Středoškolská technika 2009

Setkání a prezentace prací
středoškolských studentů na ČVUT

Zabezpečovací zařízení

Filla Jakub – Vojtášek Jindřich

SŘEDNÍ PRŮMYSLOVÁ ŠKOLA SDĚLOVACÍ TECHNIKY
Praha1, Panská 3

Anotace

Lidé měli odnepaměti potřebu chránit své vlastnictví. I my jsme se rozhodli vytvořit vhodný zabezpečovací systém, který ochrání byt, auto, chalupu nebo i kůlnu. Zvládne informovat nejen o přítomnosti cizích osob, ale i rozlišit osoby oprávněné k přístupu pomocí identifikačních čipů. Předpokladem snadného užívání je uživatelsky přívětivé rozhraní, a to nejlépe pomocí displeje. Systém je také schopný provozu po určitou dobu bez přívodu elektrické energie z veřejné sítě, kdy potřebnou energii dodává záložní zdroj.

Annotation

People have always had needs for protection of their own property. We have also decided to create suitable security system, which can protect a flat, a car, a house or even a shed. It can inform not only about presence of strange people, but even recognize people who are allowed to enter using identification chips. Assumption of easy usage is user friendly interface, preferably by display. The system is also able to work without power supply for certain period of time, when needed energy is provided by backup power source.

Obsah

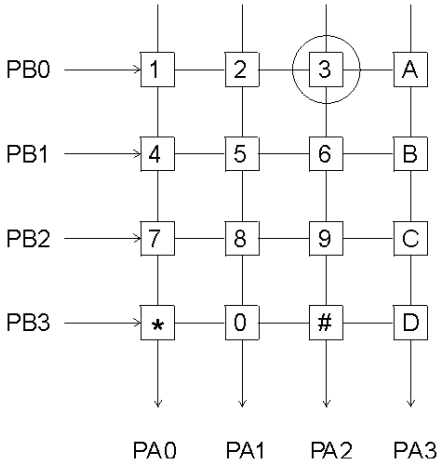
1 Úvod

Bezpečnostní systém, jak už název napovídá, musí ochránit určitý prostor před vniknutím nežádoucích osob. V hlídaném prostoru budou umístěna čidla, která budou schopna odhalit přítomnost osob či zvířat. Zaznamená-li čidlo přítomnost osoby, vyšle signál do řídící jednotky a ta spustí alarm. Vycházíme z předpokladu, že objekt není třeba střežit v případě, že se v něm nebo v jeho okolí nachází alespoň jeden z uživatelů. Je-li systém aktivní, první příchozí uživatel ho deaktivuje přiložením čipové karty a zadáním osobního hesla. V tom okamžiku bude možné se po objektu volně pohybovat. Uživatel, který opustí objekt jako poslední, čidla opět pomocí své karty a hesla aktivuje.

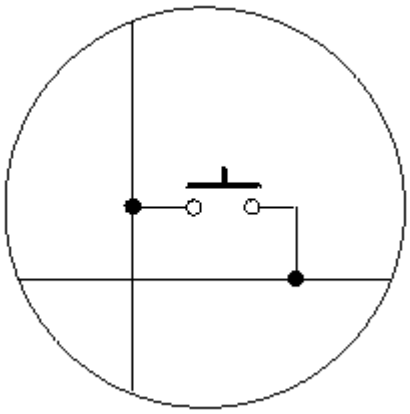
2 Hardware

2.1 Klávesnice

Pro ovládání systému jsme si vybrali maticovou klávesnici 4x4, která šetří v našem případě datové vodiče u procesoru oproti samotným tlačítkům. Potřebujeme pouze osm (při použití maticové klávesnice) datových vodičů místo šestnácti (při použití jednotlivých tlačítek). Klávesnice obsahuje čtyři vývody pro výběr řádku a čtyři vývody pro výběr sloupce (obr. 1). Při stisknutí tlačítka „3“ se spojí vodiče mezi vývody PB0 a PA2 a vytvoří vodivou cestu (obr. 2). Obslužný program poté vyhodnotí, ve kterém sloupci je stisknuto tlačítko, uloží tuto informaci a porovná s každým řádkem. Po této proceduře je jasné, jaké z těchto šestnácti tlačítek bylo vybráno. Každé jedno tlačítko má přiděleno v obslužném programu binární kód, s kterým se dále pracuje.



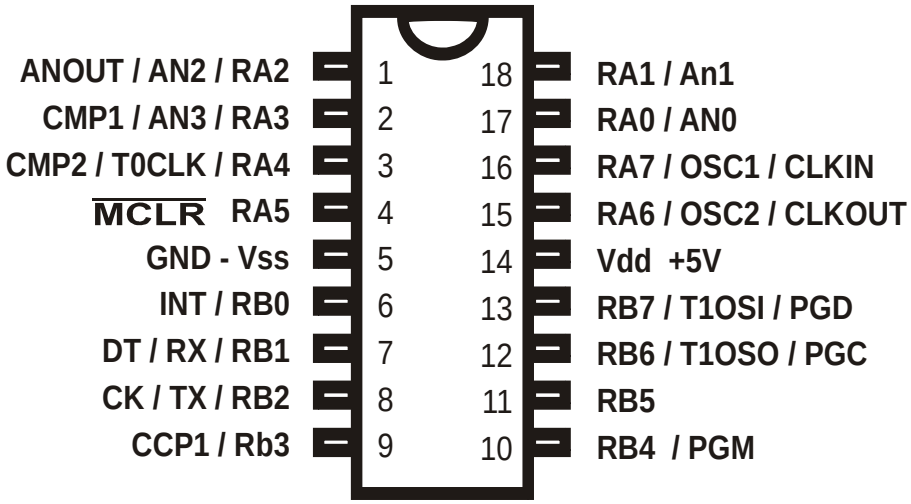
Obrázek 1 - Schéma maticové klávesnice



Obrázek 2 – Detail tlačítka

2.2 Mikroprocesor

Řídící jednotkou celého systému je jednočipový mikroprocesor PIC16F648A od firmy Microchip. Řadič funguje podle strojového kódu, bez kterého nic nevykonává. Kontrolér se programuje v jazyku Assambler, jež je považován za nejjednodušší programovací jazyk. Instrukce zapisujeme tak, jak předem definoval výrobce a poté zkompilujeme zdrojový kód do strojového, který následně nahrajeme do mikroprocesoru.



Obrázek 3 - Význam výstupů

U zvoleného procesoru jsme využili zejména tyto funkce:

- 35 jednoslovných instrukcí
- 16 vstupních/výstupních vývodů se zatížením 25mA
- 2 osmibitové čítače
- 1 šestnáctibitový čítač
- UART (sériová komunikace)
- mnoho zdrojů přerušení
- 4096 x 14 programové paměti
- 256 bajtů datové RAM paměti
- 256 bitů EEPROM paměti

2.2.1 Konfigurace a využití portů

2.2.1.1 PIC – Hlavní

TRISA - E0h

RA0-RA3 - datová sběrnice pro klávesnici (příjem)
RA4 - povoluje převzetí čísla z RA0 – RA3
RA5 - čidlo
RA6 a RA7 - krystal 11,059 MHz

TRISB - 00h

RB0-RB3 - datová sběrnice pro displej
RB4 - R/W
RB5 - RS
RB6 - E
RB7 - alarm

V závislosti na části programu může být RB1 vstupní, nebo výstupní. Kdykoli je potřeba použít blok UART musí TRISB nastavit na hodnotu 02h (tzn. vývod RB1 je vstupní).

2.2.1.2 PIC – Klávesnice

TRISA - 00h

RA0-RA3 - datová sběrnice pro klávesnici (vysílání)
RA4-RA5 - neobsazeno
RA6 - frekvence pro zvuk tlačítek
RA7 - povolení převzetí čísla po datové sběrnici

TRISB - F0h

RB0-BR3 - sloupce
RB4-RB7 - řádky

2.3 Displej

Pro zobrazování informací o stavu systému slouží znakový displej MC2004E-SYL s řadičem HD44780, který zobrazuje v rozlišení 4x20 znaků. Každému znaku je přidělena ASCII hodnota. S touto hodnotou pracujeme, chceme-li na displej přenést určitý znak.

Náš znakový displej má celkem 16 vývodů (obr. 4):

- 8 datových
- 3 ovládací
- napájení
- zem
- řízení kontrastu (trvale zapojen)
- 2 podsvětlení (anoda, katoda)

Vývod	Název	Funkce
1	Vss	Zem
2	Vcc	Napájení
3	Vee	Regulace jasu
4	RS	volí mezi instrukcemi (0) a daty (1)
5	R/W	volí mezi čtením (0) a zápisem (1)
6	E	hodinový vstup (zapisuje se na kladný impuls)
7	D0	data 0 (při 4bitové komunikaci uzemněn)
8	D1	data 1 (při 4bitové komunikaci uzemněn)
9	D2	data 2 (při 4bitové komunikaci uzemněn)
10	D3	data 3 (při 4bitové komunikaci uzemněn)
11	D4	data 4
12	D5	data 5
13	D6	data 6
14	D7	data 7
15	LED+	anoda podsvětlení
16	LED-	katoda podsvětlení

Obrázek 4 - Význam jednotlivých vývodů

Před samotným odesíláním znaků, je třeba displej inicializovat. To znamená, že je třeba odeslat soubor instrukcí (příloha 5) v určitých časových intervalech.

