zařízení limitováno pouze vnitřní pamětí mikrokontroléru, která slouží pro uložení hlavního obslužného programu. Pokud jde o energetickou náročnost vysílače, činí jeho maximální odběr, který zahrnuje zapnuté podsvětlení displeje se zápisem znaků, odběr obou diod a příslušných integrovaných obvodů, zhruba 110mA. Po celou dobu zkoušení přípravku napojeného na zdroj s indikací odběru se hodnota odebíraného proudu nenavýšila nad tuto hodnotu.

V souvislosti s prací jsem byl nucen se naučit se dvěma, mnou doposud nevyužívanými, programy. Celkově jsem postupoval od teoretického návrhu přes technickou realizaci až po naprogramování přípravku, takže se dá říci, že práce byla zaměřena na téměř všechna elektrotechnická odvětví. Prací jsem tedy získal cenné zkušenosti při návrhu a realizaci programem řízeného elektronického výrobku.

POUŽITÁ LITERATURA

Knižní zdroje:

[1] FORRÓ, D.: *MIDI komunikace v hudbě*, 1. vyd. – Praha, Grada, 1993. - 267 s. ISBN 80-85623-56-0 (brož)

[2] MATOUŠEK, D. *Práce s inteligentními displeji LCD,* 1. vyd. - Praha: BEN - technická literatura, 2006. s. 115–170. ISBN 80-7300-121-7

[3] MATOUŠEK, D. *USB prakticky s obvody FTDI. 1. díl,* 1. vyd. - Praha : BEN - technická literatura, 2003. - 270 s. : il. - ISBN 80-7300-103-9 (váz.)

Internetové zdroje:

[4] *atmel.com* [online]. [c2008] [cit. 2008-05-02]. Dostupný z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf>.

[5] *avrfreaks.net* [online]. [c2008] [cit. 2008-05-02]. Dostupný z WWW: <http://www.avrfreaks.net >.

[6] *CadSoft online* [online]. [c2008] [cit. 2008-05-02]. Dostupný z WWW: <http://www.cadsoft.de/>.

[7] *Datasheet catalog* [online]. [c2008] [cit. 2008-05-02]. Dostupný z WWW: <http://www.datasheetcatalog.com/datasheets_pdf/A/T/9/0/AT90S2313.shtml>.

[8] *gme.cz* [online]. [c2008] [cit. 2008-05-02]. Dostupný z WWW: <http://www.gme.cz/_dokumentace/dokumenty/513/513-143/dsh.513-143.1.pdf>.

