STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Mikroprocesorový výukový systém

Vojtěch Drbohlav

Jičín 2010

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor SOČ: 12. Tvorba učebních pomůcek, didaktická technologie

Mikroprocesorový výukový systém

Microprocessor learning system

Autor:	**Vojtěch Drbohlav**
Škola:	**VOŠ a SPŠ Jičín**
Konzultant:	**Ing. Vladimír Vik**

Jičín 2010

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně, použil jsem pouze podklady (literaturu, SW atd.) citované v práci a uvedené v přiloženém seznamu a postup při zpracování práce je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Jičíně dne 15. března 2010 podpis: .....

Poděkování

Chtěl bych poděkovat p. Ing. Vladimíru Vikovi a pí. Mgr. Miroslavě Polákové za jejich cenné rady a hlavně pomoc s organizačními záležitostmi kolem soutěže. Také bych rád poděkoval své rodině za jejich podporu v průběhu soutěže.

Anotace

Mikroprocesorový systém je založený na 8bitovém mikroprocesoru AT89C51RD2 od firmy Atmel. Pro signalizaci stavů jednotlivých bitů paralelního výstupního portu P2 využívá 8 červených led, které svítí, pokud je na výstupu logická 0. Další výhodou tohoto systému jsou dvě již připravená tlačítka, která je možné využít v programech psaných studenty k základnímu uživatelskému vstupu.

Dále jsou na procesoru dva konektory pro připojení vnějších periferií. Díky tomu je možné schopnosti systému dále rozšiřovat a přidávat nové funkce, jako je například deska s tlačítky a displeji nebo deska s krokovým motorem.

Klíčová slova: assembler, RD2, 8051, RD2prog, mikroprocesorový systém, ISP, In-System Programming, C++, Qt

Keywords: assembler, RD2, 8051, RD2prog, microprocessor system, ISP, In-System Programming, C++, Qt

Obsah

1 Úvod

Cílem práce bylo vytvořit mikroprocesorový výukový systém, který by usnadnil výuku programování mikroprocesorů řady 8051 v jazyce assembler. Práce byla inspirována již existujícím RD2 Kitem[6]. Zapojení celého systému bylo navrženo znovu pouze na jednostranné deske plošného spoje, tím se celé zapojení zjednodušilo.

Komunikace systému s počítačem probíhá přes sériový port. Na straně počítače komunikaci zajišťuje program RD2prog, který jsem vytvořil a přikládám na CD. K jeho vývoji mě vedlo to, že pomocí stávajícího programu nebylo možné provést prvotní naprogramování mikroprocesoru, protože po něm je nutné zapsat 00h do bytu BSB v mikroprocesoru a to nynější software neumožňoval. Další výhodou nového programu je multiplatformnost, to znamená, že může pracovat v operačních systémech Microsoft Windows i v systémech GNU/Linux.

1.1 Požadavky

- mikroprocesor AT89C51RD2 v patici PLCC44
- napájení adaptérem 9 – 15 V, signalizace zelenou LED
- propojení s PC přes sériový port
- tlačítka na bitech P0.2 a P0.3 paralelního portu P0, zbylé bity vyvedené na konektor společně s napájecím napětím U_{CC} a GND pro připojení externích periferií
- 8 červených LED na výstupu paralelního portu P2
- paralelní porty P1 a P3 vyvedené na konektor společně se signálem PSEN a RST, napájecím napětím U_{CC} a GND pro připojení externích periferií
- software pro PC umožňující naprogramování mikroprocesorového systému

2 Hardwarová část

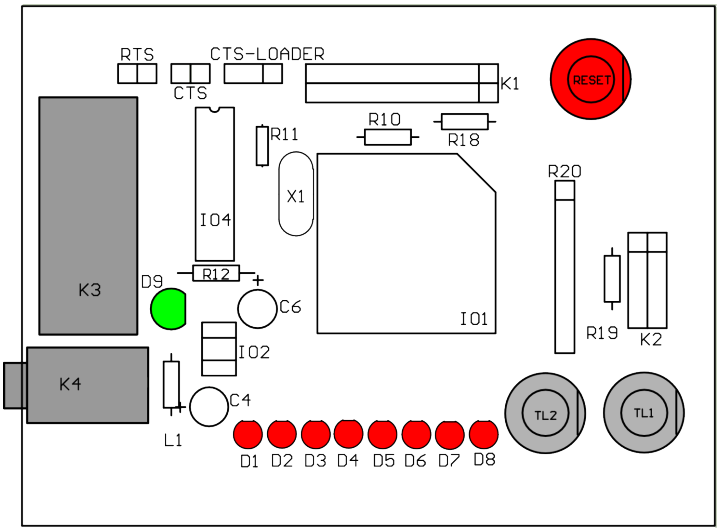
2.1 Technické parametry

- 64 kB paměti programu (programovatelná pomocí ISP, 100 000 zapisovacích cyklů)
- 2 kB paměti dat (100 000 zapisovacích cyklů)
- kmitočet 12 MHz
- dva 16bitové registry DPTR (přepínání pomocí bitu DPS = AUXR1.0)
- signalizace jednotlivých bitů P2 červenými LED
- výstupní konektor K1 (paralelní port P1 a P3)
- uživatelský konektor K2 (paralelní port P0)
- programovatelná tlačítka TL1 a TL2 (na bitech paralelního portu P0.2 a P0.3)
- RESET
  - MAX810 – automatický reset po připojení napájení
  - manuální červeným tlačítkem
  - externí – konektor K2:18, signál RST
- komunikace s PC pomocí rozhraní RS232 (MAX232)
- napájení 9 – 15 V (signalizace zelenou LED)
- rozměry desky mikroprocesorového systému – 92 x 70 mm

2.2 Popis desky mikroprocesorového systému

Základem celého systému je mikroprocesor AT89C51RD2 v pouzdře PLCC44, který je dalším z rodiny 8bitových mikroprocesorů řady 8051 od firmy ATMEL.

Mikroprocesorový systém je navržen v minimální konfiguraci vyhovující požadavkům na jednostranné desce plošných spojů o rozměrech 92 x 70 mm.



Obrázek 1: Zjednodušený obrázek desky mikroprocesorového systému

Z důvodu stability a ochrany SMD součástek připájených ze strany plošných spojů jsou v rozích desky gumové nožičky a z důvodu ochrany součástek a portů je na desku přes distanční sloupky o délce 2 cm přišroubováno plexisklo.

V konstrukci jsou převážně použity odpory a kondenzátory v provedení SMD ve velikostech 0805 a 1206. Procesor AT89C51RD2, IO1, je v pouzdře PLCC44 a je vložen do patice, což umožňuje jeho snadnou výměnu. Oscilátor procesoru je řízen krystalem X1, který pracuje na frekvenci 12 MHz. Na krystal je připojen kapacitní dělič skládající se z kondenzátorů C1 a C2. Procesor poskytuje vstupně výstupní paralelní porty P0, P1, P2 a P3.

K zajištění správného spuštění mikroprocesorového systému je v obvodu resetu zapojen integrovaný obvod IO3 MAX810 v SMD provedení v pouzdře SOT23. Tento obvod resetuje mikroprocesor po připojení napájení. Ruční reset je možné provést červeným tlačítkem TL3.

Dalším integrovaným obvodem na desce je IO4 MAX232 v základním zapojení. Obvod zajišťuje převod úrovní ze sériového portu PC na úrovně TTL.