Pro komunikaci mezi displejem a zařízením (u nás PIC), lze volit mezi 4bitovou, nebo 8bitovou sběrnicí. My, vzhledem k úspoře vývodů, jsme využili komunikaci 4bitovou. Tento typ komunikace sice šetří datové vodiče, ale čas potřebný k přenesení jednoho znaku na displej je dvojnásobný.

2.4 Čtečka karet

Naším druhým vstupním zařízením je čtečka bezkontaktních karet s čipem RFID. V našem případě se jedná o typ Netronix CTU-005 TTL.

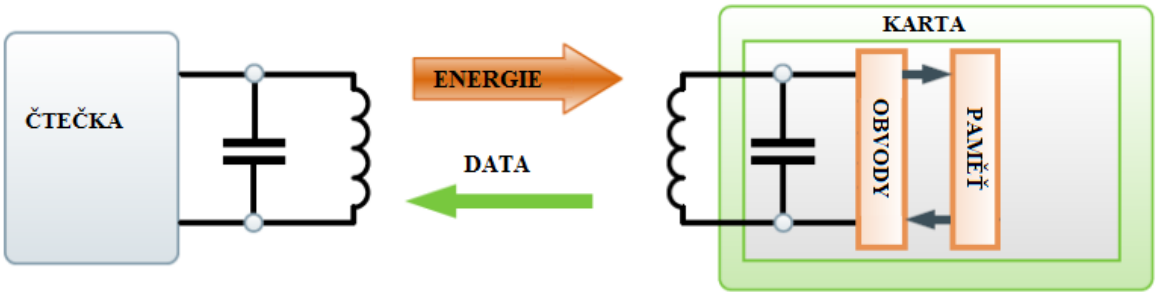
Čtečka má celkem 7 vývodů:

- Napájení
- Zem
- vývod na LED, která značí připojení k napájení
- vývod na LED, která značí úspěšné odeslání dat z čipu
- pomocný reproduktor
- TX – sériový výstup
- RX – sériový vstup

My jsme nevyužili pomocný reproduktor (jelikož čtečka sama jeden obsahuje a ten je pro naše účely vyhovující), RX a vývod na LED pro informaci o napájení.

Celé zařízení je pasivní snímač využívající nosnou frekvenci 125kHz. Princip čtečky by se dal shrnout v těchto bodech:

1. Čtečka vysílá na nosném kmitočtu elektromagnetickou vlnu, kterou karta následně přijme.
2. Indukované napětí vyvolá elektrický proud, který je usměrněn a nabíjí kondenzátor na kartě.
3. Jakmile dosáhne kondenzátor minimální potřebné úrovně napětí, spustí se řídící obvody uvnitř karty a karta začne odesílat.
4. Odrazy, které vznikají změnou impedance antény (na kartě), jsou poté čtečkou vyhodnocovány jako logické jedničky a nuly.

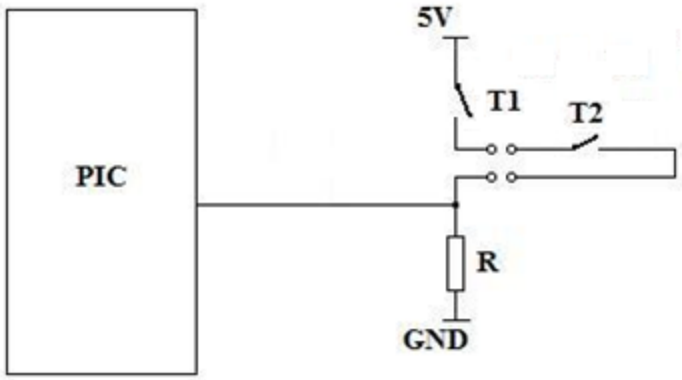


Obrázek 5 - Schéma funkce čtečky

Procesor na čtečce následně data zpracuje a odešle sériově rychlostí 9600 Baudů.

2.5 Čidlo a alarm

Naším cílem bylo zhotovit zabezpečovací systém, který si umí poradit s běžně dostupnými čidly. Většina jich funguje jednoduše - rozepne kontakt, pokud čidlo zaznamená pohyb (viz obr. 6). My jsme proto použili pouze rozpínací tlačítka, kterými funkci čidla simulujeme. První (T1) je připevněno tak, aby se kontakt rozepnul při otevření skříně, druhé je připojeno pomocí konektorů a vyvedeno ven. Program je samozřejmě schopný pracovat se skutečným čidlem.



Obrázek 6 - Schéma připojení čidla

Stejně jako čidlo je náš alarm spíše provizorní. My jsme zvolili obyčejný piezo reproduktor. Jednak kvůli rozměrům a také proto, že většina hlasitých alarmů je drahá a pro účely prezentace nevhodná.

3 Software

Jak už je uvedeno v sekci Mikroprocesor, obslužný program je psán v jazyce Assembler. Zařízení obsahuje dva PIC16F648A, přičemž jeden je určen pro obsluhu maticové klávesnice a druhý se stará o zbytek. Popis každého řádku programu by byl zbytečně dlouhý, a proto se budeme soustředit na stěžejní části a problémy. Ty by se daly rozdělit do několika kapitol, jsou to:

- klávesnice
- paměť RAM
- paměť EEPROM
- displej a jeho obsluha
- čtečka karet a UART
- čidlo
- alarm

Pro jednodušší porozumění řídícímu programu je nutné si ho rozdělit na bloky nastavení a samotný provoz. Na příloze 2 je vidět zjednodušený vývojový diagram obou částí. Sekce nastavení je znázorněna na příloze 3. Jak pracuje samotný provoz, je možné zjistit z přílohy 4.

Program pracuje následovně. Po spuštění se ověří, je-li v paměti alespoň jedna karta. Pokud ano, program skočí do samotného provozu. Pokud ne, dostaneme se do sekce nastavení, o ní ale později. V tento okamžik je čidlo neaktivní a čeká na aktivaci od jakéhokoli uživatele systému. Ověřování osob probíhá pomocí RFID karty a osobního hesla. Uživatel je vyzván k přiložení karty, ta se následně ověří. Není-li v paměti, má uživatel ještě 4 pokusy, aby přiložil správnou kartu (uloženou v paměti), jinak se spustí alarm. Je-li karta v paměti, je jasné o koho jde a jaké má heslo. Po třetím zadání špatného hesla se spustí alarm. Shoduje-li se heslo s uloženým, uživatel aktivoval čidlo (hlídá). Zaznamená-li čidlo přítomnost nežádoucí osoby v hlídaném prostoru, spustí alarm. Pro odblokování (deaktivaci) čidla je nutné opět přiložit kartu a ověřit jí heslem. Po této proceduře se program opět vrací na začátek. Tedy čeká na přiložení karty. Z této části je možnost odskočit do nastavení zmáčknutím křížku. Zde jsou dvě možnosti, vrátit se nebo pokračovat do nastavení zadáním PUK.

V nastavení lze přidat nebo odebrat uživatele. Přidávání je podobné jako ověřování. Uživatel přiloží kartu a zvolí heslo. Zde se mu přidělí ID, pomocí kterého je identifikován v případě spuštění alarmu. Nyní se uživatel může rozhodnout, jestli chce přidat dalšího uživatele. Když ne, program skočí do provozu. Další funkcí sekce nastavení je odebrání uživatele. Jednotlivý uživatel se maže pomocí ID. Je zde i možnost smazání celé paměti. Je-li po těchto procedurách v paměti alespoň jedna karta, program se na příkaz uživatele přepne do provozu.

Po spuštění alarmu si program vyžádá PUK. Po správném zadání se vrátí zpátky do provozu.

3.1 Klávesnice

3.1.1 Dekódování

Maticová klávesnice má osm vývodů. Všech osm je připojeno na PORTB s tím, že RB0 až RB3 odpovídá vývodům pro výběr sloupce a RB4 až RB7 vývodům pro výběr řádku. Výsledek se zobrazuje na RA0 až RA4. Důležitou funkcí, kterou jsme zde využili, je zapnutí pull-up rezistorů. Uvnitř procesoru jsou zdroje proudu, které „táhnou“ vstupní bity na 5V. V praxi to znamená, že pokud je vývod ponechán ve vzduchu (nezapojený), objeví se na něm logická jedna a v případě uzemnění se objeví logická nula. Porty RB0 až RB3 jsou nakonfigurovány jako výstupní a RB4 až RB7 jako vstupní.

Abychom identifikovali stisknuté tlačítko, vysíláme neustále dokola postupně logickou nulu na každý vodič, který představuje sloupec, a zároveň testujeme všechny řádky. Takto dokážeme určit pozici tlačítka, kterou program vyhodnotí a odešle ji do hlavního procesoru. Znaky A, B, C, D, # a * využíváme k ovládání menu, popřípadě k různým příkazům.

```
;prvni sloupec
                movlw   B'11111110'
                movwf   PORTB
                btfss   PORTB, 4
                goto    jedna
                btfss   PORTB, 5
                goto    ctyri
                btfss   PORTB, 6
                goto    sedm
                btfss   PORTB, 7
                goto    hvezdicka
;druhy sloupec
                movlw   B'11111101'
                movwf   PORTB
                btfss   PORTB, 4
                goto    dva
                btfss   PORTB, 5
                ….
```

Program 1 – Detekce stisku

Zde je naznačena detekce stisku. Nejprve se pošle pouze na RB0 logická nula (do prvního sloupce). Při sepnutí například čísla 1 se na kontaktu spojí námi vyslaná nula na RB0 s pull-upem na RB4, což se vyhodnotí jako logická nula. Poté se testuje na RB4 až RB7 výskyt logické nuly.

