KONSTRUKČNÍ ELEKTRONIKA A Radio

ROČNÍK VIII/2003. ČÍSLO 2

Amatérské RADIO

ŘADA B - PRO KONSTRUKTÉRY

ROČNÍK LII/2003. ČÍSLO 2

V TOMTO SEŠITĚ

KONSTRUKČNÍ ELEKTRONIKA A RADIO

Vydavatel: AMARO spol. s r. o.
Redakce: Radlická 2, 150 00 Praha 5, tel.: 2 57 31 73 11, tel./fax: 2 57 31 73 10.
Šéfredaktor ing. Josef Kellner, sekretářka redakce Eva Kelárková, tel. 2 57 31 73 14.
Ročně vychází 6 čísel. Cena výtisku 36 Kč.
Rozšiřuje PNS a. s., Transpress spol. s r. o., Mediaprint & Kapa a soukromí distributoři.
Předplatné v ČR zajišťuje Amaro spol. s r. o. - Michaela Jiráčková, Hana Merglová (Radlická 2, 150 00 Praha 5, tel./fax: 2 57 31 73 13, 2 57 31 73 12. Distribuci pro předplatitele také provádí v zastoupení vydavatele společnost Mediaservis s. r. o., Abocentrum, Moravské náměstí 12D, P. O. BOX 351, 659 51 Brno; tel: 5 4123 3232; fax: 5 4161 6160; abocentrum@mediaservis.cz; reklamace - tel.: 800 171 181.
Objednávky a predplatné v Slovenskej republike vybavuje MAGNET-PRESS Slovakia s. r. o., Teslova 12, P. O. BOX 169, 830 00 Bratislava 3, tel./fax (02) 44 45 45 59, (02) 44 45 06 97 - predplatné, (02) 44 45 46 28 - administratíva; email: magnet@press.sk
Podávání novinových zásilek povoleno Českou poštou - ředitelstvím OZ Praha (č.j. nov 6005/96 ze dne 9. 1. 1996).
Inzerci v ČR přijímá redakce, Radlická 2, 150 00 Praha 5, tel.: 2 57 31 73 11, tel./fax: 2 57 31 73 10.
Inzerci v SR vyřizuje MAGNET-PRESS Slovakia s. r. o., Teslova 12, 821 02 Bratislava, tel./fax (02) 44 45 06 93.
Za původnost a správnost příspěvků odpovídá autor (platí i pro inzerci). Nevyžádané rukopisy nevracíme.
http://www.aradio.cz; *E-mail*: pe@aradio.cz
ISSN 1211-3557, MKČR 7443

Ročník 2002 na CD ROM

Vážení čtenáři, nyní vychází nový CD ROM s ročníkem 2002 všech časopisů našeho vydavatelství.

CD ROM 2002 zahrnuje kompletní obsah časopisů Praktická elektronika A Radio, Konstrukční elektronika A Radio, Electus 2002, Amatérské radio a Stavebnice a Konstrukce za rok 2002 (inzerce je vynechána).

Vše je zpracováno ve formátu pro elektronické publikování **Adobe PDF**.

Na disku je nahrán nový prohlížecí program **Adobe Acrobat Reader 5.05CZ.** Nelze použít starší verzi 3.0, proto si musíte vždy starý prohlížeč přeinstalovat.

Po nainstalování prohlížecího programu Acrobat jsou tři možnosti otevření požadovaného časopisu. První možností je otevřít přímo soubor, např. _PE07_2002.pdf a ukáže se první strana čísla 7 Praktické elektroniky A Radia. V ní můžeme listovat pomocí šipek v liště nástrojů nebo stačí kliknout na číslo stránky v obsahu a ta se sama zobrazí.

Druhou možností je otevřít soubor _AMARO2002.pdf. Objeví se dvě stránky se všemi titulními listy jednotlivých časopisů. Stačí kliknout na jeden z nich, otevře se žádaný časopis na první straně a dále pokračujeme jako v předchozím odstavci.

Poslední možnost je otevřít soubor _OBSAH2002.pdf, objeví se známý obsah z PE 12/2002 (neobsahuje Amatérské radio) a kliknutím na číslo stránky se otevře přímo požadovaný článek.

Na zbytek místa na CD ROM jsme nahráli:

- **Katalog firmy OKI - zastupuje ji firma Spezial Electronic. Obsahuje např. procesory, obvody pro USB, Bluetooth apod.**
- **Programy ke konstrukcím uveřejněným v PE a KE.**
- **Katalog firmy PS electronic. Obsahuje katalogové listy součástek.**
- **Katalog firmy BEN - technická literatura.**

Věříme, že se vám bude nový CD ROM líbit a že jím opět rozšíříte svou elektronickou knihovnu.

Redakce

Popsaný CD ROM si lze objednat telefonicky (2 57 31 73 12 a 2 57 31 73 13) nebo poštou na dobírku, případně osobně na adrese: AMARO spol. s r. o., Radlická 2, 150 00 Praha 5. CD ROM si také bude možné zakoupit v některých prodejnách knih a součástek.

Lze si ho také objednat na Internetu: www.aradio.cz; e-mail: pe@aradio.cz

Cena CD ROM je 350 Kč + poštovné + balné. Předplatitelé časopisů u firmy AMARO mají výraznou slevu. Pouze pro ně bude CD ROM v ceně 220 Kč + poštovné + balné.

Zájemci na Slovensku si mohou CD ROM objednat u firmy Magnet-Press Slovakia s. r. o., P. O. BOX 169, 830 00 Bratislava, tel./fax (02) 444 545 59, magnet@press.sk

Z dějin vědy a techniky

Historie elektřiny a magnetizmu

Henry Cavendish

Henry Cavendish žil v letech 1731 až 1810. Byl to především chemik a patří k zakladatelům vědecké experimentální chemie. Zajímal se však také o elektrické jevy a i v této oblasti experimentoval.

V neposlední řadě to však byl také podivín, který se ani nezajímal o případnou publikaci svých objevů, které uchovával pouze ve svých pečlivě vedených laboratorních sešitech. Ty se pak nějakým způsobem po jeho smrti dostaly do archivu vědeckých laboratoří v Cambridge (mimochodem, ty dnes nesou jeho jméno).

Když se známý fyzik James Clark Maxwell stal ředitelem těchto laboratoří, začal se zajímat také o archivní materiály. Zjistil, že v detailních laboratorních popisech pokusů, které konal Cavendish, jsou zaznamenány některé významné objevy z oboru chemie a jako fyzika jej nesmírně překvapilo, že jsou tam svým způsobem odvozeny také Ohmův zákon a Coulombův zákon! K jejich objevu tedy došlo podstatně dříve (u Ohmova zákona to bylo 70 let před jeho oficiálním publikováním, u Coulombova zákona o devět let), než se v té době tradovalo; nedá se také předpokládat, že by Coulomb (Francouz) a Ohm (Němec) měli možnost ve své době čerpat z Cavendishových poznatků. Maxwell pak tento překvapivý objev zveřejnil, ale na tom, že svět uznává jako autory těchto principiálních objevů Coulomba a Ohma, se již nic nezměnilo. Oni totiž byli první, kdo tyto poznatky zveřejnili (a není pochyb o tom, že je získali na základě svých vlastních experimetů).

Z Cavendishových objevů v oblasti chemie musíme jmenovat objev „hořlavého vzduchu" (vodíku), nezávisle na jiných objevil dusík a zjistil, že mimo kyslíku a dusíku musí ve vzduchu existovat ještě něco dalšího. Teprve opakování jeho pokusů po 100 letech znamenalo objev argonu a dalších plynů. Pomocí elektrického výboje prokázal složení vody.

Elektronka a její počátky - D. Fleming, Lee de Forest

Patent na dvouelektrodovou elektronku (v té době se nazývala lampou, a upřímně řečeno, svítit se s ní dalo...), tzv. Flemingovu diodu, byl vydán 16. 1. 1904. Byl to jeden ze dvou mezníků, které sehrály významnou roli v dějinách radiotechniky.

Je třeba přiznat, že vlastně prvním, kdo přišel na efekt průchodu proudu mezi rozžhaveným vláknem žárovky a kovovou destičkou zatavenou v blízkosti tohoto vlákna, byl T. A. Edison. V té době byl D. Fleming konzultantem u jedné Edisonovy společnosti, a tak byl s pokusy seznámen, ovšem tehdy ještě nikdo nedomyslel, jaký význam tento objev jednou bude mít. Fleming se pak stal profesorem na londýnské univerzitě a poradcem firmy, kterou založil Marconi.

Patřil ke skupině nadšenců, kteří se pokoušeli překonat Atlantický oceán pomocí rádiových vln v roce 1901 a on sám byl při pokusech na straně evropského vysílače v Poldhu, zatím co Marconi na New Foundlandu. Flemingova lampa byla již tehdy v přístroji použita jako detektor!

V roce 1906 Fleming publikoval poznatky tehdejší vědy v knize „The principles of electric waves, telegraphy and telephony". Byl jedním z autorů, kteří se podíleli na sepsání proslulé „Encyklopedia Brittanica" v letech 1910 až 1912, kde se zabýval elektrotechnikou. V roce 1929 byl povýšen do šlechtického stavu.

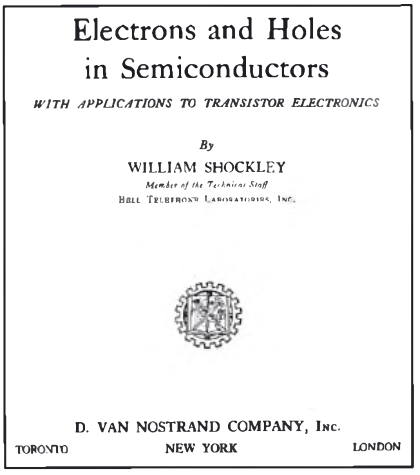
I když dioda neumožňovala zesilovat signály, s jejím využitím byla přenášena již v roce 1906 řeč i hudba na vzdálenost asi 40 km. 4. března 1906 Robert von Lieben podal přihlášku patentu na „katodové relé" - elektronku, která skutečně byla později používána v některých telefonních ústřednách místo mechanických relé. Patent pak převedl na firmu Telefunken. To však ještě stále nebyl znám zesilovací efekt - ten objevil sice též v roce 1906, ale později, Lee de Forest.

Jeho jméno vešlo do podvědomí tisíců Američanů v roce 1910, když světoznámý italský tenor Enrico Caruso (1873 až 1921) zpíval v newyorské Metropolitní opeře nadšeným posluchačům. Tento koncert byl poprvé bezdrátově přenášen pomocí rozhlasu a o přenos se nejvíce zasloužil právě americký vysokofrekvenční technik Lee de Forest.

William Shockley

William Shockley si poznamenal 24. 12. 1947 do svého pracovního sešitu, když studoval povrchové jevy v pevných látkách: „Zesiluje nejméně osmnáckrát". Tehdy zkoumal plátek krystalového germania, na který měl zapojen mikroampérmetr. Toto zjištění mu později umožnilo vypracovat teorii tranzistorového jevu a v červnu 1948 na tiskové konferenci představit první tranzistor.

Skeptici namítali, že se jedná jen o vylepšený druh detektoru, a u nás ještě v roce 1957 prof. Stránský, když


Electrons and Holes
in Semiconductors
WITH APPLICATIONS TO TRANSISTOR ELECTRONICS
By
WILLIAM SHOCKLEY
Member of the Technical Staff
BELL TELEPHONE LABORATORIES, INC.
D. VAN NOSTRAND COMPANY, INC.
TORONTO NEW YORK LONDON

Titulní list Shockleyho knihy o polovodičích z roku 1950

ukazoval na přednášce z vysokofrekvenční elektrotechniky studentům jeden z prvních hrotových tranzistorů 2N109 již vyráběný a dodávaný firmou RCA, prohlásil „... a o tomhle si někteří lidé myslí, že nahradí elektronky".

Shockley ovšem byl týmovým vědcem, již méně se mluví o dalších členech jeho týmu - Johnu Bardeenovi a Waltru Brattainovi, a mimo tohoto objevu nijak nevynikl. Naproti tomu např. Schottky dal elektrotechnice objevů mnoho.

Když se vrátíme k objevu tranzistoru, mimo počáteční oslavné ódy se vlastně nic nestalo. Průmysl elektronek dále vzkvétal, i když složitější přístroje, mezi kterými zvláště oblast výpočetní techniky postupně zaujímala stále významnější místo, začínaly mít neúnosně velké rozměry. Průlom do tohoto stavu přišel teprve asi po deseti letech, když se na trhu v USA objevil první přenosný radiopřijímač, který nepotřeboval rozměrnou anodovou baterii a okamžitě se stal módním hitem.

Mezitím se ovšem také původní tranzistory hodně změnily - místo původních hrotových se objevily nové výrobní technologie umožňující sériovou velkovýrobu a tím jejich zlevnění, místo germania přišel ke slovu křemík a další materiály.

Shockley ještě před rozšířením tranzistorových přístrojů obdržel spolu se svými spolupracovníky v roce 1956 Nobelovu cenu. Jako podnikatel však nebyl úspěšný, na dalším vývoji svého objevu se dále nepodílel, ale přijal místo profesora na známé Stanfordově univerzitě. Úspěšnější byl jeho kolega, John Bardeen, který se věnoval výzkumu supravodivosti a za práce v tomto oboru byl odměněn Nobelovou cenou podruhé.

(Dokončení na str. 39)

MĚŘICÍ PŘÍPRAVKY JAKO PERIFERIE K PC

Ing. David Matoušek

matousek@vosji.cz

Tento článek ukazuje možnosti tvorby zařízení ovládaných osobním počítačem. Zařízení se připojují k paralelnímu nebo sériovému portu (častěji) počítače a jsou řízena programy, které běží pod operačním systémem Windows 95 a vyšším. Programy pro Windows jsou vytvořeny v oblíbeném vývojovém prostředí C++ Builder od firmy Inprise (dříve Borland). Zařízení v některých případech obsahují jednočipové mikrořadiče (= mikrokontroléry = procesory), jednodušší konstrukce mikrořadič nepotřebují.

Jsou publikovány konstrukce tohoto typu: vstupně/výstupní desky, A/D převodníky k počítači, programátory mikrořadičů a pamětí E²PROM (AT989S8252, AT90S2313, 24Cxx) a další.

Před tím, než se „pustíme" do popisu konstrukce uvedených zařízení, je nutné vysvětlit některé kroky tvorby zařízení tohoto typu. Jedná se zejména o popis vývojového prostředí C++ Builder, úvodní popis mikrořadiče AT89C2051, popis paralelních a sériových portů osobního počítače a jejich ovládání.