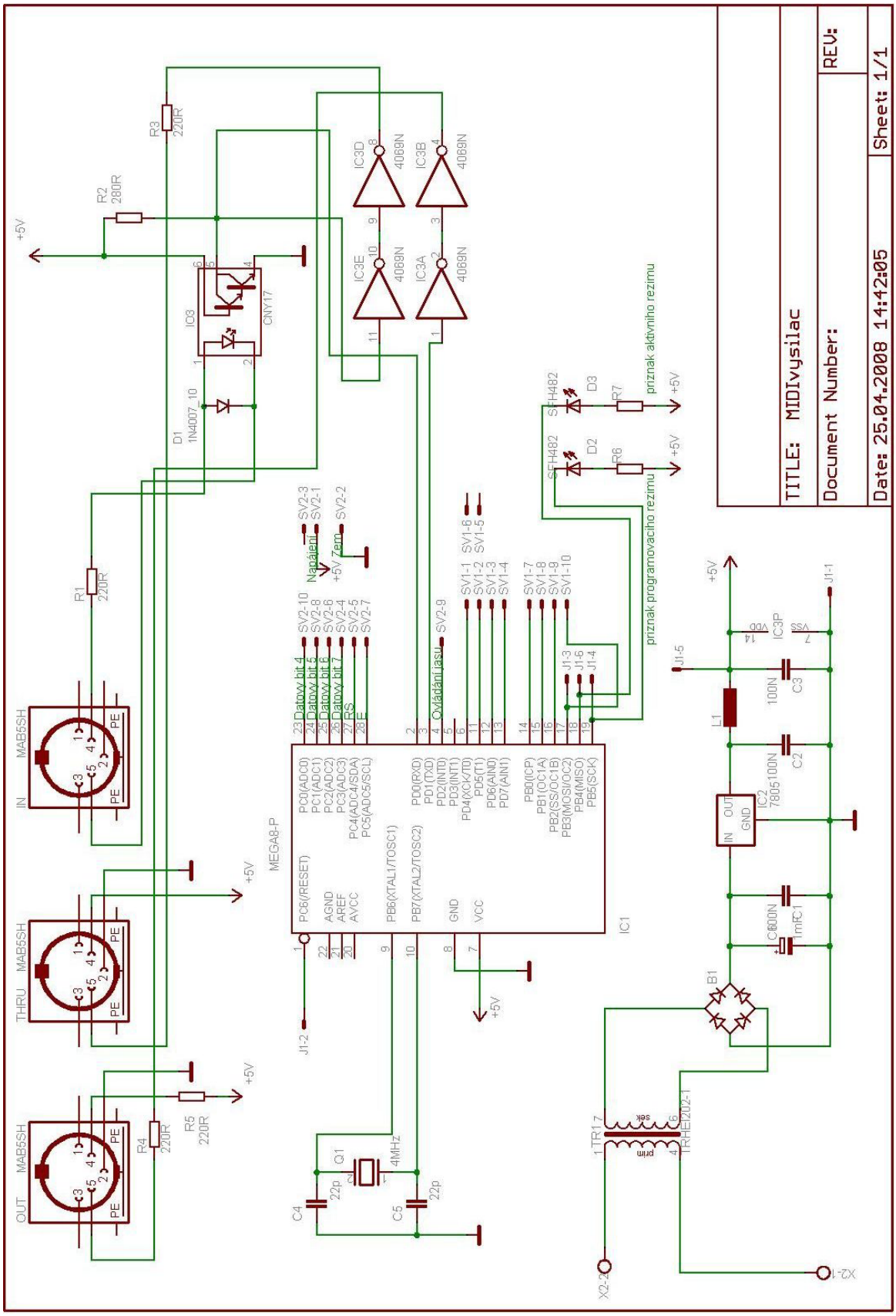
[9] SCHIMMEL, J. *Komunikační rozhraní MIDI* [online]. [2002] [cit. 2007-11-12]. Dostupný z WWW: <http://www.elektrorevue.cz/clanky/02069/index.html>.

[10] TOMAN, P. *MIDI* [online]. [16.9.2000] [cit. 2007-11-12]. Dostupný z WWW: <http://www-kiv.zcu.cz/~herout/html_sbo/midi/toc.html>.

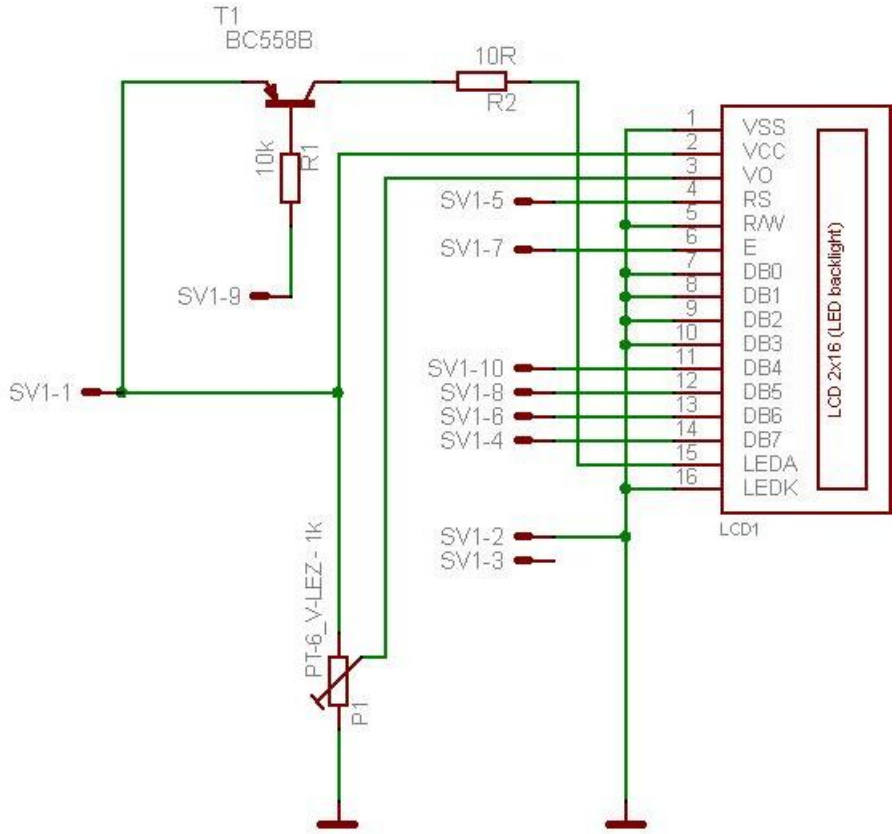
[11] *Wikipedia: Musical Instrument Digital Interface* [online]. c2001-2007 [cit. 2007-11-12]. Dostupný z WWW: http://cs.wikipedia.org/wiki/Musical_Instrument_Digital_Interface.

