VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SVĚTELNÉ NOVINY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE LUKÁŠ MIČULKA
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SVĚTELNÉ NOVINY

LIGHTNING NEWS

BAKALÁŘSKÁ PRÁCE

BACHELOR´S THESIS

AUTOR PRÁCE LUKÁŠ MIČULKA
AUTHOR

VEDOUCÍ PRÁCE Ing. TOMÁŠ HERRMAN
SUPERVISOR

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Mičulka Lukáš**
Obor: Informační technologie
Téma: **Světelné noviny**
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s vybraným mikroprocesorem (např. 89s51).
2. Navrhněte připojení procesoru k matici led a k PC pomocí RS232.
3. Diskutujte možnost programování procesoru v aplikaci (ISP).
4. Navrhněte a implementujte software pro mikroprocesor, který umožní zobrazení a rolování textu na matici led a umožní připojení přes RS232 s PC pro nahrání zpráv.

Literatura:

- Skalický P.: Mikroprocesory řady 8051, nakladatelství BEN, Praha, 2001
- Kľúčik J., Fronc V.:M ikrokontroléry ATMEL s jádrem 8051, nakladatelství BEN, Praha, 2001
- Matoušek D.: Vývojový kit USB51KIT pro AT89S51 a AT89S52, nakladatelství BEN, Praha, 2005
- Vlach J., Vlachová V.: Počítačová rozhraní přenos dat a řídicí systémy, nakladatelství BEN, Praha, 2002

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1-2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herrman Tomáš, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Licenční smlouva

Licenční smlouva je uložena v archivu FIT VUT v Brně.

Abstrakt

Zaměřením této bakalářské práce je návrh zapojení informačního panelu s maticí LED a vytvoření firmware pro osmibitový mikrokontrolér z rodiny 8051, který ji má řídit a komunikovat s PC přes rozhraní RS232. Okrajově je zde zmíněna aplikace pro PC, se kterou je možné ovládat zobrazování textu a měnit data v paměti mikrokontroléru. V rámci této práce byl sestaven a otestován funkční vzorek tohoto zařízení.

Klíčová slova

Světelné noviny, informační panel, matice LED diod, AT89S51, rozhraní RS232, I^2C, ISP

Abstract

This Bachelor thesis is focused on connection design of information board with LED diode matrix and creation of firmware for 8-bit microcontroller from 8051 family, which aim is to control the matrix and communicate with PC over RS232 interface. Marginally there is mentioned the application for PC, which can handle with text imaging and changing data in memory of microcotroller. Within this thesis there was completed and tested the working prototype of this appliance.

Keywords

Lightning news, information board, LED diode matrix, AT89S51, RS232 interface, I^2C, ISP

Citace

Mičulka Lukáš: Světelné noviny. Brno, 2008, bakalářská práce, FIT VUT v Brně

Světelné noviny

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Tomáše Herrmana.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Mičulka
8. května 2008

Poděkování

Rád bych tímto poděkoval vedoucímu mé bakalářské práce Ing. Tomáši Herrmanovi za odbornou pomoc, podnětné připomínky a konzultace.

© Lukáš Mičulka, 2008.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů..

Obsah

Obsah ... 1
Úvod ... 3
1 Návrh hardware ... 4
1.1 Matice LED ... 4
1.1.1 Časové multiplexování ... 4
1.1.2 Sériový přenos dat ... 5
1.1.3 Obvodové zapojení matice ... 5
1.2 Modul s mikrokontrolérem ... 6
1.2.1 Mikrokontrolér AT89S51 ... 7
1.2.1.1 Charakteristické rysy AT89S51 ... 7
1.2.1.2 In-System programování ... 9
1.2.2 Externí sériová paměť 24LC256 ... 10
1.2.3 Sériový port ... 10
1.2.4 Obvodové zapojení modulu ... 12
2 Program pro mikrokontrolér ... 14
2.1 Komunikace s pamětí ... 14
2.1.1 Charakteristika I^2C ... 14
2.1.2 Popis komunikace ... 15
2.1.2.1 Zápis bytu ... 15
2.1.2.2 Zápis stránky ... 16
2.1.2.3 Čtení bytu na aktuální adrese ... 16
2.1.2.4 Čtení bytu ... 17
2.1.2.5 Čtení stránky ... 17
2.1.3 Implementace komunikace ... 17
2.1.3.1 Podprogram ShiftTo ... 17
2.1.3.2 Podprogram StartBit ... 17
2.1.3.3 Podprogram StopBit ... 18
2.1.3.4 Podprogram SlaveAck ... 18
2.1.3.5 Podprogram NoAck ... 18
2.1.3.6 Podprogram AckPoll ... 18
2.1.3.7 Podprogram ByteWrite ... 18
2.1.3.8 Podprogram ByteReadCycle ... 19
2.1.3.9 Podprogram ByteRead ... 19
2.1.3.10 Podprogram ByteCurrentRead ... 19

2.2 Rutina pro zobrazování ... 20
2.2.1 Zobrazení bufferu na matici LED ... 20
2.2.2 Pohyb a načítání textu ... 22
2.2.2.1 Pohyb ... 22
2.2.2.2 Načítání textu ... 23
2.2.3 Speciální funkce ... 24
2.2.3.1 Přehled funkcí ... 24
2.2.3.2 Spouštění funkcí pomocí speciálního znaku v textu ... 26
2.3 Rutina pro sériový port ... 26
2.3.1 Nastavení parametrů komunikace ... 27
2.3.2 Popis komunikace ... 27
2.3.2.1 Příkazy s operandem ... 27
2.3.2.2 Příkazy bez operandu ... 29
2.3.2.3 Dotazy ... 30
3 Ovládací aplikace pro PC ... 32
4 Závěr ... 33
Literatura ... 34
Seznam příloh ... 35

Úvod

V dnešním světě se stále více projevuje hlad člověka po informacích. Jejich cesta k člověku a forma jejich prezentace je velmi rozmanitá. Jednou z možných variant jsou informační panely, které mohou informaci vizuálně promítnout. Jedná se přitom o velmi univerzální a pro člověka velmi přijatelnou formu získání informace. Dnes se již s takovými panely setkáme prakticky kdekoli, od proměnných značek na dálnici, jízdních řádů na zastávkách hromadné dopravy, čekáren v ordinacích až po použití v reklamě. Panely jsou většinou připojeny pomocí vnějšího rozhraní k PC či k jinému ovládacímu prvku, a tím je lze jednoduše aktualizovat.

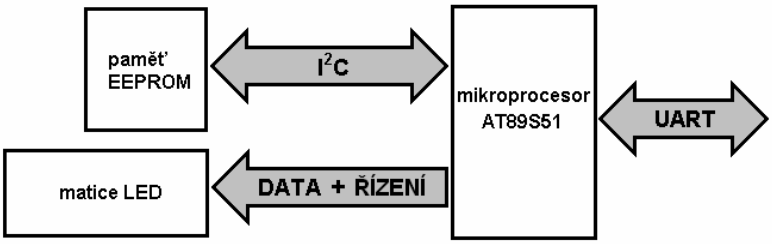
Tato bakalářská práce se tedy zabývá zobrazováním textu na světelném panelu sestaveném z matice LED diod, který je řízen mikroprocesorem AT89S51 firmy ATMEL. Text, který po panelu roluje, je zadán z PC. Pro přenos dat mezi mikrokontrolérem a PC je požadována sériová komunikace pomocí RS232. Nahraná data by měla být uložena do permanentní paměti, aby se zamezilo jejich ztrátě při výpadku proudu. Na straně PC slouží ke komunikaci specializovaný software.

V první kapitole je popsáno obvodové zapojení matice LED a modulu s mikrokontrolérem a jejich vzájemné propojení. Kapitola 2 obsahuje popis programového vybavení mikrokontroléru, všech jeho řídících funkcí a implementovaných datových přenosů. V následné kapitole je zmíněn ovládací program pro PC.

1 Návrh hardware

V této kapitole bude popsán návrh a realizace funkčního vzorku, který je součástí této práce. Návrh je zaměřen hlavně na efektivnost a jednoduchost.

Celý panel se skládá z několika důležitých celků. Blokové schéma na obrázku 1.1 zachycuje hlavní konstrukční prvky a jejich propojení.



Obr. 1.1 Blokové schéma panelu

Pro zjednodušení konstrukce je tento panel rozdělen na 2 části:

- matice LED - obsahuje LED diody a komponenty
- modul s mikrokontrolérem - zbývající část včetně externí paměti

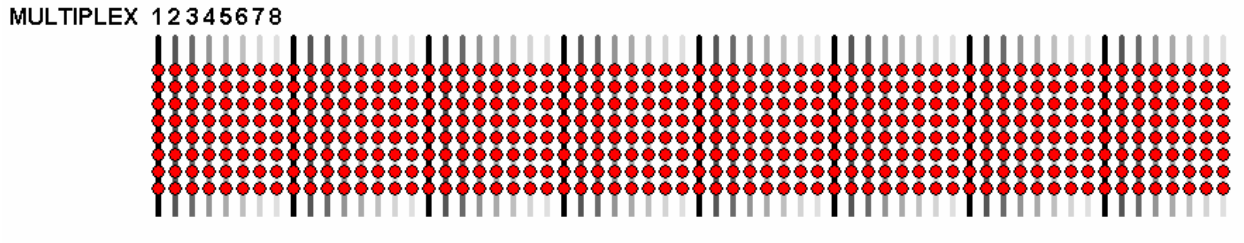
1.1 Matice LED

Cílem je navrhnout takové zapojení, které umožňuje mikroprocesorem ovládat velký počet LED diod. Proto se v následujících sekcích zmiňuji o metodách, jakými lze snížit počet vodičů k ovládání matice, které je nutné připojit k výstupům mikrokontroléru.

1.1.1 Časové multiplexování

Jelikož má matice obsahovat 512 LED diod, není možné pro každou z nich použít samostatný výstup mikrokontroléru. Proto je nutné použít časového multiplexu, kdy nesvítí všechny světla najednou, ale matice je rozdělena na menší části, které se postupně střídavě rozsvěcují. Je přitom nutná relativně vysoká frekvence, aby se toto střídavé rozsvěcování jevilo oku, jako kdyby všechna světla svítila stále. Tento princip se při podobných zapojeních používá často, právě s ohledem na možnou velkou úsporu potřebných ovládacích vodičů.

V případě tohoto zapojení, kde je zmiňovaných 512 světel organizovaných do 8 řádku po 64 sloupcích, se jeví jako nejvhodnější varianta rozdělení do multiplexů podle obrázku 1.2.



Obr. 1.2 Časové multiplexování v matici

Z obrázku je patrno, že vždy najednou svítí právě 8 sloupců. Tento počet je vybrán z důvodu, že s obsluhou více multiplexů by již pravděpodobně nebylo možné dosáhnout dostatečné frekvence s použitým kmitočtem mikrokontroléru.

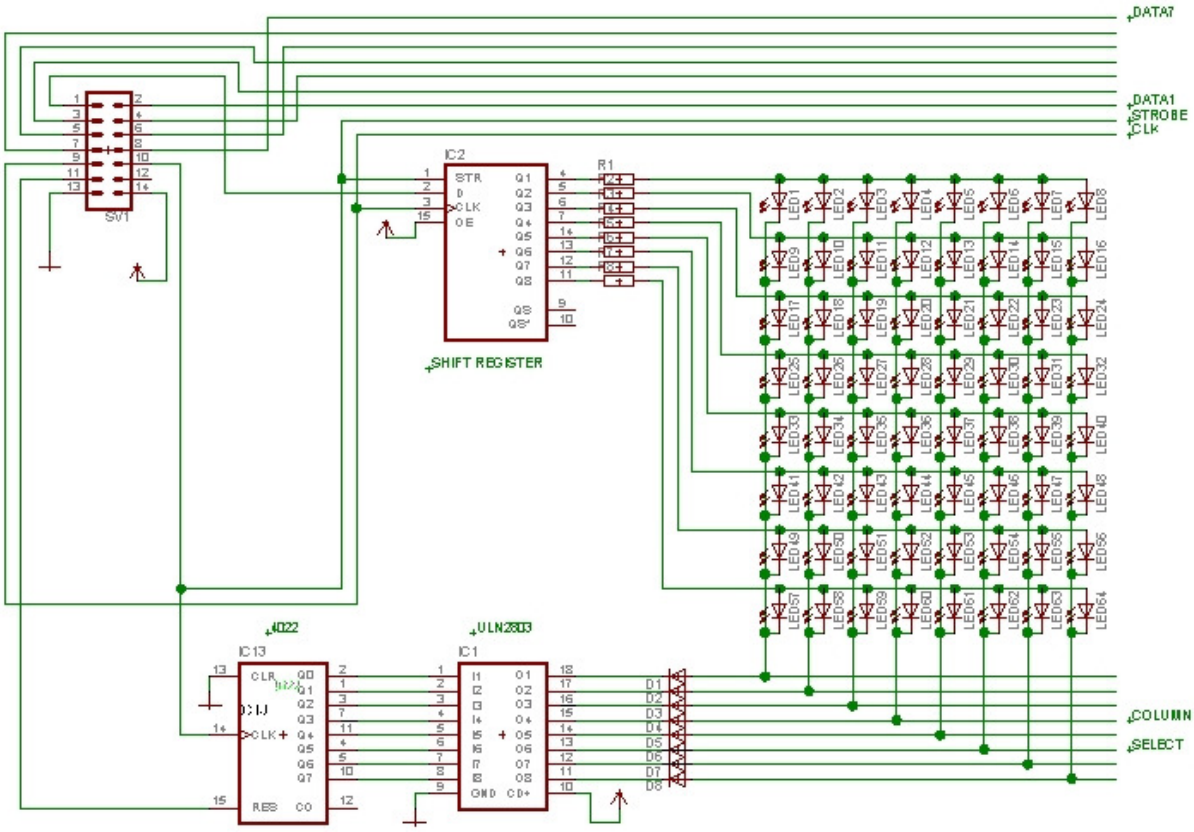
1.1.2 Sériový přenos dat

I přes použití časového multiplexování je stále nutno použít při každém rozsvícení části světel 64 vodičů. Toto číslo lze podstatně snížit změnou z přenosu dat paralelními vodiči na hodinami řízený přenos sériový. Potom je však nutné vložit mezi vodiče a světla prvek, který převede sériovou informaci zpět na paralelní. Tím se v našem případě stane posuvný 8bitový registr CD4094 [2]. Ten se skládá ze 3 částí: posuvného registru, úložného registru a třístavových budičů na výstupech. K nahrávání sériově přenášeného bytu do posuvného registru dochází při nástupné hraně na vodiči CLK. Pro přesun bytu z posuvného registru do úložného registru a tedy na výstupy je zapotřebí nástupná hrana signálu STROBE. Takto lze snížit počet potřebných datových vodičů na 8, přibudou pouze 2 vodiče řídící.