1. Vývojové prostředí C++ Builder

Vývojové prostředí C++ Builder velmi významně zjednodušuje vývoj aplikací pro operační systémy Windows 95 a vyšší.

Jak už je zřejmé z názvu, je toto prostředí určeno pro vývoj aplikací zapsaných v programovacím jazyce C++. Přičemž se opírá o poslední standardizaci ANSI C++, takže je plně kompatibilní s jinými C++ překladači. Navíc připojuje poměrně velké množství nově zavedených klíčových slov, které jsou potřebné pro podporu rychlého návrhu aplikací.

Po spuštění C++ Builderu se zobrazí prázdný **formulář** (viz obr. 1.1), který vlastně odpovídá hlavnímu oknu vytvářené aplikace. Další formuláře lze přidávat položkou menu **File|New Form**.

Dalším prvkem C++ Builderu je **paleta komponent**. Ta obsahuje různé komponenty (např. tlačítka, editační políčka, posuvníky apod.), které potřebujeme pro tvorbu aplikace. Stačí si takovou komponentu přetáhnout na plochu formuláře a ona se pak objeví ve výsledném okně.

Posledním a patrně nejdůležitějším nástrojem C++ Builderu je **objekt inspektor**. Toto okno slouží pro pohodlnou editaci vlastností a událostí komponent.

Vlastností komponenty rozumíme například její barvu (Color), typ použitého písma (Font), titulek (Caption) či rozměry a umístění (Left, Top, Width a Height). Vlastnosti se nastavují v záložce **Properties**.

Událostí komponenty rozumíme specifickou metodu (funkci), která se má vykonat v okamžiku, kdy nastane určitá událost. Může se například jednat o událost OnClick. Ta nastane, pokud uživatel klikne levým tlačítkem myši na dané komponentě (tato událost se hojně používá u tlačítek nebo položek menu). Události se generují v záložce **Events**. Po zapsání jména události se do editoru zdrojového textu (na obr. 1.1 je schován za formulářem) vloží příslušná definice a uživatel může okamžitě napsat reakci na příslušnou událost.

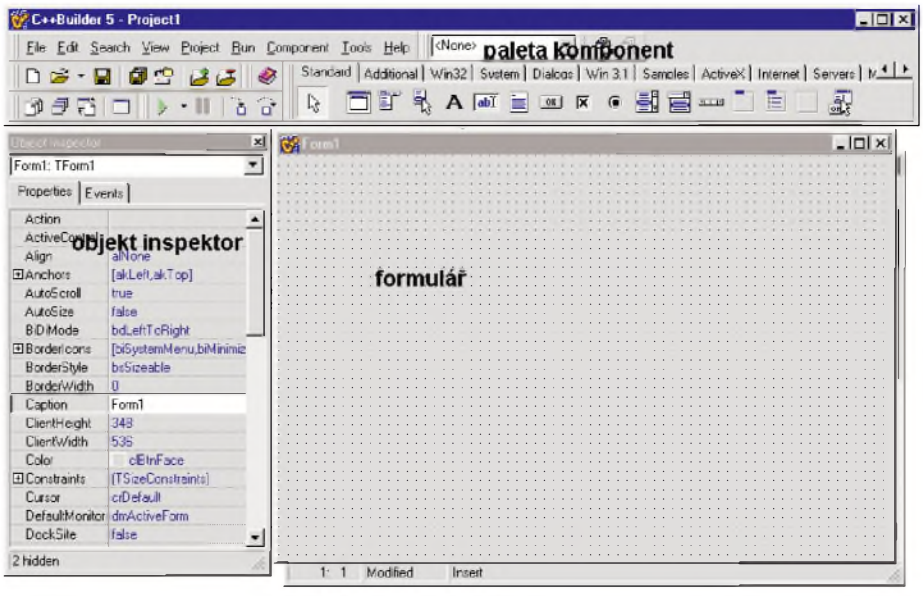
Např. na obr. 1.2 je vloženo tlačítko se jménem **Button1**, kterému byl změněn titulek na **Konec** (také byl změněn font). Dále jsem vytvořil událost **KonecClick**, kam jsem zapsal volání metody **Close**, která zavře formulář. Tím se ukončí celá aplikace.

Knihovna VCL

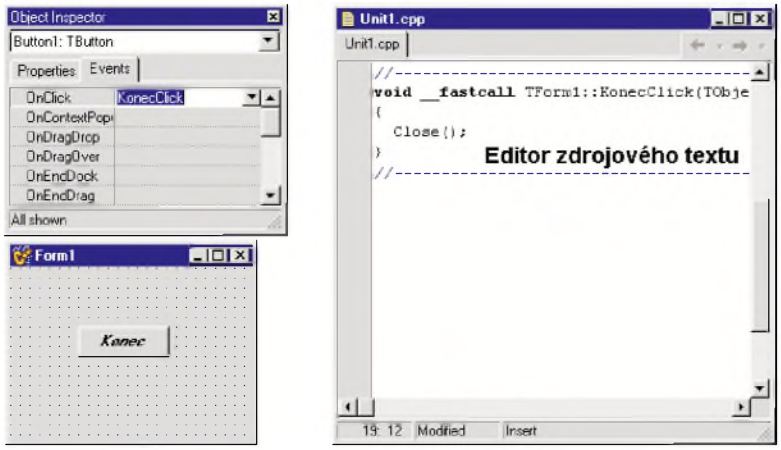
Základem C++ Builderu je knihovna označená jako **VCL** - Visual Component Library. Tato knihovna obsahuje veškeré komponenty, které je možno vkládat do formuláře v době návrhu. Definuje však také množství tříd, které zjednodušují programování pod operačním systémem Windows.

Komponenty jsou v paletě komponent uspořádány do skupin podle podobnosti. Například záložka **Standard** obsahuje nejčastěji používané komponenty (Label - popisek, Button - tlačítko, Edit - editační políčko, MainMenu - hlavní menu, PopupMenu - místní menu, CheckBox - zaškrtávací políčko, ScrollBar - posuvník, ListBox - seznam a mnoho dalších).

Další třídy např. zjednodušují práci se soubory (TFileStream) nebo s dynamickou pamětí (TMemoryStream), se systémovou databází Registry (TRegistry), se schránkou (TClipboard), s bitmapovými obrázky (TBitmap) nebo s obrázky ve formátu JPEG (TJPEGImage).



Obr. 1.1. Vývojové prostředí C++ Builder



Obr. 1.2. Práce s událostmi a editorem zdrojového textu

Windows API

Windows API je knihovna funkcí pro vytváření aplikací pod operačním systémem Windows. Tato knihovna je součástí operačního systému a je tedy vytvořena přímo společností **Microsoft**. Nápovědu k funkcím Windows API získáme po aktivaci položky menu **Help|Windows SDK Help**.

Funkce Windows API musíme použít např. pro přístup k paralelnímu nebo sériovému portu počítače (také lze používat specializované komponenty, které jsou nabízeny na některých internetových stránkách).

Verze C++ Builderu

C++ Builder se vyvíjel již od roku 1997, jeho vývoj však pokračuje i v současnosti (v roce 2003 se očekává verze 7). Jednotlivé verze se liší nabízenými možnostmi podpory programování. Zohledňují se například nové verze operačních systémů (Windows 98 resp. Windows 2000 a další).

Dále se jednotlivé verze dělí na kompilace **Personal** (dříve Standard), **Professional** a **Enterprise**. Od verze C++ Builderu 6 není možno ve verzi Personal vytvářet volně prodejné aplikace (tato verze je vlastně určena pro výuku programování pomocí C++ Builderu). Vyšší kompilace podporují tvorbu databázových aplikací a aplikací pro Internet.

Z hlediska zapálených amatérů je jasné, že vystačíme s kompilací Personal. Pokud se rozhodnete aplikace prodávat, musíte si zakoupit kompilaci Professional nebo Enterprise (tyto kompilace jsou však podstatně dražší než Personal).

Další literatura

Je jasné, že výše uvedený popis nedává dostatečně silné základy proto, abyste mohli okamžitě začít programovat (i když programovaní s použitím C++ Builderu je poměrně jednoduché). Spíše se jedná o informaci pro ty, kteří si pouze postaví publikovaná zařízení a mají mít rámcovou představu o tvorbě ovládacích programů.

Další informace lze načerpat např. z [1], [2] a [3]. Základním pramenem je [1], další ukazují pokročilé rysy programování za pomoci funkcí Windows API a DirectX.

Další překladače

Pro vývoj aplikací pro operační systém Windows lze pochopitelně používat i jiná vývojová prostředí. Zmíním se krátce pouze o dvou zástupcích:

Delphi je rovněž produktem společnosti Inprise a umožňuje programovat pod Windows v programovacím jazyce Object Pascal. Opírá se o stejnou množinu komponent a pomocných tříd jako C++ Builder. Já si před lety zvolil **C++ Builder** proto, že C++ se ukazoval jako progresivnější programovací jazyk.

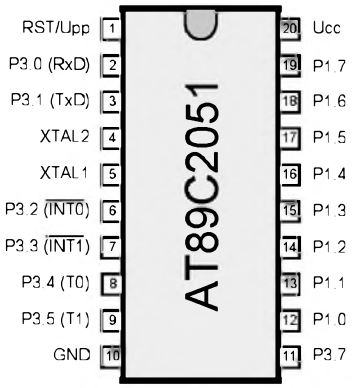
Visual C++ je produktem společnosti Microsoft. Jedná se o vývojové prostředí pro vývoj aplikací pomocí programovacího jazyka C++. Přiznejme si ale, že vizuální návrh (přestože je deklarovaný v názvu) není příliš podporován a ve srovnání proti C++ Builderu neobstojí. Proto si myslím, že přes jiné výhody tohoto prostředí je pro začátečníka poměrně nevýhodný.

2. Stručný popis mikrořadiče AT89C2051

V této kapitole se seznámíme s klíčovými vlastnostmi mikrořadiče Atmel AT89C2051, který je použit v mnoha následujících konstrukcích. Jeho blokové schéma je na obr. 2.4. Podrobnější popis lze nalézt např. v [4].

Mezi jeho základní vlastnosti patří:

- Programová paměť Flash velikosti 2 KB, zaručovaný počet přeprogramování je 1000 cyklů (konstrukce programátoru je uvedena ve [4] nebo [5]).
- Datová RAM o kapacitě 128 B.
- Napájecí napětí v rozsahu 2,7 až 6 V.
- Mikrořadič může ovládat 15 vstupně/výstupních linek, které mohou přímo budit LED (zkratový proud jedné linky je 20 mA, součet proudů všech výstupů nesmí překročit 80 mA).
- Analogový komparátor.
- Dva šestnáctibitové čítače/časovače.
- Programovatelný sériový kanál.



Obr. 2.1. Zapojení vývodů mikrořadiče AT89C2051 v pouzdru DIP 20

Všechny tyto vlastnosti umožňují používat mikrořadič AT89C2051 v systémech řízených sériovým portem počítače PC, ve kterých vystačíme s menším počtem vývodů.

Zapojení vývodů mikrořadiče

Na obr. 2.1 je uvedeno zapojení jednotlivých vývodů mikrořadiče AT89C2051 v pouzdru DIP 20:

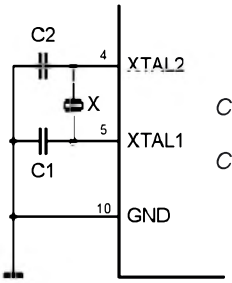
Ucc a **GND** slouží pro připojení napájecího napětí (v rozsahu 2,7 až 6 V).

XTAL1 a XTAL2 slouží pro připojení krystalu. Pokud místo krystalu použijeme vnější hodinový signál, ponecháme XTAL2 nezapojený a hodinový signál připojíme na vývod XTAL1 (viz obr. 2.2 a obr. 2.3.).

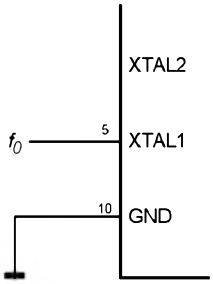
RST je nulovací vstup. Přivedeme-li na tento vstup úroveň „log. 1" alespoň po dobu dvou strojových cyklů (každý strojový cyklus trvá 12 hodinových cyklů), vyvoláme reset mikrořadiče.

Port 1 (P1.0 až P1.7) obsahuje 8 vstupně/výstupních linek. Vývody P1.2 až P1.7 jsou opatřeny vnitřními zdvihacími rezistory (pull-up), které zajišťují definovanou logickou úroveň („log. 1") těchto vývodů i v případě, že nejsou nikam připojeny. Vývody P1.0 a P1.1 nejsou zdvihacími rezistory opatřeny. Je-li třeba zajistit definovanou logickou úroveň těchto vstupů, musí být zdvihací rezistory připojeny z vnějšku. Vývod P1.0 (AIN0) je neinvertujícím vstupem a P1.1 (AIN1) je invertujícím vstupem vnitřního analogového komparátoru (výstup komparátoru je k dispozici na z vnějšku nedostupném vývodu P3.6).

Port 3 (P3.0 až P3.5 a P3.7) obsahuje 7 vstupně/výstupních linek. Všechny jsou opatřeny zdvihacími rezistory. Linka P3.6 není dostupná z vnějšku a je připojena na výstup analogového



Obr. 2.2. Připojení krystalu. C1 = C2 = 30 ±10 pF pro krystaly, C1 = C2 = 40 ±10 pF pro keramické rezonátory



Obr. 2.3. Použití vnějšího zdroje hodinového signálu f_0

komparátoru. Port 3 sdružuje také funkce spojené s vnitřními periferiemi (viz tab. 2.1).

Tab. 2.1. Alternativní význam vývodů portu P3

Vývod	Druhá funkce
P3.0	RxD (sériový vstup)
P3.1	TxD (sériový výstup)
P3.2	INT0non (vstup vnějšího přerušení číslo 0)
P3.3	INT1non (vstup vnějšího přerušení číslo 1)
P3.4	T0 (vnější vstup čítače/ /časovače číslo 0)
P3.5	T1 (vnější vstup čítače/ /časovače číslo 1)

Sériový kanál

Mikrořadič **AT89C2051** obsahuje plně duplexní sériový kanál (příjem i vysílání může probíhat současně).

Pro práci se sériovým kanálem jsou určeny registry:

- Řídicí registr **SCON** (konfiguruje vlastnosti sériového kanálu).
- Datový registr **SBUF** (slouží pro příjem/vysílání znaku).
- Bit **SMOD** v registru PCON ovlivňuje přenosovou rychlost.