Pokud se neobjeví, znamená to, že stisknuté tlačítko je v jiném sloupci a program skočí do další části, kde odešle nulu na RB1 (do druhého sloupce). Testovací rutina je pro všechny sloupce stejná, pouze na konci (v případě, že není nic stisknuto) je odesláno 0FFh.

Pokud se objeví nula, program jí zaznamená a skočí na návěští určeného znaku. Zde se připraví hodnota stisknutého tlačítka na odeslání do řídící jednotky.

3.1.2 Odeslání čísla

Máme číslo připravené k odeslání. K jeho bezchybnému přenesení je potřeba ošetřit zákmity a několikanásobné přečtení čísla. Problém se zákmity vyřeší přibližně 50 µs trvající čekací smyčka, do které program skočí ihned po zápisu přenášeného čísla na PORTA. Po 50 µs je na vývodech RA0 - RA3 stabilní číslo. Vyšleme tedy přes řídící vodič signál, že je číslo připraveno k převzetí. Jedná se o RA7, ten musí být v tomto případě v logické jedna. Když uživatel drží tlačítko, je na vývodech RB4 - RB7 hodnota různá od Fh. Až při uvolnění tlačítka se na RB4 - RB7 objeví Fh, což je signálem, že je stisk dokončen a na RA7 se objeví logická nula. To je signálem pro hlavní PIC, že může čekat na další číslo. V hlavním PIC se přijatá čísla ukládají do paměti RAM a dále se s nimi pracuje.

Různý zvuk tlačítek obstarává přerušení od čítače TMR0. Konstanta, symbolizující určitý tón, je nahrávána v závislosti na stisknutém tlačítku. V přerušení se invertuje RA6, díky tomu máme na zmíněném vývoduu obdélníkový signál.

Pro výpočty frekvencí jsme použili program TonReg, který po dosazení hodnoty komorního a (440Hz) dopočítá ostatní tóny.

3.2 Paměť RAM

Do této paměti se ukládají data, která jsou dočasně potřeba pro chod programu - při odpojení napájení jsou ztracena. Pro přehledné ukládání dat je potřeba si každou používanou osmibitovou buňku v paměti pojmenovat. Definování vlastních registrů vypadá takto:

„jméno registru“ equ „adresa buňky“

3.2.1 Registry a jejich funkce

TMP, TMP_2

Dočasné registry sloužící k uložení dat potřebných pro složitější operace.

N_TMP

Obsahuje příchozí znak z klávesnice, po každém stisknutí se přepisuje.

N1, N2, N3, N4

Do těchto registrů se v případě výzvy k zadání hesla uloží příchozí znaky z klávesnice.

STAT_0

Povoluje a zakazuje skoky do podprogramů. Funkce každého bitu je popsána v příloze 6.

ERREG

Zvyšuje se o jedna pokaždé, když uživatel zadá špatné heslo. Maximální počet špatně zadaných hesel v řadě je tři. Když je hodnota v registru rovna třem, inkrementuje se hodnota v registru TERREG.

TERREG

Je-li v tomto registru hodnota různá od nuly, spustí se alarm.

COUNT, COUNT2, ODPOCET1, ODPOCET2

Počítací registry se používají pokaždé, když je v programu nutno provést tutéž rutinu několikrát.

CRD_N1, CRD_N2

Nachází uplatnění v části programu, kde se porovnávají DATA karty, která byla právě přiložena, s DATA karet uložených v paměti EEPROM. V CRD_N1 je hodnota z paměti EEPROM a CRD_N2 obsahuje hodnotu právě načtené karty.

CRD_ERR

Počítá chyby při porovnávání karet.

INT_DEP

Po skoku do přerušení slouží tento registr jako záloha WREG.

EEPROM_ADR

Když program zapisuje do EEPROM, zde je adresa, na kterou se znak zapíše.

CRD

Počítá přiložené karty, které nejsou v paměti. Když je v tomto registru hodnota 05h, spustí se alarm.

CRD_CNT

Při porovnávání právě přijaté karty s kartami v paměti se kontroluje několika bajtové číslo. Po každém zkontrolovaném bajtu se CRD_CNT sníží o jedna.

DSP, DSP_2

Vysílá-li se znak na displej, nachází se nejdříve zde, poté se vyšle.

D_BFLAG

Doslova říká. Teď je možné poslat znak na displej. Je volno.

HEX, DEC_1, DEC_10, DEC_100, SMAZ

Pomocné registry, které slouží pro převod z hexadecimální soustavy na desítkovou a naopak.

DSP_ADR

Hodnota, která po odeslání do displeje určí pozici kurzoru.

3.3 Paměť EEPROM

U PICu slouží paměť EEPROM (do konce podkapitoly 3 jen jako paměť) zejména pro uchování dat i po odpojení napájení. Této skutečnosti můžeme využít pro uložení karet a k nim náležících hesel. Další, ale ne méně významnou funkcí, jsou kontrolní a funkční registry. Jejich funkce jsou popsány níže. Na mapě paměti (příloha 4) jsou vidět adresy jednotlivých registrů.

DATA

Po přidání karty se do těchto registrů uloží unikátní pětibajtové identifikační číslo čipové karty.

N1 – N4

Když program uloží kartu do paměti, tak vyzve uživatele k zadání uživatelského hesla. To se ukládá právě zde. První zmáčknuté číslo se nachází v N1, poslední v N4.

CTRL

Zde se nachází adresa, na kterou se uloží následující karta. Po přidání karty se inkrementuje o 10h.

INIT

Po prvním korektním uložení karty a hesla do paměti se sem uloží hodnota 0x01.

CNT

Obsahuje počet celkově přidaných karet, nehledě na přepisování paměti (nuluje se pouze při mazání celé paměti). Pomocí tohoto registru se generuje ID.

ID

Slouží pro identifikaci uživatele. Každému novému uživateli je přiděleno pořadové číslo.

LL

Po každém zamknutí se do tohoto registru uloží ID uživatele, který systém zamknul.
V případě spuštění alarmu se zobrazí na displej.

3.3.1 Zápis

Na prog. 2 je vidět, jak se data, která jsou před voláním podprogramu EEPROM_zapis v registru WREG, uloží do registru EEDATA na adresu, jejíž hodnota se nachází v registru EEADR. Hodnotu do EEADR je nutné nastavit před skokem do podprogramu. Po vyslání zapisovací sekvence program testuje bit EEIF. Pokud je v něm logická jedna, podprogram se ukončí. V našem případě se negeneruje přerušení - je zakázáno bitem GIE.

```
EEPROM_zapis
        bsf             STATUS, 5
        movwf           EEDATA
        bcf             STATUS, 5
        movf            EEPROM_ADR, 0
        bsf             STATUS, 5
        movwf           EEADR
        bsf             EECON1, 2
        movlw           55h
        movwf           EECON2
        movlw           0AAh
        movwf           EECON2
        bsf             EECON1, 1
        bcf             STATUS, 5
EEPROM_zapis_1
        btfss           PIR1, 7
        goto EEPROM_zapis_1
        bcf             PIR1, 7
        return
```

Program 2 - Syntaxe pro zápis do paměti EEPROM

3.3.2 Čtení

Čtení z paměti EEPROM je poněkud jednodušší. Stačí nastavit adresu, ze které chceme číst do registru WREG. Ta se následně přepíše do EEADR. Vyšle se požadavek na čtení. Dále se pouze přečte EEDATA a přepíše se do WREG – v něm je hodnota, kterou jsme chtěli z paměti přečíst.

```
EEPROM_precist
        bsf             STATUS, 5
        movwf           EEADR
        bsf             EECON1, 0
        movf            EEDATA, 0
        bcf             STATUS, 5
        return
```

Program 3 - Syntaxe pro čtení z paměti EEPROM

3.3.3 Mazání celé paměti

Mazání paměti není nic jiného než zapisování hodnoty 0FFh do buňky, kterou chceme smazat. Chceme-li smazat celou paměť, stačí do každé její buňky zapsat tuto hodnotu. V praxi to vypadá, že se smaže první buňka a EEADR se zvýší o jedna. Tak to pokračuje do té doby, než je hodnota v EEADR rovna celkové velikosti paměti. Pokud nastane tento případ, procesor se resetuje.

```
EEPROM_vymazat
        call        LCDSMAZ
        call        EEPROM_mazani
        clrf        EEPROM_ADR
EEPROM_vymazat_0
        movlw       0FFh
        call        EEPROM_zapis
        movf        EEPROM_ADR
        sublw       0FFh
        btfsc       STATUS, 2
        goto        reset
        incf        EEPROM_ADR, 1
        goto        EEPROM_vymazat_0
```