1.1.3 Obvodové zapojení matice

V této části je pozornost věnována obvodovému zapojení matice LED. Jeho část s jedním segmentem světel je zobrazena na obrázku 1.3. Návrh plošného spoje je uveden v příloze.

Z obrázku vyplývá, jak jsou LED diody organizovány do segmentů po 8 sloupcích. LED diody v jednom segmentu a v jednom řádku mají vždy anodu spojenou přes rezistor k odpovídajícímu výstupu výše zmíněného posuvného registru. U všech LED diod v jednom sloupci jsou zase spojeny katody a podle čísla sloupce v segmentu připojeny k odpovídajícím vodičům ve sběrnici pro výběr sloupců. Tu řídí 8bitový čítač 1 z n CD4022 [3], který pracuje tak, že při nástupné hraně hodinového signálu pošle na odpovídající vodič logickou 1 a ostatním logickou 0. Mezi čítačem a sběrnicí však ještě musí být umístěn IO ULN2803 [4], který je potřeba pro zesílení proudu, aby bylo možné najednou rozsvítit až 64 LED diod. Kromě toho má ještě funkci invertoru. Pro ochranu LED diod jsou zde umístěny diody, které mají zamezit, aby LED diody nezničila obrácená polarita napětí.



Obr. 1.3 Schéma zapojení hlavní části matice LED

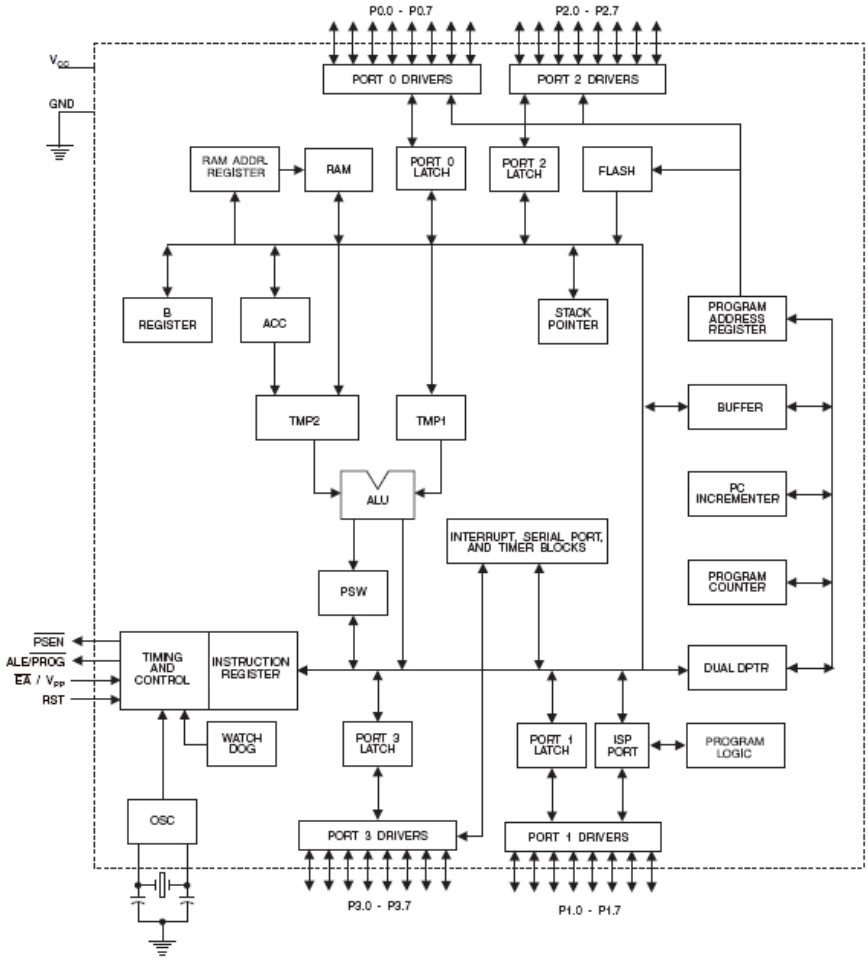
Rozsvěcování LED diod v aktuálním sloupci v každém segmentu probíhá následovně. Na datové vstupy všech posuvných registrů jsou přivedeny logické úrovně pro diody v prvních řádcích. Poté následuje nástupní hrana signálu CLK pro posuvné registry. Stejný postup se opakuje ještě sedmkrát, dokud nejsou v posuvných registrech všech obvodů hodnoty pro všechny LED diody v zobrazovaných sloupcích. Poté následuje nástupní hrana signálu STROBE u posuvných registrů, se kterou se nahraná data přesunou na výstupy. Vodič STROBE je připojen i na hodinový vstup čítače, kde stejná nástupní hrana způsobí aktivaci dalšího vodiče na sběrnici pro výběr řádku. V tento moment, kdy jsou na výstupy posuvných registrů zapsány nové hodnoty a kdy je na katody LED diod v jednom z 8 sloupců v každém segmentu přivedena logická 0, se rozsvítí LED diody v těchto sloupcích, které mají na anodě logickou 1.

1.2 Modul s mikrokontrolérem

Jádrem tohoto modulu je mikrokontrolér, který má kromě řízení matice LED na starosti i komunikaci přes RS232 s PC a také ukládání a načítání dat z externí paměti.

1.2.1 Mikrokontrolér AT89S51

Řídícím mikrokontrolérem je zvolen AT89S51 firmy ATMEL řady 8051 [1]. Mezi jeho hlavní výhody patří hlavně cenová dostupnost, jednoduché programování a také to, že poskytuje všechny potřebné zařízení a periferie.



Obr. 1.4 Blokové schéma AT89S51

1.2.1.1 Charakteristické rysy AT89S51

- kompatibilní architektura s rodinou jednočipových mikroprocesorů MCS-51
- 4kB Flash paměti pro program, až 1000 programovacích cyklů
- In-System Programming(ISP)
- 128B RAM paměti pro proměnné
- pracuje s hodinovým kmitočtem 0 HZ až 33 MHz
- 32 vstupně/výstupních linek
- 2 16-bitové čítače/časovače
- plně duplexní UART sériový kanál
- úsporné režimy

Instrukční soubor

Tento mikrokontrolér stejně jako ostatní z řady 8051 používá CISC instrukční architekturu. Nabízí tedy širokou paletu instrukcí, které mohou být jednobytové, dvoubytové či tříbytové.

1. Instrukce pro přenos dat – univerzální instrukce MOV realizuje přenosy bitů i bytů ze zdrojové na cílovou adresu. Pracují s pamětí, registry i akumulátorem. K práci se zásobníkem slouží instrukce PUSH a POP, jejichž jediným operandem je vždy akumulátor. Dále lze použít instrukce pro přenos využívající střadač jako XCH, XCHD, MOVX či MOVC. MOVX slouží k přesunu dat mezi vnější paralelní paměti dat a akumulátorem a MOVC pro přenos konstant uložených v paměti programu tamtéž. Kromě instrukce MOV s operandem DPTR, která slouží k přenosu 16bitové adresy do tohoto registru, je přenášeno vždy jen jedno slovo.
2. Aritmetické instrukce – pomocí instrukční sady mohou být realizovány všechny 4 základní aritmetické operace. Instrukce pracují s 8bitovými čísly bez znaménka, při použití příznaku přetečení OV lze použít i čísla v doplňkovém kódu. Existuje zde i podpora pro BCD.
3. Logické instrukce – je nabízena možnost provádění všech základních logických operací, posunů a rotací. Obsahují jeden nebo dva operandy. V prvním případě je operandem akumulátor, v druhém je jeden z operandů v akumulátoru a druhý v paměti dat.
4. Řídící instrukce – obsahuje celou škálu podmíněných i nepodmíněných skoků, které mohou sloužit ke skákání na krátké 8bitové, střední 11bitové či dlouhé 16bitové vzdálenosti v kódu. Také u volání podprogramu a rutin po přerušeních a u návratů z nich se rozlišuje délka skoků. Pomocí instrukcí CJNE a DJNZ lze jednoduše realizovat cykly.

Organizace paměti

Z pohledu paměti se jedná o mikrokontrolér s Hardwardskou architekturou, tedy program a data mají své oddělené paměti. Toto oddělení umožňuje adresovat každou paměť zvlášť jen 8bitovou adresou, čímž se velmi zjednodušuje práce 8bitové centrální procesorové jednotce. Pro případ nepřímého 16bitového adresování slouží registr DPTR.

1. Paměť programu – program je uložen v 4kB paměti FLASH, kterou lze programovat pomocí ISP. Na nulté stránce je umístěn vektor přerušení. Každá buňka programové paměti může být adresována 16bitovým programovým čítačem (PC). Lze jím adresovat i externí paměť.
2. Paměť pro data – je tvořena RAM pamětí a její rozdělení znázorňuje obrázek 1.5. Její velikost je 128 bytů. Spodní část je zabrána 16 univerzálním registry ve 4 bankách, dalších 16 bytů lze adresovat bitově a zbytek pouze po jednotlivých slovech.

Obr. 1.5 Mapa paměti pro data

Vstupy a výstupy

Vstupně výstupní linky mikrokontroléru jsou rozděleny po 8 do 4 portů a porty 1 až 3 jsou opatřeny pull-up rezistory. S porty se pracuje pomocí instrukcí pro přesun dat, lze s nimi provádět bytové i bitové operace. Některé linky mají kromě své vstupně výstupní funkce i funkce jiné (viz [1]).

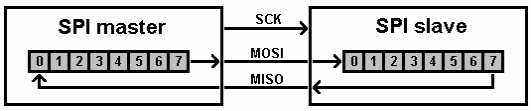
Přerušovací systém

Přerušovací systém AT89S51 může obsahovat až 5 zdrojů přerušení. Jsou jimi 2 přerušení od čítačů/časovačů, externích vstupů a sériového portu. Když je přerušení aktivováno, systém podle obsahu vektoru přerušení volá příslušnou rutinu pro jeho ošetření. Pokud dojde k aktivaci více přerušení najednou, přednost dostává zdroj s vyšší prioritou. Tu lze nastavit pomocí příslušného registru či ponechat pevně přednastavenou.

1.2.1.2 In-System programování

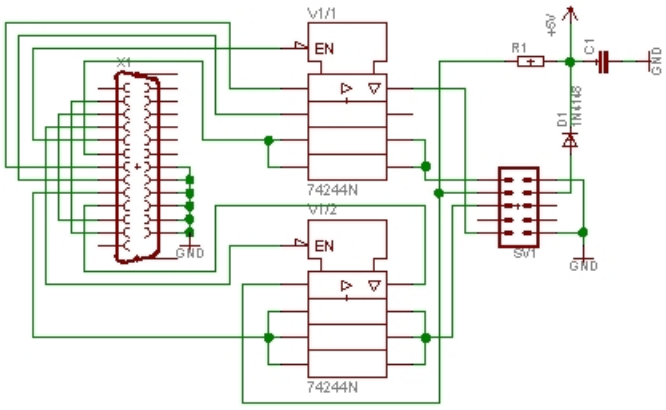
Jedná se o metodu, která dovoluje programování přímo v cílovém zapojení. Odpadá tak nutnost při potřebě naprogramování mikrokontrolér ze zařízení vyjmout a naprogramovat jej v programátoru.

FLASH paměť pro program se programuje pomocí SPI. K tomu má mikrokontrolér určeny vývody označené jako MOSI, MISO a SCL, které jsou na portu 1. Při programování je na RESET mikrokontroléru přivedena logická 1. Poté již probíhá sériový přenos dat z PC do mikrokontroléru řízený hodinovým signálem z SCL. Schéma přenosu mezi masterem, v tomto případě PC, a slavem je na obrázku 1.6. Po úspěšném naprogramování je RESET vrácen do logické 0 a program je spuštěn.



Obr. 1.6 Přenos dat pomocí SPI

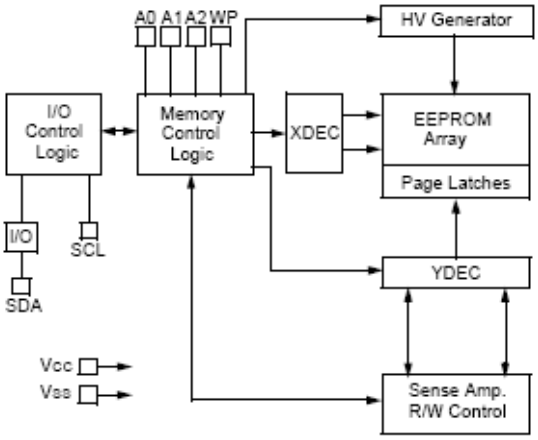
V tomto zapojení je pro programování využit paralelní port PC a proto musí být zhotoven i programátor nebo spíše interface, jehož schéma je uvedeno na obrázku 1.7.