Registr SBUF slouží pro příjem/vysílání znaku. Zápis znaku do SBUF způsobí jeho vyslání (při správné konfiguraci sériového kanálu). Podobně čtením SBUF získáme přečtený znak (pokud je příjem znaku povolen).

Úlohu bitů řídicího registru SCON vysvětluje obr. 2.5.

Nejčastěji je používán režim (mód) 1 (SM1 = 1, SM0 = 0) - viz obr. 2.6. Jedná se o osmibitový asynchronní přenos dat. Bity se vysílají na TxD (P3.1) a přijímají na RxD (P3.0). Přenos začíná start-bitem („log. 0"), následuje 8 datových bitů (v pořadí od nejméně významného k nejvíce významnému) a poslední je stop-bit („log. 1"). Přenosová rychlost je dána přetečením časovače 1.

Pro časovač 1 nastavený do režimu (módu) 2 je přenosová rychlost *PR*:

$$PR = (2^{SMOD}/32)\cdot[f_0/(12\cdot(256 - TH1))],$$

kde:

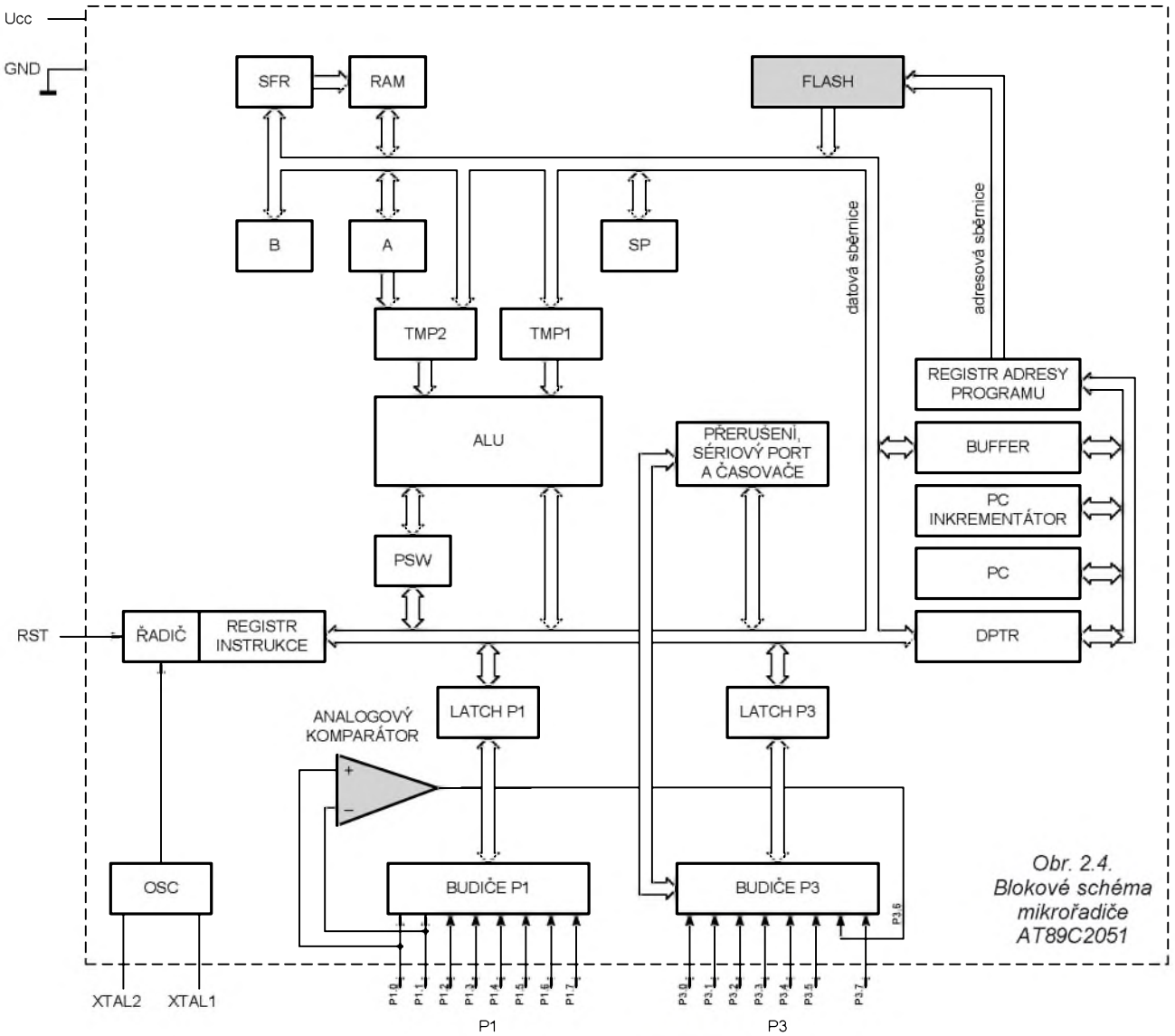
f_0 je hodinový kmitočet mikrořadiče (až 24 MHz),

TH1 je obsah registru TH1 časovače 1.

Bit **SMOD** v registru PCON umožňuje zdvojnásobit přenosovou rychlost v režimech 1, 2 a 3.

Přenosové rychlosti vysílače a přijímače při asynchronním přenosu (režimy 1, 2 a 3) nemusí být shodné, nesmí se však vzájemně lišit o více než ±5 %!

V tab. 2.2 jsou hodnoty registru TH1, který je použit jako osmibitový časovač udávající přenosovou rychlost



Obr. 2.4. Blokové schéma mikrořadiče AT89C2051

Tab. 2.2. Hodnoty TH1 a SMOD pro různé přenosové rychlosti a krystaly

f_0 = 11,059 MHz

PR [Bd]	TH1	SMOD	δ_{PR} [%]
300	64	1	–0,002
600	160	1	–0,002
1200	208	1	–0,002
2400	232	1	–0,002
4800	244	1	–0,002
9600	250	1	–0,002
19200	253	1	–0,002

f_0 = 12 MHz

PR [Bd]	TH1	SMOD	δ_{PR} [%]
300	48	1	+0,2
600	152	1	+0,2
1200	204	1	+0,2
2400	230	1	+0,2
4800	243	1	+0,2
9600	–	–	–
19200	–	–	–

f_0 = 24 MHz

PR [Bd]	TH1	SMOD	δ_{PR} [%]
300	48	0	+0,2
600	48	1	+0,2
1200	152	1	+0,2
2400	204	1	+0,2
4800	230	1	+0,2
9600	243	1	+0,2
19200	–	–	–

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0, SM1 – režim sériového kanálu:

mód	SM0	SM1	režim	přenosová rychlost
0	0	0	8bitový posuvný registr	OSC/12
1	0	1	8bitový asynchronní přenos	čítač/časovač 1
2	1	0	9bitový asynchronní přenos	OSC/64 nebo OSC/32
3	1	1	9bitový asynchronní přenos	čítač/časovač 1

SM2 – povolení tzv. víceprocesorové komunikace

REN – povolení příjmu

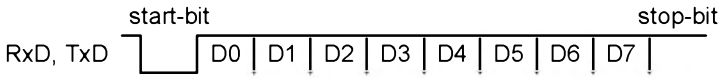
TB8 – vysílaný 9. bit (v režimech 2,3)

RB8 – přijatý 9. bit (v režimech 2,3)

TI – indikace vyprázdnění vysílacího registru,
v režimu 0 je aktivován na konci vysílání 8. bitu,
v režimech 1, 2 a 3 je aktivován na začátku stop-bitu
tento bit se nastaví hardwarově, nuluje se programově

RI – indikace naplnění přijímacího registru,
v režimu 0 je aktivován po přijetí 8. bitu,
v režimech 1, 2 a 3 je aktivován uprostřed stop-bitu
tento bit se nastaví hardwarově, nuluje se programově

Obr. 2.5. Registr SCON



Obr. 2.6. Režim (mód) 1

sériového kanálu. Hodnoty platí pro režim 1 (osmibitový přenos bez parity). Tabulka je sestavena pro krystaly použité v uvedených konstrukcích. Další informace přesahují rámec tohoto článku a jsou uvedeny např. v [4].

3. Popis paralelních a sériových portů počítače PC

V této kapitole jsou popsány jednotlivé standardy paralelních a sériových portů osobního počítače.

SPP - Standard Parallel Port (standardní paralelní port)

SPP odpovídá původnímu standardu jednosměrné komunikace z počítače na tiskárnu, který je také často označován jako **CENTRONICS**. Data jsou vysílána paralelně jako osmice bitů, jejich tok je řízen několika vodiči - viz tab 3.1.

Fyzicky se paralelní port ovládá přístupem na tři porty. První port má bázovou adresu označenou jako BA, další dva porty mají adresy BA+1 a BA+2. První z portů (s adresou BA) ovládá výstupní osmibitová data. Druhý port (adresa BA+1) je vstupní, k dispozici je 5 bitů. Třetí port (adresa BA+2) je výstupní a obsahuje 4 bity.

Časování zápisu na paralelní port podle standardu SPP je na obr. 3.1.

Data se s určitým předstihem připojí na vodiče D0 až D7 a potvrdí se aktivací výstupu STBnon (STBnon přejde do úrovně „log. 0"). Pokud není výstupní zařízení schopno data okamžitě zpracovat, aktivuje vstup BUSYnon. Nakonec aktivuje vstup ACKnon, kterým potvrdí schopnost přijmout další bajt.

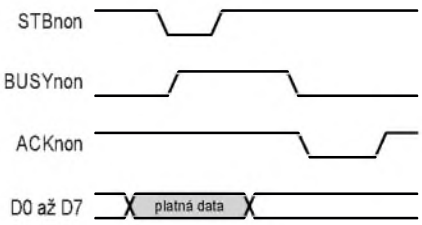
EPP - Enhanced Parallel Port (paralelní port s rozšířenými možnostmi)

Počítače vyrobené po roce 1995 jsou obvykle vybaveny paralelním portem, který je schopen pracovat v normálním režimu (SPP) nebo v rozšířeném režimu (EPP).

Volba režimu paralelního portu je v případě vestavěných portů zajištěna pomocí programu SETUP.

Standard EPP definuje obousměrný přenos dat mezi počítačem a připojenou periferií rychlostí až 2 MB/s. Periferie může používat až 256 vstupních a 256 výstupních registrů, protože po datové sběrnici lze přenášet nejen data, ale i osmibitovou adresu.

EPP používá stejný konektor jako SPP. Význam většiny signálů EPP je velmi podobný jako u SPP.



Obr. 3.1. Časování zápisu na SPP

Tab. 3.1. Vývody SPP a jejich význam

Vývod	Adresa/bit	Název	Směr
1	BA+2/0	STBnon	výstup
2	BA/0	D0	výstup
3	BA/1	D1	výstup
4	BA/2	D2	výstup
5	BA/3	D3	výstup
6	BA/4	D4	výstup
7	BA/5	D5	výstup
8	BA/6	D6	výstup
9	BA/7	D7	výstup
10	BA+1/6	ACKnon	vstup
11	BA+1/7	BUSYnon	vstup
12	BA+1/5	PE	vstup
13	BA+1/4	SEL	vstup
14	BA+2/1	AUTOFEED	výstup
15	BA+1/3	ERRnon	vstup
16	BA+2/2	INITnon	výstup
17	BA+2/3	SELIN	výstup
18 až 25	-	GND	-

Tab. 3.2. Vývody EPP a jejich význam

Vývod	Název	Směr	Význam
1	WRITEnon	výstup	směr toku dat (WRITEnon = 0, zápis; WRITEnon = 1, čtení
2 až 9	D0 až D7	vst./výst.	obousměrná datová sběrnice
10	INT	vstup	vstup přerušení (aktivní je vzestupná hrana)
11	WAITnon	vstup	řídí komunikaci (přenos začíná při WAITnon = 0 a končí při WAITnon = 1)
12	-	-	nepoužito
13	-	-	nepoužito
14	DATASTBnon	výstup	indikuje přenos dat (aktivní je stav „log. 0“)
15	-	-	nepoužito
16	RESETnon	výstup	reset periferie (aktivní je stav „log. 0“)
17	ADDRSTBnon	výstup	indikuje přenos adresy (aktivní je stav „log. 0“)
18 až 25	GND	-	signálová zem

Tab. 3.3. Adresy pro ovládání paralelního portu SPP/EPP

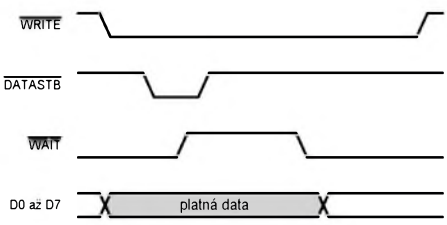
Adresa portu	Význam	Směr
Báze+0	Data (SPP)	výstup
Báze+1	Stav (SPP)	výstup
Báze+2	Řízení (SPP)	výstup
Báze+3	Adresa (EPP)	vst./výst.
Báze+4	Data (EPP)	vst./výst.

V tab. 3.3 jsou jednotlivé adresy, které se používají pro ovládání paralelního portu (SPP i EPP).

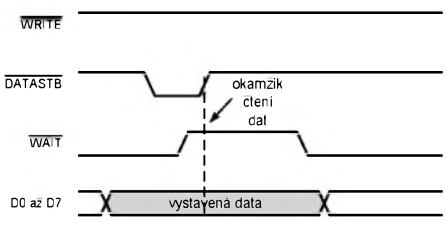
Na tomto místě si uvedeme časovací operace pouze pro čtení a zápis dat (více nebude použito):

Zápis na port Báze+4 vyvolá zápis dat. EPP provede automaticky následující operace:

- WRITEnon přejde do aktivního stavu „log. 0“ a tím indikuje zápis.
- Data se vystaví na vodičích D0 až D7.
- Platnost dat se potvrdí sestupnou hranou signálu DATASTBnon.
- Nyní se vyčkává, až periferie vrátí signál WAITnon do stavu „log. 1“ (tímto signálem si periferie prodlužuje dobu potřebnou pro příjem a zpracování dat)
- signály DATASTBnon a WRITEnon se vrátí do neaktivního stavu „log. 1“.



Obr. 3.2. Časování zápisu dat na EPP



Obr. 3.3. Časování čtení dat z EPP

Čtení z portu Báze+4 vyvolá čtení dat. EPP provede automaticky následující operace:

- WRITEnon zůstává ve stavu „log. 1“ (jedná se o čtení).
- Počítač PC žádá o data sestupnou hranou signálu DATASTBnon.
- Nyní se vyčkává, až periferie vrátí signál WAITnon do stavu „log. 1“ (tímto signálem si periferie prodlužuje dobu potřebnou pro vyslání dat).
- Periferie vloží data na vodiče D0 až D7.
- Data jsou čtena náběžnou hranou signálu DATASTBnon.