Porty P1 a P3 jsou spolu se signálem PSEN, napájecím napětím U_{CC} a GND vyvedeny na konektor K1 (tabulka 1). Tento konektor může být použit například pro připojení desek s displeji, případně dalších periferií. Stavy jednotlivých bitů portu P2 jsou signalizovány

červenými LED, které jsou připojeny přes odpor na napájecí napětí U_{CC}. Svítí tedy jen pokud je na výstupu portu logická 0. K rozsvícení každé LED byl zvolen proud I_F 5 mA. Prahové napětí diody U_F je 1,7 V, odpor byl tedy vypočítán ze vztahu

$$R = \frac{U_{CC} - U_F}{I_F}$$

a po zaokrouhlení byl vybrán odpor 680 Ω.

Na bitech P0.2 a P0.3 jsou podle požadavků dvě tlačítka šedé barvy, která spínají jednotlivé bity portu do logické 0. Zbylé bity portu P0 jsou společně s napájecím napětím a GND vyvedeny na konektor K2 (tabulka 2) a připojeny přes odporovou síť typu RR na napájecí napětí.

Číslo pinu	**Signál**	**Číslo pinu**	**Signál**
1	P1.0	2	P0.1
3	P1.2	4	P0.3
5	P1.4	6	P0.5
7	P1.6	8	P0.7
9	P3.0 (RxD)	10	P3.1 (TxD)
11	P3.2 (INT0)	12	P3.3 (INT1)
13	P3.4 (T0)	14	P3.5 (T1)
15	P3.6 (WR)	16	P3.7 (RD)
17	PSEN	18	RST
19	GND	20	Ucc

Tabulka 1: Zapojení konektoru K1

Číslo pinu	**Signál**
1	P0.0
2	P0.1
3	×
4	×
5	P0.4
6	P0.5
7	P0.6
8	P0.7
9	GND
10	UCC

Tabulka 2: Zapojení konektoru K2

Dále jsou na desce tři propojky: CTS–LOADER, T1–RTS a INT1–CTS. Slouží k propojení některých signálů mikroprocesoru se sériovým portem PC. Pomocí propojky CTS–LOADER je možné signál PSEN propojit s GND, tím ho nastavíme trvale na 0, nebo se sériovým portem PC, a tím umožníme jeho softwarové ovládání. Podle tohoto signálu se mikroprocesor při resetu rozhoduje, zda bude spuštěn námi zapsaný program nebo tzv. bootloader (jeho bližší popis je v kapitole Softwarová část). Propojením s GND zajistíme, že při každém resetu bude spuštěn bootloader.

Zbylé dvě propojky, T1–RTS a INT1–CTS, slouží k propojení signálů mikroprocesoru T1 a RTS se signály sériového portu RTS a CTS přes IO4 (MAX232). Díky tomuto nastavení je možné obsluhovat signály T1 a INT1 z PC.

Napájecí napětí je k mikroprocesorovému systému připojeno přes konektor K4. Za ním je zapojen jednoduchý filtr složený z kondenzátoru C12 a tlumivky L1, dále dioda D1, která slouží jako ochrana při přepólování vstupního napětí.

Další součástkou desky je stabilizátor IO2 78L05, který má na vstupu filtr z kondenzátorů C3 a C4. Na jeho výstupu je připojen filtr z C5 a C6 a 5mm zelená LED D9, připojená přes odpor R9, signalizující aktivní napájení.

Při návrhu plošného spoje byly z důvodu úspory místa ohnuty nepoužívané piny č. 1, 12, 23 a 34 u patice pro procesor.

Na celé desce je využita pouze jedna drátová propojka, a to k propojení GND sériového portu s GND mikroprocesorového systému.

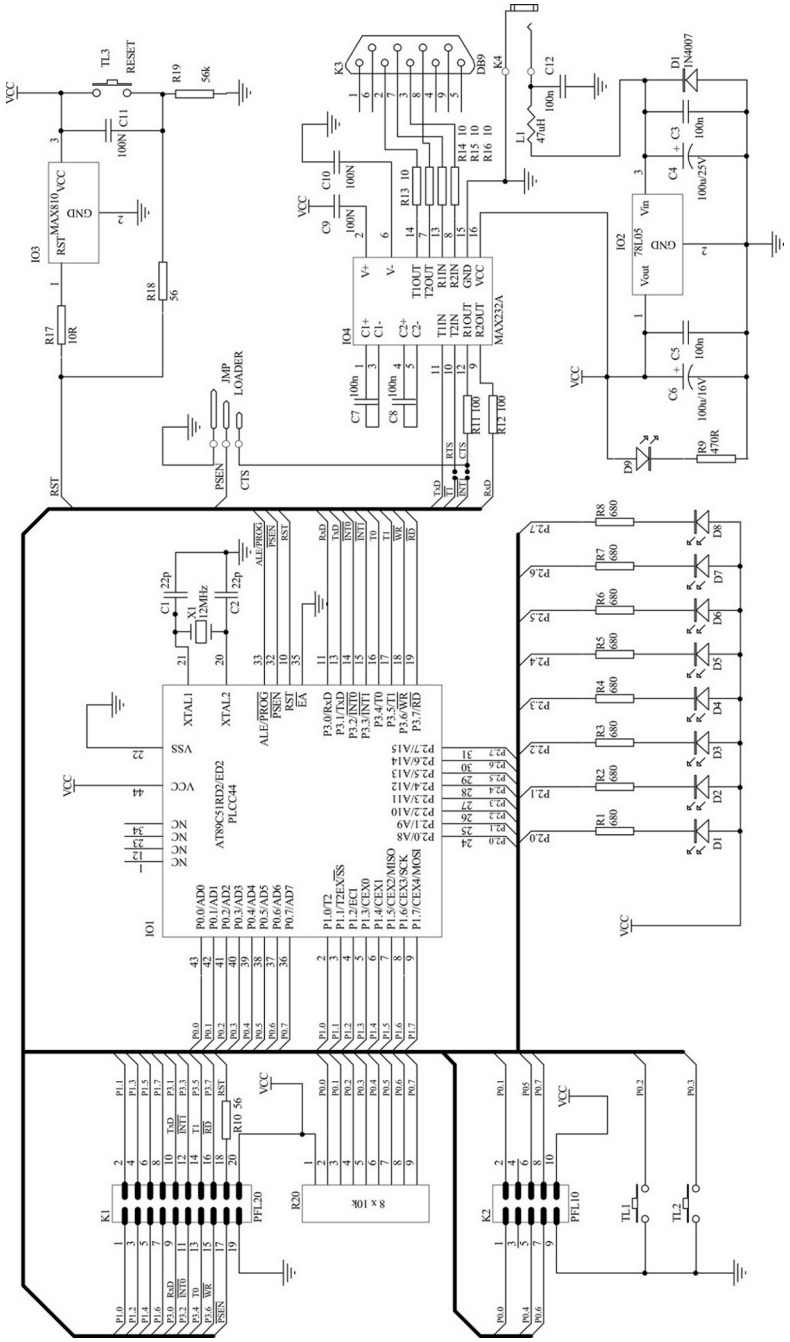
2.3 Návrh a výroba plošného spoje

Schéma (obrázek 2 a plošný spoj (obrázek 3 byly nakresleny v trial verzi programu Altium Designer.

Obraz plošného spoje vytoiskneme na laserové tiskárně na pauzovací papír nebo průhlednou fólii, poté se přiloží na fotocitlivou vrstvu desky plošného spoje a UV zářením se osvítí. Vzdálenost, ze které se UV záření nechá působit, a doba působení jsou závislé na podmínkách prostředí a zdroji záření. Já jsem UV záření nechal působit ze vzdálenosti cca 50 cm po dobu cca 4,5 minuty. Je při tom třeba dbát na ochranu očí a používat ochranné brýle s UV filtrem.