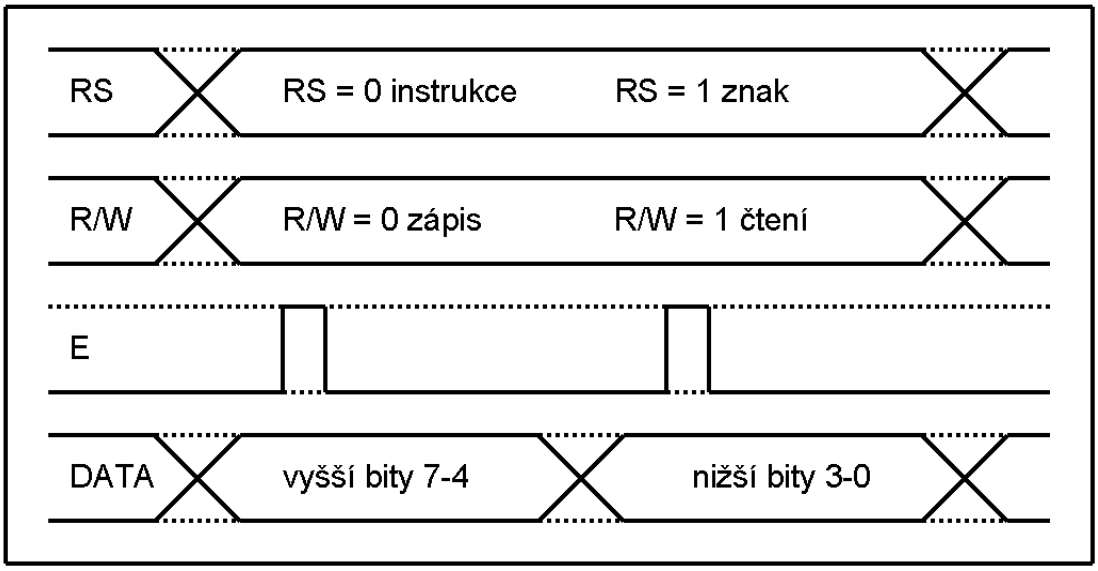
Program 4 - Mazání EEPROM

3.3.4 Mazání jednotlivého uživatele

Princip smazání jednotlivého uživatele by se dal nastínit takto. Všechny registry, které souvisí s uživatelem (mazaným) se smažou a nahradí se registry uživatele, který zabírá část paměti, jejíž počáteční hodnota je o 0x0F větší. Tím se mezera po smazaném uživateli posune o jeden řádek níž (viz příloha 5). Toto se provádí, tak dlouho než se ona smazaná část posune do prostoru paměti, která není využívána. V případě plného obsazení na konec. Dále se musí snížit registr CTRL. Taktéž je zapotřebí zvýšit hodnotu v registru COUNT, který je nezbytný pro generaci ID.

3.4 Displej a jeho obsluha

Většina znakových LCD používá řadič HD44780, patří mezi ně i MC2004E-SYL, který jsme použili. Kvůli úspoře vývodů byla zvolena 4bitová komunikace.



Obrázek 7 – Graf komunikace s displejem

3.4.1 Čtení

Využíváme-li možnosti čtení z displeje, jedná se pouze o testování „Busy flagu“ (dále BF). V tomto případě je RS v logické nule a R/W v logické jedničce. Čtení se odstartuje kladným pulzem na E. Po přečtení se testuje stav na DB7. Je-li v logické jedničce, značí probíhající proces v displeji, není tedy možné do něj cokoli posílat. Podprogram BUSYFLAG tedy čeká až je DB7 v logické nule. Když je, tato část se ukončí.

3.4.2 Zápis instrukce

Instrukce lze zapisovat takto. Zkontrolujeme BF, podle kterého řídící jednotka zjistí, jestli je možno na displej posílat další data. Je-li tomu tak, RS a R/W se nastaví do logické nuly a kladným pulzem na E se zapíší vyšší 4 bity, poté následují i nižší 4 bity. V našem programu jsme používali především smazání displeje a posunutí kurzoru (nastavení počáteční adresy v DDRAM). Pro zápis instrukcí slouží podprogram LCDPRIKAZ - do WREG se nahraje binární vyjádření instrukce a ta se vyšle.

3.4.3 Zápis znaku

Zapisuje-li se znak, RS je v logické jedničce a R/W v nule. Po testu BF se opět zapíší vyšší 4 bity a odešlou se kladným pulzem na E. Hned po nich následují nižší 4 bity, které se také odešlou kladným pulzem na E. Na displej zapisujeme podprogramem LCDCHAR, ASCII hodnota se nahraje do WREG a zavolá se zmíněný podprogram.

3.4.4 Komunikace a inicializace

Jako první věc po zapojení napájení je třeba displej inicializovat. Během inicializace se nastaví typ komunikace, směr psaní znaků, pozice kurzoru a blikání kurzoru. My, jelikož jsme zvolili komunikaci 4bitovou, musíme první instrukci vyslat jednou v 8bitové (displej je tak nastaven po zapojení napájení) a následně už ve čtyřbitové komunikaci. Odešleme tedy všechny potřebné instrukce (viz Zápis instrukce) a tím pro nás inicializace končí.

Samotná komunikace probíhá následovně. Po nastavení RS a R/W se osmibitové číslo se rozdělí na vyšší a nižší bity. Vyšší bity je možné odeslat ihned, ale nižší je potřeba přesunout na místo vyšších. Lze použít funkci LLF, RRF nebo SWAPF, my jsme použili SWAPF (převrátí horní a dolní bity v registru). V tomto bodě už nic nebrání odeslání nižších bitů - opět kladným pulzem na E.

3.4.5 Podprogramy

Pokaždé vypisovat hodnotu instrukce nebo znaku do WREG, by bylo poněkud neefektivní. Proto jsme v programu vytvořili několik dalších podprogramů, které práci s displejem ulehčí. Vzhledem k tomu, že používáme většinou jenom dvě instrukce, nebyl problém vytvořit každé z nich vlastní podprogram. Podprogram ADR změní pozici kurzoru na displeji. LCDSMAZ smaže displej a posune kurzor na začátek DDRAM. U zápisu znaků bylo nutné se uchýlit k většímu zjednodušení. Chceme-li zapsat znak na displej, postará se o to funkce CALL XX s tím, že X je samotný znak. Když je potřeba zapsat A, stačí napsat CALL AA. Na návěští XX, je vypsaná ASCII hodnota znaku, ta se přesune do WREG a potom se zavolá výše zmíněný LCDCHAR.

3.5 Čtečka karet

Čtečka CTU 005 TTL od firmy Netronix, kterou jsme vybavili náš systém, odesílá po každém přijetí karty 11bajtový rámec (viz obr. 8).

1 bajt	1 bajt	1 bajt	5 bajtů	1 bajt	2 bajty
0x01	délka rámce	0x01	DATA	0xFF	kontrolní součet

Obrázek 8 – výstupní rámec čtečky

Po mnohonásobném testování se ukázalo, že není třeba využít kontrolní součtu, jelikož čtení karet probíhá bezchybně. Délka rámce a ostatní konstantní části rámce, vyjma prvního bajtu, už po přijetí nepoužíváme – pro naše účely jsou redundantní.

3.5.1 UART

Pro komunikaci čtečky a ovládacího programu jsme využili blok UART. Čtečka vysílá data rychlostí 9600 Bd, tuto rychlost je nutné nastavit v řídícím programu. Větší přesnost jsme zajistili použitím krystalu 11,0592 MHz. Je-li zapotřebí přijmout kartu, zapne se přerušení, a když uživatel přiloží kartu, odskočí program na adresu 004. Zde se ukáže, je-li přerušení skutečně vyvoláno příjmem UART. Když ano, otestuje se první bajt na hodnotu 0x01. Pokud je první příchozí bajt skutečně 0x01 (viz obr. 8), začne se rámec zapisovat do paměti RAM. Zde se s ním dále pracuje. Právě přijatý rámec se ukládá na adresy 40h – 4Bh. Po přijetí celého rámce se UART opět připraví pro příjem.

3.6 Čidlo a alarm

V programu není čidlo nic jiného než podprogram, který testuje RA5. V závislosti na části programu se čidlo buď testuje, nebo přeskakuje (viz prog. 5). Jestli program čidlo přeskočí, závisí na bitu ONOFF v registru STAT_0 – na logickou nulu přeskakuje a na jedničku snímá.

```
cidlo
        btfss       STAT_0, 0
        return
        btfsc       PORTA, 5
        return
        goto        cidlo_block
```

Program 5 – testování portu RA5

Výstup pro alarm se nachází na vývodu RB7. Zde je možné program konfigurovat podle požadavků uživatele. V případě potřeby zde může být logická jedna, nula, frekvence, nebo frekvence s PWM (např. pro ovládání servomotoru). My jsme zvolili frekvenci přibližně 337 Hz.

4 Oživení

4.1 Návrh desky plošných spojů

Základem pro výrobu desky bylo navrhnutí schématu a desky v programu EAGLE 5.3. Pro snadnější upevnění do krabičky, jsme rozdělili systém na 2 desky.

4.1.1 Signalizační lišta

Tato malá část desky zjednodušuje přístup k jednotlivým napájením a obsahuje:

- relé RM40
- 2 signalizační LED (zelená a červená, včetně ochranných odporů)
- usměrňovací diodu 1N4007
- zenerova dioda BZX55
- 9V stabilizátor 7809
- 5V stabilizátor 7805
- 2 vypínače
- čtyři keramické nepolarizované kondenzátory 100nF
- konektor pro připojení signalizace čidla z PIC
- konektor pro připojení akumulátoru
- konektor pro připojení napájení ze zdroje
- konektory pro připojení napájení hlavní desky, čidla a čtečky
- konektory pro případné další potřebné napájení 5V

Vstupní napětí může být až 30V, které je stabilizátorem sníženo na 9V. Poté následuje zelená LED, signalizující připojení síťového adaptéru a usměrňovací dioda (slouží především k úbytku napětí). V této části je napětí 8,2V, kterým nabíjíme baterii vedenou na 5V stabilizátor. Relé, připojené ihned na začátku, přepíná mezi červenou a zelenou LED. Pokud se odpojí síťový adaptér, relé rozpojí první kontakt a rozsvítí se červená. Ta se v případě nízkého napětí na baterii uzavře zenerovou diodou a zhasne, což znamená, že baterie je téměř vybitá a systém se zanedlouho vypne. Za 5V stabilizátorem jsou už pouze konektory na napájení čtečky, desky a signálu pro čidlo. Mezi stabilizátory jsou ještě umístěny dva vypínače. Jeden je vyvedený na skříň a druhý je přímo na desce. Ten při přepnutí slouží k trvalému zapnutí systému (nelze jej vypnout vypínačem umístěným na boku skříně).