UART - Universal Asynchronous Serial Port (asynchronní sériový port)

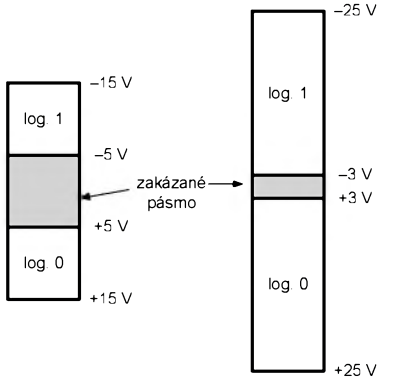
Osobní počítače jsou vybaveny obvykle dvěma asynchronními sériovými kanály (COM1, COM2). Pokud nepoužíváte klasickou sériovou myš (máte myš do zásuvky PS/2 nebo USB) máte tak k dispozici dva sériové kanály.

Vývody sériového portu na konektoru CANNON9 jsou v tab. 3.4.

Přenos dat probíhá po vodičích TxD (výstup) a RxD (vstup). Je-li zvolen formát 8 datových bitů bez parity a jeden stopbit, je situace stejná jako u mikrořadiče AT89C2051 v režimu 1 (viz obr. 2.6).

Tab. 3.4. Vývody sériového portu (konektor CANNON9)

Vývod	Název	Směr
1	RLSD	vstup
2	RxD	vstup
6	DSR	vstup
8	CTS	vstup
9	RING	vstup
5	GND	zem
3	TxD	výstup
4	DTR	výstup
7	RTS	výstup



Obr. 3.4. Definice úrovní RS-232C pro vstupy (vlevo) a výstupy (vpravo)

Přímé řízení sériového portu

Ostatní linky slouží pro řízení modemu, lze je však použít i pro přímé řízení připojeného zařízení. Tak máme k dispozici tři výstupy (TxD, RTS, DTR) a čtyři vstupy (RLSD, DSR, CTS, RING). Vývod RxD nelze v režimu přímého řízení použít.

Napěťové úrovně sériového kanálu

Připomeňme, že sériový kanál pracuje s úrovněmi RS-232C. Vstupy uvažují stav „log. 1“ jako napěťové úrovně -3 až -25 V a stav „log. 0“ jako úrovně +3 až +25 V. Na výstupech je stav „log. 1“ v rozsahu -5 až -15 V a stav „log. 0“ v rozsahu +5 až +15 V - viz obr. 3.4.

4. Ovládání portů pomocí C++ Builderu

V této kapitole jsou uvedeny způsoby ovládání paralelních a sériových portů pomocí aplikace vytvořené ve vývojovém prostředí C++ Builder.

Funkce Window API - Řízení paralelního portu podle standardu SPP

Pro ovládání paralelního portu podle standardu SPP se používají následující funkce Windows API: CreateFile, WriteFile a CloseHandle.

Funkce **CreateFile** otevře paralelní port a získá tzv. handle, ten se pak používá při volání dalších služeb. Nejdůležitější je možnost volit paralelní port (pokud máte k počítači připojeno více paralelních portů) uvedením jeho jména.

Funkce **WriteFile** zapisuje na paralelní port určený svým handle (získá se právě předchozím voláním funkce CreateFile) zvolený počet bajtů.

Funkce **CloseHandle** zavírá port určený svým handle a odevzdává jej tak operačnímu systému. Pokud zapome-

nete funkci CloseHandle zavolat, zavře se port automaticky při ukončení běhu aplikace, která jej používala.

Dále je uveden příklad kódu, který otevře paralelní port LPT1, pošle na něj jeden bajt s hodnotou 57h (57 hexadecimálně) a poté jej zavře:

```
DWORD zapsano;      //počet úspěšně zapsaných bajtů
BYTE Data=0x57;     //zapisovaná hodnota
//otevření LPT1:
HANDLE lpt=CreateFile("LPT1",GENERIC_WRITE,0,NULL,
        OPEN_EXISTING,0,NULL);
//zápis:
WriteFile(lpt,&Data,1,&zapsano,NULL);
//zavření portu:
CloseHandle(lpt);
```

K uvedenému kódu se sluší dodat několik poznámek:

Při otevírání portu funkcí **CreateFile** je požadován přístup pro zápis (čtení z SPP portu nemá význam), to odpovídá symbolu **GENERIC_WRITE**. Symbol **OPEN_EXISTING** se používá pro porty a značí, že se systém pokusí otevřít existující port (pokud bude existovat; u souborů lze soubor založit, pokud neexistoval).

Funkce **WriteFile** přijímá zapisovaná data přes vyrovnávací paměť (buffer). Proto nelze zapisovanou hodnotu zadat přímo, ale musí být uložena do proměnné Data. Hodnota 1 označuje, že buffer **Data** čítá jediný bajt. Proměnná **zapsano** je použita pro získání informace o počtu skutečně zapsaných bajtů (v našem případě ji není třeba testovat).

Z principu lze paralelní port ovládat i zápisem do jeho ovládacích portů instrukcemi **in** a **out**. Takové řešení je však nejen komplikované, ale především nebezpečené. Nezabrání totiž možným kolizím zápisů mezi více programy (představme si tisk na tiskárnu a současný přístup na stejný port). Naproti tomu otevření portu přes CreateFile je možné jen tehdy, pokud daný port nevlastní jiná aplikace (např. i správce tisku). Teprve po zavření portu přes CloseHandle může jeho vlastnictví získat jiná aplikace.

Podrobnější popis uvedených funkcí naleznete buď v nápovědě Windows SDK (v angličtině) nebo v [5].

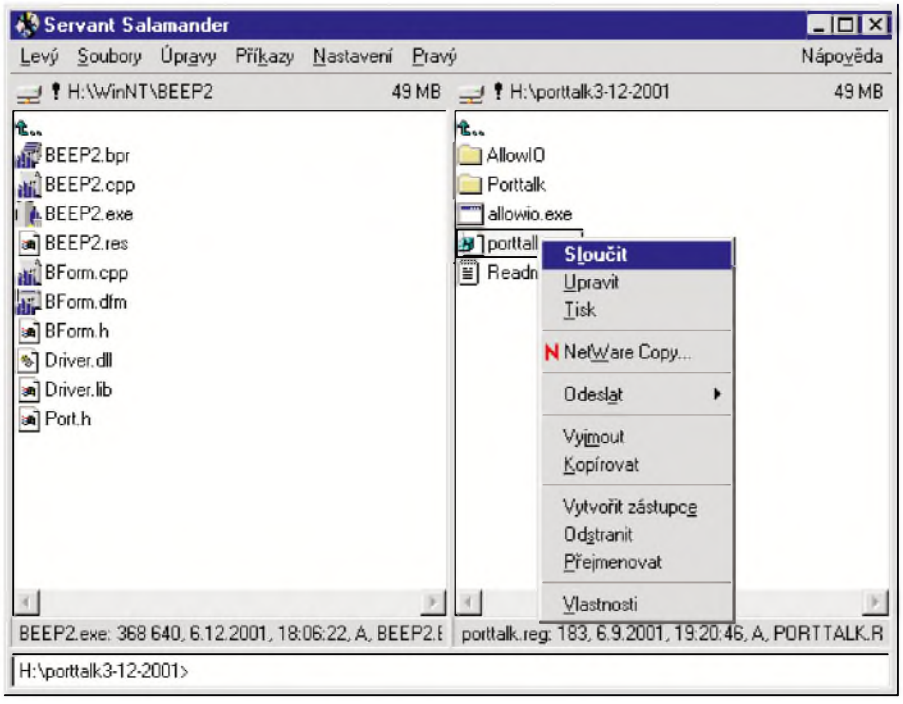
Ovladač PortTalk - Řízení paralelního portu podle standardu EPP

Uvedené funkce pro práci s paralelním portem podle standardu SPP nejsou bohužel použitelné pro ovládání paralelního portu podle standardu EPP. Zde se Microsoft moc „nevyznamenal“, a tak musíme přistupovat přímo k portům počítače. Jak bylo uvedeno výše, je tato akce poměrně nebezpečná.

Naštěstí se však můžeme zaregistrovat jako vlastník portu (voláním **CreateFile**). Funkce ReadFile a WriteFile však nebudou fungovat správně (jsou napsány pro SPP port) a místo toho musíme port řídit instrukcemi **in** a **out**.

S tím je však spojena ochrana operačního systému proti nevhodně napsaným programům. Tato ochrana se uplatňuje na platformě NT (operační systémy typu Windows NT, Windows 2000, Windows XP). Pokusí-li se totiž naše aplikace vykonat instrukci **in** nebo **out**, bude násilně ukončena operačním systémem.

Jedinou možností je používat ovladač pracující na úrovni jádra (přípona SYS). Tomuto ovladači je dovolen přímý přístup k portům procesoru. Takovým ovladačem je PortTalk.



Obr. 4.1. Sloučení souboru PORTTALK.REG s Registry

Instalace PortTalk do systému

Instalace ovladače **PortTalk** začíná zkopírováním souboru **PORTTALK.SYS** do systémového adresáře (název je obvykle C:\WINNT\SYSTEM32\Services).

Potom již jen stačí poklepat na ikonu souboru **PORTTALK.REG** v nějakém souborovém manažeru nebo vybrat příkaz **Sloučit** z místní nabídky (viz obr. 4.1). Tím bude obsah souboru **PORTTALK.REG** sloučen se systémovou databází Registry. Systém pak bude vědet, že po restartu má zavést ovladač **PortTalk** do paměti.

Nakonec je nutné počítač restartovat, aby se PortTalk mohl zavést do paměti.

Po restartu lze spustit aplikaci **Ovládací panely** a vybrat zařízení. V zobrazeném seznamu lze nalézt běžící ovladač **PortTalk** (viz obr. 4.2).

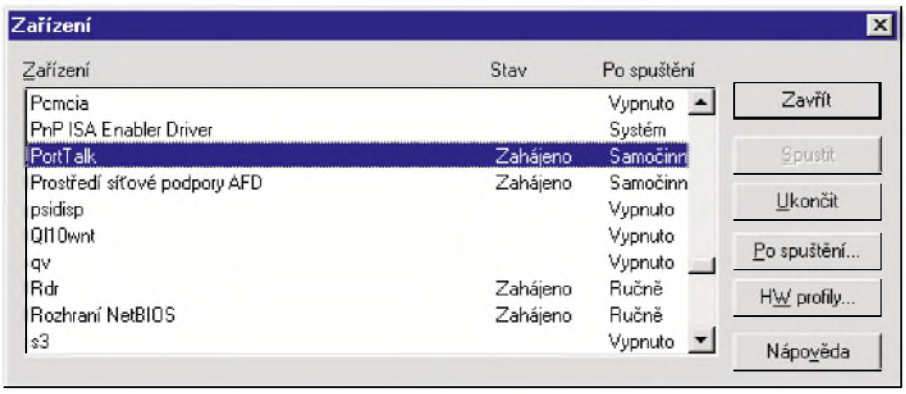
Pro zjednodušení a také proto, abychom zabránili konkurenčnímu přístupu více programů na stejný port, byla vytvořena dynamická knihovna **DRIVER.DLL**, která využívá služeb poskytovaných ovladačem **PortTalk**.

Stručný popis metod třídy TPort

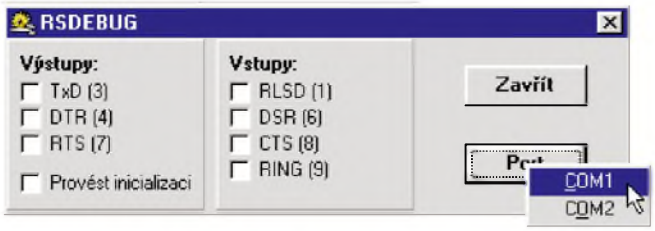
Pro použití knihovny **DRIVER.DLL** resp. třídy **TPort** je třeba podat krátký popis jejího používání:

- **__fastcall TPort(unsigned short MinPortId, unsigned short MaxPortId)** - konstruktor, žadá o porty v rozsahu **MinPortId** až **MaxPortId**. Volání konstruktoru selže (vyvolá se výjimka) v těchto případech:
  - **MaxPortId** < **MinPortId** (ale může být **MinPortId** rovno **MaxPortId**; pak žádáme o jediný port),
  - nelze alokovat systémové zdroje,
  - nelze otevřít ovladač **PORTTALK.SYS**.
- **void __fastcall OutPort(unsigned short PortId, BYTE Value)** - zapíše hodnotu **Value** na port **PortId**. Volání této metody selže (vyvolá se výjimka) v těchto případech:
  - **PortId** nespadá do intervalu zadaného při volání konstruktoru,
  - porty jsou odpojeny předchozím voláním metody **Close**.
- **BYTE __fastcall InPort(unsigned short PortId)** - čte stav portu určeného adresou **PortId**. Volání této metody selže (vyvolá se výjimka) v těchto případech:
  - **PortId** nespadá do intervalu zadaného při volání konstruktoru,
  - porty jsou odpojeny předchozím voláním metody **Close**.
- **void __fastcall Close()** - odpojí porty z knihovny DRIVER.DLL, dále nejsou použitelné. Volejte tuto metodu před destrukcí dané instance, jinak zůstanou porty alokovány.

Další infromace o ovladači PortTalk.sys a knihovně Driver.dll naleznete v [5].



Obr. 4.2. Seznam zařízení



Obr. 4.3. Volba portu pro přímé řízení

Třída TSerial - Použití asynchronního přenosu dat sériovým kanálem

Protože často ovládám zařízení připojená přes sériový kanál, rozhodl jsem se vytvořit speciální třídu, která zjednoduší zápisy operací. Uživatel pak ani neví, že používá funkce Windows API. Třída **TSerial** byla poprvé publikována v [2], na tomto místě uvádím pouze krátký popis zaměřený hlavně na nové vlastnosti.