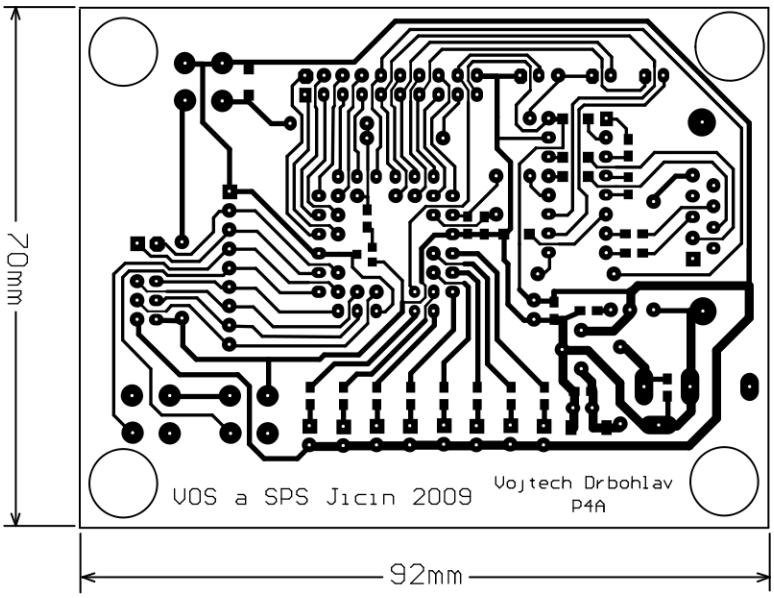
Následuje vyvolání desky plošného spoje v louhu sodném (NaOH), poté její opláchnutí vodou a osušení. Dalším krokem je leptání desky v chloridu železitém ($FeCl_3$). Při této činnosti je třeba používat ochranné brýle a gumové rukavice. Po vyleptání následuje opět omytí vodou a vyvrtání děr. Posledním krokem je očištění plošného spoje lihem a natření kalafunou rozpuštěnou v lihu, která slouží jako ochranný lak a usnadňuje pájení.

2.3.1 Schéma zapojení



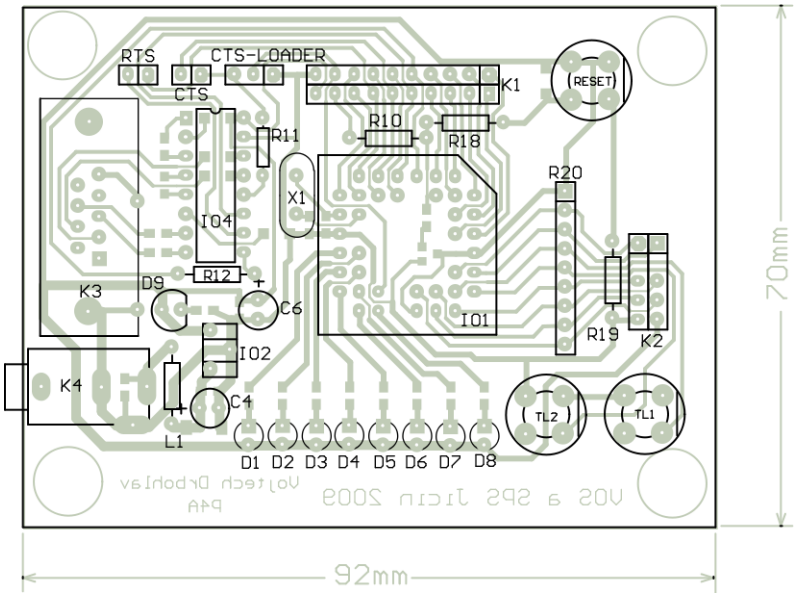
Obrázek 2: Schéma zapojení

2.3.2 Plošný spoj



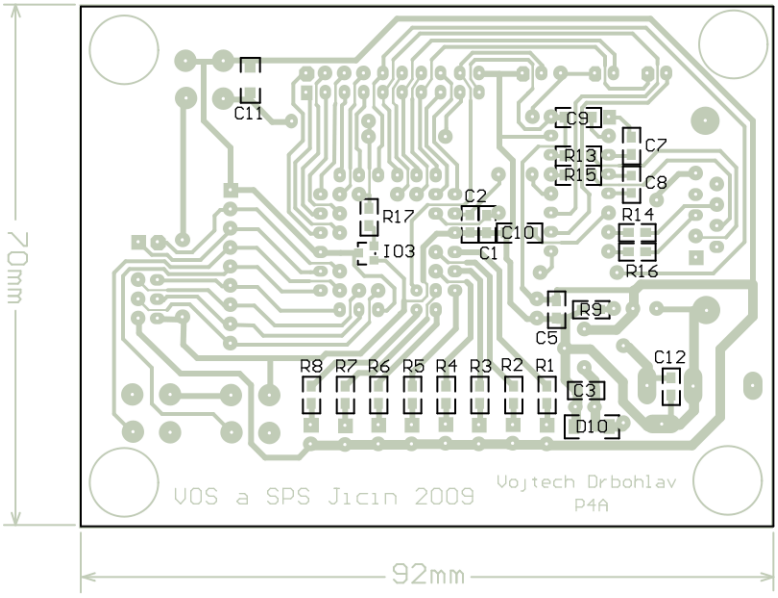
Obrázek 3: Plošný spoj

2.3.3 Rozmístění součástek – horní strana



Obrázek 4: Rozmístění součástek – horní strana

2.3.4 Rozmístění SMD součástek – strana spojů



Obrázek 5: Rozmístění SMD součástek – strana spojů

2.3.5 Seznam použitých součástek

Rezistory		
R1 až R8	R0805	680R
R9	R0805	470R
R10	MRR	1K
R11, R12	MRR	100R
R13, R15	R1206	10R
R14, R16, R17	R0805	10R
R18	MRR	470R
R19	MRR	56K
R20	RR	8X10K

Kondenzátory (keramické)		
C1, C2	CK0805	22P
C3, C5, C7, C8, C12	CK0805	100N
C9, C10, C11	CK1206	100N

Kondenzátory (elektrolytické)	
C4	E100M/25V
C6	E100M/16V

Tlumivka	
L1	TL.47uH

Diody		
D1-8	L-LTL4221N	3 mm, červená
D9	L-53GD	5 mm, zelená
D10	1N4007	SMA
		DO-214AC
Krystal		
X1	QM	12.000MHz
Integrované obvody		
IO1	AT89C51RD2/ED2	pouzdro PLCC44
IO2	78L05	pouzdro TO29
IO3	MAX810	pouzdro SOT23
IO4	MAX232IN	
Ostatní (patice, konektory, tlačítka)		
patice pro IO1	PLCC44Z	
TL1, TL2	P-DT6GR	šedé
TL3	P-DT6RT	červené
K1, K2	S2G20	2 řady po 20 pinech
konektorová lišta	S1G20	1 řada po 20 pinech
	S1G10	1 řada po 10 pinech
K3	CAN9Z90	konektor pro RS232
K4	K375A	napájecí konektor

Tabulka 3: Seznam součástek

2.4 Oživení systému

Před osazením mikroprocesoru proběhla kontrola přítomnosti napájení na příslušných místech na desce a na příslušných pinech patice pro procesor. Dále byl změřen odběr proudu bez osazeného mikroprocesoru. Po kontrole a ověření napájení byl osazen mikroprocesor a byl znovu změřen odběr proudu.

Poté byl mikroprocesorový systém připojen k počítači a bylo provedeno prvotní naprogramování, při kterém je nutné zapsat hodnotu 00h do bytu BSB, aby byl po resetu spuštěn zapsaný program.

3 Softwarová část

Mikroprocesor AT89C51RD2 podporuje tzv. *In-system programming*, dále jen ISP, díky tomu je možné s procesorem snadno komunikovat bez nutnosti jeho vyndávání z desky a používání speciálního hardwaru.