SEZNAM PŘÍLOH

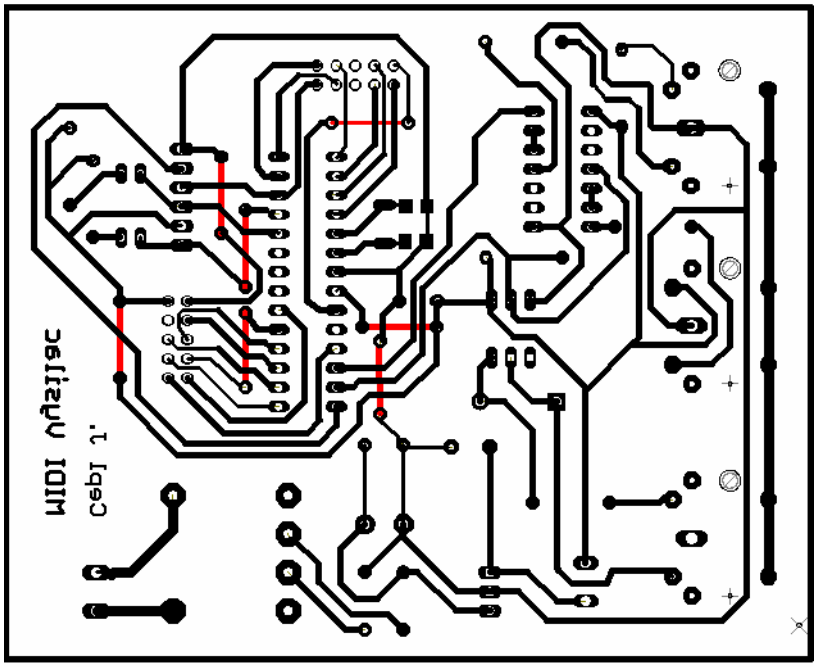
Příloha 1: **Schéma MIDI vysílače (program EAGLE Layout Editor 4.16 Light)**