Většina nastavení parametrů sériového kanálu je řízena pomocí vlastností (R/O - pouze pro čtení, W/O - pouze pro zápis):

- **TBaudRate BaudRate** baudová rychlost přenosu, možné hodnoty jsou určeny výčtovým typem **TBaudRate** (od 110 do 256 000 Bd),
- **TParity Parity** parita přenosu, možné hodnoty jsou určeny výčtovým typem **TParity** (pOdd - lichá, pEven - sudá, pMark - značená, pNo - žádná),
- **TStopBits StopBits** počet stop-bitů, možné hodnoty jsou určeny výčtovým typem **TStopBits** (sb10 - 1 stop-bit, sb15 - 1,5 stop-bitu, sb20 - 2 stop-bity),
- **TByteSize ByteSize** délka bajtu, možné hodnoty jsou určeny výčtovým typem **TByteSize** (od 4 do 8 bitů v bajtu),
- **bool CTS** (R/O) stav linky CTS (Clear To Send),
- **bool DSR** (R/O) stav linky DSR (Data Set Ready),
- **bool RING** (R/O) stav linky RI (Ring Indicator),
- **bool RLSD** (R/O) stav linky RLSD (Receive Line Signal Detect),
- **bool DTR** (W/O) nastaví linku DTR (novinka),
- **bool RTS** (W/O) nastaví linku RTS (novinka),
- **DWORD InputQueue** (R/O) délka vstupního bufferu v bajtech,
- **DWORD OutputQueue** (R/O) délka výstupního bufferu v bajtech,
- **DWORD ReadIntervalTimeout**, **ReadTotalTimeoutMultiplier**, **ReadTotalTimeoutConstant**, **WriteTotalTimeoutMultiplier WriteTotalTimeoutConstant** řízení time-outu při čtení a zápisu (podrobný popis viz [2], [6]).

Metody ovládají založení kanálu a vysílání resp. přijímání znaků:

- **__fastcall TSerial(int Number)**; konstruktor; **Number** určuje pořadové číslo kanálu, se kterým chceme pracovat (např.: Number = 2 pro COM2),
- **int __fastcall WriteByte(Byte byte)**; zapíše jeden bajt **byte** do sériového kanálu. Vrací počet bajtů, které se zapsaly (1 - značí úspěch),
- **int __fastcall ReadByte(Byte* byte)**; načte jeden bajt ze sériového kanálu do **byte**. Vrací počet přečtených bajtů (1 - značí úspěch),
- **void __fastcall PurgeInput()**; vyprázdní vstupní buffer sériového kanálu (dříve přijaté znaky nebudou přečteny voláním **ReadString** nebo **ReadChar**),
- **void __fastcall SetupComm(DWORD InQueue, DWORD OutQueue)**; nastaví velikosti vstupního (InQueue) a výstupního (OutQueue) bufferu sériového kanálu v bajtech.

Třída TSPort - Přímé řízení sériového portu

Pro účely přímého řízení sériového portu byla vytvořena třída **TSPort**, která byla publikována již v [2].

Na tomto místě pouze krátce připomeneme klíčové metody a vlastnosti (R/O - pouze pro čtení, W/O - pouze pro zápis).

Metody:

- **__fastcall TSPort(int Number)**; konstruktor. **Number** udává pořadové číslo sériového kanálu, se kterým chceme pracovat (například pro COM2 zadáme Number = 2),
- **__fastcall TSPort()**; konstruktor. Najde první volný sériový kanál. Jeho handle a pořadové číslo jsou dostupné v **Handle** a **Number**,
- **__fastcall ~TSPort()**; destruktor, zavře sériový kanál.

Vlastnosti:

- **HANDLE Handle** (R/O) handle otevřeného sériového kanálu, použijte pro přímé volání funkcí Win API,
- **int Number** (R/O) pořadové číslo sériového kanálu (například pro COM2 je Number = 2),
- **bool CTS** (R/O) čtení stavu linky CTS,
- **bool DSR** (R/O) čtení stavu linky DSR,
- **bool RING** (R/O) čtení stavu linky RING (RI),
- **bool RLSD** (R/O) čtení stavu linky RLSD (DCD),
- **bool DTR** (W/O) zápis stavu linky DTR,
- **bool RTS** (W/O) zápis stavu linky RTS,
- **bool TxD** (W/O) zápis stavu linky TxD.

RSDEBUG - Ladicí program pro přímé řízení sériového portu

Pro první testy přímého řízení sériového portu („tahání za drátky“) byl vytvořen program RSDEBUG (obr. 4.3).

Stiskem tlačítka **Port** se zobrazí seznam dostupných portů, vybereme ten, který chceme ovládat. Jeho název se pak objeví na tlačítku místo výchozího titulku Port. Potom již jen nastavujeme výstupní linky v panelu **Výstupy** a sleduje vstupní linky v panelu **Vstupy**.

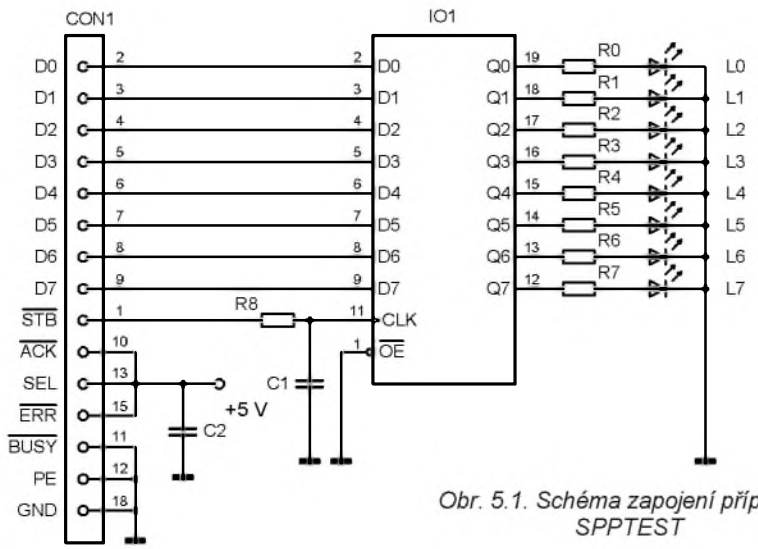
Je-li políčko **Provést inicializaci** zaškrtnuto, nastaví se vybrané hodnoty po výběru portu. V opačném případě, se inicializace podle aktuálních hodnot neprovede.

5. Příklad práce s paralelním portem podle standardu SPP

Přípravek **SPPTEST** předvádí základní práci s paralelním portem podle standardu SPP.

Zapojení přípravku je na obr. 5.1. Datové vodiče jsou připojeny na vstupy osmibitového registru IO1, hodinový signál je připojen na STBnon (zápis se provede náběžnou hranou). Integrační článek *R8*, *C1* odstraňuje rušivé impulzy. Indikační LED jsou připojeny klasicky přes omezovací rezistory mezi jednotlivé výstupy a zem (LED svítí při stavech „log. 1“ na výstupech IO1).

Řídicí vstupy SPP portu jsou napojeny tak, aby zajišťovaly úspěšnou komunikaci.



Obr. 5.1. Schéma zapojení přípravku SPPTEST

Všechny součástky přípravku SPPTEST jsou umístěné na desce s jednostrannými plošnými spoji.

Na obr. 5.2 je obrazec plošných spojů a na obr. 5.3 je rozmístění součástek na desce.

Použité LED jsou červené (R) o průměru 5 mm nejlevnějšího typu v ceně asi 1 Kč za kus. Pro IO1 je vhodné použít objímku, aby jej bylo možné přemístit i do dalších konstrukcí.

Seznam součástek

(cena asi 60 Kč)

R0 až R8	330 Ω	9 ks
C1	220 pF	1 ks
C2	100 nF	1 ks
L0 až L7	LED, R, 5 mm	8 ks
IO1	74HCT574	1 ks
CON1	CAN 25 V 90	1 ks
deska s plošnými spoji SPPTEST		

Pro řízení zápisu z PC na přípravek **SPPTEST** je vytvořen program **SPPTEST.EXE**.

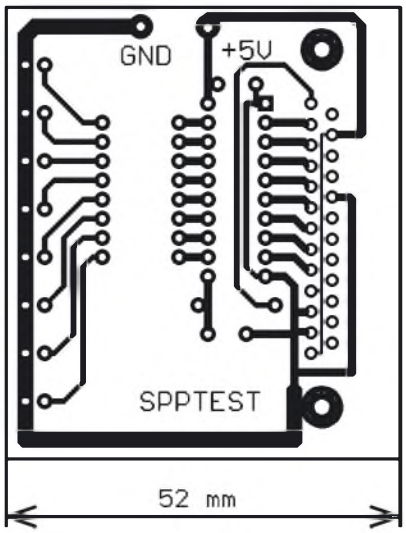
Program SPPTEST.EXE

HLFORM.H:

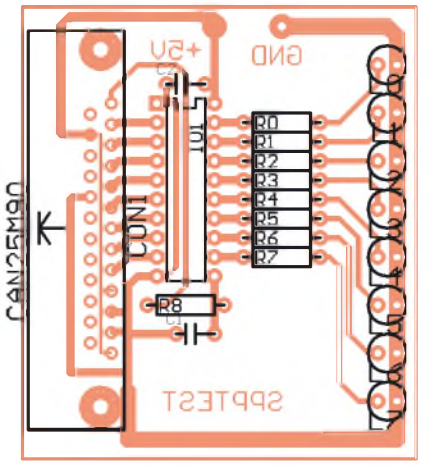
```
//---------------------------------------
#ifndef HlFormH
#define HlFormH
//---------------------------------------
.
.
//---------------------------------------
class TMainForm : public TForm
{
__published:    // IDE-managed Components
.
.
private:        // User declarations
  HANDLE lpt1;  //pro práci s LPT1
public:         // User declarations
  __fastcall TMainForm(TComponent* Owner);
  __fastcall ~TMainForm();
};
//---------------------------------------
extern PACKAGE TMainForm *MainForm;
//---------------------------------------
#endif
```

HLFORM.CPP:

```
#include <vcl.h>
#pragma hdrstop

#include "HlForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TMainForm *MainForm;
//---------------------------------------
__fastcall TMainForm::TMainForm
        (TComponent* Owner)
        : TForm(Owner)
{
  //otevření portu LPT1:
  lpt1=CreateFile(
    "LPT1",
    GENERIC_READ|GENERIC_WRITE,
    0,
    NULL,
    OPEN_EXISTING,
    0,
    NULL);

  //test chyby:
  if(lpt1==INVALID_HANDLE_VALUE)
    throw Exception("LPT1 není
        k dispozici");
}
//---------------------------------------
__fastcall TMainForm::~TMainForm()
{
  //zavření portu:
  CloseHandle(lpt1);
}
//---------------------------------------
void __fastcall TMainForm::VystupyClick
        (TObject *Sender)
{
  //zápis dat na port:
  DWORD d=0;

  //sestavení dat:
  Byte Data=
    128*OUT7->Checked
    +64*OUT6->Checked
    +32*OUT5->Checked
    +16*OUT4->Checked
    +8*OUT3->Checked
    +4*OUT2->Checked
    +2*OUT1->Checked
    +OUT0->Checked;

  //zápis:
  WriteFile(lpt1,&Data,1,&d,NULL);

  //test chyby:
  if(d!=1)
    MessageBox(Handle,
      "Zápis selhal",
      "SPPTEST",
      MB_ICONHAND);
}
```



Obr. 5.2. Obrazec plošných spojů přípravku SPPTEST (měř.: 1 : 1)



Obr. 5.3. Rozmístění součástek na desce přípravku SPPTEST

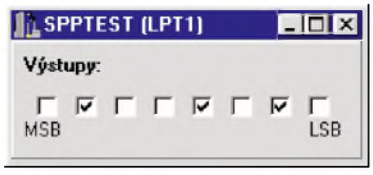
Volání funkce **CreateFile** zjišťuje, zda je port LPT1 dostupný. Pokud např. současně probíhá tisk na tiskárně připojené k portu LPT1 nebo pokud tuto aplikaci spustíte dvakrát, nebude port dostupný. Operační systém tímto způsoben brání možným kolizím. Test úspěšného otevření portu je proveden testování získaného handle **lpt1**.

Zavření portu zajistí funkce **CloseHandle**. Port se poté stává dostupný dalším aplikacím.

Zápis na port je proveden funkcí **WriteFile**.

Okno, které se objeví na obrazovce monitoru PC při běhu programu SPPTEST.EXE je na obr. 5.4.

Fotografie přípravku SPPTEST je na obálce tohoto časopisu.



Obr. 5.4. Ovládací program SPPTEST.EXE v akci

6. Příklad práce s paralelním portem dle standardu EPP

Přípravek **EPPTEST** umožňuje obousměrnou komunikaci s paralelním portem PC podle standardu EPP.

Schéma přípravku **EPPTEST** je na obr. 6.1.

Vstup do PC je realizován obvodem IO3. Tento obvod (74HCT245) pracuje jako obousměrný budič sběrnice. Zde je použit jako jednosměrný budič, který umožňuje číst stav vstupů D0 až D7 přes EPP port ve vstupním režimu.

Obvod IO3 pracuje jako oddělovač, který se aktivuje pouze v případě, že je WRITEnon = 1 (čímž je indikováno čtení). Protože je ovládací vstup Gnon IO3 aktivní ve stavu „log. 0", musí se signál WRITEnon před přivedením na vstup Gnon IO3 negovat (k negaci je použito jedno z hradel obsažených v obvodu IO1). Možné zákmity odstraňuje integrační článek *R9*, *C2*.

Výstup z PC je realizován obvodem IO2 (jedná se o podobné zapojení jako u přípravku **SPPTEST**). Zápis do registru (74HCT574) je odvozen od signálů WRITEnon a DATASTBnon. Z předchozího popisu vyplývá, že k zápisu dojde, když je WRITEnon = 0 a současně DATASTBnon = 0. Hodinový signál registru (CLK) je sestaven jako logický součet obou signálů (nejdříve se vytvoří negovaný logický součet, který se pak zneguje). Rušivé impulsy jsou filtrovány integračním článkem *R8*, *C1*.

Obvod IO1 je čtveřice hradel NOR (74HCT02). Jsou použita pro vytvoření aktivačních signálů pro obvody IO2 a IO3. Operace NOR se mi jevila jako nejvýhodnější (vystačil jsem s jediným obvodem).

Kondenzátory C3 a C4 blokují napájecí napětí všech tří IO.

Všechny součástky přípravku EPPTEST jsou umístěné na desce s jednostrannými plošnými spoji.

Na obr. 6.2 je obrazec plošných spojů a na obr. 6.3 je rozmístění součástek na desce.

Použité LED jsou červené o průměru 5 mm nejlevnějšího typu v ceně asi 1 Kč za kus. Pro všechny IO je vhodné použít objímky, aby je bylo možné přemístit i do dalších konstrukcí.