Výpočet ochranných rezistorů:

$$R_{zel} = \frac{9 - 1,9}{0,02} = 355\Omega$$

$$R_{cer} = \frac{8,2 - 1,6}{0,02} = 330\Omega$$

4.1.2 Hlavní deska

Osazení hlavní desky je následující:

- dvě patice - 18 vývodů (pro mikroprocesory)
- dva keramické nepolarizované kondenzátory 22pF
- krystal 11,059 MHz
- piezo reproduktor s ochranným odporem
- konektor pro připojení alarmu
- konektor pro připojení čidla
- konektor pro připojení výstupu z čtečky
- konektor na výstup z UARTu
- čtyři konektory pro připojení displeje (4x4 vývody)
- dva konektory pro připojení klávesnice (2x4 vývody)
- konektor se zámkem pro připojení napájení

4.2 Výroba desky plošných spojů

4.2.1 Osvícení a leptání

Návrhy hlavní desky a signalizační lišty jsme nejprve zrcadlově vytiskli na transparentní fólii. Poté jsme pomocí osvěcovače (upravený skener s UV lampami) osvěcovali fólii, po dobu dvou minut, na fotocitlivý plošný spoj. Takto osvětlená deska byla ponořena do roztoku hydroxidu sodného (1,5%) pouze na pár vteřin, aby došlo k odstranění vrchní fotocitlivé vrstvy. Po opláchnutí jsme umístili plošný spoj do lázně s chloridem železitým tak, aby měděná fólie byla směrem dolů. Leptali jsme, dokud nebyl návrh vidět skrz. Nakonec jsme desku opláchli vodou a vyčistili lihem.

4.2.2 Vrtání a pájení

Všechny díry na desce měly stejný průměr a byly vyvrtány pomocí stojanové vrtačky s vrtákem o průměru 1mm. Poté jsme desku natřeli kalafunovým lakem pro větší přilnavost, osadili a připájeli jednotlivé součástky.

4.2.3 Propojovací kabely

Abychom předešli případným problémům s nefunkční deskou a vyndáváním kabelů, použili jsme dva typy počítačový konektorů. Jedná se o bíle dvouvodičové konektory se zámkem, které chrání před nesprávným zapojením napájení a klasické černé počítačové konektory.

5 Závěr

Při programování a oživování jsme postupovali po dílčích krocích. Nejdříve jsme vybrali periferii (displej, klávesnice, čtečka), k té napsali program a oživili ji. Po vytvoření obslužného programu ke každé periferii jsme začali s psaním hlavního (řídícího) programu, do kterého jsme je vložili. První testy funkčnosti proběhly v programu Proteus, následovalo oživení na pultu.

Největší potíže vznikly s přenesením programu a zapojení na desku. První desku jsme osvítili a vyleptali zrcadlově (naštěstí jsme na to přišli ještě před pájením), na druhé chybělo napájení pro čtečku a celá deska se z nepochopitelného důvodu v programu EAGLE neuzemnila. Při třetím pokusu jsme vložili na desku text, abychom se lépe orientovali při osvícení, použili konektory se zámkem (často jsme přehazovali zem s napájením) a celé schéma řádně překontrolovali.

Posledním krokem bylo mechanické řešení. Po zakoupení krabičky, pravděpodobně pro rozvodnou skříň, jsme do ní vyřezali otvory pro displej, klávesnici a různé drobnosti. Tištěné spoje jsme do krabičky umístili pomocí distančních sloupků. Během realizace nás napadlo umístit tlakový vypínač na víko krabičky – spustí alarm při jejím otevření.

Vzhledem k narůstající ceně během výroby, jsme se rozhodli nekupovat žádné čidlo ani hlasitou sirénu. Sirénu jsme nahradili výstupem z PIC zapojeným na piezo reproduktor. Když se spustí alarm, tak reproduktor vydává zvuk o frekvenci 337 Hz. Čidlo jsme simulovali rozpínacím tlačítkem – princip je stejný jako u většiny komerčně využívaných čidel.

Při závěrečném testování se ukázalo, že řídící program obsahuje drobné chyby, které ovšem nebyly zásadní pro jeho samotnou funkčnost. Chyby jsme opravili a přeprogramovali řídící jednotku. Jako další komplikace se ukázalo přehřívání stabilizátoru 7809. Po namontování pasivního chladiče se teplota dostala do únosných mezí.

Závěrem je možno říci, že se nám, i přes problémy které nastaly, podařilo sestavit plně funkční výrobek.

Seznam zkratek

ASCII - American Standard Code for Information Interchange (Americký standardní kód pro výměnu informací)
PIC - Programmable Interface Controller (Programovatelný mikrořadič)
UART – Universal Asynchronous Reciever Transmitter (Univerzální asynchronní přijímač a vysílač)
RAM – Random Access Memory (Paměť s libovolným přístupem)
EEPROM - Electronically Erasable Programmable Read-Only Memory (Elektronicky mazatelná programovatelná paměť)
EAGLE - Easily Applicable Graphical Layout Editor (Program pro výrobu desek plošných spojů)
PWM – Pulse Wide Modulation (Pulsně-šířková modulace)
GND – Ground (Uzemnění)
ID – Identification (Identifikace)
RFID – Radio-Frequency Identification (Radio-frekvenční identifikace)
PUK – Personal Unblocking Key (Soukromý odblokovávací klíč)
LED – Light-Emitting Diode (Světlo vyzařující dioda)
DDRAM – Displey Data RAM (Paměť pro data displeje)
CGRAM – Character Generator RAM (Paměť pro uložení vlastního znaku)

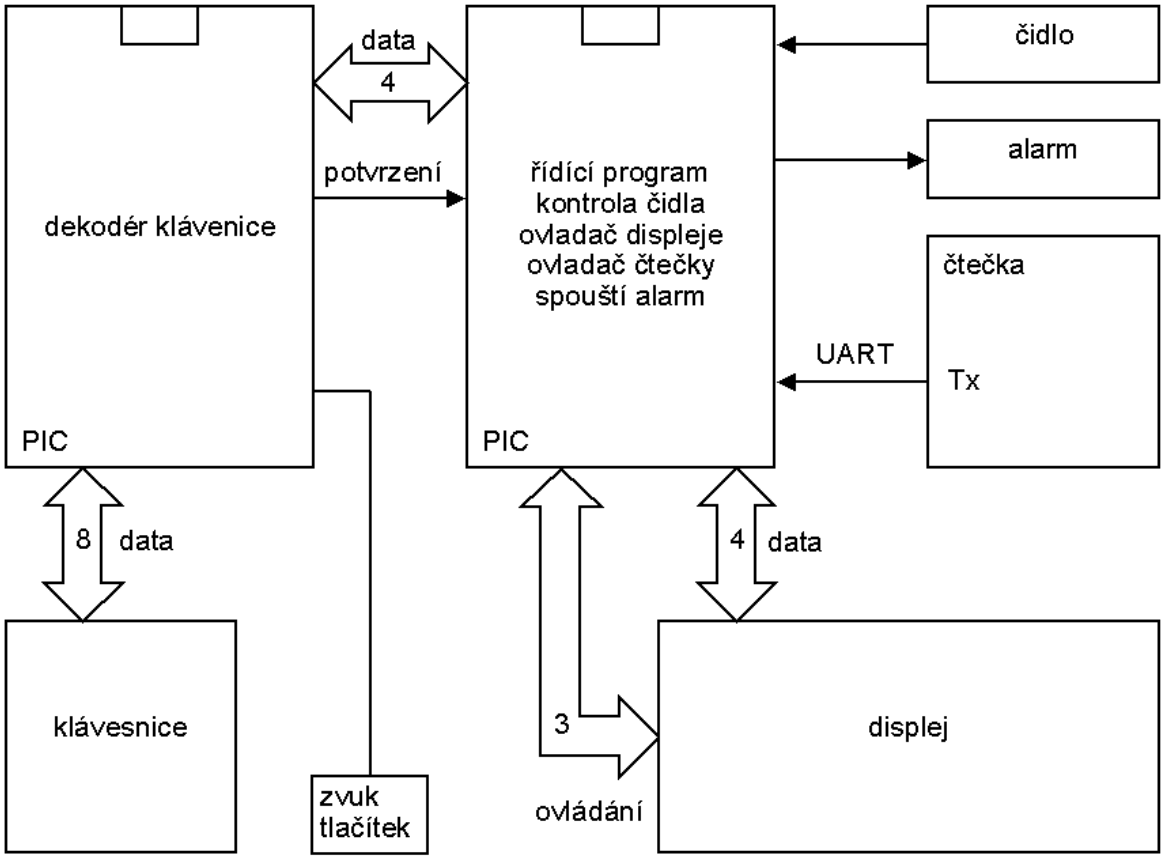
Seznam použité literatury

http://access.feld.cvut.cz/
http://www.elektronika.kvalitne.cz/ATMEL/necoteorie/LCDmatice.html
http://www.pandatron.cz
http://www.cmail.cz/doveda/
Tomáš Kubalík: Knížka na procesor PIC 16F27A / 628A / 648A
Datasheet Netronix CTU-005
Datasheet PIC16F648A
Datasheet MC2004E-SYL