Obr. 1.7 Programátor ISP přes LPT

1.2.2 Externí sériová paměť 24LC256

Jelikož použitý mikrokontrolér neobsahuje žádnou vnitřní permanentní paměť pro data, musí být k němu přidána vnější. Z nabízených možností je vybrána varianta sériové paměti EEPROM s označením 24LC256 firmy Microchip [5], která se jeví jako nejvýhodnější díky poměru cena versus kapacita. Jedná se o 256kB paměť, kde komunikace mezi mikrokontrolérem a pamětí probíhá po sběrnici I^2C, která je velmi šetrná k počtu potřebných vodičů, které jsou pouze dva, jeden datový a druhý hodinový. Protokol I^2C bude ještě podrobně popsán v kapitole 2. Paměť je stránkována po 64 bytech. Dokáže pracovat s 400kHz hodinovou frekvencí, což pro tyto účely postačuje. Na sběrnici I^2C je možné připojit až 8 těchto pamětí díky 3 adresným vstupům (A0-A2), data navíc mohou být chráněna pomocí vstupu write protect (WP). Na obrázku 1.8 je znázorněno blokové schéma.



Obr. 1.8 Blokové schéma paměti 24LC256

Zde je výčet všech operací, které lze s pamětí provádět:

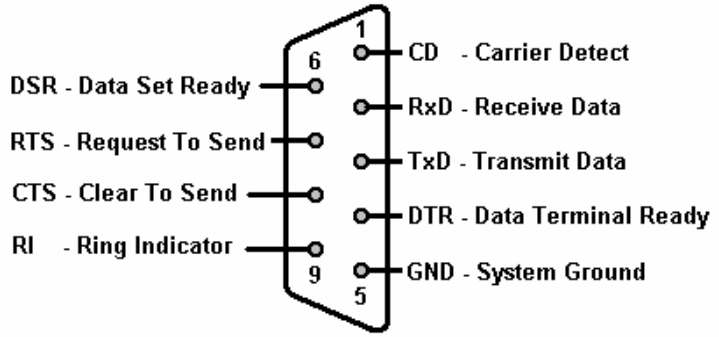
- zápis jednoho bytu
- zápis celé stránky
- čtení jednoho bytu
- čtení jednoho bytu na aktuálně nastavené adrese
- sekvenční čtení bytů

1.2.3 Sériový port

Pro propojení panelu a PC je vybrán sériový port COM. Sériová linka se pro toto použití výborně hodí, protože poskytuje dostatečnou rychlost a oproti rozhraní USB je podstatně jednodušší. Jako nevýhoda se může naopak jevit maximální délka kabelu jen 15 metrů.

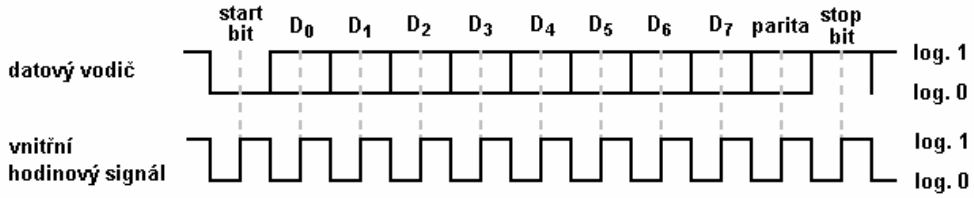
Sériový port je součástí periferních rozhraní většiny počítačů již přes 20 let. Dnes jej ale už pro větší rychlost vytlačuje modernější USB. Data se po sériové lince přenáší sériově po jednom vodiči pro každý směr. Jedná se tedy o full-duplex přenos, kdy je možné zároveň číst i odesílat. Přenos

probíhá asynchronně, což znamená, že zde není žádný hodinový signál. Kromě přenosových vodičů disponuje port ještě několika signály pro domlouvání spojení, tzv. hand-shakingu. Ten se dnes již ale většinou neprovádí. Na obrázku 1.9 je zobrazeno zapojení sériového portu na 9-ti pinovém konektoru D-Sub, který je v PC nejčastější.



Obr. 1.9 Zapojení sériového portu na 9 pin D-Sub konektoru

Jelikož zde není hodinový signál, musí být všechna propojená zařízení dohodnuta na rychlosti přenášení, která se udávájí v bitech za sekundu, a na synchronizačních a kontrolních bitech. Ty se přidají k přenášeným datům a vytvoří rámec (viz obrázek 1.10). Počet datových bitů je nutno také předem dohodnout, jelikož kromě běžně používaných 8 se může přenášet i 6 či 7 bitů. Mezi synchronizační bity patří start bit, což je nulový bit, který předchází datovým bitům v rámci a upozorňuje příjemce na příchozí data, a jeden až dva stop bity, které rámec ukončují. Pro jednoduchou kontrolu dat slouží paritní bit umístěný mezi daty a stop bity. Parita je jednoduchým kontrolním mechanizmem nenáročným na výkon. Vysílací zařízení spočítá počet log. 1 v datových bitech a doplní paritní bit tak, aby zůstal v součtu s ním počet jedničkových bitů podle nastavení buď sudý nebo lichý. Parita není ovšem povinná. Posledním parametrem přenosu, který je nutno nastavit, je řízení toku dat. Řízení slouží jako hand-shaking, sériový port je k němu vybaven několika signály. V dnešní době už je toto ale většinou zbytečné, proto jej lze i vypnout.



Obr. 1.10 Rámec s vnitřním hodinovým signálem zařízení

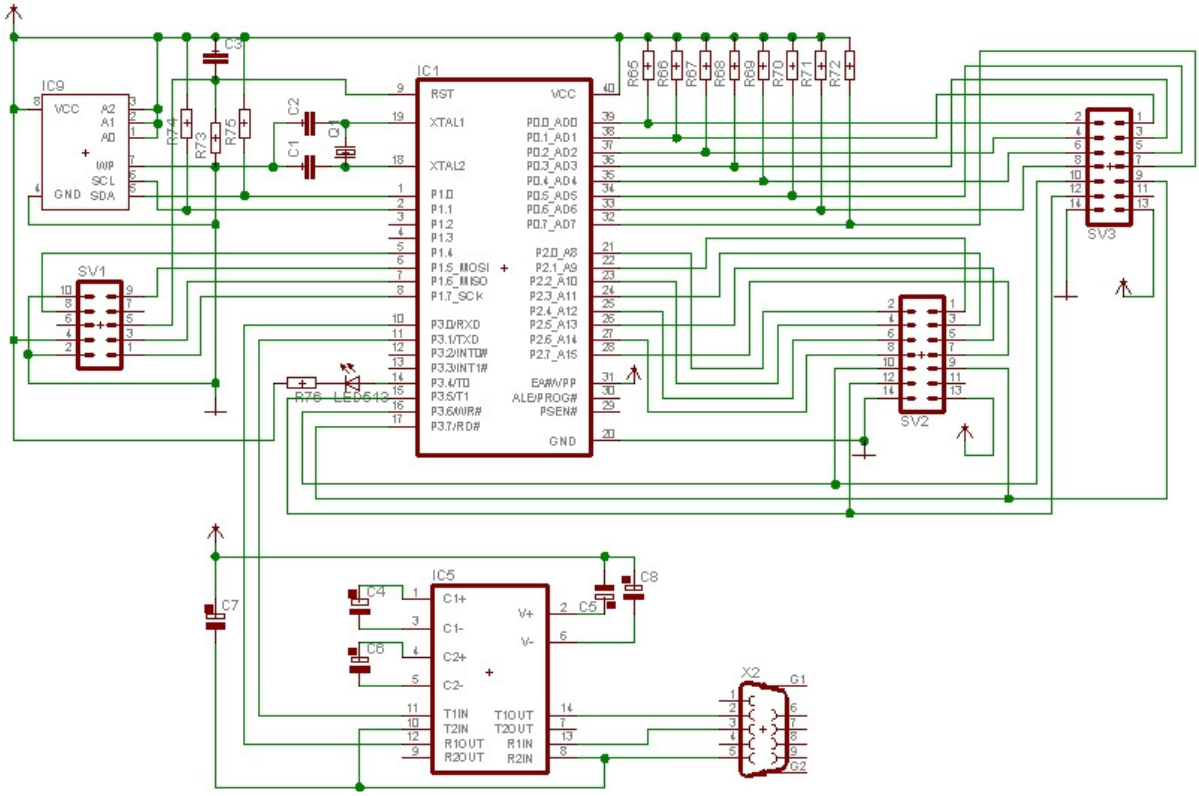
PC je pro sériovou komunikaci vybaven obvodem UART (universal asynchronous receiver transmitter). Ten při odesílání přebírá ze sběrnice PC paralelně data, serializuje je a odesílá po sériové lince. Při příjmu provádí přesně opačný postup. Jelikož ale přenos probíhá velmi rychle a ne vždy je možné je ihned odeslat, existuje zde ještě buffer, ve kterém se ukládají do fronty, aby se neztratili.

Na straně mikrokontroléru se nachází také obvod UART, který je ovšem vybaven pouze bufferem pro jeden byte. Tento buffer, označený jako SBUF je zde dvakrát, jeden pro záchyt přijatého bytu a druhý pro byte odesílaný. Sériový port lze nastavit jako zdroj hardwarového přerušení, které může být vyvoláno při odeslání bytu na linku či při přijetí bytu. Sériový kanál mikrokontroléru neobsahuje žádné příznaky pro indikaci chyby přenosu (žádná parita). Nepoužívá se zde ani hardwarové řízení toku dat, takže ke komunikaci stačí pouze 2 vodiče - RxD a TxD (viz obr. 1.9).

Jelikož je sériový port PC navržen podle standardu RS-232, který využívá pro přenos jiné napěťové úrovně než dávají výstupy mikrokontoléru, je nutné použít ještě převodník napětí. Tím se stane integrovaný obvod MAX232 firmy Maxim [6], který je schopný převést napěťové úrovně výstupů sériového portu na mikrokontroléru na vyšší díky nábojové pumpě.

1.2.4 Obvodové zapojení modulu

Obvodové zapojení modulu s mikrokontrolérem je uvedeno na obrázku 1.11.



Obr. 1.11 Schéma zapojení modulu s mikrokontrolérem

Jádrem obvodu je pochopitelně mikrokontrolér. Vývod RESET je připojen na RC článek, který zaručuje, že po připojení napětí bude na vývodu po krátkou dobu logická 1. Ta způsobí řádný restart mikrokontroléru, který je nutný, protože některé jeho části mohou být po startu v nedefinovaném stavu a mohlo by dojít k havárii programu. Na vývody XTAL1 a XTAL2 je připojen krystal doplněný

dvěma kondenzátory určující hodinovou frekvenci pro mikrořadič. Jeho hodnota však není závislá jen na dovoleném rozmezí mikrokontroléru. Jelikož je z ní následně odvozená i frekvence vnitřního časovače, který určuje frekvenci hodin u sériového portu, může nabývat jen určitých hodnot. Vstupně výstupní linky jsou využity téměř v celém rozsahu. Port 0 je použit pro sériové posílání dat do 8 posuvných registrů v matici LED. Tento port však neobsahuje vnitřní pull-up rezistory, proto musejí být nahrazeny vnějšími. Port 1 je zčásti obsazen sběrnicí I^2C vedoucí ke vnější paměti, další část zase slouží pro ISP. Port 2 obsahuje vývody sériového portu, které vedou do převodníku napětí. Tento port využívají i 3 řídící vodiče pro matici LED, jedná se o hodinový signál a signál STROBE pro posuvné registry a nulovací signál pro čítač 1 z n. Port 3 je použit jako záložní k portu 0, slouží také pro testování. V budoucnu by jej bylo možné použít pro druhou matici LED.

Adresní vstupy externí paměti jsou připojeny na kladné napětí, což znamená, že adresa paměti na sběrnici I^2C bude 7. Ochrana proti zápisu není využita, a proto je tento vývod trvale připojen k zemi. Vývody s hodinovým a datovým signálem jsou opatřeny pull-up rezistory.

2 Program pro mikrokontrolér

V této kapitole je popsán program pro mikrokontrolér. Kód je psán v assembleru s využitím nabízeného instrukčního souboru na bázi 8051. Překlad probíhal na překladači asm51.

Program je pro přehlednost rozdělen do několika souborů:

- news.asm - hlavní soubor, obsahuje rutiny a inicializace
- proc.asm - obsahuje pomocné podprogramy
- serial.asm - obsahuje podprogramy týkající se sériového portu
- memory.asm - obsahuje podprogramy pro komunikaci přes I^2C sběrnici

Celý program je založen na přerušovacím systému mikrokontroléru. Hlavní smyčka programu se proto skládá pouze z jediné skokové instrukce, které provádí skok sama na sebe. Hlavní smyčce programu předchází počáteční inicializace proměnných, načtení uložených informací o programu z paměti a nastavení sériového portu a časovačů. Přerušení jsou volána při přerušení od sériového portu a časovače 0. Podle vektoru přerušení poté následuje zpracování jednotlivých rutin pro tyto situace.

2.1 Komunikace s pamětí

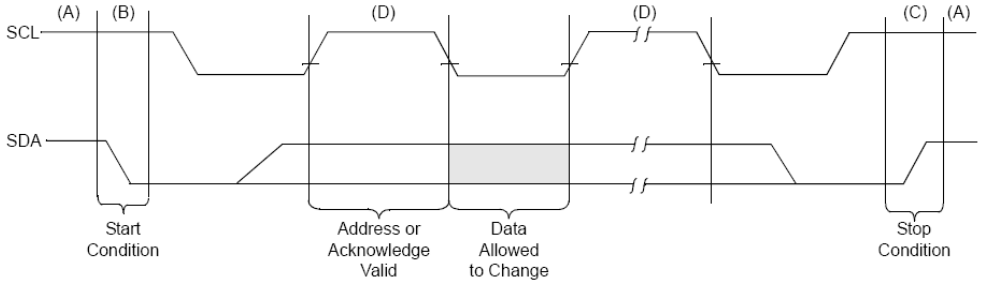
Jak už zde bylo uvedeno, paměť 24LC256 je spojená s mikrokontrolérem sběrnicí I^2C.