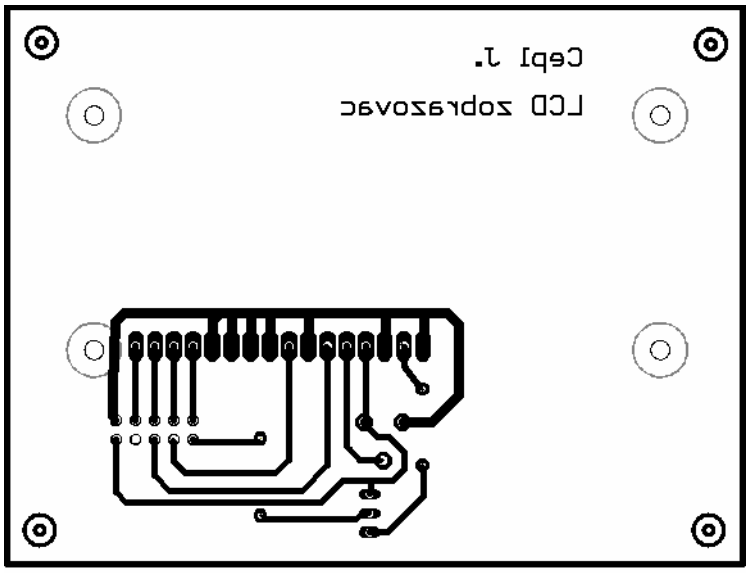
Příloha 2: **Schéma LCD zobrazovače (program EAGLE Layout Editor 4.16 Light)**



Příloha 3: **Deska plošného spoje MIDI vysílače (program EAGLE Layout Editor 4.16 Light, pohled ze strany součástek)**



Příloha 4: **Deska plošného spoje LCD zobrazovače (program EAGLE Layout Editor 4.16 Light, pohled ze strany součástek)**



Příloha 5: Seznam použitých součástek

Označení ve schématu	*Označení součástky*	*Hodnota*	*Popis*
B1	DB107	-	Diodový můstek
C1	CK 100N/63V	100nF	Keramický kondenzátor
C2	CK 100N/63V	100nF	Keramický kondenzátor
C3	CK 100N/63V	100nF	Keramický kondenzátor
C4	CK0603 22P/50V NPO	22pF	SMD kondenzátor
C5	CK0603 22P/50V NPO	22pF	SMD kondenzátor
C6	Jamicon 1000µF 16V	1mF	Elektrolytický kondenzátor
D1	1N 4007_10	-	Polovodičová dioda
D2	LED 5mm Red Point	-	LED dioda červená
D3	LED 5mm 02GT	-	LED dioda zelená
IC1	ATMEL MEGA8-16PU	-	Mikrokontrolér, 28pinů
IC2	7805	-	Stabilizátor 5V
IC3	4069N	-	IO 6 x invertor
IN	DIN 5 P ZP90	-	DIN5 konektor
IO3	SHARP PC900V	-	Optočlen
J1	PSH02-06PG	-	Vidlice 6pin
L1	09P-331K	330µH	Tlumivka
OUT	DIN 5 P ZP90	-	DIN5 konektor
Q1	Q 4MHZ	4 MHz	Krystal 4MHz
R1	RRU 220R, 0207	220Ω	Rezistor 10mm
R2	RRU 300R, 0207	300Ω	Rezistor 10mm
R3	RRU 220R, 0207	220Ω	Rezistor 10mm
R4	RRU 220R, 0207	220Ω	Rezistor 10mm
R6	RRU 220R, 0207	220Ω	Rezistor 10mm
R7	RRU 220R, 0207	220Ω	Rezistor 10mm
SV1	LPV-10	-	Svorkovnice, 10 pinů + protikus
SV2	LPV-10	-	Svorkovnice, 10 pinů + protikus
THRU	DIN 5 P ZP90		DIN5 konektor
TR1	TRHEI202-1X9	0,5VA	Transformátor do DPS
X2	ARK306 2P	-	Svorkovnice 0,5mm
LCD zobrazovač			
LCD1	AMT1602B	-	2 x 16 zobrazovací displej LCD
P1	PT655K002.2	2,5kΩ	Odporový trimr ležatý
R1	MRR 10K, 0207	10kΩ	Rezistor 10mm
R2	RRU 10R, 0207	10Ω	Rezistor 10mm
SV1	LPV-10	-	Svorkovnice, 10 pinů + protikus
T1	BC558B	-	Tranzistor
Ostatní			
-	Paralelní 10žilový kabel	0,5m	-
-	Maticová klávesnice	Velleman	-
-	-	-	Patice DIL 28
-	-	-	Patice DIL 14
-	-	-	Patice DIL 6