Předpokladem pro úspěšnou komunikaci je jen propojení mikroprocesorového systému s PC sériovým kabelem a použití softwaru, který podporuje ISP. Pro tento účel jsem naprogramoval aplikaci s názvem RD2prog.

3.1 Princip komunikace mikroprocesorového systému s PC – bootloader

Na straně PC komunikaci zajišťuje již zmínený RD2prog a na straně mikroprocesorového systému tzv. *bootloader*, což je program uložený v mikroprocesoru. V mikroprocesorech T89C51RD2 je uložen v posledním kB paměti programu, v novějších mikroprocesorech AT89C51RD2 je uložen ve zvláštní paměti typu ROM.

Bootloader je spuštěn po resetu mikroprocesoru, pokud je obsah bytu BSB nenulový nebo je signál PSEN nastaven do logické 0.

Obsah bytu BSB je nenulový po zakoupení nového mikroprocesoru. Z toho vyplývá, že je nutné po prvotním naprogramování tento byte vynulovat, aby bylo možné spustit námi zapsaný program. Pokud by nebyl byte BSB vynulován, spouštěl by se po každém resetu bootloader.

Signál PSEN lze trvale nastavit do logické 0 propojkou CTS–LOADER a to zapojením propojky do polohy LOADER. Toto nastavení způsobí, že se opět po každém resetu spustí bootloader, není tedy možné spustit zapsaný program. Zapojením propojky do polohy CTS dojde k propojení signálu PSEN se sériovým portem PC přes integrovaný obvod MAX232, čímž je umožněno softwarové ovládání signálu PSEN.

Program RD2prog tedy automaticky nastaví signál do logické 0 před začátkem programování a vyžádá si reset mikroprocesorového systému, aby došlo ke spuštění bootloaderu. Po ukončení zápisu je signál nastaven opět do logické 1 a následujícím resetem se spustí zapsaný program.

3.1.1 Synchronizace mikroprocesorového systému s PC

Po spuštění bootloaderu před zápisem programu nebo odesláním jiného příkazu do mikroprocesoru je nutné provést synchronizaci mikroprocesorového systému a PC. Během synchronizace se mikroprocesor pokusí spočítat nastavenou přenosovou rychlost sériového portu.

Toto je možné díky opakovanému odesílání ASCII kódu znaku U z PC. Mikroprocesor přijatá data zpracuje a odešle je zpátky. V PC jsou tato data vyhodnocena, a pokud je přijat ACII kód znaku U 12× v řadě, byla synchronizace úspěšná a je možné přejít k zápisu vlastního programu, nebo odeslání příkazu.

Pokud synchronizace selže, je to pravděpodobně tím, že na straně mikroprocesorového systému selhala detekce přenosové rychlosti sériového portu, a je nutné tuto rychlost změnit v nastavení sériového portu. Použitelné rychlosti jsou odvozeny od frekvence použitého krystalu. S krystalem 12 MHz je možné použít rychlosti 2400 kHz, 4800 kHz, 9600 kHz, 19200 kHz a 38400 kHz.

3.1.2 Protokol ISP

Vlastní komunikace PC a mikroprocesorového systému probíhá pomocí protokolu ISP. Přes sériový port jsou odesílány tzv. *rámce*. Struktura rámce je založená na formátu Intel HEX (tabulka 4).

Začátek rámce	**Délka dat**	**Offset**	**Typ rámce**	**Data**	**Checksum**
1 B	1 B	2 B	1 B	různá	1 B

Tabulka 4: Rámec protokolu ISP a velikost jeho částí

První část rámce musí vždyt obsahovat znak : (dvojtečka).

Druhá část, délka dat, obsahuje počet bytů v datové části rámce. Maximální délka dat, kterou dokáže mikroprocesor AT89C51RD2 zpracovat, je 128 B.

Offset je část používaná pouze při zápisu do paměti programu. Určuje, od jaké adresy budou zapsána data. V jazyce assembler se tato adresa nastavuje pomocí direktivy překladače `org`. Adresa je 16bitová.

Typ rámce je identifikační číslo příkazu, který chceme vykonat. Mezi příkazy patří například zápis do paměti programu, čtení paměti programu, vymazání paměti programu apod.

Předposlední částí rámce jsou vlastní data. Obsah této části se mění v závisloti na použitém příkazu. Při zápisu programu do mikroprocesoru jsou to například kódy instrukcí a jejich operandy.

Poslední částí je kontrolní součet celého rámce. Pokud není správný, mikroprocesor neprovede požadovaný příkaz. Kontrolní součet je dvojkový doplněk součtu všech předchozích bytů (kromě počáteční dvojtečky) vyjádřený 1bytovým číslem.

3.2 Program RD2prog

RD2prog slouží k zápisu do paměti programu mikroprocesorového systému a také usnadňuje vývoj programů v assembleru, protože dokáže tyto programy kompilovat, aniž by bylo nutné externě spouštět kompilátor. Zároveň zvýrazňuje řádky s případnými chybami v kódu.

Program je multiplatformní, to znamená, že funguje jak na operačních sytémech Miscrosoft Windows, tak i na operačním systému GNU/Linux. Je naprogramován v jazyce C++ s využitím frameworku Qt od firmy Nokia. Důvodem pro vytvoření nového programu bylo to, že stávající program neumožňoval vynulování bytu BSB, takže nebylo možné provést prvotní naprogramování mikroprocesorového systému. Dalšími důvody byly nepřehlednost starého programu, pomalé zvýrazňování syntaxe jazyka assembler a další drobnosti.

3.2.1 Rozhraní programu

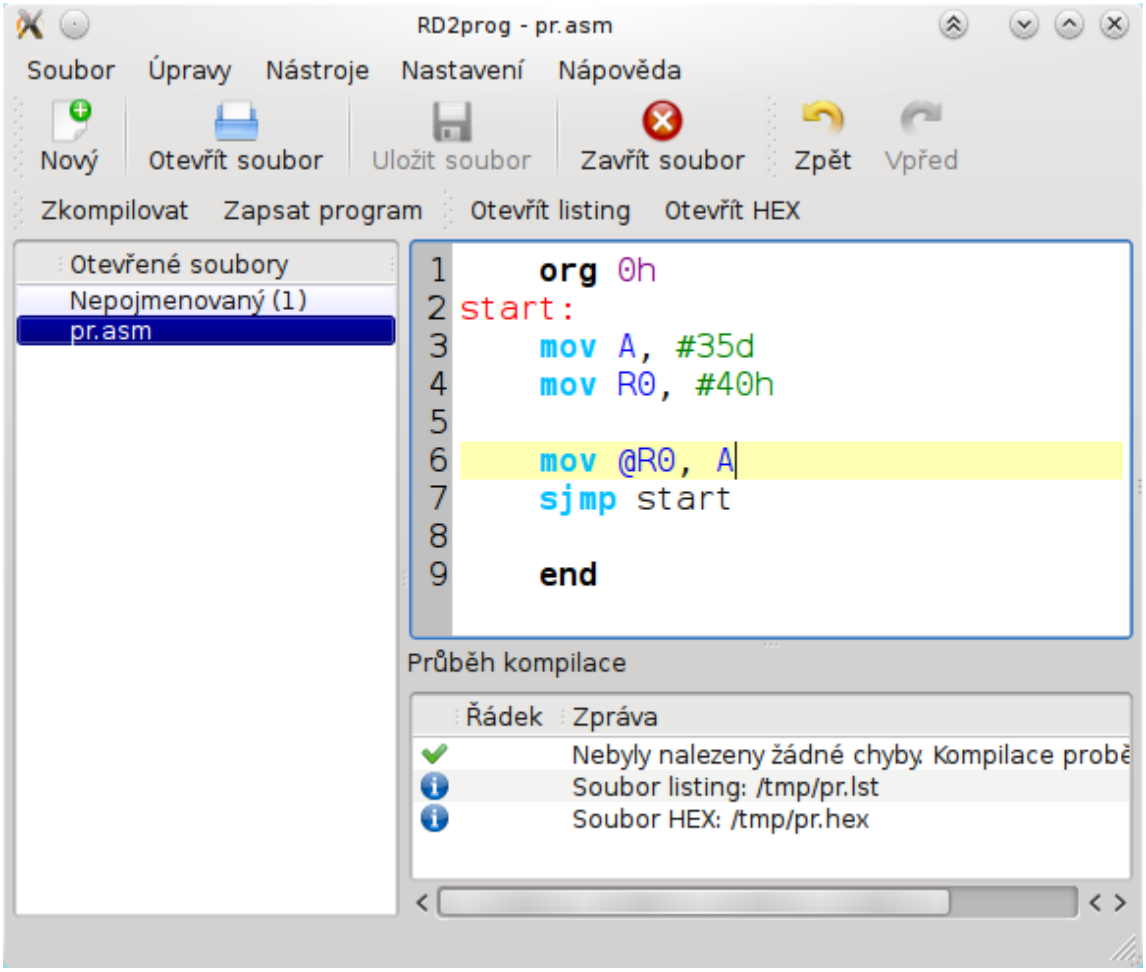
Hlavní okno programu (obrázek 6) obsahuje hlavní menu a panel nástrojů s nejpoužívanějšími položkami menu z důvodu usnadnění a urychlení práce. Dále je okno rozděleno na tři části. Vpravo je seznam otevřených souborů. V levé horní části je editor zdrojového kódu a ve spodní části jsou vypisovány informace o průběhu kompilace programů.