2.1.1 Charakteristika I^2C

Tato sběrnice umožňuje připojení jednoho či více procesorů (master) k periferním zařízením (slave). Všechny prvky na sběrnici jsou propojeny 2 vodiči. Prvním z nich je datový vodič SDA, který slouží pro oba směry toku dat, a dalším je vodič SCL s hodinovým signálem. Ten smí generovat pouze zařízení označené jako master (v tomto případě mikrokontrolér). Jelikož je na sběrnici možné připojit i více periferních zařízení, tak je nutné, aby každé mělo svou unikátní adresu. Ta je poslána při zahájení přenosu na začátku paketu. Pro situaci, kdy na stejnou sběrnici zapisuje více zařízení, je I^2C vybaveno detekcí kolizí. Oba vodiče sběrnice jsou zapojeny jako otevřený kolektor. Z toho také vyplývá nutnost připojení pull-up rezistorů ke kladnému napětí, což zajistí logickou 1 v klidovém stavu na obou vodičích. I^2C slouží k připojování spíše pomalejších periferií, protože přenos může probíhat pouze v jednom směru a maximální frekvence je omezena podle verze na 100kHz nebo 400kHz. Použitá paměť podporuje rychlost 400kHz.

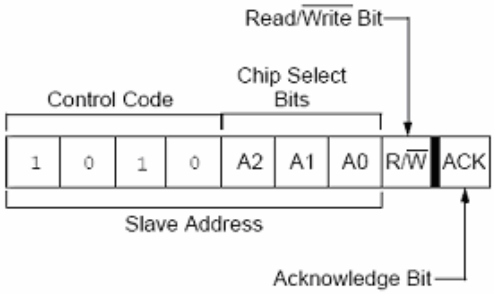
2.1.2 Popis komunikace

V protokolu I^2C jsou přesně definovány situace, podle kterých mohou zařízení poznat začátek a konec přenosu či klidový stav. Pokud žádné zařízení nevysílá, je SDA i SCL v log. 1. Jedině v tomto stavu lze zahájit přenos. Ten musí začínat startovací podmínkou, která je určena přechodem SCL z klidového stavu do log. 0. Konec přenosu zase indikuje stop podmínka, při které je nejprve datový a poté i hodinový vodič nastaven do log. 1. Mezi těmito podmínkami musí probíhat změna úrovní na datovém vodiči pouze při SCL v log. 0. Situace je znázorněna na obrázku 2.1.



Obr. 2.1 Zahájení a ukončení přenosu po I^2C

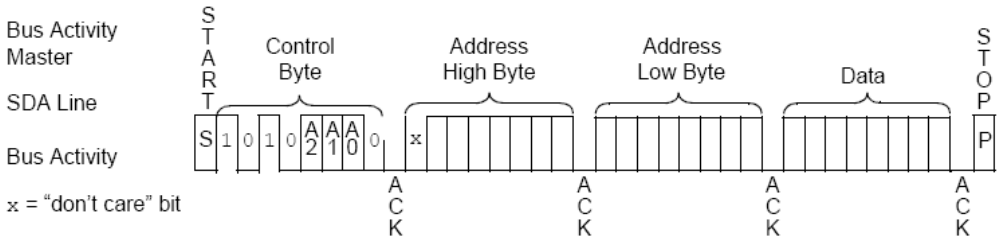
Po zahájení musí přijít kontrolní byte, který podle prvních 7 bitů určí, pro které slave zařízení je paket určen. Z této adresy jsou první čtyři byty dány pevně již výrobcem, zbývající je možno zvolit libovolně. Posledním bitem je určeno, zda se bude provádět čtení či zápis. Po kontrolním bytu musí slave potvrdit přijetí pomocí bitu ACK, tedy uvedením SDA do log. 0. I v tomto případě je generování nástupné hrany signálu SCL pro přečtení ACK v režii mastera. Toto potvrzování probíhá po každém odeslaném bytu.



. Obr. 2.2 Kontrolní byte s potvrzovacím bitem

2.1.2.1 Zápis bytu

Při zápisu bytu je nutné po start bitu a kontrolním bytu s příznakem pro zápis poslat 15-bitovou adresu požadované paměťové buňky. Ta je rozdělena na dva byty, po každém následuje ACK bit. Po adrese se odesílá datový byte od nejméně významného bitu. Přenos končí stop bitem.

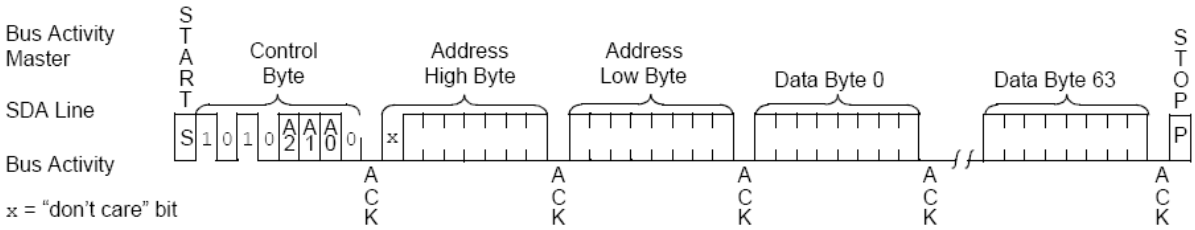


Obr. 2.3 Zápis bytu

Nyní by mohl nastat problém, pokud by hned za touto operací následovala další a zápis by v paměti ještě nebyl dokončen. Proto se zde vkládá dotazování na potvrzení zápisu (acknowledge polling). To probíhá tak, že po ukončení operace zápisu následuje další pokus o zápis, ten ovšem zůstává jen u poslání start bitu a příslušného kontrolního bytu. Pokud tento fiktivní zápis slave potvrdí, tak už je předchozí zápis hotov a lze provádět další operace.

2.1.2.2 Zápis stránky

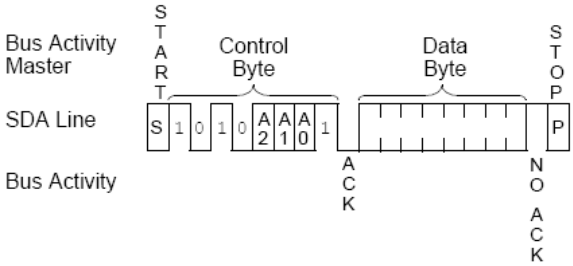
Do paměti lze zapisovat nejen po jednotlivých bytech, ale také po 64-bytových stránkách. K tomu slouží podobná konstrukce jako u zapisování bytu, pouze datových bytů je najednou odesláno 64.



Obr. 2.4 Zápis stránky

2.1.2.3 Čtení bytu na aktuální adrese

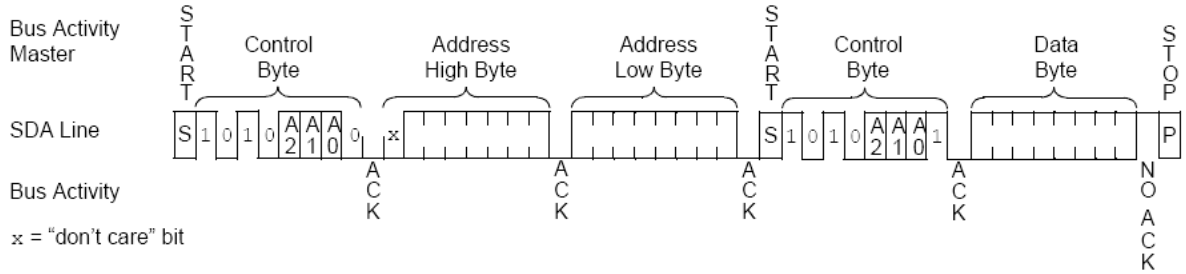
Tato operace slouží ke čtení bytu na adrese, která je nastavena předchozí operací. Každá operace čtecí či zapisovací po svém ukončení zanechá v paměti používanou adresu, kterou inkrementuje. Z tohoto důvodu se tato operace používá k sekvenčnímu čtení. Sestává se ze start bitu, kontrolního bytu s příznakem pro čtení a datového bytu, kdy ovšem vodič SDA neovládá master, ale slave. Zde je jediná výjimka, kde po tomto bytu nenásleduje bit ACK, ale rovnou stop bit.



Obr. 2.5 Čtení bytu na aktuální adrese

2.1.2.4 Čtení bytu

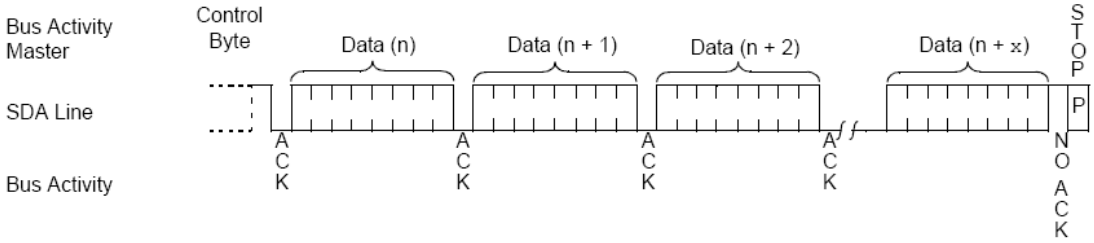
Pro přečtení bytu na požadované adrese je nutné použít fiktivní zápis na adresu, která má být přečtena. Fiktivní zápis se opět skládá ze start bitu a kontrolního bytu s příznakem pro zápis, po němž ale následuje další kontrolní byte tentokrát s příznakem pro čtení a poté datovým bytem a stop bitem. Pasáž od druhého kontrolního bytu se shoduje s předchozí operací.



Obr. 2.6 Čtení bytu na určité adrese

2.1.2.5 Čtení stránky

Stejně jako při zápisu existuje i u čtení možnost načíst celou stránku paměti najednou. Princip je také stejný, kdy se ke čtení stránky použije stejný postup jako u čtení bytu na určité adrese s tím rozdílem, že čtených datových bytů je 64. Zde je ovšem nutné počítat kromě posledního u všech datových bytů s potvrzovacím ACK bitem.



Obr. 2.7 Lišící se část operace čtení stránky

2.1.3 Implementace komunikace

Tato část komunikace využívající sběrnice I^2C je uložena v souboru memory.asm. Z operací uvedených v minulé části ale jsou implementovány jen některé. Čtení stránky či její zápis se ukazuje jako nepoužitelné z důvodu nedostatku interní paměti dat mikrokontroléru.

2.1.3.1 Podprogram ShiftTo

Pomocný podprogram posílá sériově na sběrnici I^2C byte uložený v akumulátoru od nejméně významného bitu.

2.1.3.2 Podprogram StartBit

Pomocný podprogram vytvoří na sběrnici startovací podmínku pro přenos.

2.1.3.3 Podprogram StopBit

Pomocný podprogram vytvoří na sběrnici stop podmínku pro signalizaci ukončení přenosu.

2.1.3.4 Podprogram SlaveAck

Pomocný podprogram slouží pro získání potvrzení o přijetí bytu od paměti. Nejdříve uvede SDA do klidového stavu a po impulsu na SCL přečte stav SDA do C (carry bit).

2.1.3.5 Podprogram NoAck

Pomocný podprogram slouží pro uvedení SDA do klidového stavu před stop bitem. Nahrazuje ACK u operací týkajících se čtení. SDA je uvedeno do klidového stavu a následuje impuls na SCL.

2.1.3.6 Podprogram AckPoll

Pomocný podprogram zjišťuje, zda je v paměti již dokončena předchozí operace. Pokud není, cyklus se opakuje. Opakování je možné nejvíce čtyřicetkrát. Konstanta ukládaná do akumulátoru představuje kontrolní byte s příznakem pro zápis.

```
AckPoll:
    mov   r7, #40h
AckPollCycle:
    djnz  r7, AckPollCont
    jmp   AckPollEnd
AckPollCont:
    call  StartBit
    mov   a, #10101110b
    call  ShiftTo
    call  SlaveAck
    jc    AckPollCycle
AckPollEnd:
    call  StopBit
    ret
```

2.1.3.7 Podprogram ByteWrite

Podprogram slouží k zápisu bytu uloženému v proměnné `data_byte` do paměti na adresu, jejíž horní byte je umístěn v proměnné `high_address` a spodní byte v `low_address`. Po ukončení zápisu se ještě provádí kontrola dotazováním na potvrzení.

```
ByteWrite:
    call  StartBit
    mov   a, #10101110b
    call  ShiftTo
    call  SlaveAck
    mov   a, high_address
    call  ShiftTo
    call  SlaveAck
    mov   a, low_address
    call  ShiftTo
    call  SlaveAck
    mov   a, data_byte
```

```
        call  ShiftTo
        call  SlaveAck
        call  StopBit
        call  AckPoll
        ret
```

2.1.3.8 Podprogram ByteReadCycle

Pomocný podprogram čte ze sběrnice sériově byte a ukládá je do proměnné. Probíhá tím způsobem, že po každém hodinovém impulsu na SCL je přečtena hodnota na SDA pomocí shift instrukce uložena do proměnné `data_byte`.