Příloha 6: Zdrojový kód hlavního programu

Hlavní program (soubor midi.c)

```
#define F_CPU 4000000

#include <util/delay_basic.h>
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include "usart.h"
#include "lcd_h.h"
#include "midi.h"

volatile byte x,y ;
volatile byte keys;  volatile byte menu ;
 volatile byte compare;
volatile byte counter;
volatile uint16_t my_timer;

char buffer[17];
byte data[30] EEMEM;

int main (void)
{
        byte my_button;

        DDRC = 0xFF;
        DDRD = 0x0F;
        DDRB = 0xFF;

        lcd_init();
        lcd_clrscr();
        light_on();
        USART_Init(MYUBRR);
        TCCR0 |= (1<<CS02);
        TIMSK |= (1<<TOIE0);
        UCSRB |= (1<<RXCIE);
        sei();

while(1)
        { leds_func(leds_on);

        menu = main_menu;
        view_lcd(menu);

        my_button = get_button(1);

        if(my_button == 0)
          { leds_func(red_on);
                menu=choise_PG_CC;
                view_lcd(menu);
                my_button = get_button(1);
                 if(menu==choise_PG_CC)
                        {
                                if(my_button == 0)
PG_setting();
                                 else
                                {
                                if(my_button == 1)
CC_setting();

else menu = main_menu;
                                }
                                                }

                                }
                        else
                                {
                                if(my_button == 1)
                                 _active();
                                else menu = main_menu;
                                }
        }

}
ISR(TIMER0_OVF_vect)
{

my_timer++;

byte  column;
keys = 0;

TCNT0 = INIT_TIMER0;

KEY_OUT=(0b11111110 &
KEY_OUT)|0b00001110 ;
column = KEY_IN|0x0F;

if (column!=0xFF) keys = ~column|0x08;

KEY_OUT=(0b11111101 &
KEY_OUT)|0b00001101 ;
column = KEY_IN|0x0F;
if (column!=0xFF) keys = ~column|0x01;

KEY_OUT=(0b11111011 &
KEY_OUT)|0b00001011 ;
column = KEY_IN|0x0F;
if (column!=0xFF) keys = ~column|0x02;

KEY_OUT=(0b11110111 &
KEY_OUT)|0b00000111 ;
column = KEY_IN|0x0F;
if (column!=0xFF) keys = ~column|0x04;
```

```
if(keys==0) counter++;

        if(counter==20)
                {
                compare=1;
                counter=0;
                }
}

ISR(USART_RXC_vect)
{
USART_Transmit(UDR);
}


void _active(void)
{
        byte temp;

        leds_func(green_on);
        menu=active_mode;
        view_lcd(menu);

while(1)
        {
        temp=get_button(10);
        if(temp==10) break;

        USART_Transmit(eeprom_read_byte(&da
ta[temp*3]));
        USART_Transmit(eeprom_read_byte(&da
ta[temp*3+1]));
        if((eeprom_read_byte(&data[temp*3]) &
0xF0)==0xB0)

        USART_Transmit(eeprom_read_byte(&da
ta[temp*3+2]));

        }
}

void CC_setting(void)
{       byte buffer_CC[4],temp;

        menu=num_channel_CC;
        view_lcd(menu);

        temp=read_byte(15);
        if(temp==0xFF) return;
         else buffer_CC[0]=temp;

        menu++;
        view_lcd(menu);

        temp=read_byte(127);
        if(temp==0xFF) return;
         else buffer_CC[1]=temp;

        menu++;
        view_lcd(menu);

        temp=read_byte(127);
        if(temp==0xFF) return;
         else buffer_CC[2]=temp;

        menu++;
        view_lcd(menu);

        temp=read_byte(9);
        if(temp==0xFF) return;
         else buffer_CC[3]=temp;

        view_lcd(saving);

        eeprom_write_byte(&data[buffer_CC[3]*3
],(buffer_CC[0] & 0x0F)|0b10110000);
        eeprom_write_byte(&data[buffer_CC[3]*3
+1],buffer_CC[1]);
        eeprom_write_byte(&data[buffer_CC[3]*3
+2],buffer_CC[2]);

        my_delay_ms(1000);
}

void PG_setting(void)
{
        byte buffer_PG[3],temp;

        menu=num_channel_PG;
        view_lcd(menu);

        temp=read_byte(15);
        if(temp==0xFF) return;
         else buffer_PG[0]=temp;

        menu++;
        view_lcd(menu);

        temp=read_byte(127);
        if(temp==0xFF) return;
         else buffer_PG[1]=temp;

        menu++;
        view_lcd(menu);

        temp=read_byte(9);
        if(temp==0xFF) return;
         else buffer_PG[2]=temp;


        view_lcd(saving);
        my_delay_ms(1000);

        eeprom_write_byte(&data[buffer_PG[2]*3
],(buffer_PG[0] & 0x0F)|0xC0);
        eeprom_write_byte(&data[buffer_PG[2]*3
+1],buffer_PG[1]);

}

byte read_byte(byte valu)
```

```
{
        uint16_t tem;
        byte tem_2;

while(1)
        {
        tem=get_button(10);
        if(tem==10) return 0xFF;
        change_value(menu,tem);
        tem_2=get_button(11);
        if(tem_2==10) return 0xFF;
        if(tem_2==11) goto jump;
        tem=tem*10+tem_2;

        if(tem>valu) goto jump;
        change_value(menu,tem);

        tem_2=get_button(11);
        if(tem_2==10) return 0xFF;
        if(tem_2==11) goto jump;
        tem=tem*10+tem_2;
        if(tem>valu) goto jump;
        change_value(menu,tem);

        while(get_button(11)!=11);

        jump:if(tem<=valu) break;
                else
                        {
                        lcd_clrscr();

        strcpy_P(buffer,PSTR("Neplatne cislo"));
                        lcd_puts(buffer);
                        my_delay_ms(1000);
                        view_lcd(menu);
                        }
        }

        return (byte)tem;

}

void my_delay_ms(uint16_t tim)
{
        my_timer=0;
        while(my_timer!=(tim/5)){};        }


byte get_button(byte max_number)
{
byte button=0;
byte temp=1;

while(temp)
        {
        if(keys!=0 && compare==1)
                {
                compare = 0;
                switch(keys)