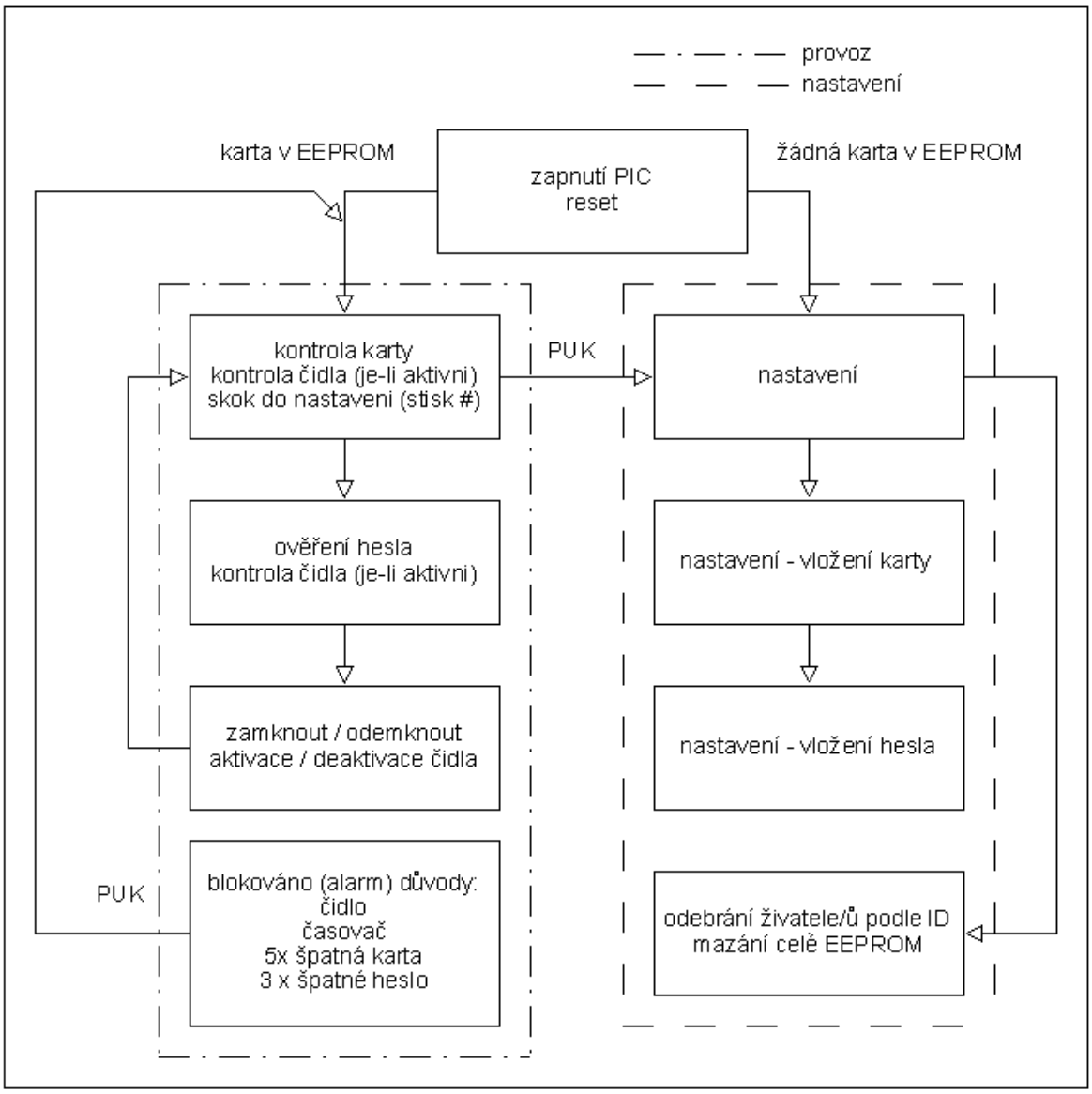
Seznam použitých programů

EAGLE 5.3.0
Microsoft Word
Proteus ISIS 6
Microchip MPLAB IDE 7.41
Asix UP
WinPic Programmer
DIA
TonReg