2.1.3.9 Podprogram ByteRead

Podprogram slouží k načtení bytu do proměnné `data_byte` z paměti na adrese, jejíž horní byte je umístěn v proměnné `high_address` a spodní byte v `low_address`. Po ukončení zápisu se ještě provádí kontrola dotazováním na potvrzení. Konstanty ukládané do akumulátoru představují v prvním případě kontrolní byte s příznakem pro zápis a v druhém kontrolní byte s příznakem pro čtení.

```
ByteRead:
    call  StartBit
    mov   a, #10101110b
    call  ShiftTo
    call  SlaveAck
    mov   a, high_address
    call  ShiftTo
    call  SlaveAck
    mov   a, low_address
    call  ShiftTo
    call  SlaveAck
    call  StartBit
    mov   a, #10101111b
    call  ShiftTo
    call  SlaveAck
    call  ByteReadCycle
    call  NoAck
    call  StopBit
    ret
```

2.1.3.10 Podprogram ByteCurrentRead

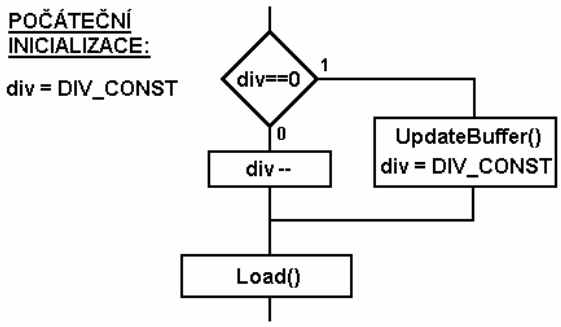
Podprogram slouží k načtení bytu do proměnné `data_byte` z paměti na adrese, která je v ní aktuálně nastavena po předchozí operaci.

```
ByteCurrentRead:
    call  StartBit
    mov   a, #10101111b
    call  ShiftTo
    call  SlaveAck
    call  ByteReadCycle
    call  NoAck
    call  StopBit
    ret
```

2.2 Rutina pro zobrazování

Všechny operace týkající se zobrazování jsou řešeny v této rutině, která je vyvolána při přerušení od časovače 0. Ten je pomocí svých ovládacích registrů nastaven do módu 16-bitového čítače, který používá vnitřní hodinový signál a čítá nahoru. Počet tiků do vyvolání přerušení je pevně dán konstantou zapsanou do modulo registru, přičemž frekvence je 1/12 hodinového signálu kmitočtu mikrořadiče. V tomto případě při frekvenci 11,059 MHz je časový interval mezi tiky asi 1µs.

Na obrázku 2.8 je zobrazen zjednodušený blokový diagram, z kterého je patrno, že při každém přerušení časovače je zobrazen obsah bufferu na matici LED funkcí Load (lépe řečeno jeden jeho multiplex) a jednou za určitý počet cyklů daný konstantou DIV_CONST je zavolána i funkce pro úpravu bufferu UpdateBuffer, která slouží k načítání nových sloupců do bufferu a jejich posun v něm tak, aby se text po promítnutí na matici posunul o jeden sloupec v určeném směru. Tyto funkce budou probrány dále. Tento způsob volání podprogramu UpdateBuffer byl zvolen v důsledku nedostatku časovačů mikrokontroléru. Ty jsou zde jen dva, jeden je použit zde a druhý slouží sériovému portu k vytváření hodinového signálu. Kromě této základní konstrukce je v této rutině nutné na začátku zálohovat na zásobník a na konci opět vyzvednout některé proměnné, protože ty mohou být používány i druhou rutinou pro sériový port. Ta má ovšem nižší prioritu, takže by mohlo nastat při přerušení od časovače situace, že by bylo nutné přerušit obsluhu přerušení sériového portu. V ten moment by mohlo dojít k přepsání některých proměnných, které musí být z různých důvodů společné.



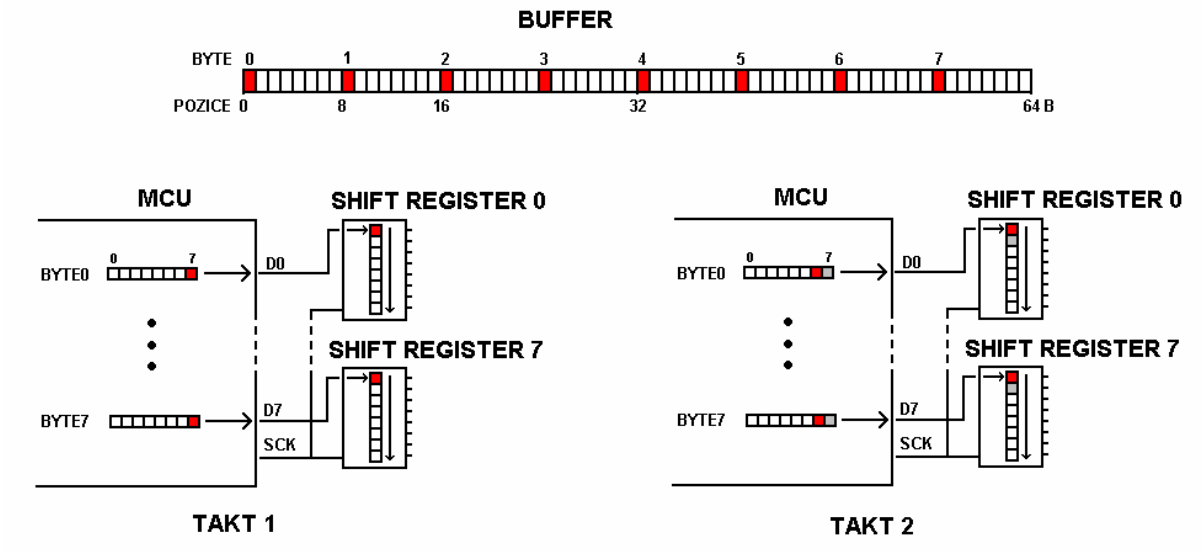
Obr. 2.8 Zjednodušený blokový diagram rutiny pro zobrazování

2.2.1 Zobrazení bufferu na matici LED

Operace se zobrazováním na matici tvoří základ celého programu. Jak už bylo vysvětleno v části 1.1, není možné zobrazovat najednou na celé matici LED. Proto tento algoritmus řeší pouze rozsvícení LED diod v jednom multiplexu. Postupným voláním této funkce pro všechny multiplexy v rychlém sledu je možné alespoň fiktivně pro lidské oko dosáhnout rozsvícení všech LED v matici, které mají svítit.

Pro uchování informace o právě zobrazovaných znacích je nutné tuto informaci uložit do paměti. Pro tyto data označené jako buffer bylo alokováno ve vnitřní paměti RAM 64 bytů. Každý

byte bufferu představuje vzor pro jeden sloupec na matici LED a každý bit tohoto bytu je vzorem pro jednu LED z tohoto sloupce. Pro pochopení operací potřebných k poslání dat do matice je nutné představit si, jak jsou v bufferu organizovány jednotlivé byty. Rozmístění je zobrazeno v horní části obrázku 2.9. Zvýrazněné paměťové buňky zde odpovídají sloupcům jednoho multiplexu Tyto byty je nutné sériově přenést do posuvných registrů. Tato operace je zobrazena ve spodní části obrázku 2.6. Nejdříve jsou získány nejvyšší bity jednotlivých bytů. Z těchto osmi bitů je vytvořen byte, který je poslán na vstupně výstupní port, na který jsou připojeny datové vodiče posuvných registrů (obr. 1.3). Poté následuje impuls hodinového signálu, kterým řídí mikrokotrolér posuvné registry. V tomto momentu jsou přeneseny první bity. Celý tento krok se poté opakuje ještě sedmkrát. Jakmile dojde k přenesení všech 8 bitů do všech registrů a přivedení jejich logických hodnot na výstupy, bude svítit celý jeden multiplex. Pro rozsvícení dalších platí stejný postup, jen jsou na začátku vybrány následující byty po právě přenesených z bufferu.



Obr. 2.9 Způsob organizace bytů v bufferu v paměti a situace při prvních dvou krocích nahrávání dat pro sloupce jednoho multiplexu do posuvných registrů

Implementace algoritmu je umístěna v podprogramu Load v souboru news.asm. U tohoto podprogramu je velmi kladen důraz na efektivnost a rychlost algoritmu, aby bylo možné dosáhnout požadované frekvence u zobrazování na matici.

Pro správné fungování celého podprogramu je důležitá proměnná `visible_col` obsahující číslo sloupce, který je právě v jednotlivých segmentech matice rozsvěcován. Po sečtení s adresou bufferu v paměti a pomocné proměnné `new_col`, jejíž význam bude objasněn v následující části, vznikne adresa sloupce v prvním segmentu. Další segmenty leží na adresách zvětšených vždy o násobek osmi.

```
mov   a, visible_col
add   a, new_col
anl   a, #007h
add   a, #buffer

mov   r0, a
mov   num0, @r0
add   a, #8
```

Tímto způsobem je získáno z bufferu všech 8 bytů s informacemi pro zobrazované sloupce. Následně je z bytů pomocí rotace přemístěn požadovaný bit do příznaku carry. Hodnota z něj je již vystavena na výstup a po hodinovém impulsu CLK zapsána do posuvného registru. Akce je opakována ještě sedmkrát (pro d1 - d7).

```
mov   a, num0
rlc   a
mov   num0, a
mov   d0, c
setb  clk
clr   clk
```

Poté následuje impuls potvrzovací signálu STROBE, který v posuvných registrech způsobí převedení přijatých hodnot na výstupy. Pomocná proměnná `visible_col` je zvětšena o jednu, která způsobí při příštím spuštění podprogramu rozsvícení dalšího multiplexu.

```
setb  strobe
clr   strobe
inc   visible_col
anl   visible_col, #007h
```

2.2.2 Pohyb a načítání textu

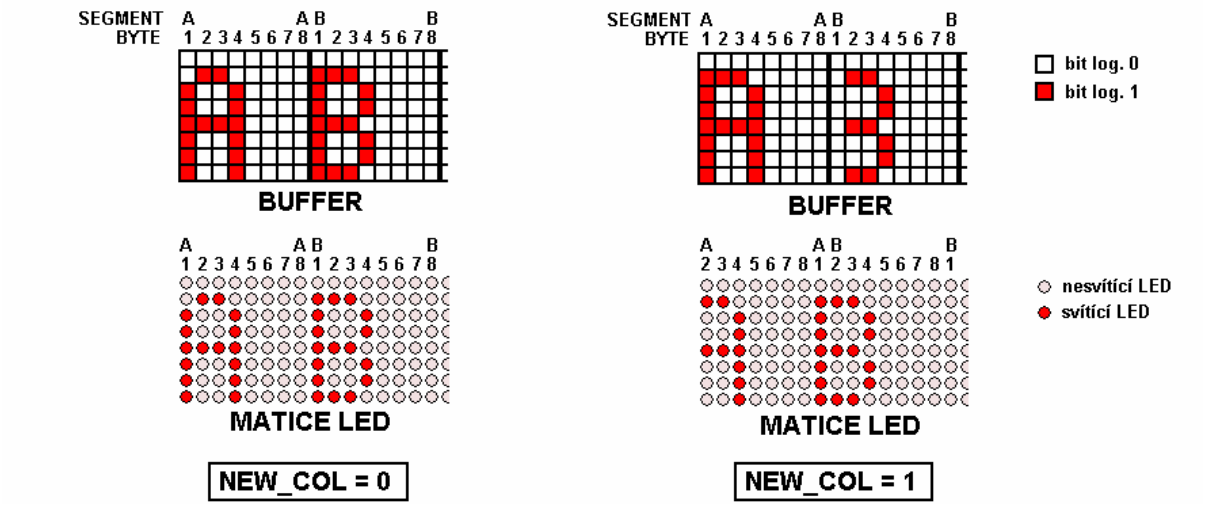
Implementace těchto částí je umístěná v podprogramu UpdateBuffer() v souboru proc.asm.

2.2.2.1 Pohyb

Pohyb textu je implementován pro 4 směry. Kromě horizontálního pohybu zprava doleva je možné, aby text roloval po matici zleva doprava nebo vertikálně oběma směry. Jelikož všechny ostatní směry vychází právě z pravolevého pohybu, bude zde pospán právě ten.

Pohyb na matici LED se provádí posunem sloupců o jednu pozici ve směru pohybu. Tato operace s bufferem by byla ale příliš náročná a neefektivní. Proto je zvoleno řešení, kdy zůstávají téměř všechny byty bufferu na svém místě. Je zde použita proměnná `new_col`, která představuje ukazatel na byte v segmentu, který bude pro podprogram Load (viz 2.2.1) představovat první sloupec daného segmentu na matici. Pohyb je tedy zajišťován nejenom v tomto podprogramu, ale i podprogramem Load a společnou proměnnou `new_col`. Na obrázku 2.10 jsou znázorněny dva kroky pohybu. Je zde vidět, že při posunu znaků na matici nedochází k posunu všech bytů v bufferu. Přesouvají se pouze byty umístěné na indexu odpovídající proměnné `new_col` v každém segmentu a

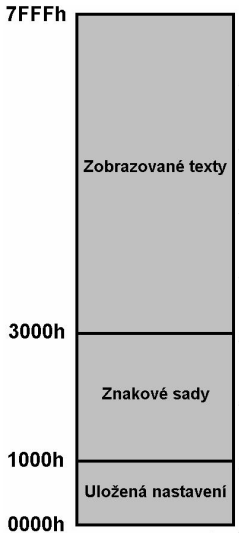
to do sousedního segmentu ve směru pohybu textu. Sloupec na konci je zahozen a nový sloupec na začátku je získán z externí paměti z rozkladových sloupců načítaného znaku v požadované znakové sadě (viz 2.2.2.2).



Obr. 2.10 Ilustrace provázání bufferu v paměti a zobrazovaného textu na matici

2.2.2.2 Načítání textu

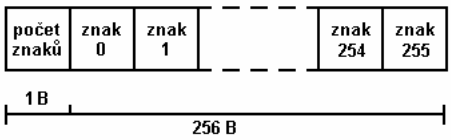
Všechna data jako texty, znakové sady a další nastavení jsou uloženy v externí paměti. Její organizace je znázorněna na obrázku 2.11. V paměti je vyhrazeno místo pro 80 různých textů a 8 znakových sad. K načtení nového sloupce, který bude vložen na místo posledního sloupce v matici, se používá následující postup.