                        {
                case key_0:        button=0;break;
                case key_1:        button=1;break;
                case key_2:        button=2;break;
                case key_3:        button=3;break;
                case key_4:        button=4;break;
                case key_5:        button=5;break;
                case key_6:        button=6;break;
                case key_7:        button=7;break;
                case key_8:        button=8;break;
                case key_9:        button=9; break;
                case key_grid:     button=10;break;
                case key_asterisk: button=11;break;
                case key_A: button=12 ;break;
                case key_B:        button=13;break;
                case key_C:        button=14;break;
                case key_D:        button=15;break;
                                default:break;
                                }
                        if(button<=max_number)
temp=0;
                        }

        }

  return button;

}

void change_value(byte my_menu,byte value)
{

        switch(my_menu)
                {
        case num_channel_PG:
                lcd_gotoxy(0,1);
                char_to_string(value);
                break;
        case num_program_PG:
                lcd_gotoxy(0,1);
                char_to_string(value);
                break;
        case save_un_key_PG:
                lcd_gotoxy(9,1);

                                char_to_string(value);
                                break;
         case num_channel_CC:
                                lcd_gotoxy(0,1);
                                char_to_string(value);
                                 break;
         case num_controler_CC:
                                lcd_gotoxy(0,1);
                                char_to_string(value);
                                 break;
        case value_controler_CC:
                                lcd_gotoxy(0,1);
                                char_to_string(value);
                                 break;
        case save_un_key_CC:
                                lcd_gotoxy(9,1);
```

```
                        char_to_string(value);
                        break;

        default: break;
                }
}


void char_to_string(byte number)
{
        if((number / 100)!=0) lcd_putc('1');
                else lcd_putc(' ');
        if((number / 10)!=0)
lcd_putc((number/10)%10 + 0x30);
                else lcd_putc(' ');
        lcd_putc(number % 10 + 0x30);

}

void view_lcd(byte window)
{
        lcd_clrscr();

        switch(window)
                {
                case main_menu:

        strcpy_P(buffer,PSTR("Progr. rezim
(0)"));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);

        strcpy_P(buffer,PSTR("Aktiv. rezim
(1)"));
                        lcd_puts(buffer);
                        break;

                case choise_PG_CC:

        strcpy_P(buffer,PSTR("Zprava PG
(0)"));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);

        strcpy_P(buffer,PSTR("Zprava CC
(1)"));
                        lcd_puts(buffer);
                        break;

                case active_mode:

        strcpy_P(buffer,PSTR("**Aktivni
rezim*"));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);

        strcpy_P(buffer,PSTR("**********Esc
#"));
                        lcd_puts(buffer);
                        break;

                case num_channel_PG:

        strcpy_P(buffer,PSTR("cislo kanalu:   "));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);
                        break;

                case num_program_PG:

        strcpy_P(buffer,PSTR("cislo programu:
"));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);
                        break;

                case save_un_key_PG:

        strcpy_P(buffer,PSTR("Ulozit pod      "));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);

        strcpy_P(buffer,PSTR("klavesu:        "));
                        lcd_puts(buffer);
                        break;

                case num_channel_CC:

        strcpy_P(buffer,PSTR("cislo kanalu:   "));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);
                        break;

                case num_controler_CC:

        strcpy_P(buffer,PSTR("cislo kontroleru"));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);
                        break;

                case value_controler_CC:

        strcpy_P(buffer,PSTR("hodnota kotrol.
"));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);
                        break;

                case save_un_key_CC:

        strcpy_P(buffer,PSTR("Ulozit pod      "));
                        lcd_puts(buffer);
                        lcd_gotoxy(0,1);

        strcpy_P(buffer,PSTR("klavesu:        "));
                        lcd_puts(buffer);
                        break;

                case saving:
```

```
strcpy_P(buffer,PSTR("*** ukladam ****"));
                                lcd_puts(buffer);
                                lcd_gotoxy(0,1);

        strcpy_P(buffer,PSTR("**********
*"));
                                lcd_puts(buffer);
                                break;

                        default: break;
                        }


}

void light_on(void)
{
PORTD &=(0b11111011);
}

void light_off(void)
{
PORTD |= 0b00000100 ;
}

void leds_func(byte leds)
{
switch(leds)
        {
        case red_on:    PORTB = (PORTB &
0b11011111)|0b00010000; break;
        case green_on:  PORTB = (PORTB &
0b11101111)|0b00100000;         break;
        case leds_on:   PORTB = PORTB &
0b11001111;                     break;
        default: break;
        }
}
```