Hlavní menu má 5 podmenu: *Soubor*, *Úpravy*, *Nástroje*, *Nastavení* a *Nápověda*.

V podmenu *Soubor* je možné vytvářet nové soubory, otevírat soubory uložené na disku nebo vybrat jeden z 10 naposledy otevřených souborů. Dále je možné soubory ukládat a zavírat.

Podmenu *Úpravy* obsahuje běžné položky jako *Zpět*, *Vpřed*, *Kopírovat*, *Vyjmout*, *Vložit*, *Smazat* a *Vybrat vše*. Pomocí tohoto menu je možné pracovat s právě otevřeným souborem.

Nástroje obsahují tři položky, a to: *Zkompilovat* (provede kompilaci zdrojového kódu a zobrazí informace o průběhu kompilace ve spodní části okna, případně zvýrazní chyby v otevřeném souboru), *Zapsat program* (otevře dialog pro zápis programu do mikroprocesorového systému, viz obrázek 11) a *Vymazat celý mikroprocesor* (uvede mikroprocesor do továrního nastavení, čehož lze využít například při nechtěném zablokování bezpečnostním

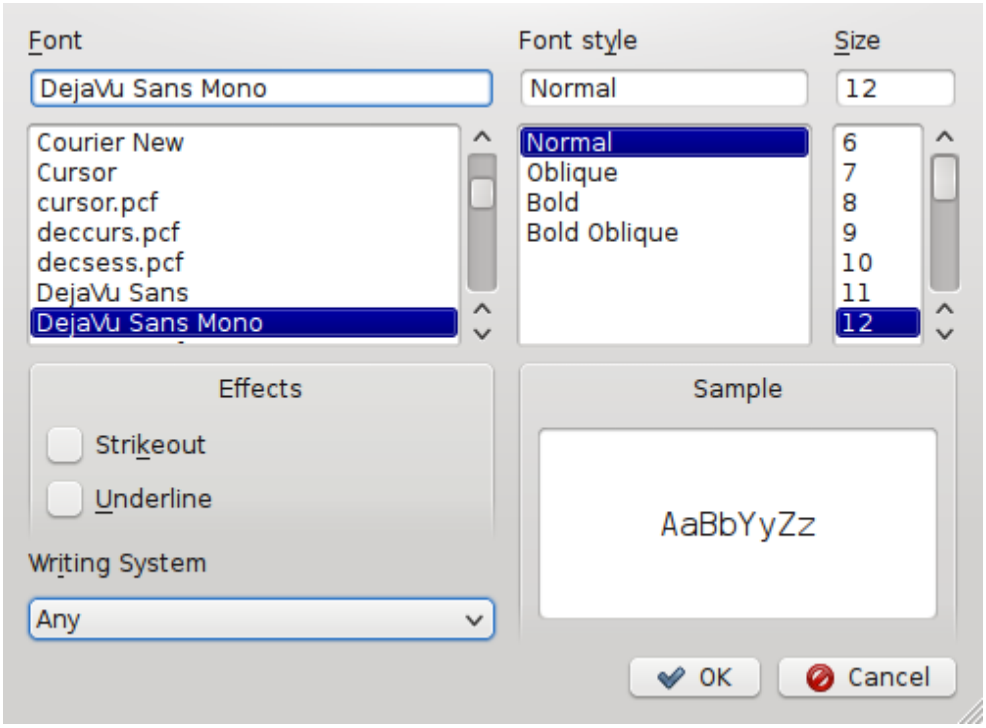


Obrázek 6: Hlavní okno programu RD2prog

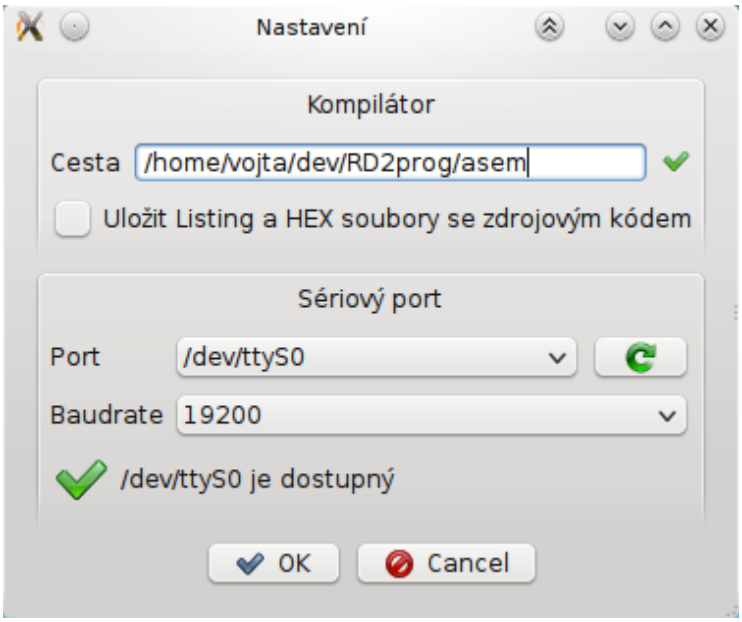
bitem nebo pokud chceme celý čip rychle vymazat – vymazání celého mikroprocesoru trvá přibližně 6 vteřin).

Podmenu *Nastavení* umožňuje nastavení fontu (obrázek 7) pro editor zdrojového kódu a nastavení programu RD2prog (obrázek 8), kde je možné změnit cestu ke kompilátoru nebo nastavit sériový port používaný pro komunikaci s mikroprocesorovým systémem. Dialog pro výběr fontu se může na různých operačních systémech mírně lišit.

V podmenu Nápověda jsou pouze dvě položky, informace o programu RD2prog a o frameworku Qt.