Seznam součástek

(cena asi 90 Kč)

R0 až R9	330 Ω	10 ks
C1, C2	220 pF	2 ks
C3, C4	100 nF	2 ks
L0 až L7	LED, R, 5 mm	8 ks
IO1	74HCT02 (74LS02)	1 ks
IO2	74HCT574 (74LS574)	1 ks
IO3	74HCT245 (74LS245)	1 ks
CON1	CAN 25 V 90	1 ks
deska s plošnými spoji EPPTEST		

Při přímém přístupu je nutné kromě názvu portu znát jeho adresu. Tyto údaje jsou sice uloženy v operačním systému (v proměnných BIOSu), ale přístup do této části paměti je chráněn.

Proto jsem se rozhodl použít inicializační soubor, který obsahuje jméno portu, jeho bázovou adresu a periodu, se kterou se snímá stav vstupů. Soubor jsem pojmenoval **EPPTEST.INI**. V sekci **Port** jsou uvedeny položky udávající jméno portu (**Jmeno**) a jeho adresu (**Adresa**). V sekci **TIMER** je uveden klíč **Interval**, který definuje interval mezi dvěma čtecími operacemi v milisekundách.

EPPTEST.INI:

```
[Port]
Jmeno="LPT1"
Adresa=0x378

[TIMER]
Interval=100
```

Pro přímý přístup k portům je třeba použít knihovnu **DRIVER.DLL**. Proto se do projektu musí připojit importní knihovna **DRIVER.LIB** (položkou menu **Project|Add To Project**) a do zdrojového souboru vložit hlavičkový soubor **PORT.H**.

Připomeňme, že knihovna **DRIVER.DLL** se musí nacházet buď v systémovém adresáři nebo v adresáři aplikace.

Pro řízení přenosu dat mezi PC a přípravkem **EPPTEST** je vytvořen program **EPPTEST.EXE**.

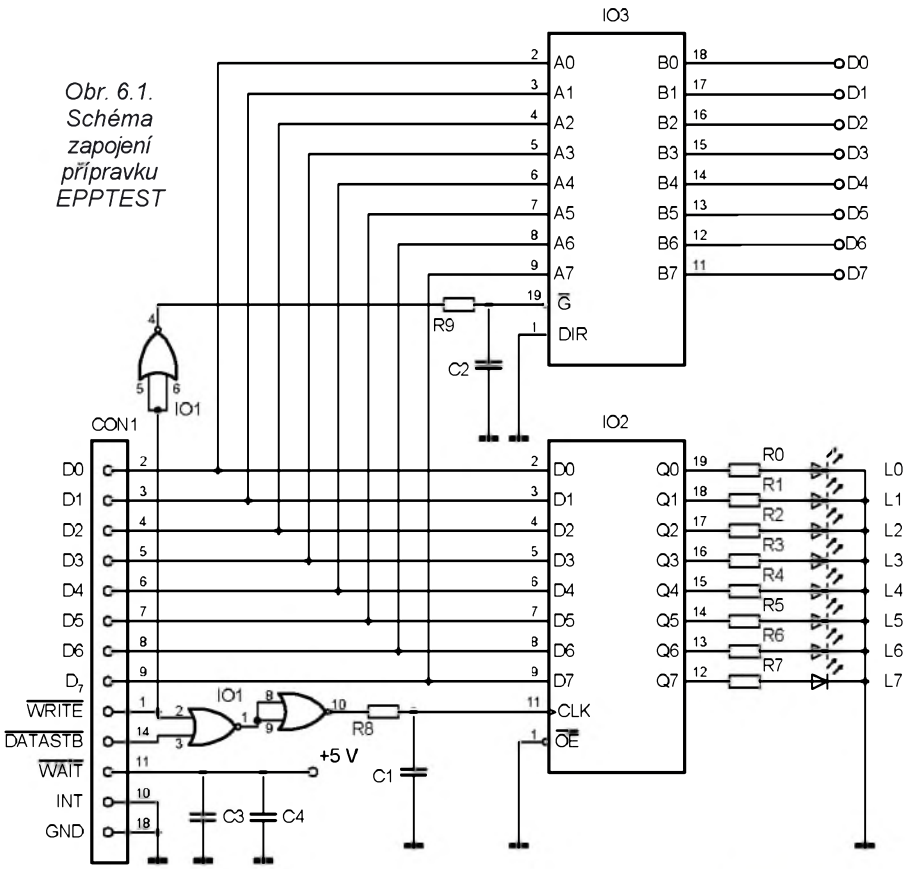
Program EPPTEST.EXE

HLFORM.H:

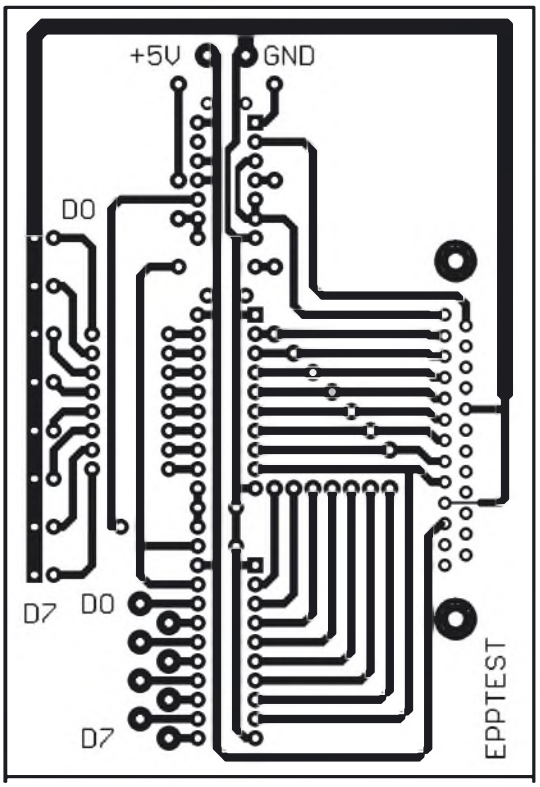
```
//---------------------------------------
#ifndef HlFormH
#define HlFormH
//---------------------------------------
#include <Classes.hpp>
.
.
//---------------------------------------
#include "Port.h"        //přímý přístup
        //na port
//---------------------------------------
class TMainForm : public TForm
{
__published:    // IDE-managed Components
.
.
private:        // User declarations
  HANDLE hPort; //handle LPT
  TPort *Port;  //přímý přístup na port
  int Baze;     //bázová adresa portu
public:         // User declarations
  __fastcall TMainForm(TComponent* Owner);
  __fastcall ~TMainForm();
};
//---------------------------------------
extern PACKAGE TMainForm *MainForm;
//---------------------------------------
#endif
```

HLFORM.CPP:

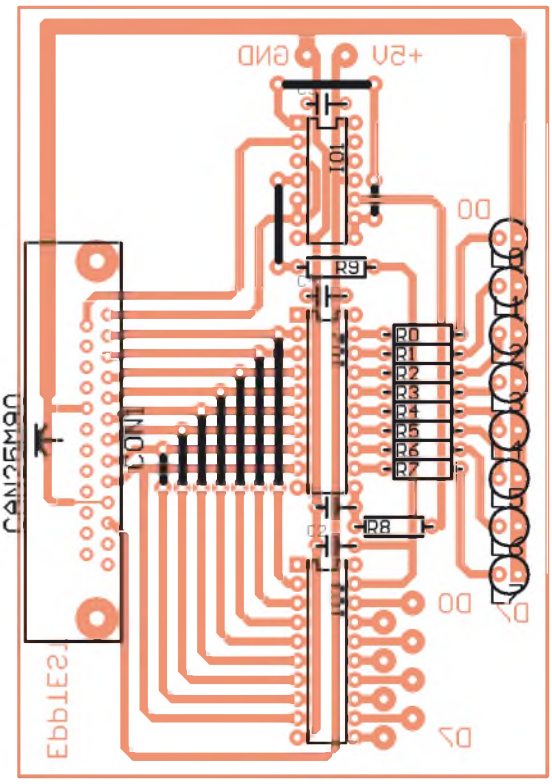
```
//---------------------------------------
#include <vcl.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "HlForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TMainForm *MainForm;
//---------------------------------------
```



Obr. 6.1. Schéma zapojení přípravku EPPTEST



Obr. 6.2. Obrazec plošných spojů přípravku EPPTEST (měř.: 1 : 1, kratší rozměr desky je 70 mm)



Obr. 6.3. Rozmístění součástek na desce přípravku EPPTEST

```
__fastcall TMainForm::TMainForm
        (TComponent* Owner)
        : TForm(Owner)
{
  //nastavení parametrů z EPPTEST.INI:
  TIniFile *ini=new TIniFile
        (GetCurrentDir()
    +"\\EPPTEST.INI");
  AnsiString Jmeno=ini->ReadString("PORT",
    "Jmeno","LPT1");
  int Adresa=ini->ReadInteger("PORT",
    "Adresa",0x378);
  int Interval=ini->ReadInteger("TIMER",
    "Interval",100);
  if(Interval<=0)
    Interval=100;
  delete ini;

  //osvojení LPT:
  hPort=CreateFile(
    Jmeno.c_str(),
    GENERIC_READ|GENERIC_WRITE,
    0,
    NULL,
    OPEN_EXISTING,
    0,
    NULL);

  //test neúspěchu:
  if(hPort==INVALID_HANDLE_VALUE)
    throw Exception(Jmeno+
        "není k dispozici");
  Baze=Adresa;  //bázová adresa
  Casovac->Interval=Interval;  //perioda
        časovače
  Port=new TPort(Baze+4,Baze+4);
        //otevření portu

  //vypsání parametrů: jméno portu, adresa,
        //perioda:
  Caption=AnsiString("EPPTEST (")
    +Jmeno
    +AnsiString(", 0x")
    +IntToHex(Baze,3)
    +AnsiString(", ")
    +Interval
    +AnsiString("ms)");
}
//---------------------------------------
__fastcall TMainForm::~TMainForm()
{
  //odevzdá port systému:
  CloseHandle(hPort);
  //zavře přímý přístup:
  Port->Close();
  delete Port;
}
//---------------------------------------
void __fastcall
        TMainForm::VystupyClick
        (TObject *Sender)
{
  //reakce na změnu výstupu:

  //sestavení dat pro odeslání:
  Byte Data=
    128*OUT7->Checked
    +64*OUT6->Checked
    +32*OUT5->Checked
    +16*OUT4->Checked
    +8*OUT3->Checked
    +4*OUT2->Checked
    +2*OUT1->Checked
    +OUT0->Checked;

  //odeslání dat:
  Port->OutPort(Baze+4,Data);
}
//---------------------------------------
void __fastcall
        TMainForm::AktivaceCasovace
        (TObject *Sender)
{
  //přetečení časovače-aktualizace vstupů:

  //čtení dat:
  Byte Data=Port->InPort(Baze+4);

  //sestavení informace pro zobrazení:
  IN7->Checked=Data&0x80;
  IN6->Checked=Data&0x40;
  IN5->Checked=Data&0x20;
  IN4->Checked=Data&0x10;
  IN3->Checked=Data&0x08;
  IN2->Checked=Data&0x04;
  IN1->Checked=Data&0x02;
  IN0->Checked=Data&0x01;
}
```

Program nejdříve získá výsadní přístup k portu funkcí **CreateFile** (pokud je již spuštěna jiná aplikace, která tento port používá, dojde k chybě). Poté se pokusí získat přímý přístup k portu Báze+4. I v tomto případě je testováno, zda jiná aplikace nemá takový přístup.

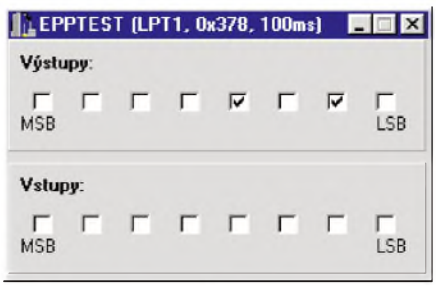
Vlastní čtení a zápis na port je jednoduché. Po změně stavu políček v panelu **Výstupy** se vyvolá událost **VystupyClick**, složí se bajt pro vyslání a volá se metoda **TPort::OutPort**, která zápis provede.

Čtení je realizováno časovačem. Nastavený interval v milisekundách určuje, jak často se vstup čte. Po přetečení časovače, který odměření intervalu zajišťuje, je volána metoda **TPort::InPort**. Získaná data se rozloží na jednotlivé bity a podle nich se aktualizuje stav políček v panelu **Vstupy**.

Konec aplikace představuje odevzdání portu knihovně **DRIVER.DLL** a operačnímu systému (**CloseHandle**).

Okno, které se objeví na obrazovce monitoru PC při běhu programu EPPTEST.EXE je na obr. 6.4.

Fotografie přípravku EPPTEST je na obálce tohoto časopisu.



Obr. 6.4. Ovládací program EPPTEST.EXE v akci

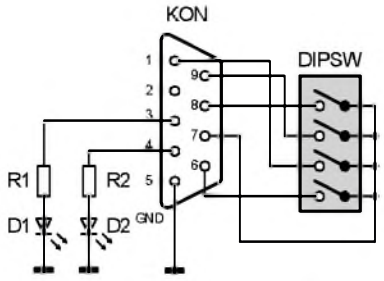
7. COMTEST - zkouška přímého řízení sériového portu

Úvodní seznámení s možnostmi přímého řízení sériového portu provedeme na přípravku COMTEST.

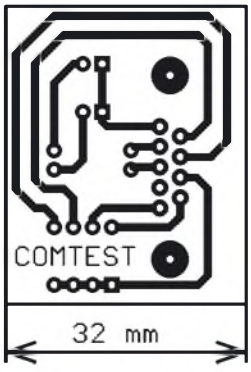
Schéma přípravku COMTEST je na obr. 7.1.

LED D1 a D2 jsou přes omezovací rezistory R1 a R2 připojeny na linky **TxD** (*3*) a **DTR** (*4*). LED svítí, přivedeme-li na příslušnou linku napětí +12 V (programově to řešíme zápisem **1** do vlastností TxD nebo DTR instance třídy **TSPort**).

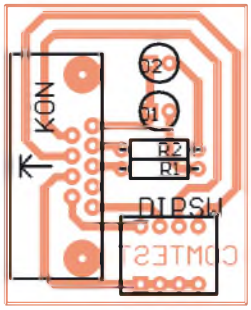
Dále je na přípravku čtyřnásobný spínač DIP, který má vývody připojené na linky CTS (*8*), RING (*9*), RLSD (*1*) a DSR (*6*). Druhé konce jsou spojeny a přivedeny na vývod RTS (*7*). Stavy spínačů se pak testují tak, že na vývod RTS nejdříve přivedeme napětí +12 V (RTS = 1) a sejmeme stav jednotlivých vstupů. Potom na vývod DTR přivedeme -12 V (RTS = 0) a sejmeme stav vstupů znovu. Linka, jejíž stav je 1 při RTS = 1 a 0 při RTS = 0 odpovídá sepnutému spínači. Linka, jejíž stav se při změně RTS nemění, odpovídá rozpojenému spínači.