Konstanty a definice funkcí (soubor midi.h)

```
typedef unsigned char byte;

#define INIT_TIMER0 172
#define KEY_OUT         PORTB
#define KEY_IN PIND

#define key_0   0x28
#define key_1   0x11
#define key_2   0x21
#define key_3   0x41
#define key_4   0x12
#define key_5   0x22
#define key_6   0x42
#define key_7   0x14
#define key_8   0x24
#define key_9   0x44
#define key_A   0x81
#define key_B   0x82
#define key_C   0x84
#define key_D   0x88
#define key_grid 0x48
#define key_asterisk        0x18


#define main_menu                       0
#define  active_mode                    1
#define  choise_PG_CC                   2
#define  num_channel_PG                 3
#define  num_program_PG                 4
#define  save_un_key_PG                 5
#define  num_channel_CC                 6
#define  num_controler_CC               7
#define  value_controler_CC             8
#define  save_un_key_CC                 9
#define  saving                         10

#define red_on   0
#define green_on 1
#define leds_on  2

void light_on(void);
void light_off(void);
void view_lcd(byte);
void char_to_string(byte);
void change_value(byte ,byte );
void leds_func(byte );
void _active(void);
void PG_setting(void);
void CC_setting(void);
void my_delay_ms(uint16_t );
byte get_button(byte );
read_byte(byte);
```