Obr. 2.11 Mapa rozdělení externí datové paměti

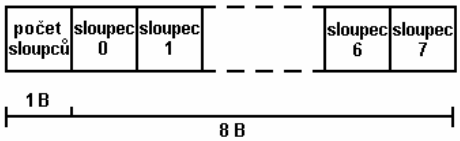
Nejdříve je nutné určit znak, jehož je částí. Ty jsou postupně načítány z vybraného textu. Formát takového uloženého textu je vidět na obrázku 2.12. Na první pozici je uložena délka textu, která může dosahovat až 255 znaků. Dále se nachází hodnoty znaků, které by se daly rozdělit na zobrazitelné a speciální znaky. Zobrazitelné znaky, které mají hodnoty od 0 do 127, mají své

rozkladové sloupce umístěny ve znakové sadě. Speciální znaky slouží k speciálním účelům, jako je třeba nastavení znakové sady apod. Jejich podrobnější popis je v části 2.2.3.



Obr. 2.12 Formát textu v paměti

Vybraný zobrazitelný znak slouží jako ukazatel do paměti do znakové sady na místo, kde jsou umístěny jeho rozkladové sloupce. Formát jednoho záznamu ve znakové sadě je zobrazen na obrázku 2.13. Opět je zde na první pozici byte, který nese informaci o počtu, tentokráte sloupců. Pro každý znak je sice ve znakové sadě vyhrazeno pouze 8 bytů, lze však využít prostoru pro více záznamů za sebou pro uložení dat o jednom dlouhém znaku. Délka takového znaku je potom omezena na 255.



Obr. 2.13 Formát znaku v paměti

Po přečtení ze znakové sady je žádaný sloupec představovaný jedním bytem uložen v bufferu na místo, které bude zobrazeno jako poslední sloupec v matici. Poté je proměnná ukazující na aktuální sloupec inkrementována. Pokud její hodnota přesáhne počet sloupců ve znaku, je vynulována a inkrementována proměnná ukazující na znak v textu. Pokud ta překročí počet znaků v textu, je vynulována a celý text načítán znovu.

2.2.3 Speciální funkce

Zobrazení, pohyb a načítání textu lze ovlivnit pomocí několika implementovaných funkcí. Spouštění a parametry těchto funkcí lze nastavit pomocí speciální znaku uloženého v právě zobrazovaném textu. Některé funkce jdou nastavit také přímo pomocí sériového portu. O této možnosti je zmínka v části 2.3.

2.2.3.1 Přehled funkcí

Zde je stručný popis všech funkcí.

Změna znakové sady

Jelikož zobrazitelný znak může nabývat pouze jedné ze 128 hodnot, je přidána možnost zvolit další znakové sady. Těch je možné v paměti uložit 8 a tím se počet znaků rozšíří na 1024. Číslo znakové

sady je uloženo v proměnné po celou dobu načítání z jednoho textu. Pokud dojde k její změně, zůstává tato nová hodnota platná do další změny nebo do konce textu.

Odkaz na jiný text

Tímto speciálním znakem v textu lze zřetězit několik jednotlivých textů a obejít tak omezení každého textu na 255 znaků. Tento znak má smysl pouze při vložení jako poslední znak textu, protože ihned po jeho načtení je načítání znaků přesměrováno na odkazovaný text.

Změna rychlosti pohybu textu

Rychlost rolování textu lze nastavit buď speciálním znakem či přímo přes sériový port. Změny rychlosti je dosaženo tím, že se změní proměnná obsahující počet cyklů, za který je volán podprogram UpdateBuffer (viz obr. 2.8).

Změna směru pohybu textu

Tato změna směru znamená, že text je bude rolován horizontálně nebo vertikálně. Jeho orientace je pak nastavována zvlášť. Horizontální pohyb zde byl již popsán. Vertikální probíhá tak, že nejprve je do bufferu v 64 krocích načten text jako při horizontálním pohybu, který je potom pomocí měnící se masky a rotace bitů v jednotlivých bytech v bufferu v 16 krocích postupně zobrazován. Tím vzniká dojem, že text naroluje a odroluje ve vertikálním směru.

Změna orientace pohybu textu

Pokud je požadováno, aby text roloval zleva doprava či ve vertikálním směru zdola nahoru, pak je nutné nastavit tento příznak. U horizontální změny orientace textu je v programu dosaženo tím, že jsou byty v bufferu, na něž ukazuje proměnná `new_col`, přesouvány zleva doprava, u vertikální se změní směr při rotaci bitů v jednotlivých bytech bufferu.

Otočení znaku

Slouží k otočení znaků kolem své svislé osy. Při načítání sloupců znaku se začíná na posledním a pokračuje až k prvnímu.

Čekání na stisknutí tlačítka

Při nastavení tohoto příznaku přestane text rolovat, dokud není zmáčknuto tlačítko připojené na vývod 3 na portu 3 mikrokontroléru.

Konec řádku

Tato funkce najde uplatnění především při vertikálním pohybu textu. Zabraňuje tomu, aby byl text, který se celý nevejde najednou na matici, rozdělen uprostřed znaku. Toto ovšem musí hlídat uživatel a vložit na patřičná místa textu tento speciální znak. Tohoto chování je dosaženo tak, že je od prvního sloupce prvního znaku v textu v pomocné proměnné počítán počet sloupců. Ten je vždy po 64 (velikost matice) nulován. Pokud je nastaven příznak konce řádku, je místo nových sloupců z textu

doplňován na místo nového sloupce prázdný. Jakmile je pomocná proměnná nulována, je vynulován i tento příznak.

Nulování konce řádku

Slouží pro vynulování pomocné proměnné, která počítá počet sloupců textu. Její použití je vhodné společně s příznakem konce řádku, kdy bude při rolování do textu vložena mezera o velikosti matice, tedy 64 prázdných sloupců.

2.2.3.2 Spouštění funkcí pomocí speciálního znaku v textu

V tabulce 2.1 jsou vyobrazeny formáty speciálních znaků, které při načtení z textu způsobí zavolání příslušné funkce.

Speciální znak (7 … 0)	Funkce
1 1 A_5 A_4 A_3 A_2 A_1 A_0	odkaz na text, adresa odkazovaného textu na A_0 až A_5
1 0 1 S_4 S_3 S_2 S_1 S_0	změna rychlosti pohybu textu, rychlost na S_0 až S_4
1 0 0 1 X X X X	nevyužito
1 0 0 0 1 C_2 C_1 C_0	změna znakové sady, číslo sady na C_0 až C_2
1 0 0 0 0 1 1 1	čekání na stisknutí tlačítka
1 0 0 0 0 1 1 0	změna směru pohybu textu
1 0 0 0 0 1 0 1	změna orientace pohybu textu
1 0 0 0 0 1 0 0	otočení znaků
1 0 0 0 0 0 1 1	konec řádku
1 0 0 0 0 0 1 0	nulování konce řádku
1 0 0 0 0 0 0 1	nevyužito
1 0 0 0 0 0 0 0	nevyužito

Tabulka 2.1 Speciální znaky

2.3 Rutina pro sériový port

Tato rutina obsluhuje přerušení, které pochází od sériového portu. Toto přerušení je voláno ve dvou případech. Prvním je vyprázdnění bufferu sériového portu mikrokontroléru po odeslání bytu, druhým přečtení bytu ze sériové linky do tohoto bufferu. Rozlišení těchto dvou původů přerušení je možné pomocí registru SCON, ve kterém je příznak RI nastavovaný při přijetí a příznak TI při

odeslání bytu. Tato rutina reaguje pouze na přijetí bytu. V případě, že přerušení vyvolalo odeslání bytu, je rutina ukončena.

2.3.1 Nastavení parametrů komunikace

Před použitím sériového kanálu mikrokontroléru je nutné nastavit jeho parametry. K tomu slouží několik registrů. Nejdůležitějším je řídící registr SCON, kde je potřeba nastavit jeden ze 4 režimů. Pro toto zapojení je vybrán 8bitový asynchronní režim řízený zabudovaným časovačem 1. Dalším důležitým parametrem je rychlost přenosu dat. Ten je závislý na hodinové frekvenci mikrořadiče, obsahu registru TH1 a bitu SMOD v registru PCON. Rychlost, která vzejde z těchto parametrů musí odpovídat některé ze standardních hodnot, které lze nastavit u obvodu UART v PC. Zde je rychlost i vzhledem k frekvenci oscilátoru nastavena na dostačujících 19200 Bd.

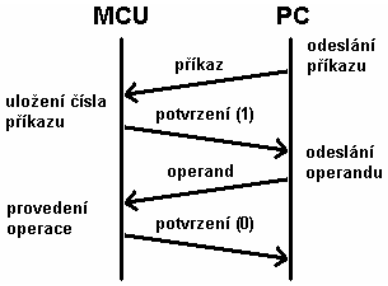
Vzorec pro přenosovou rychlost (PR): $\mathbf{PR} = \frac{2^{SMOD}}{32} \bullet \frac{f_0}{12(256 - TH1)}$ kde f_0 je frekvence mikrořadiče

2.3.2 Popis komunikace

Veškerá komunikace s modulem je řízena program na PC. Pokud je potřeba, PC vyšle zprávu, mikrokontrolér ji přijme, zpracuje a odešle odpověď. Zprávy z PC se dělí na 3 typy, příkaz s operandem, příkazy bez operandu a dotazy.

2.3.2.1 Příkazy s operandem

Jedním typem zpráv jsou příkazy, které vyžadují odeslání operandu v další zprávě. Po tomto dotazu je z mikrokontroléru odesláno potvrzení v podobě hodnoty 1. Zároveň je do pomocné proměnné uloženo číslo dotazu. Při přijetí dalšího bytu, kterým je operand, je podle typu dotazu provedena požadovaná operace s využitím tohoto operandu. Po úspěšném provedení je odeslána odpověď v podobě hodnoty 0. Pokud je operand chybný nebo dojde k jiné chybě, je odpověď rovna 255. Průběh takové komunikace je znázorněn na obrázku 2.14.



Obr. 2.14 Komunikace mezi mikrokontrolérem a PC u příkazu s operandem

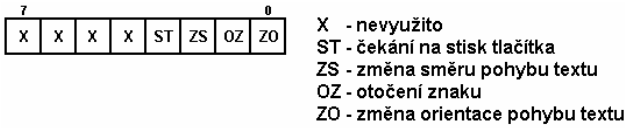
Zde je přehled všech příkazů s operandem. Číslo v závorkách značí číslo příkazu.

Nastavení čísla textu (1)

Do proměnné `text_num`, která obsahuje číslo právě zobrazovaného textu, je uložen obsah operandu. Zároveň je text právě zobrazený na matici vymazán a spuštěn nový.

Změna nastavení (2)

Obsah operandu je uložen do proměnné `settings` a tím dojde ke změně některých nastavení. Obsah proměnné `settings` je na obrázku 2.15.



Obr. 2.15 Proměnná `settings`

Nastavení rychlosti (3)

Do proměnné `div_const_low` obsahující počet cyklů, za který je volán podprogram UpdateBuffer, je uložen obsah operandu.

Nastavení čítače (4)

Čítač neboli proměnná `serial_counter` slouží zjednodušení operací, které jsou prováděny sekvenčně. Jeho obsah je nahrazen operandem.

Nastavení pracovního čísla znaku (5)

Proměnné označené jako pracovní jsou použity pouze v této rutině, mají ale stejný význam jako podobně pojmenované proměnné při zobrazování. Používají je další operace. V této je nahrazeno operandem pracovní číslo znaku v proměnné `serial_char_num`. Operand smí být pouze v rozsahu 0 až 128.

Nastavení pracovního čísla textu (6)

Nahrazení pomocné pracovní proměnné s číslem textu `serial_text_num` hodnotou operandu. Ten musí být v rozmezí 0 až 79.

Nastavení pracovního čísla znakové sady (7)

Nahrazení pomocné pracovní proměnné s číslem znakové sady `serial_charset_num` hodnotou operandu. Ten musí být pouze v rozmezí 0 až 7.

Nastavení znaku v textu (8)

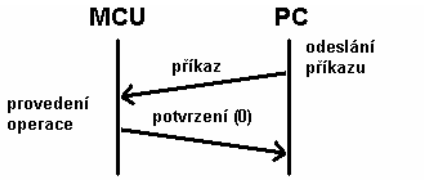
Tento příkaz způsobí zápis hodnoty operandu do paměti na místo, které je dáno součtem adresy textu s číslem odpovídajícím proměnné `serial_text_num` v paměti a obsahem čítače `serial_counter`. Zde je vidět použití čítače, který se po uložení hodnoty sám inkrementuje a je možné ho použít ihned pro zápis dalšího bytu.

Nastavení sloupce znaku ve znakové sadě (9)

S tímto příkazem lze nahradit hodnotu sloupce znaku ve znakové sadě hodnotou operandu. Místo v paměti, kam bude obsah operandu uložen, se počítá jako součet adresy znakové sady s číslem z proměnné `serial_charset_num`, čísla znaku z proměnné `serial_char_num` vynásobené 8, tedy počtem sloupců znaku a obsahem čítače `serial_counter`. Opět je možné ihned použít tento příkaz znovu pro zápis následujícího sloupce.

2.3.2.2 Příkazy bez operandu

Druhým typem zpráv z PC jsou příkazy, které nevyžadují operand. Operace je v mikrokontroléru ihned provedena a odeslána odpověď o úspěšném dokončení s hodnotou 0.



Obr. 2.16 Komunikace mezi mikrokontrolérem a PC u příkazu bez operandu

Zde je přehled všech příkazů bez operandu.