Obrázek 7: Změna fontu editoru zdrojového kódu



Obrázek 8: Nastavení programu RD2prog

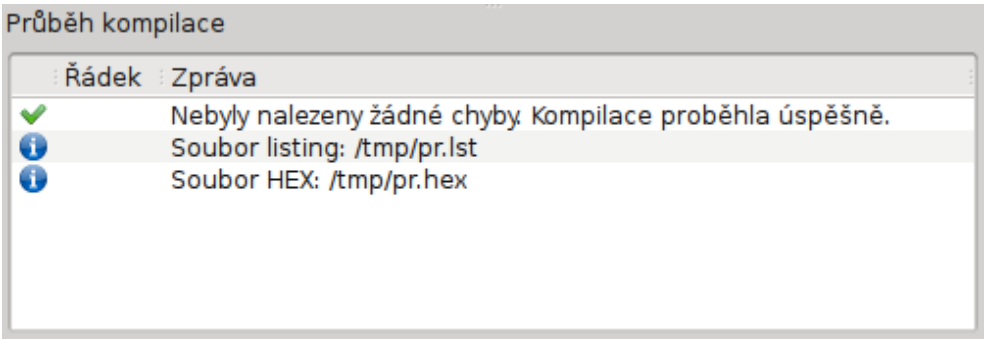
3.2.2 Kompilace zdrojového kódu

Pokud napíšeme zdrojový kód programu nebo libovolný otevřeme a chceme ho zkompilovat, stačí v panelu nástrojů kliknout na tlačítko *Zkompilovat*, nebo z menu *Nástroje* vybrat

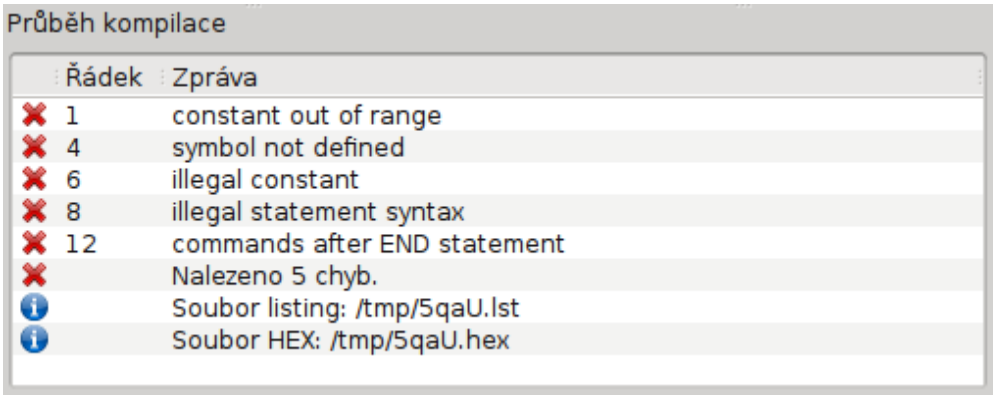
položku *Zkompilovat*. Když kompilaci spustíme, jsou ve spodní části hlavního okna zobrazeny informace o průběhu kompilace (obrázek 9 a 10).

Pokud byly během kompilace nalezeny nějaké chyby, jsou vypsána čísla řádků, na kterých se chyby nachází a zobrazeny jejich krátké popisy (obrázek 10). Řádky s chybami jsou také červeně zvýrazněny v editoru kódu.

Dále jsou vypisovány cesty k souborům listing a HEX. Pokud si některý ze souborů chceme prohlédnout, stačí dvakrát kliknout na řádek s cestou k souboru, nebo použít panel nástrojů.



Obrázek 9: Zobrazení informací o průběhu úspěšné kompilace



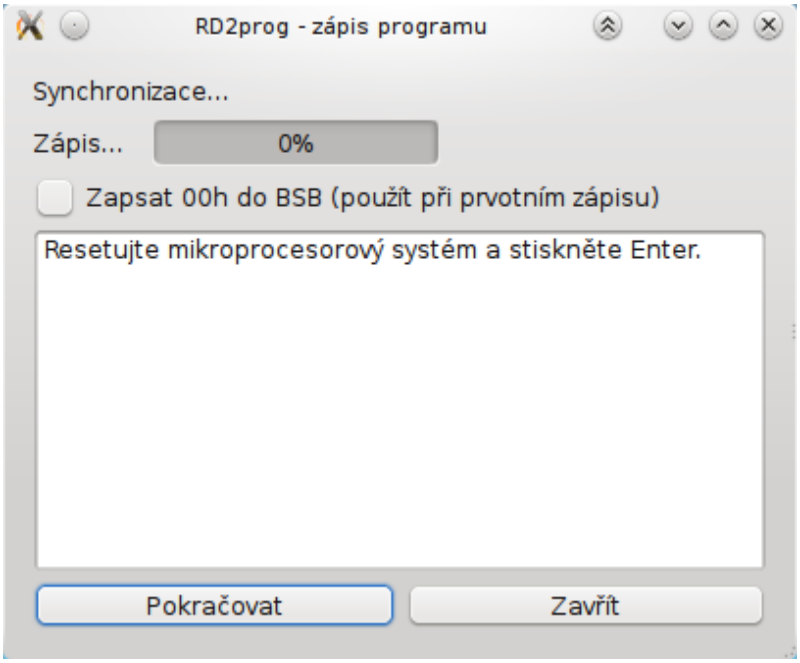
Obrázek 10: Zobrazení informací o průběhu kompilace s chybami

3.2.3 Zápis programu do mikroprocesorového systému

Po úspěšné kompilaci je program připraven k zapsání do mikroprocesorového systému. Před tímto krokem je nutné mikroprocesorový systém propojit s PC pomocí sériového kabelu a připojit ho k napětí napájecím adaptérem, připojeným na konektor K4.

Jestliže je mikroprocesorový systém připojen, je možné spustit dialog pro zápis programu (obrázek 11) kliknutím na tlačítko *Zapsat program* v panelu nástrojů nebo vybráním položky *Zapsat program* z menu *Nástroje*.

Pokud je v mikroprocesorovém systému mikroprocesor, do kterého probíhá zápis poprvé, je nutné zatrhnout volbu *Zapsat 00h do BSB*, aby bylo možné později spustit zapsaný program.



Obrázek 11: Dialog pro zápis programu do mikroprocesorového systému

Prvním krokem zápisu programu je reset mikroprocesorového systému. Po resetu bude spuštěn bootloader, který zajistí komunikaci mikroprocesorového systému s PC. Dalším krokem je stisknutí tlačítka *Pokračovat*, nebo stisknutí klávesy Enter.

Následuje synchronizace mikroprocesorového systému a PC, poté zápis vlastního programu, případně zápis bytu BSB. Pokud synchronizace selže, je nutné změnit nastavenou přenosovou rychlost sériového portu, případně zkontrolovat kabel, kterým je mikroprocesorový systém připojen k PC.

Po úspěšném zápisu následuje další reset mikroprocesorového systému, nyní však nebude spuštěn bootloader, ale právě zapsaný program.

3.3 Ukázkový program

Program demonstruje funkčnost LED na paralelním portu P2 a tlačítek TL1 a TL2 na základní desce. Dále funkčnost konektoru K1, na který je připojena deska s displeji a tlačítky, a konektoru K2, na který je připojena deska s krokovým motorem a dalšími tlačítky.

Po spuštění programu svítí jedna LED na paralelním portu P2 a ta se posouvá od bitu 0 k bitu 7 s intervalem 0,5 s. Na desce s displeji svítí tři nuly a krokový motor stojí.

3.3.1 Ovládání ukázkového programu

- **LED na paralelním portu P2 a tlačítka na základní desce**
  - Stiskem TL1 dojde ke zvýšení počtu svítících LED na paralelním portu P2, pokud svítí sedm LED, rozsvítí se opět pouze jedna.
  - Stiskem TL2 se bude měnit prodleva mezi posunutím svítících LED z 0,5 s na 1 s a poté na 2 s. Po třetím stisku se změní směr posouvání LED a dojde ke snížení prodlevy na 0,5 s.
- **Deska s displeji**
  - Stiskem tlačítka TL1 dojde ke spuštění stopek, dalším stiskem se tyto stopky zastaví a dalším vynulují.
  - Pokud stopky načítají 99,9 s rozbliká se na displejích znak F, stiskem TL1 se opět displeje vynulují.
  - Tlačítko TL2 funguje pouze, pokud na displejích svítí hodnota 0. Stiskem toho tlačítka se spouští režim čítače, ve kterém každý stisk inkrementuje hodnotu svítící na displejích o 1. Tlačítkem TL1 je možné opět displeje vynulovat.
- **Deska s krokovým motorem**
  - Stiskem TL1 se motor roztočí.
  - Stiskem TL2 dojde k zastavení motoru.