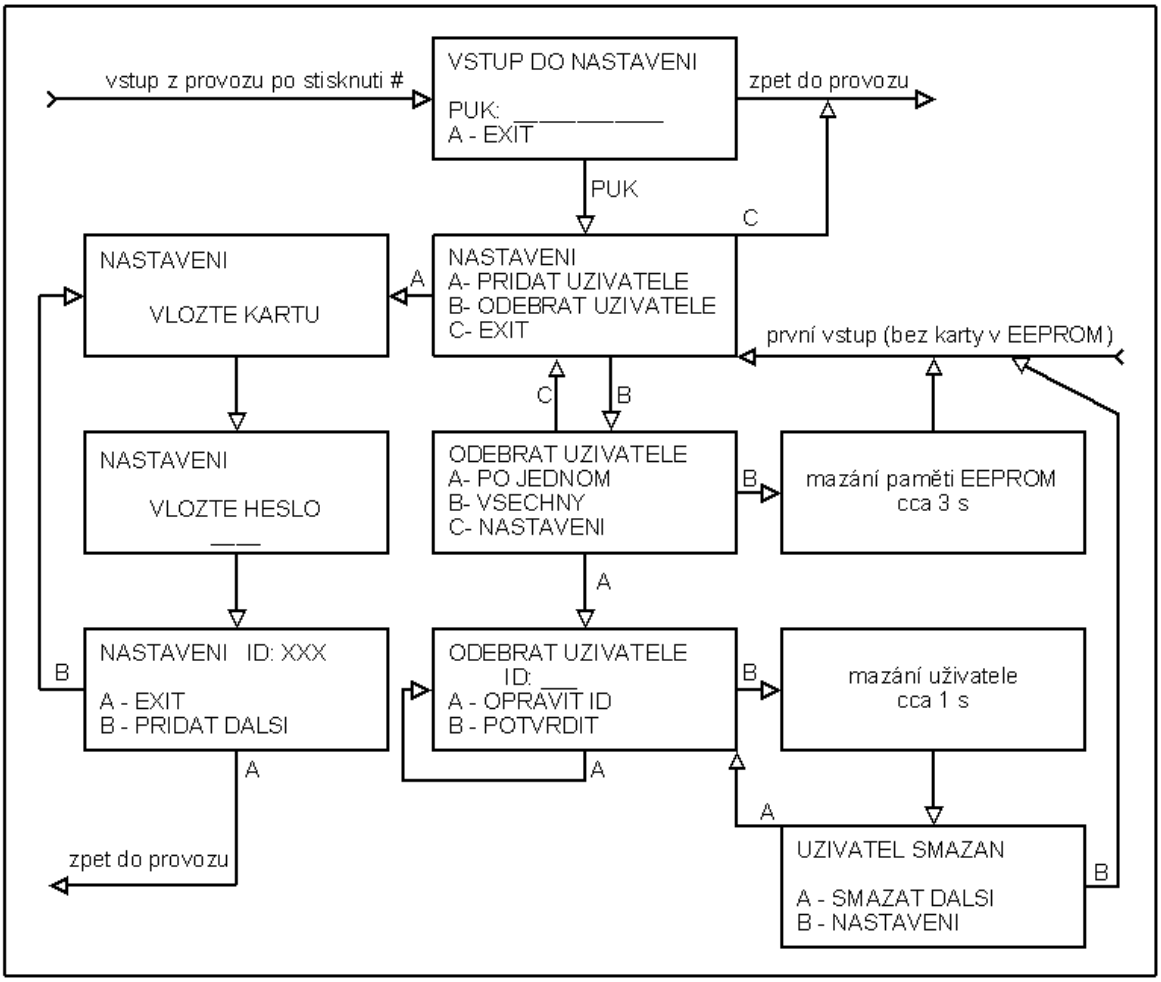
Seznam příloh



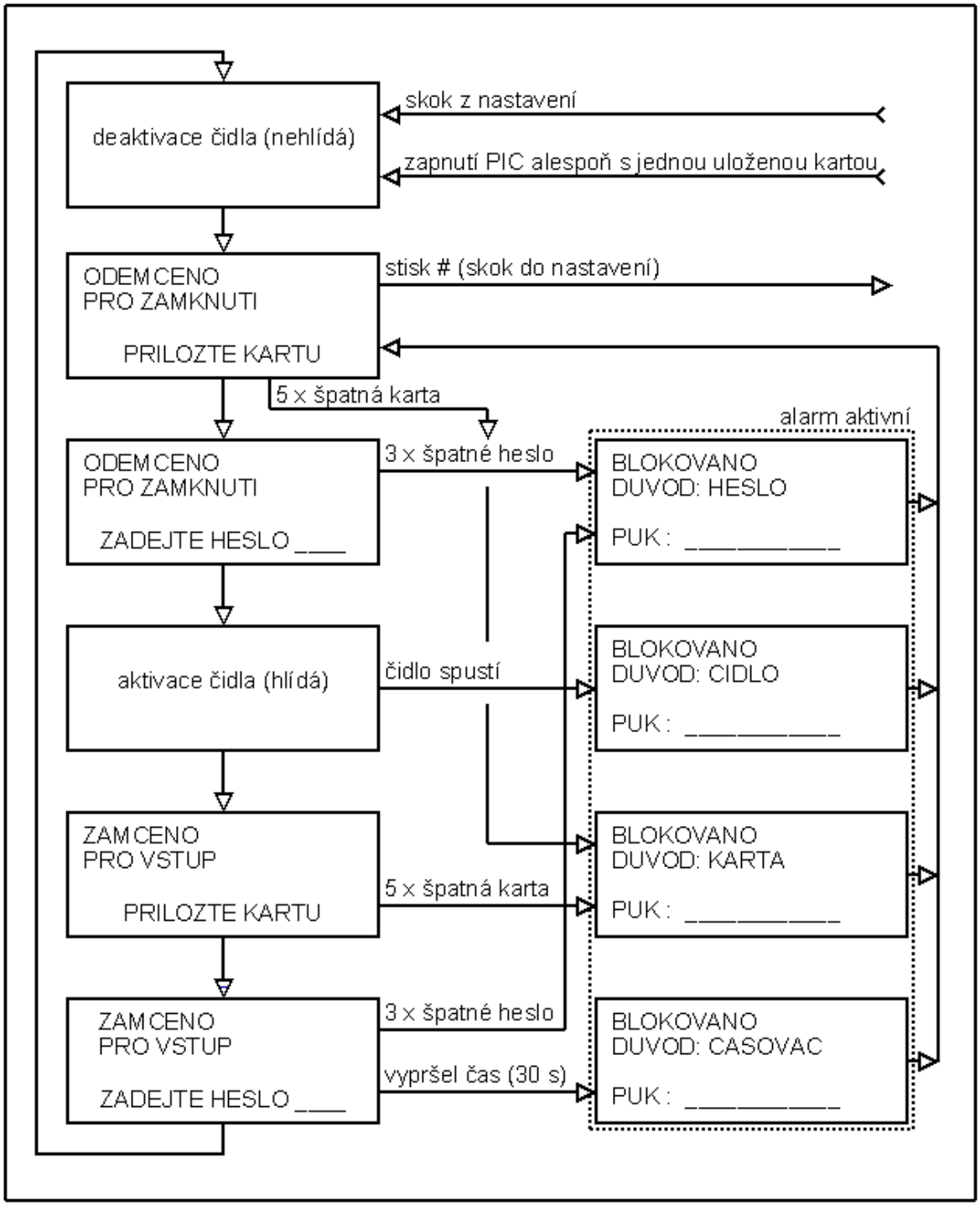
Příloha 1 - Schéma uspořádání a komunikace hardware



Příloha 2 - Zjednodušený vývojový diagram



Příloha 3 - Vývojový diagram - blok nastavení



Příloha 4 - Vývojový diagram - blok provoz

EEPROM	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	CTRL
0x10	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	INIT
0x20	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	CNT
0x30	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	LL
0x40	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0x50	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0x60	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0x70	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0x80	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0x90	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0xA0	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0xB0	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0xC0	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0xD0	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0xE0	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--
0xF0	0x01	0x0B	0x01	DATA					ID	CRCH	CRCL	N1	N2	N3	N4	--

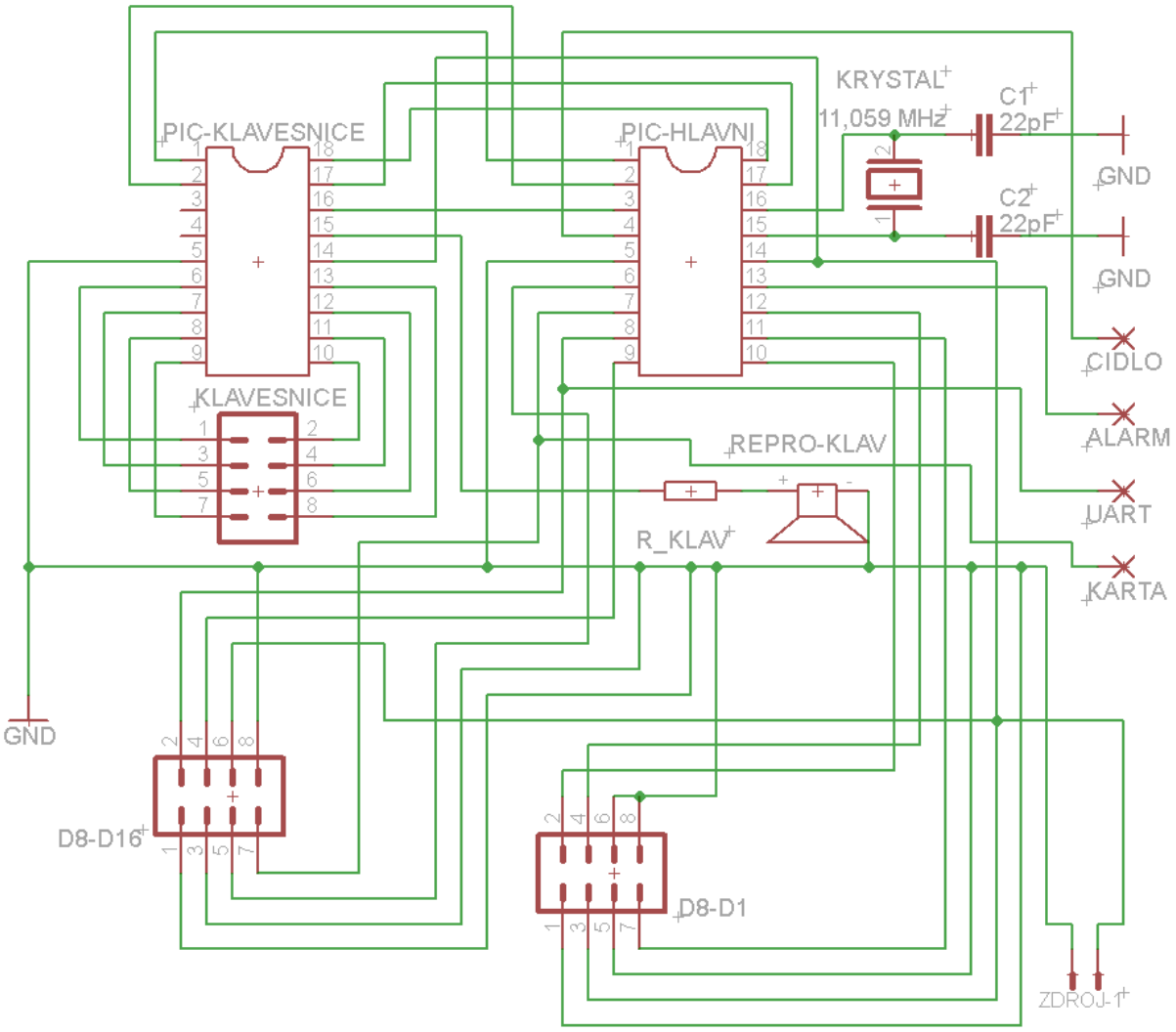
Příloha 5 - Mapa paměti EEPROM

Význam funkce	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Doba
smaže disp. a nastaví kurzor na začátek	0	0	0	0	0	0	0	0	0	1	1,64 ms
nastaví kurzor na začátek	0	0	0	0	0	0	0	0	1	x	1,64 ms
směr posuvu kurzoru I/D (0=vlevo, 1=vpravo), posuv textu S (0=ne, 1=ano)	0	0	0	0	0	0	0	1	I/D	S	40 us
D - zapne displej, C - zapne kurzor, B - zapne blikání kurzoru	0	0	0	0	0	0	1	D	C	B	40 us
1x posune (S/C=0 kurzor, S/C=1 text) směrem (R/L=0 vlevo, R/L=1 vpravo)	0	0	0	0	0	1	S/C	R/L	x	x	40 us
inicializace: DL=0 4-bit, DL=1 8-bit mód N=0 jednořádkový, N=1 dvouřádkový disp. F=0 font 5x8, F=1 font 5x10	0	0	0	0	1	DL	N	F	x	x	40 us
přepnutí na zápis do CGRAM	0	0	0	1	adresa v CGRAM						40 us
přepnutí na zápis do DDRAM	0	0	1	adresa v DDRAM							40 us
čtení příznaku BF (BF=0 příjem povolen, BF=1 řadič zaneprázdněn), čtení adresy v DDRAM	0	1	BF	adresa v DDRAM							0
zápis dat do CGRAM nebo DDRAM	1	0	data								40 us
čtení dat z CGRAM nebo DDRAM	1	1	data								40 us