Vymazání textu (10)

Tento příkaz má za následek, že text s číslem z proměnné `serial_text_num` se bude jevit jako prázdný. Ve skutečnosti dojde pouze k vynulování prvního bytu v textu, který obsahuje počet znaků.

Vymazání znakové sady (11)

Dojde v smazání znakové sady, jejíž číslo je dáno proměnnou `serial_charset_num`. Opět nedochází ke smazání všech buněk paměti, na kterých daná znaková sada leží, ale pouze vynulování prvních bytů u všech 128 znaků.

Restartování rolování textu (12)

Způsobí vymazání bufferu a tím i zhasnutí všech LED diod matice a nastavení příznaku pro spuštění nového textu.

Vymazaní bufferu (12)

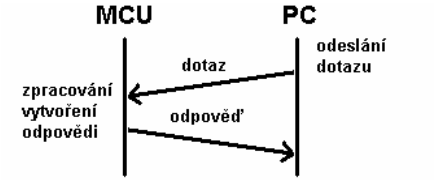
Pouze vymaže buffer, načítaní textu pokračuje dál.

Zápis defaultní znakové sady (13)

Zapíše do znakové sady s číslem z proměnné `serial_charset_num` hodnoty z defaultní znakové sady, které jsou uloženy jako konstanty v paměti programu.

2.3.2.3 Dotazy

Posledním typem zpráv, které mohou z PC do mikrokontroléru přijít, jsou dotazy, které požadují zaslání hodnoty proměnné či obsahu paměti v odpovědi. Ta je ve většinou jednobytová, ve 2 případech je ale tato odpověď vícebytová z důvodu zvýšení rychlosti a omezení režie operace. Je zde využito vícebytového bufferu v obvodu UART na straně PC.



Obr. 2.17 Komunikace mezi mikrokontrolérem a PC u dotazu

Zde je opět přehled všech dotazů. Většina jich tvoří protiklad k nastavovacím příkazům s operandem.

Získání čísla textu (129)

Jako odpověď pošle obsah proměnné `serial_text_num`.

Získání nastavení (130)

Odpovědí bude obsah proměnné `settings`.

Získání rychlosti (131)

Odpovědí bude obsah proměnné `div_const_low`.

Získání hodnoty čítače (132)

Odpovědí bude obsah proměnné `serial_counter`.

Získání hodnoty pracovního čísla znaku (133)

Odpovědí bude obsah proměnné `serial_char_num`.

Získání hodnoty pracovního čísla textu (134)

Odpovědí bude obsah proměnné `serial_text_num`.

Získání hodnoty pracovního znakové sady (135)

Odpovědí bude obsah proměnné `serial_charset_num`.

Získání hodnoty znaku z textu (136)

Protiklad k zápisu znaku do textu. S využití proměnných `serial_text_num` a `serial_counter` spočítá adresu a přečte hodnotu znaku. Tu pošle jako odpověď.

Získání hodnoty sloupce znaku ze znakové sady (137)

Opět tvoří protiklad k zápisu sloupce znaku. Adresa je spočtena z proměnných `serial_charset_num`, `serial_char_num` a `serial_counter` a hodnota sloupce odeslána jako odpověď.

Získání textu (138)

Na tento dotaz je postupně odesláno všech 256 bytů, které tvoří záznam textu. Číslo textu je určeno proměnnou `serial_text_num`. Jelikož se jedná o relativně náročnou operaci, je při ní zastaven pohyb textu a zakázáno další přerušení od sériového portu. V ten moment přistupuje k externí paměti pouze tato rutina a může využít její rychlé sekvenčního čtení.

Získání znakové sady (139)

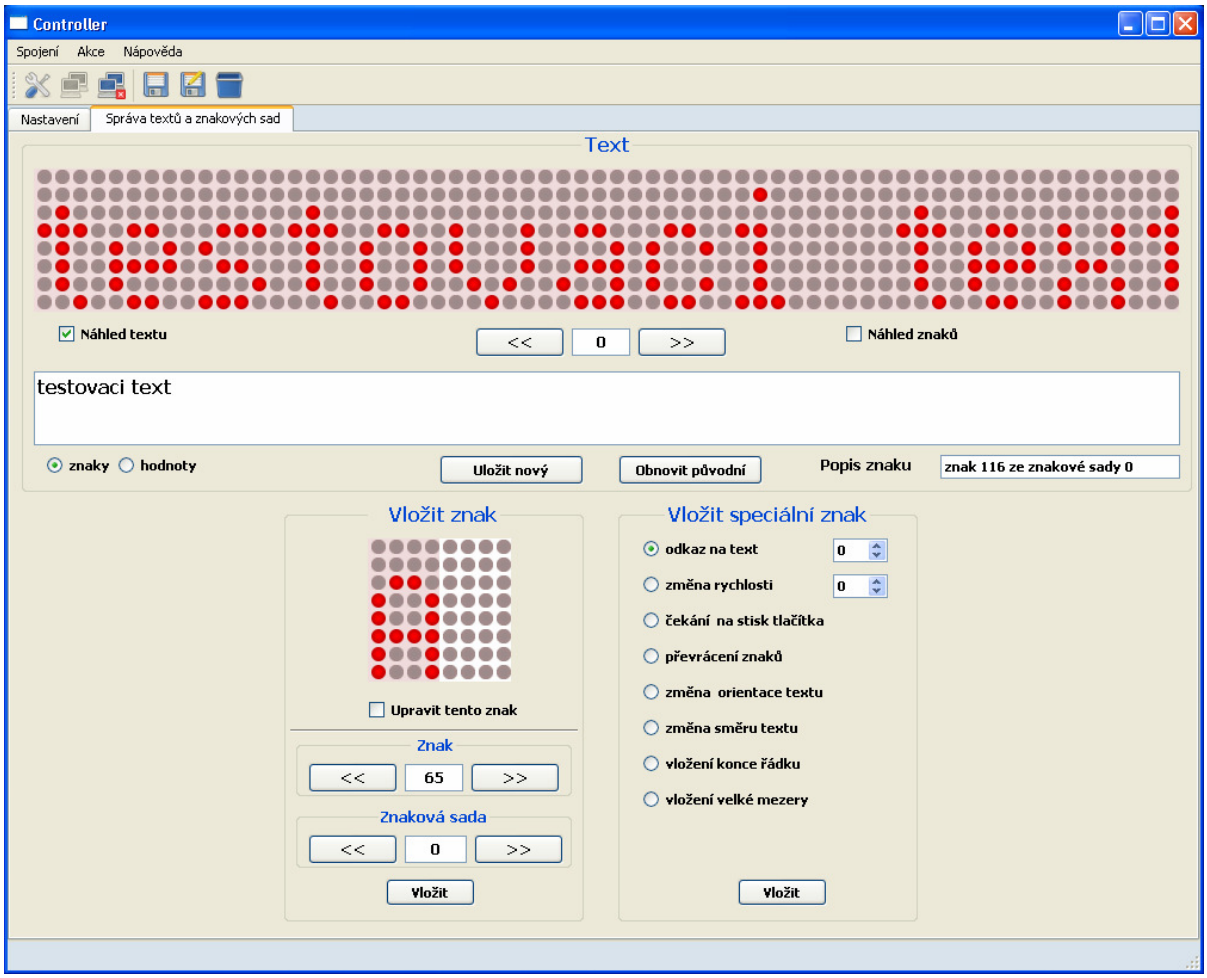
Tento dotaz způsobí odeslání všech 1024 bytů znakové sady určené proměnnou `serial_charset_num`. Opět je při odesílání zastaven pohyb textu a zakázáno přerušení od sériového portu.

3 Ovládací aplikace pro PC

Pro ověření funkčnosti informačního panelu byla realizována ovládací aplikace Controller pro PC. Ta byla naprogramována v jazyce C++ a určena pro operační systém Windows. Pro tvorbu GUI byl využit multiplatformní toolkit Qt verze 4.3 vyvíjený společností Trolltech [11].

Aplikace umožňuje nahrávání textů a znakových sad do panelu i jejich čtení a také změnit nastavení a parametry. Disponuje také simulátorem matice LED, takže je možné si upravovaný text před uložením do paměti v panelu nejdříve prohlédnout. Další výhodou je implementování ukládání dat přečtených z paměti v panelu do paměti aplikace a při vypnutí programu do souboru. Tak je možné při opakované potřebě použití dat, jako třeba textu či znakové sady, zabránit nutnosti je znovu stahovat.

Po spuštění aplikace je nutné zadat parametry připojení k panelu přes sériový kanál. Stiskem potvrzovacího tlačítka se spustí hlavní okno programu. Pro práci přímo s panelem je nutné se nejdříve připojit. Pokud je linka volná, následuje zobrazení všech komponent programu.

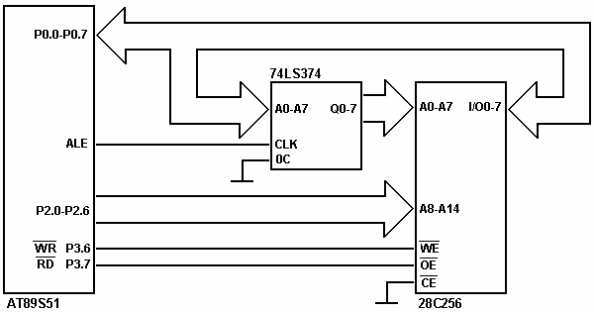


Obr. 3.1 Hlavní okno aplikace Controller

4 Závěr

Zde prezentované řešení se ukázalo jako plně funkční a splňující zadané požadavky. Podařilo se vyřešit problém se zapojením velkého množství světel na omezený počet vstupně/výstupních linek mikrokontroléru a to bez nepříjemných vedlejších efektů, jakými mohou být při použití časového multiplexování blikání sloupců při nedostatečné frekvenci.

Z pohledu dalšího vývoje by mohla být například externí sériová paměť EEPROM pro data připojená k mikrokontroléru nahrazena externí paralelní paměti, která je řádově rychlejší. Tu by bylo možné použít i pro proměnné programu, které jsou v nynějším zapojení z důvodu rychlého přístupu pouze v interní paměti a jimiž je paměť již zaplněna. S větším množstvím rychlejší paměti by bylo možné implementovat další funkce či například použít nyní nevyužitý port 2 pro připojení další matice LED stejné velikosti. Hlavní nevýhodou připojení této paralelní paměti je složitější návrh zapojení a využití vyššího počtu vstupně/výstupních linek mikrokontroléru. Příklad zapojení paralelní paměti 28C256 k mikrokontroléru je na obrázku 4.1.



Obr. 4.1 Připojení paralelní paměti 28C256

Dalším možným rozšířením by mohlo být rozšíření ovládací aplikace i na jiné platformy. Díky použití toolkitu Qt je GUI možné použít i jinde, pro rozšíření by bylo nutné pouze použít jiné ovládací funkce pro sériový port.

Literatura

[1] Atmel: AT89S51 Datasheet, 2005.

[2] Texas Instruments: CD4094B Datasheet, 2003.

[3] Texas Instruments: CD4022B Datasheet, 2004.

[4] SGS-Thomson Microelectronics: ULN2803 Datasheet, 1997.

[5] Microchip: 24AA256/24LC256/24FC256 Datasheet, 2007.

[6] Maxim: MAX232 Datasheet, 2003.

[7] Matoušek, D.: Práce s mikrokontroléry ATMEL AT89C2051 - 1. díl. 2. vydání. BEN, Praha, 2002. ISBN 80-7300-094-6.

[8] Fronc, V., Kľúčik, J.: Mikrokontroléry ATMEL s jádrem 8051. 1. vydání. BEN, Praha, 2001. ISBN 80-7300-008-3.

[9] Axelson, J.: Serial Port Complete. 3. vydání. Lakeview Research, Madison, 2000. ISBN 0-9650819-2-3.

[10] Dalheimer, M. K.: Programming with Qt. 1. vydání. O´Reilly, Cambridge, 2002. ISBN 0-596-00064-2.