3.3.2 Zdrojový kód

```
;;; nastaveni promennych

smer              equ 21h
citani            equ 22h
rucniCitani       equ 23h
zTecka            equ 24h
blink             equ 25h
blinkState        equ 26h
motor             equ 27h

predvolba         equ 50h
tmp               equ 51h

tlac1             equ P0.2
tlac2             equ P0.3
dTlac1            equ P3.2
dTlac2            equ P3.3
tlacM1            equ P0.6
tlacM2            equ P0.7

; banka 0
zaklad            equ R4
pocetRotaci       equ R2
pocetTlac2        equ R3
pocetDTlac1       equ R1

; banka 1
dispDesitky       equ P3.6
dispJednotky      equ P3.5
dispDesetiny      equ P3.4
dispInc           equ R2
desetiny          equ R3
jednotky          equ R4
desitky           equ R5
blinkInterval     equ R6

; banka 2
motorPocet        equ R2

        org 0h
        sjmp init ; skok na inicializaci mikroprocesoru

        org 0Bh
        ljmp cit0 ; preruseni pri preteceni citace/casovace 0

        org 1Bh
```

```
        ljmp cit1 ; preruseni pri preteceni citace/casovace 1

        org 30h

;;; kody cislic pro zobrazeni na displeji

cislice:
         ; 0,         1,         2,         3,         4
        db 00010010b, 11010111b, 01001010b, 01000110b, 10000111b
        ;  5,         6,         7,         8,         9
        db 00100110b, 00100010b, 01010111b, 00000010b, 00000110b
pF:
        ;  F
        db 00101011b

;;; inicializace mikroprocesoru

        org 50h
init:
        clr RS1
        clr RS0

        mov SP, #0EFh
        mov zaklad, #01111111b
        mov A, zaklad

        mov R0, #1
        mov predvolba, #10

        clr smer
        clr citani
        clr zTecka
        setb rucniCitani
        clr blink
        clr blinkState
        clr motor

        mov pocetRotaci, #0
        mov pocetTlac2, #0
        mov pocetDTlac1, #1

        setb tlac1
        setb tlac2
        setb dTlac1
        setb dTlac2
        setb tlacM1
        setb tlacM2
```

```
        setb RS0
        mov desetiny, #0
        mov jednotky, #0
        mov desitky, #0
        mov DPTR, #cislice
        mov dispInc, #1
        mov blinkInterval, #80
        clr RS0

        mov 89h, #10001b
        mov TH0, #0
        mov TL0, #1
        setb EA
        setb ET0
        setb ET1
        setb TR0
        setb TR1

;;; hlavni smycka programu, zde probiha kontrola stisku tlacitek

program:
        jb tlac1, programTlac2 ; zpracovani stisku TL1 na zakladni desce

        lcall prPosunuti
        lcall prSrovnaniPozice

        lcall prSpozdeni

programTlac2:
        jb tlac2, programDTlac1 ; zpracovani stisku TL2 na zakladni desce
        lcall prTlac2
        lcall prSpozdeni

programDTlac1:
        jb dTlac1, programDTlac2 ; zpracovani stisku TL1 na desce s displeji
        lcall prDTlac1
        lcall prSpozdeni

programDTlac2:
        jb dTlac2, programTlacM1 ; zpracovani stisku TL2 na desce s displeji
        lcall prPrictiDisp
        lcall prSpozdeni

programTlacM1:
        jb tlacM1, programTlacM2 ; zpracovani stisku TL1 na desce s motorem
        setb motor ; povoli otaceni motoru
        lcall prSpozdeni
```

```
programTlacM2:
        jb tlacM2, program ; zpracovani stisku TL2 na desce s motorem
        clr motor ; zakaze otaceni motoru
        lcall prSpozdeni
        sjmp program

;;; preruseni citace/casovace 0
;;; posun rozsvycenych LED na portu P2

cit0:
        mov TH0, #03Ch
        mov TL0, #0AFh

        djnz R0, cit0Konec
        mov R0, predvolba

        mov P2, A
        jb smer, cit0Doprava
        rl A
        sjmp cit0Pokracuj
cit0Doprava:
        rr A

cit0Pokracuj:
        inc pocetRotaci
        cjne pocetRotaci, #9, cit0Konec
        mov pocetRotaci, #0

cit0Konec:
        reti

;;; preruseni citace/casovace 1
;;; zobrazovani na displeji
;;; otaceni krokoveho motoru

cit1:
        mov TH1, #0E7h
        mov TL1, #095h
        setb RS0 ; citac 1 pouziva druhou banku registru

        jnb blink, cit1Dale
        djnz blinkInterval, cit1BlinkKonec
        mov blinkInterval, #80
        cpl blinkState
cit1BlinkKonec:
        jb blinkState, cit1PreskocInc
        setb dispDesetiny
        setb dispJednotky
```

```
        setb dispDesitky
        sjmp cit1Konec

cit1Dale:
        jnb citani, cit1PreskocInc
        djnz dispInc, cit1PreskocInc
        mov dispInc, #16

        inc desetiny
        cjne desitky, #9, cit1Dale2
        cjne jednotky, #9, cit1Dale2
        cjne desetiny, #9, cit1Dale2
        clr citani
        inc desitky
        inc jednotky
        inc desetiny
        setb blink
        clr zTecka
        sjmp cit1PreskocInc
cit1Dale2:
        cjne desetiny, #10, cit1PreskocInc
        mov desetiny, #0
        inc jednotky
        cjne jednotky, #10, cit1PreskocInc
        mov jednotky, #0
        inc desitky
cit1PreskocInc:
        push ACC
        jnb dispDesetiny, cit10
        jnb dispJednotky, cit11
        setb dispDesitky
        clr dispDesetiny
        lcall prDispDesetiny
        sjmp cit1POP
cit10:
        setb dispDesetiny
        clr dispJednotky
        lcall prDispJednotky
        sjmp cit1POP
cit11:
        setb dispJednotky
        clr dispDesitky
        lcall prDispDesitky
        sjmp cit1POP
cit1POP:
        pop ACC
cit1Konec:
        clr RS0
```

```
        jnb motor, cit1Kon
        lcall prMotor
cit1Kon:
        reti

;;; podprogram pro zobrazeni desetin displeji

prDispDesetiny:
        mov A, desetiny
        movc A, @A+DPTR
        mov P1, A
        ret

;;; podprogram pro zobrazeni jednotek na displeji

prDispJednotky:
        mov A, jednotky
        movc A, @A+DPTR
        jnb zTecka, prDispJednotkyBezTecky
        anl A, #11111101b ; v modu stopek pridame des. carku
prDispJednotkyBezTecky:
        mov P1, A
        ret

;;; podprogram pro zobrazeni desitek na displeji

prDispDesitky:
        mov A, desitky
        movc A, @A+DPTR
        mov P1, A
        ret

;;; podprogram pro posouvani sviticich LED na portu P2

prPosunuti:
        cjne zaklad, #1, prPosunutiPosun
        mov A, #01111111b
        sjmp prPosunutiKonec
prPosunutiPosun:
        mov A, zaklad
        clr C
        rrc A
prPosunutiKonec:
        mov zaklad, A
        ret

;;; podprogram pro srovnani pozice rozsvicenych LED po pridani dalsi LED
```

```
prSrovnaniPozice:
        mov tmp, pocetRotaci
        cjne pocetRotaci, #0, prSrovnaniPoziceCyklus
        ret
prSrovnaniPoziceCyklus:
        jb smer, prSrovnaniPoziceDoprava
        rl A
        sjmp prSrovnaniPozicePokracuj
prSrovnaniPoziceDoprava:
        rr A
prSrovnaniPozicePokracuj:
        djnz pocetRotaci, prSrovnaniPoziceCyklus
        mov P2, A
        mov pocetRotaci, tmp
        ret

;;; podprogram pro TL2 na zakladni desce

prTlac2:
        inc pocetTlac2
        cjne pocetTlac2, #1, prTlac2Neni1
        mov predvolba, #20
        ret
prTlac2Neni1:
        cjne pocetTlac2, #2, prTlac2Neni2
        mov predvolba, #40
        ret
prTlac2Neni2:
        mov pocetTlac2, #0
        mov predvolba, #10
        cpl smer
        ret

;;; podprogram pro TL1 na desce s displeji

prDTlac1:
        jnb rucniCitani, prDTlac1Inc
        setb RS0
        cjne desetiny, #0, prDTlac1Nulovani
        cjne jednotky, #0, prDTlac1Nulovani
        cjne desitky, #0, prDTlac1Nulovani
        clr RS0
        sjmp prDTlac1Inc
prDTlac1Nulovani:
        mov pocetDTlac1, #0
prDTlac1Inc:
        inc pocetDTlac1
```

```
prDTlac1Porovnani:
        cjne pocetDTlac1, #1, prDTlac1Neni1
        setb RS0
        mov desetiny, #0
        mov jednotky, #0
        mov desitky, #0
        clr RS0
        clr zTecka
        clr blink
        setb rucniCitani
        ret
prDTlac1Neni1:
        cjne pocetDTlac1, #2, prDTlac1Neni2
        setb citani
        setb zTecka
        clr rucniCitani
        ret
prDTlac1Neni2:
        clr citani
        mov pocetDTlac1, #0
        ret

;;; podprogram pro TL2 na desce s displeji, mod citace

prPrictiDisp:
        jnb rucniCitani, prPrictiDispKonec
        setb RS0
        inc desetiny
        cjne desitky, #9, prPrictiDispDale
        cjne jednotky, #9, prPrictiDispDale
        cjne desetiny, #9, prPrictiDispDale
        clr rucniCitani
        clr RS0
        mov pocetDTlac1, #0
        setb RS0
        sjmp prPrictiDispKonec
prPrictiDispDale:
        cjne desetiny, #10, prPrictiDispKonec
        mov desetiny, #0
        inc jednotky
        cjne jednotky, #10, prPrictiDispKonec
        mov jednotky, #0
        inc desitky
prPrictiDispKonec:
        clr RS0
        ret

;;; podprogram pro otoceni krokoveho motoru
```

```
prMotor:
        setb RS1
        inc motorPocet
        cjne motorPocet, #1, prMotorNeni1
        setb P0.0
        setb P0.1
        sjmp prMotorKonec
prMotorNeni1:
        cjne motorPocet, #2, prMotorNeni2
        clr P0.1
        sjmp prMotorKonec
prMotorNeni2:
        cjne motorPocet, #3, prMotorNeni3
        clr P0.0
        sjmp prMotorKonec
prMotorNeni3:
        mov motorPocet, #0
        setb P0.1
        sjmp prMotorKonec
prMotorKonec:
        clr RS1
        ret

;;; podprogram spozdeni, pouzivan vzdy po stisku tlacitka
;;; osetri zakmity spinacu

prSpozdeni: ; R5-7
        mov R5, #2
        mov R6, #0FFh
        mov R7, #0FFh
pp:
        djnz R7, pp
        djnz R6, pp
        djnz R5, pp
        ret

        end
```