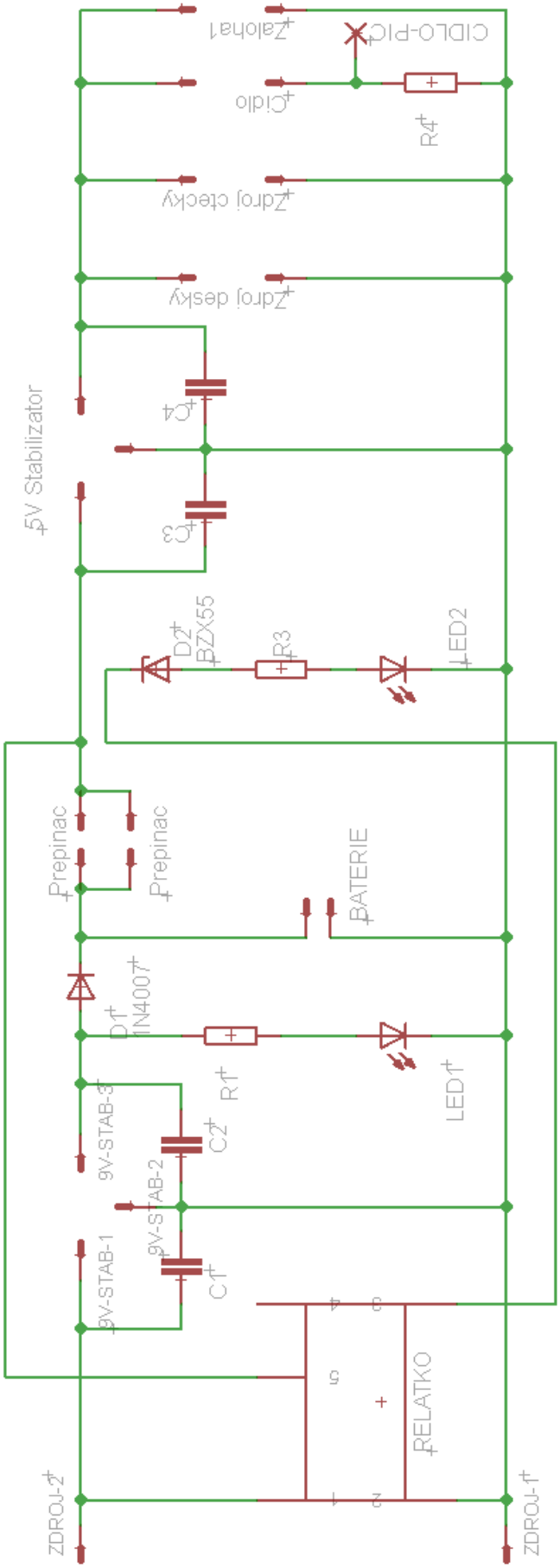
Příloha 6 - Tabulka instrukcí pro řadič HD44780

STAT_0							
7	6	5	4	3	2	1	0
CRD_FLG	ALARM		NAST	PUK	EEPROM_PLNO	CRD_EEPROM	ONOFF
						čidlo sníma	1
						čidlo nesníma	0
						karta se uloží pouze do RAM 0x40 - 0x4A	1
						přiložená karta se uloží na adresu jejíž hodnota je v reg. EEPROM_ADR	0
					pamět EEPROM je plná, jestliže se přidá nová, přepíše se nejdříve přidaná karta	1	
					pamět EEPROM není plná	0	
				z PUKu lze vyskocit	1		
				z PUK lze vykocit pouze jeho zadanim	0		
			skok do nastavení povolen	1			
			skok do nastavení zakázán	0			
		1					
		0					
	1	nahoru (slouží pro generaci tónu alarmu)					
	0	dolu (slouží pro generaci tónu alarmu)					
1	Konec přijmu UART (karta)						
0	všechny bity přijaté po UART nebyly ještě uloženy						

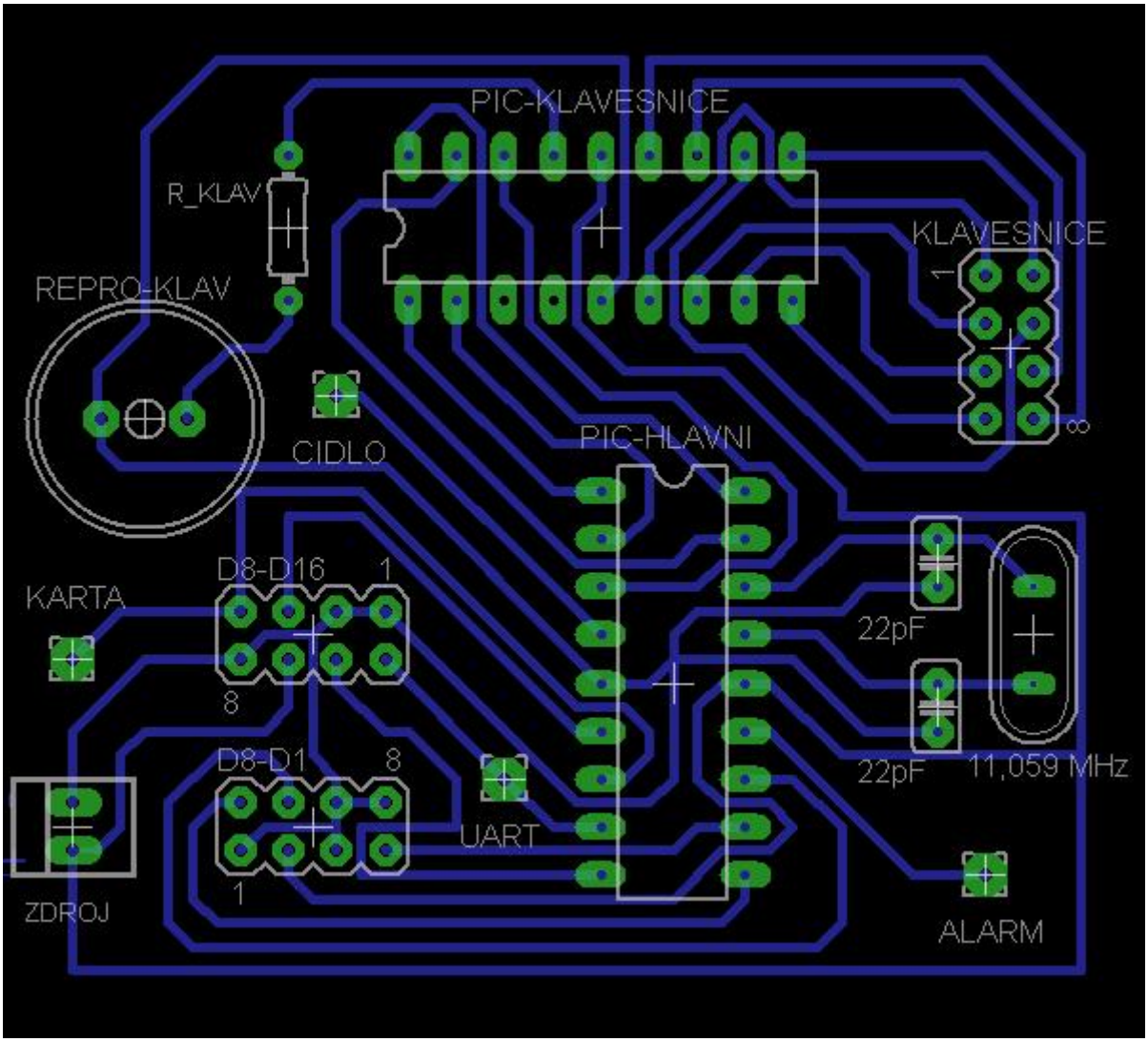
Příloha 7 - Význam bitů v registru STAT_0



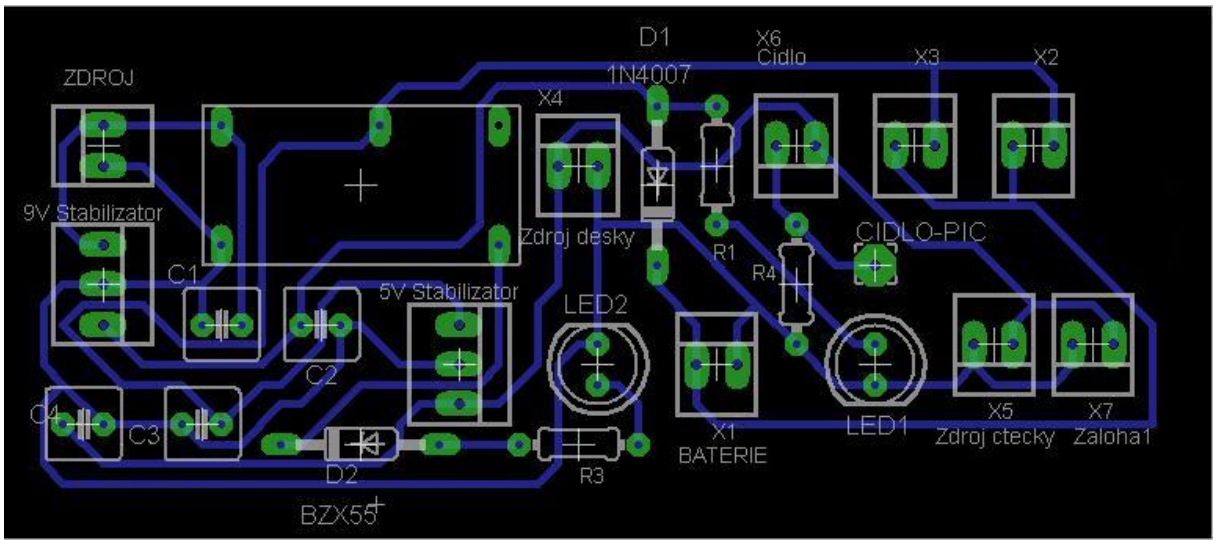
Příloha 8 - Schéma - Hlavní deska


Zalohai
CIDLO-PIC
Cidlo
R4
Zdroj ctecky
Zdroj desky
5V Stabilizator
C4
C3
D2
BZX55
R3
LED2
Prepinac
Prepinac
BATERIE
D1
1N4007
R1
LED1
9V-STAB-3
C2
9V-STAB-2
9V-STAB-1
C1
RELATKO
ZDROJ-2
ZDROJ-1

Příloha 9 - Schéma - Signalizační lišta



Příloha 10 - Deska plošných spojů - Hlavní deska



Příloha 11 - Deska plošných spojů - Signalizační lišta