[11] Trolltech: Qt Reference Documentation. 2008. Dokument dostupný na URL http://doc.trolltech.com/4.3/index.html (duben 2008)

[12] Herring, D.: AndroDNA - Qt tutorial. 2006. Dokument dostupný na URL http://www.androdna.com/programming/QtTutorial/(duben 2008)

[13] Microsoft: MSDN: Serial Communication in Win32. 2008. Dokument dostupný na URL http://msdn2.microsoft.com/library/ms810467.aspx (duben 2008)

[14] DH servis: Sběrnice I^2C, o co jde a jak pracuje. 2008. Dokument dostupný na URL http://www.dhservis.cz/iic.htm (duben 2008)

[15] Zapletal, R.: Mfc a C++: Komunikace pomocí sériového portu. 2005. Dokument dostupný na URL http://mfc.chytrak.cz/sercomm.php (duben 2008)

Seznam příloh

Příloha 1. Schémata a návrhy desek plošných spojů

Příloha 2. Funkční vzorek

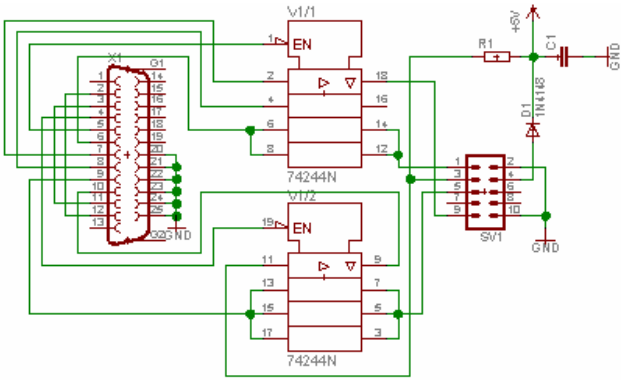
Příloha 3. Obsah CD

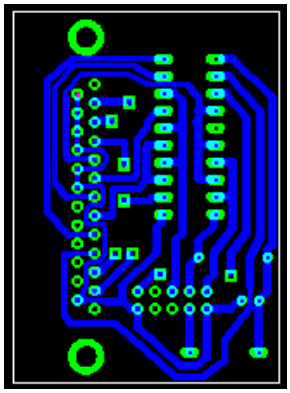
Příloha 4. CD

Příloha 1

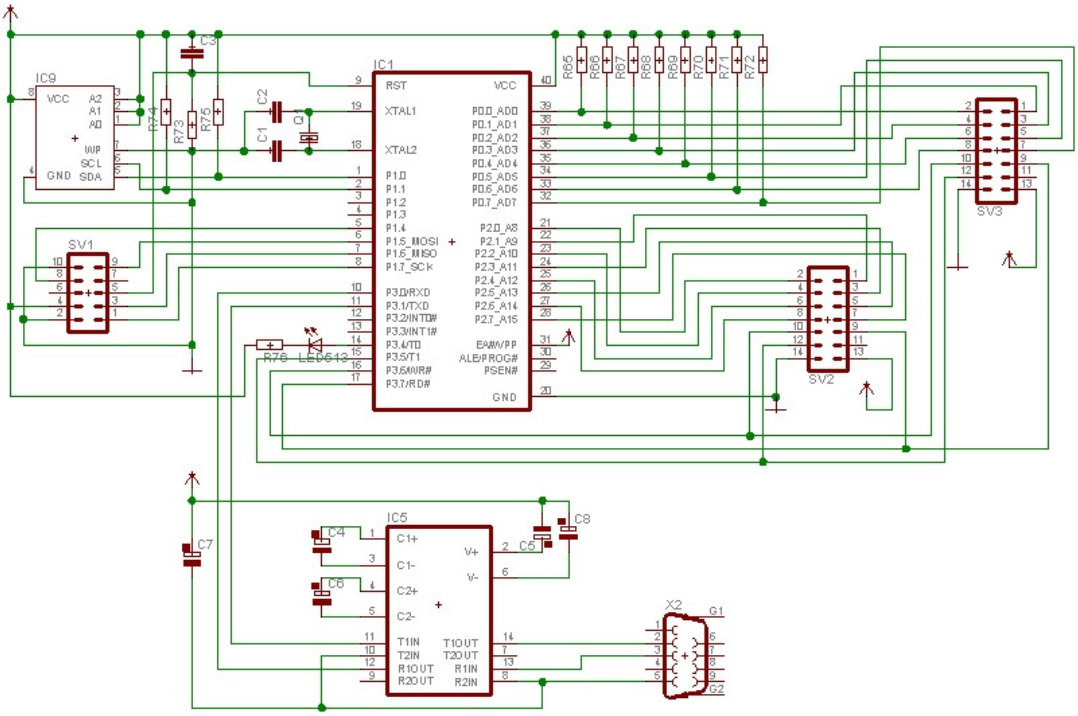
Schémata a návrhy desek plošných spojů

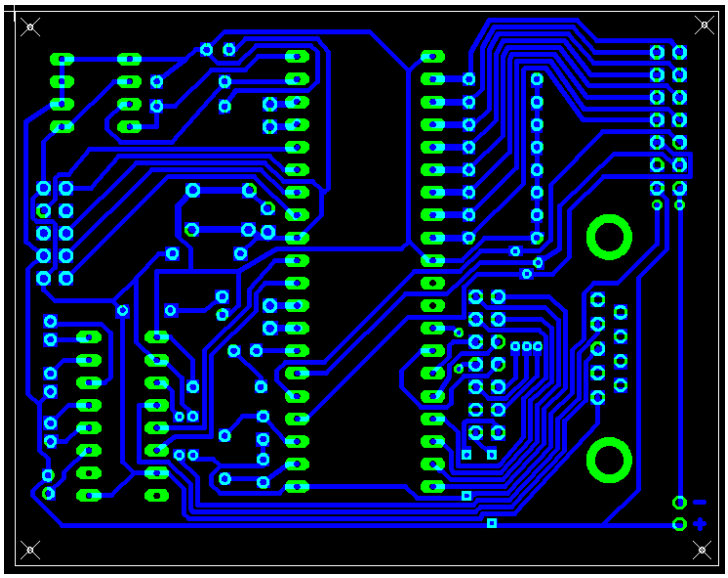
Programátor



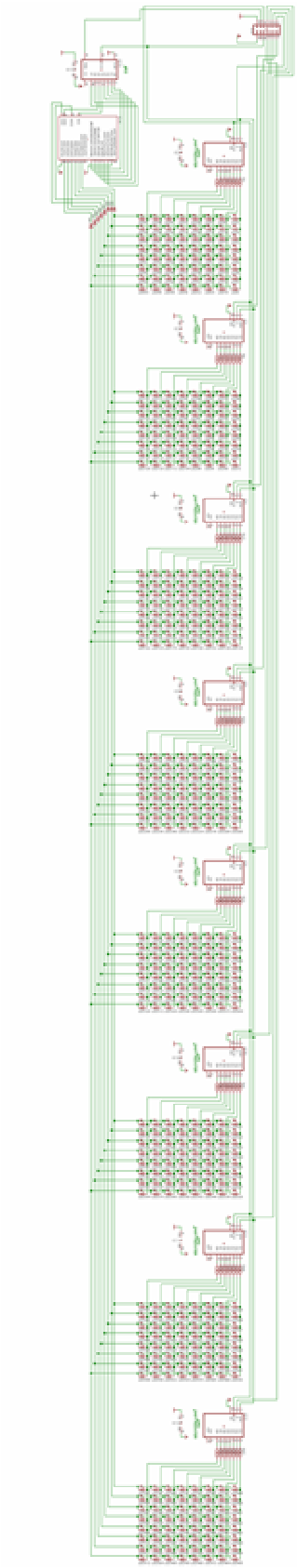


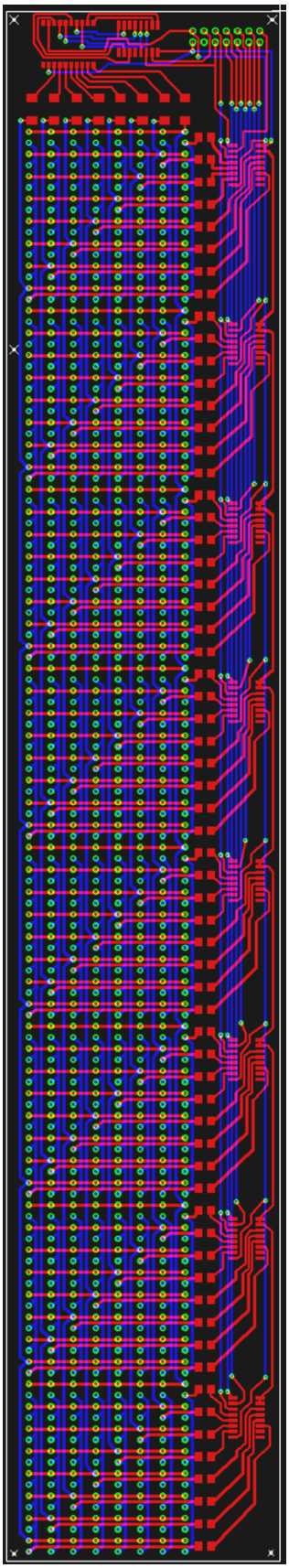
Modul s mikrokontrolérem





Matice LED diod

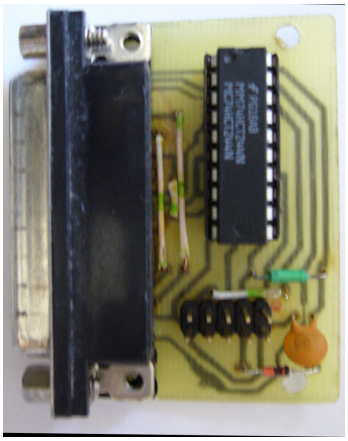


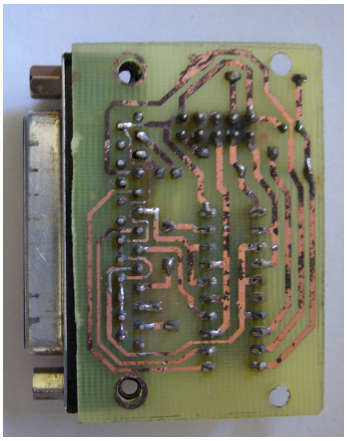


Příloha 2

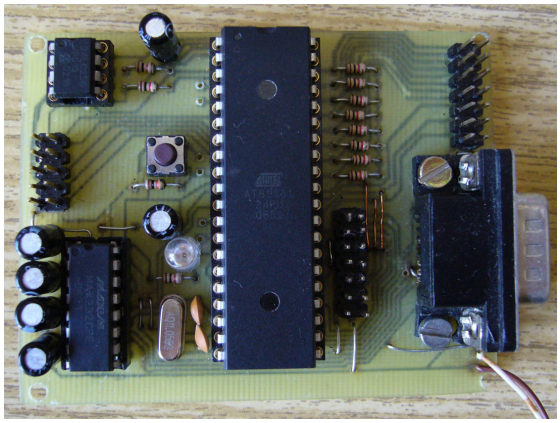
Funkční vzorek

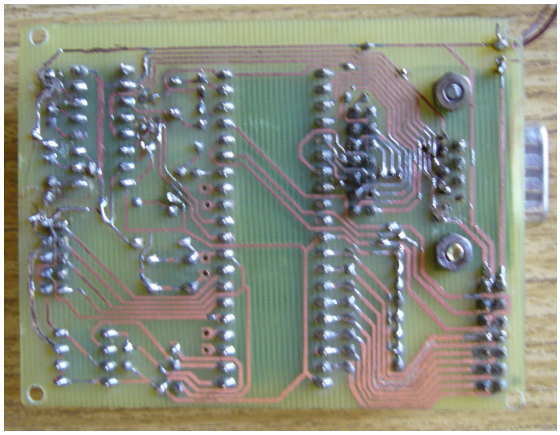
Programátor



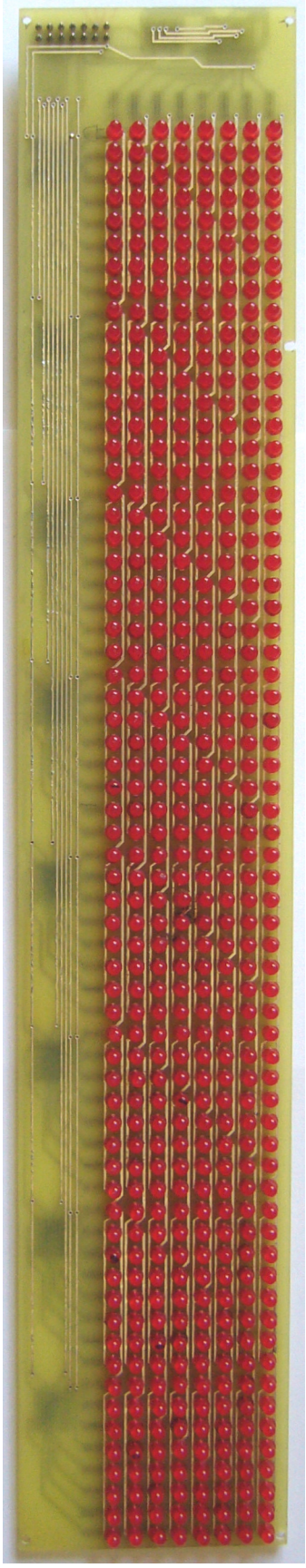


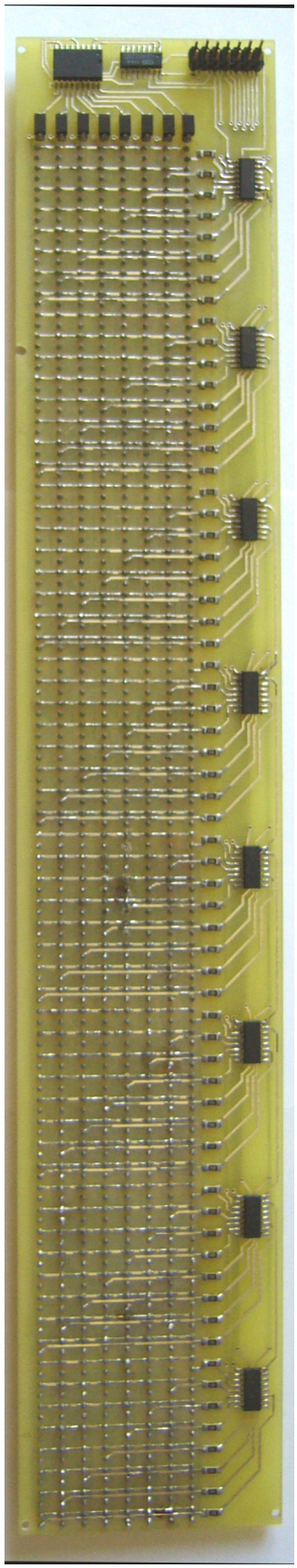
Modul s mikrokontrolérem





Matice LED diod





Příloha 3

Obsah CD

Na přiloženém CD lze najít následující adresáře:

- Adresář dokumentace - obsahuje technickou zprávu ve formátu PDF.
- Adresář firmware - obsahuje zdrojové soubory pro program mikrokontroléru v assembleru a přeložený soubor ve formátu hex.
- Adresář controller - obsahuje zdrojové soubory pro ovládací aplikaci na PC v jazyce C++ a přeloženou verzi pro MS Windows.
- Adresář schémata - obsahuje schémata a návrhy desek plošných spojů vytvořených v programu Eagle 4.16.
- Adresář foto - obsahuje fotografie funkčního vzorku v plném rozlišení.