4 Závěr

Myslím si, že požadavkům na mikroprocesorový výukový systém jsem vyhověl a vytvořil jsem učební pomůcku, která pomůže studentům procvičit si programování v jazyce assembler i s praktickým vyzkoušením jejich práce. Díky výstupu na LED diody, případně díky dalším připojeným periferiím je možné vytvářet opravdu kreativní a zajimavé příklady využívající tento mikroprocesorový systém.

Program RD2prog vytvořený v rámci této práce navíc usnadní kompilaci zdrojového kódu a případně následnou opravu chyb díky interaktivnímu zobrazení průběhu kompilace a zvýrazňování řádků, kde se chyba vyskytuje. Program je také velmi jednoduše rozšiřitelný díky dostupnosti zdrojového kódu. Jednoduše lze přidat například funkci pro čtení paměti programu nebo dat mikroprocesoru.

Seznam tabulek

Seznam obrázků

Seznam použitých pramenů, literatury a aplikací

[1] *Vladimír Šubrt:* **ATMEL AVR – vývoj aplikací**
BEN – technická literatura, Praha 2002

[2] *Vladimír Šubrt:* **Jednočipové mikropočítače INTEL 8048 – 8096**
Grada, Praha 1992

[3] *Petr Skalický:* **Mikroprocesory řady 8051**
BEN – technická literatura, Praha 1997

[4] *Brian W. Kernighan, Dennis M. Ritchie:* **Programovací jazyk C**
Computer Press, 2006

[5] *Stephen Prata:* **Mistrovství v C++**
Computer Press, 2004

[6] *RD2 Kit ze serveru hw.cz*
`http://hw.cz/Produkty/ART130-Programujte-v-C---RD2-Kit.html`

[7] *Datasheet k mikroprocesoru AT89C51RD2*
`http://www.atmel.com/dyn/resources/prod_documents/doc4235.pdf`

[8] *Popis rozdílů mezi mikroprocesory T89C51RD2 a AT89C51RD2*
`http://www.atmel.com/dyn/resources/prod_documents/doc4239.pdf`

[9] *Obecné informace o bootloaderu a ISP*
`http://www.atmel.com/dyn/resources/prod_documents/doc7716.pdf`

[10] *Datasheet k integrovanému obvodu MAX810*
`http://datasheets.maxim-ic.com/en/ds/MAX803-MAX810Z.pdf`

[11] *Datasheet k integrovanému obvodu MAX232*
`http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf`

[12] *Dokumentace frameworku Qt od firmy Nokia, verze 4.6*
`http://qt.nokia.com/doc/4.6/`

[13] *Dokumentace knihovny libudev*
http://www.kernel.org/pub/linux/utils/kernel/hotplug/libudev/

[14] *Popis zařízení v Linuxu*
http://www.kernel.org/pub/linux/docs/device-list/devices.txt

[15] *Knihovna MSDN, dokumentace Windows API*
http://msdn.microsoft.com/en-us/library/

[16] *Seriál Qt 4 – psaní grafických programů*
http://www.abclinuxu.cz/serialy/qt-4-psani-grafickych-programu

[17] *Seriál Jak na LaTeX*
http://www.root.cz/serialy/jak-na-latex/

[18] *Trial verze programu Altium Designer*
http://altium.com/products/altium-designer/en/

[19] *Aplikační framework Qt a vývojové nástroje*
http://qt.nokia.com/products

[20] *GDB: The GNU Project Debugger*
http://www.gnu.org/software/gdb/

[21] *TeX Live*
http://www.tug.org/texlive/

[22] *Kile*
http://www.kde.org/applications/office/kile

[23] *GIMP – the GNU Image Manipulation Program*
http://www.gimp.org/

[24] *Inkscape*
http://www.inkscape.org/

[25] *KSnapshot*
http://www.kde.org/applications/graphics/ksnapshot