Obr. 7.1. Schéma zapojení přípravku COMTEST



Obr. 7.2. Obrazec plošných spojů přípravku COMTEST (měř.: 1 : 1)



Obr. 7.3. Rozmístění součástek na desce přípravku COMTEST

Takové zapojení spínačů je vlastně jediné možné, protože jinak by spínače musely být buzeny z vnějšího zdroje, což by nebylo příliš výhodné. Tento nápad jsem později našel v [7]. Napadlo mě tedy to samé, jako autora této skvělé knihy.

Všechny součástky přípravku COMTEST jsou umístěné na desce s jednostrannými plošnými spoji.

Na obr. 7.2 je obrazec plošných spojů a na obr. 7.3 je rozmístění součástek na desce.

Použité LED jsou červené o průměru 5 mm nejlevnějšího typu v ceně asi 1 Kč za kus.

Fotografie přípravku COMTEST je na obálce tohoto časopisu.

Seznam součástek

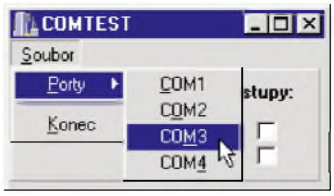
(cena asi 40 Kč)

R1, R2	1 kΩ	2 ks
D1, D2	LED, R, 5 mm	2 ks
KON	CAN 9 Z 90	1 ks
DIPSW	DIP 4x	1 ks
deska s plošnými spoji COMTEST		

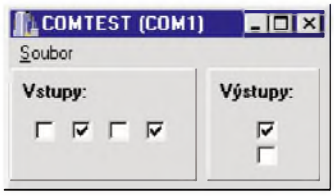
Ovládací program pro přípravek **COMTEST** je poměrně jednoduchý, ale přesto ukazuje mnoho užitečných zápisů pro tvorbu dalších programů.

Především se jedná o generování položek menu pro **Porty**. Zjištění aktuálně dostupných portů je provedeno voláním funkce **CreateFile**, jména souborů jsou COM1 až COM256. Funkce **CreateFile** vrací pro případ dostupného portu platný handle tohoto portu. Pro případ, že port není k dispozici (neexistuje nebo je již použit jinou aplikací), vrací hodnotu **INVALID_HANDLE_VALUE**.

Takto otevřený port je po zápisu odpovídající položky do menu nutno vrátit systému voláním funkce **CloseHandle**. Jinak by selhal pokus o vytvoření instance třídy **TSPort** (port by si zabrala sama aplikace). Výběr portů ilustruje obr. 7.4.



Obr. 7.4. Aplikace zobrazuje dostupné porty



Obr. 7.5. Aplikace v akci

Aplikace obsahuje dvě skupiny (viz obr. 7.5):

První skupina (Vstupy:) sleduje stavy vstupů postupem, který byl popsán výše (změna RTS a sledování odezvy). Vzorkování je prováděno časovačem, který má periodu 100 ms.

Druhá skupina (Výstupy:) odpovídá výstupům, ovládá tedy obě LED.

Pro řízení přenosu dat mezi PC a přípravkem **COMTEST** je vytvořen program **COMTEST.EXE**.

Program COMTEST.EXE

HLFORM.H:

```
#ifndef HlFormH
#define HlFormH
...
...
class TForm1 : public TForm
{
...
...
private:        // User declarations
  TSPort *Port; //ukazatel pro práci
        //s TSPort
public:         // User declarations
  __fastcall TForm1(TComponent* Owner);
  __fastcall ~TForm1();
};
...
...
#endif
```

HLFORM.CPP:

```
#include <vcl.h>
#include <stdio.h>
#pragma hdrstop
#include "HlForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------
__fastcall TForm1::TForm1
        (TComponent* Owner)
        : TForm(Owner)
{
  Port=NULL;    //instance zatím
        //nevytvořena
  Caption="COMTEST";
}
//---------------------------------------
__fastcall TForm1::~TForm1()
{
  if(Port)      //uvolni, pokud byla
        //založena instance
    delete Port;
}
//---------------------------------------
void __fastcall TForm1::AktivaceCasovace
        (TObject *Sender)
{
  //aktualizace skupiny Vstupy:
  if(Port){
    bool Bity[2][4];
    //pole s políčky k zatržení:
    TCheckBox *Policka[4]=
        {cbDSR,cbRLSD,cbRING,cbCTS};

    //RTS=0, sejmuti stavu:
    Port->RTS=0;
    Bity[0][0]=!Port->DSR;
    Bity[0][1]=!Port->RLSD;
    Bity[0][2]=!Port->RING;
    Bity[0][3]=!Port->CTS;
    Sleep(10);  //počká 10 ms

    //RTS=1, sejmuti stavu:
    Port->RTS=1;
    Bity[1][0]=Port->DSR;
    Bity[1][1]=Port->RLSD;
    Bity[1][2]=Port->RING;
    Bity[1][3]=Port->CTS;
```

```
    //vyhodnocení:
    for(int i=0;i<4;i++)
      Policka[i]->Checked=
        Bity[0][i]&Bity[1][i];
  }
}
//---------------------------------------
void __fastcall TForm1::cbDTRClick
       (TObject *Sender)
{
  //ovládá D2:
  if(Port)
    Port->DTR=cbDTR->Checked;
  else
    MessageBox(Handle,"Vyberte
        port!","COMTEST",MB_ICONHAND);
}
//---------------------------------------
void __fastcall TForm1::cbTxDClick(TObject
*Sender)
{
  //ovládá D1:
  if(Port)
    Port->TxD=cbTxD->Checked;
  else
    MessageBox(Handle,"Vyberte
        port!","COMTEST",MB_ICONHAND);
}
//---------------------------------------
void __fastcall TForm1::VyberPortu(TObject
*Sender)
{
  //výběr portu přes menu:
  TMenuItem *mi=dynamic_cast
        <TMenuItem*>(Sender);

  //zruší stávají port:
  if(Port){
    delete Port;
    Port=NULL;
  }

  //vybere nový:
  mi->Checked=true;
  Port=new TSPort(mi->Tag);     //v Tag je
        //číslo portu
  Caption=AnsiString("COMTEST (COM")
    +mi->Tag+AnsiString(")");
}
//---------------------------------------
void __fastcall TForm1::Soubor1Click
       (TObject *Sender)
{
  //zobrazí dostupné porty:
  HANDLE h;
  char p[7];
  TMenuItem *mi;

  //smaže stávající:
  Porty->Clear();

  //přidá nové:
  for(int i=1;i<=256;i++){
    sprintf(p,"COM%i",i);
    h=CreateFile(p,
        GENERIC_READ|GENERIC_WRITE,
        0,NULL,OPEN_EXISTING,0,NULL);
    if(h!=INVALID_HANDLE_VALUE){
      //port existuje-přidáme do menu:
      mi=new TMenuItem(Porty);
      Porty->Add(mi);
      mi->Caption=p;
      mi->Tag=i;        //ulož číslo portu
      mi->OnClick=VyberPortu;
    }
    CloseHandle(h);     //zavři port
  }
}
//---------------------------------------
void __fastcall TForm1::Konec1Click
       (TObject *Sender)
{
  Close();
}
```

8. ADC8DIR - levný A/D převodník k počítači

Při měření napětí pomocí počítače PC potřebujeme A/D převodník, který měřené napětí převede do číslicové formy. Dále je popsána konstrukce takového A/D převodníku s použitím obvodu TLC549.

Stručný popis obvodu TLC549

Obvod TLC549 od firmy Texas Instruments (v ceně asi 65 Kč) pracuje jako osmibitový A/D převodník se sériovým výstupem, který je řízen sběrnicí MicroWire. Existuje také podobný obvod ADC0831, který je však poněkud dražší.

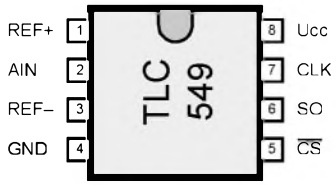
Zapojení vývodů obvodu TLC549 v pouzdru DIP 8 je na obr. 8.1.

Ucc je napájecí napětí (3 až 6 V).

GND je signálová zem.

AIN je analogový vstup (vstupní napětí smí být v rozmezí REF+ až REF-).

REF+ a **REF-** jsou referenční napětí (REF+ musí být do +2,5 V; REF- je obvykle uzemněn, max. +2,5 V).



Obr. 8.1. Zapojení vývodů obvodu TLC549 v pouzdru DIP 8

CLK, **SO**, a **CSnon** jsou signály sběrnice MicroWire.

Časování obvodu TLC549 je na obr. 8.2. Při CSnon = 1 probíhá v obvodu A/D převod, při CSnon = 0 je možné přečíst výsledek převodu.

Při měření napětí pomocí obvodu TLC549 postupujeme takto:

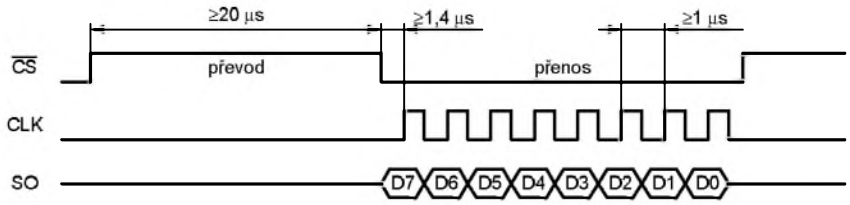
- Nejdříve je nutno po dobu alespoň 20 µs (po dobu převodu) uvést CSnon do stavu „log. 1".
- Potom aktivujeme přenos dat uvedením CSnon do stavu „log. 0". Na výstupu SO se objeví nejvíce významný bit (D7) převedeného napětí.
- Hodinovými impulsy na vstupu CLK čteme další bity (významově nižší).

Přípravek ADC8DIR

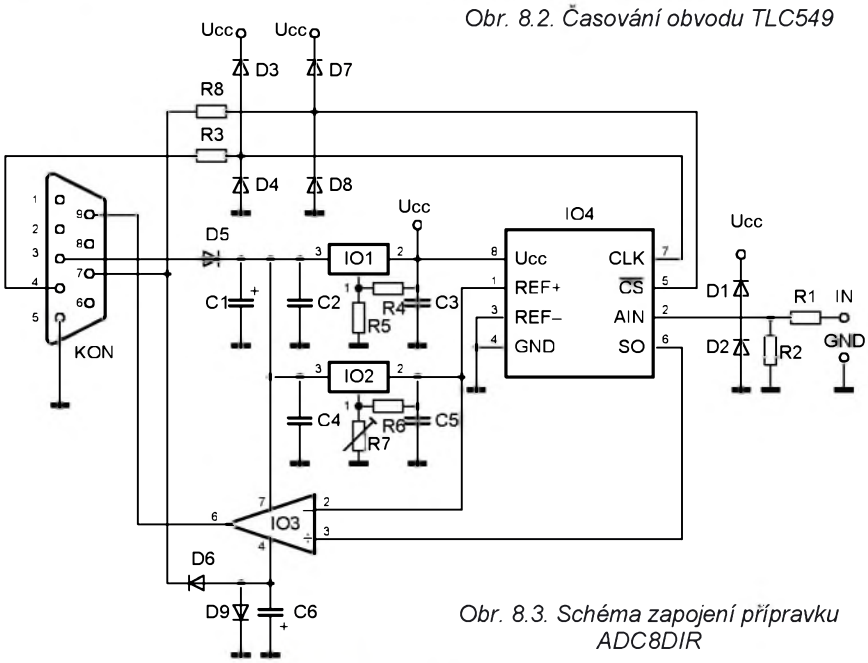
Aby se mohl obvod T**LC549** připojit k sériovému portu, je třeba přidat několik vnějších součástek. Tak vznikl přípravek **ADC8DIR**.

Schéma přípravku ADC8DIR je na obr. 8.3.

Přípravek je napájen přímo ze sériového portu, takže nepotřebujeme vnější zdroj. Napájecí napětí (přibližně +5 V) pro A/D převodník T**LC549** (IO4) je získáno z linky TxD pomocí diody D5 (nelze totiž zabránit přepólování), vyhlazovacího kondenzátoru C1 a stabilizátoru IO1 (byl použit typ **LM317L**, který má poměrně malý odběr).



Obr. 8.2. Časování obvodu TLC549



Obr. 8.3. Schéma zapojení přípravku ADC8DIR

Z linky TxD je také odvozeno **referenční** napětí pro A/D převodník. Jako zdroj referenčního napětí je opět použit stabilizátor **LM317L** (IO2), jeho výstupní napětí je však nastavitelné trimrem R7. Referenční napětí by mělo být **+2,5 V** (změříme je mezi vývody *1* a *4* IO4).

Řízení A/D převodníku (signály CSnon a CLK) je zajištěno linkami DTR (ovládá CLK) a RTS (ovládá CSnon). Převod z úrovní RS-232 C na TTL je uskutečněn pomocí rezistorů R3 a R8 a diod D3, D4, D7 a D8. Někdy se místo páru obyčejných diod používá jedna Zenerova dioda, takové řešení bylo často používáno v [7]. Já jsem od této možnosti ustoupil, protože Zenerovy diody mají podstatně větší parazitní kapacitu přechodu a tak pro zajištění dostatečně strmých výstupních signálů vyžadují použít rezistory s malým odporem. To jsem si nemohl dovolit, protože linka RTS je také použita pro získání záporného napájecího napětí pro převodník, který převádí úroveň TTL na RS-232 C.

Tento převodník úrovně je tvořen operačním zesilovačem **TL061** (IO3) s malým příkonem, který je zapojen jako komparátor. Na neinvertující vstup IO3 je přivedeno referenční napětí A/D převodníku (2,5 V) a na invertující vstup signál SO (digitální výstup převodníku). Je-li SO = 0, je na lince RING napětí zhruba +12 V. Je-li SO = 1, je na lince RING napětí zhruba -12 V.

Záporné napětí se sbírá z linky RTS přes diodu D6 a vyhlazuje se kondenzátorem C6. Dioda D9 chrání vstupy operačního zesilovače IO3 v okamžiku, kdy je RTS = 1 (na lince RTS je kladné napětí, kondenzátor C6 se po čase vybije a záporný napájecí vývod IO3 by „visel ve vzduchu").

Měřicí rozsah A/D převodníku je definován velikostí referenčního napětí na vstupu REF+ a také poměrem odporů rezistorů R1 a R2. Pro uvažované hodnoty součástek (*R1* = *R2* = 200 kΩ) je měřicí rozsah 0 až 5 V. Diody D1 a D2 pracují jako omezovače vstupního napětí a brání poškození převodníku příliš velkým (nebo záporným) vstupním napětím.

Všechny součástky přípravku ADC8-DIR jsou umístěné na desce s jednostrannými plošnými spoji.

Na obr. 8.4 je obrazec plošných spojů a na obr. 8.5 je rozmístění součástek na desce.

Pro IO3 a IO4 je vhodné použít objímky, aby je bylo možné přemístit i do dalších konstrukcí.

Fotografie přípravku ADC8DIR je na obálce tohoto časopisu.

Seznam součástek

(cena asi 100 Kč)

R1, R2	200 kΩ	2 ks
R3, R8	10 kΩ	2 ks
R4, R6	1,2 kΩ	2 ks
R5	3,9 kΩ	1 ks
R7	5 kΩ, trimr PT10H	1 ks
C1, C6	100 µF/16 V	2 ks
C2 až C4	100 nF	4 ks
D1 až D9	1N4148	9 ks
IO1, IO2	LM317L	2 ks
IO3	TL061	1 ks
IO4	TLC549	1 ks
KON	CAN 9 Z 90	1 ks
deska s plošnými spoji ADC8DIR		

Pro ovládání přípravku **ADC8DIR** byla vytvořena aplikace **DIRADC8**. Ke konfiguraci se používá inicializační soubor **DIRADC8.INI**, který definuje číslo použitého sériového portu a periodu měření v milisekundách. Jedná se o klíče **Port** a **Interval** (viz níže).

DIRADC8.INI:

```
[PORT]
Port=1
[TIMER]
Interval=55
```

Měření probíhá přesně podle časového diagramu na obr. 8.2 (převody A/D jsou spouštěny pomocí časovače, periodu převodů udává výše uvedený klíč Interval).

Linka TxD je trvale v úrovni „log. 1", protože jsou z ní napájeny všechny integrované obvody.

Dále vytvoříme dostatečně dlouhý impuls na vývodu CSnon (připomeňme, že tento vývod je ovládán linkou RTS). Nejprve je CSnon ve stavu „log. 1", po přechodu do stavu „log. 0" je převodník připraven vysílat data. Vzhledem k tomu, že z linky RTS se zároveň získává záporné napájecí napětí pro komparátor IO3, je vše připraveno na přijetí dat.

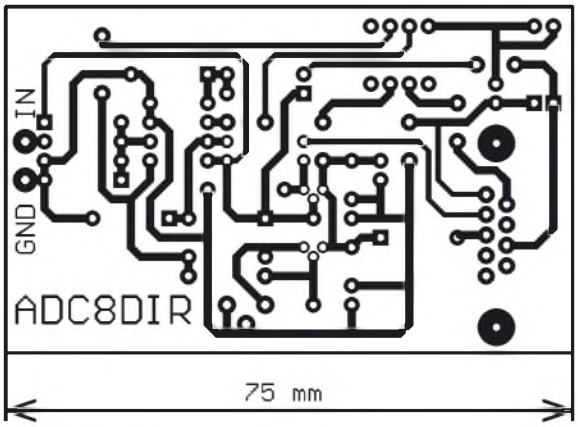
Nyní se musí vytvářet impulsy na řídicím vstupu CLK (je ovládán linkou DTR) a číst přijaté bity z linky RING (data z výstupu SO pouze procházejí komparátorem IO3, který převádí jejich úrovně z TTL na RS-232 C, signál však není invertován). Bity se pomocí operátoru posuvu << posouvají směrem doleva, první přijatý bit totiž odpovídá nejvyššímu bitu výsledku (pochopitelně, protože převodník TLC549 pracuje na principu postupné aproximace).

Chod aplikace ilustruje obr. 8.6.

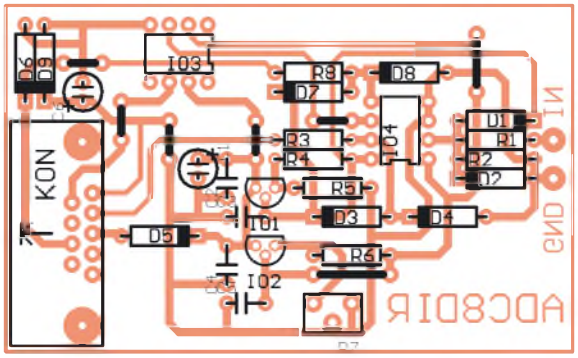
Aplikace DIRADC8

HLFORM.CPP:

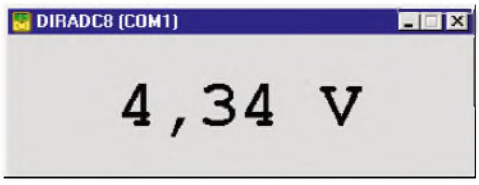
```
#include <vcl.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "HlForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormular *Formular;
//---------------------------------------
__fastcall TFormular::TFormular
      (TComponent* Owner)
      : TForm(Owner)
{
  int CisloPortu;
  //načtení konfigurace:
  TIniFile *ini=new TIniFile
      (GetCurrentDir()
    +"\\DIRADC8.INI");
  CisloPortu=ini->ReadInteger
      ("PORT","Port",1);
  Port=new TSPort(CisloPortu);
  Casovac->Interval=ini->ReadInteger
      ("TIMER","Interval",
      Casovac->Interval);
  delete ini;
```



Obr. 8.4. Obrazec plošných spojů přípravku ADC8DIR (měř.: 1 : 1)



Obr. 8.5. Rozmístění součástek na desce přípravku ADC8DIR



Obr. 8.6. Aplikace v akci

```
  //zapne napájení pro TLC549 a TL061:
  Port->TxD=1;
  Port->RTS=0;

  Caption=AnsiString
        ("DIRADC8 (COM")+CisloPortu+")";
  Application->Title=Caption;
  Napeti->Caption="";
}
//---------------------------------------
__fastcall TFormular::~TFormular()
{
  Port->TxD=0;
  Port->RTS=1;
  delete Port;
}
//---------------------------------------
void __fastcall TFormular::CasovacPretekl
        (TObject *Sender)
{
  //aktualizace měřeného údaje:
  Port->DTR=0;
  Port->RTS=1;  //CS=1
  Sleep(1);     //ustálení
  Port->DTR=1;
  Port->RTS=0;  //CS=0, záporný pól TL061
        //aktivován

  //vlastní čtení:
  BYTE Hodnota=0;
  for(int i=0;i<8;i++){
    //čtení jednoho bitu:
    Hodnota|=Port->RING;
    //a posuv doleva:
    if(i<7)
      Hodnota<<=1;
    //CLK=1:
    Port->DTR=1;
    Sleep(1);
    //CLK=0:
    Port->DTR=0;
    Sleep(1);
  }
  Port->RTS=1;  //CS=1

  //zobrazení výsledku:
  Napeti->Caption=FormatFloat(
    "0.00",Hodnota/256.0*5)+" V";
}
```

Další vylepšení programu

Program lze dále vylepšit např. zobrazováním naměřených údajů v časovém grafu, zobrazováním naměřených údajů v seznamu spolu s přesným časem měření, ukládáním naměřených údajů do diskových souborů atd. atd.

Zde se čtenářům otevírají bohaté možnosti pro vlastní uplatnění.

9. Přípravek DIR8VV

Přípravek **DIR8VV** umožňuje přenášet data mezi PC a osmi binárními vstupy a osmi binárními výstupy, přičemž pro komunikaci mezi přípravkem a PC je použit sériový port PC.

Pro převod dat ze sériové do paralelní formy a naopak jsou použity posuvné registry typu 74HCT595 a 4021, které si nejdříve stručně popíšeme.

SIPO 74HCT595

Obvod **74HC595** je posuvný registr typu **SIPO** (Serial In-Parallel Out), který lze použít pro zmnožení výstupů. Podobný je i obvod **4094**, který však má menší výstupní proud.

Vnitřní zapojení obvodu 74HCT595 je na obr. 9.1.

SI je vstup sériových dat.

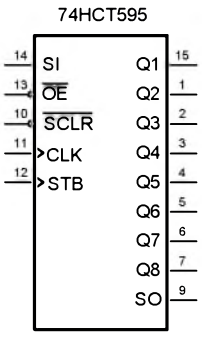
SO slouží pro kaskádní řazení obvodů.

CLK je vstup hodinového signálu (aktivní je vzestupná hrana).

STB je strobovací vstup (aktivní je vzestupná hrana).

OEnon je ovladač třístavového výstupního budiče (aktivní ve stavu „log. 0“).

Zvláštností obvodu je přítomnost vstupu **SCLRnon**, který slouží pro nulování posuvného (nikoliv záchytného registru). Tento vstup je aktivní ve stavu „log. 0“ (při normální činnosti musí být SCLRnon = 1). Tento vstup se obvykle používá pro inicializaci posuvného registru (nezajistí však inicializaci výstupů).



Obr. 9.1. Vnitřní zapojení obvodu 74HCT595

První poslaný bit se objeví na výstupu **Q8**, poslední poslaný bit na výstupu **Q1**.

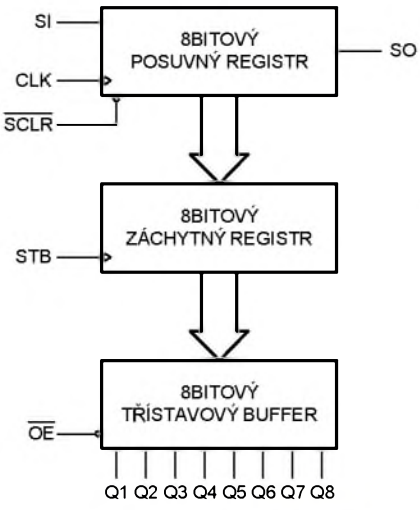
Kladné napájecí napětí ***Ucc*** se přivádí na vývod *16*, zem **GND** je na vývodu *8*.

Hlavní výhodou obvodu 74HCT595 oproti obvodu 4094 je velký proud, který lze odebrat z výstupu (±35 mA), a poměrně velký ztrátový výkon (500 mW). Proto je tento obvod vhodný např. pro přímé řízení LED.

Cena obvodu 74HCT595 je asi 20 Kč. Verze 74HC595 stojí asi 30 Kč a provedení 74LS595 neuvěřitelných 250 Kč!!!

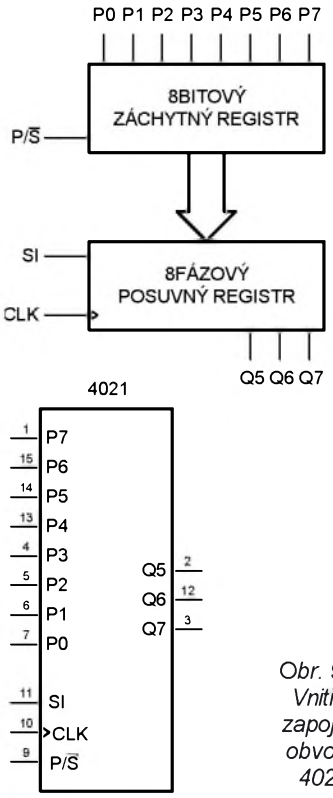
PISO 4021

Obvod **4021** je posuvný registr typu **PISO** (Parallel In-Serial Out), který lze použít pro zmnožení vstupů.



Vnitřní zapojení obvodu 4021 je na obr. 9.2.

Režim činnosti je volen vstupem **P/Snon**. Pro P/Snon = 1 se vstupní paralelní data nahrají do záchytného registru. Při P/Snon = 0 je možné číst jeho obsah. Na výstupu **Q7** je k dispozici nejdříve hodnota odpovídající vstupu **P7**, vzestupnými hranami impulsů na vstupu **CLK** se údaje v posuvném registru posouvají zleva doprava, a tak postupně čteme další bity. Čtení je ukončeno po sedmé vzestupné hraně hodin (čteme P0). Další vzestupná hrana CLK způsobí čtení vstupu **SI**, a to umožňuje např. kaskádně spojovat obvody 4021.



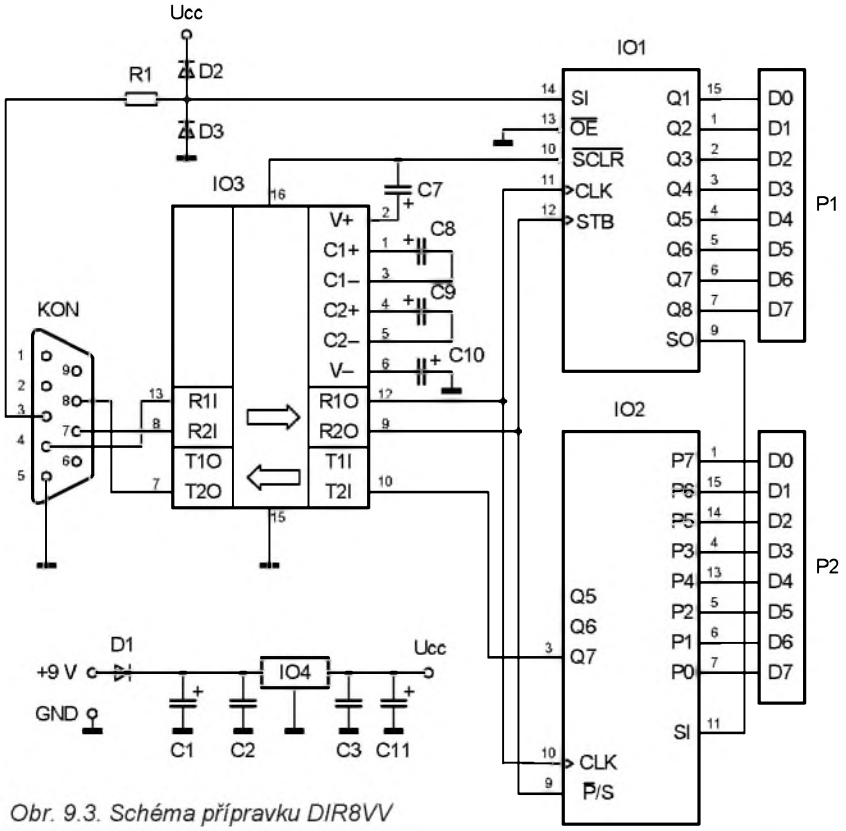
Obr. 9.2. Vnitřní zapojení obvodu 4021

Zapojení přípravku DIR8VV

Na rozdíl od předchozího přípravku ADC8DIR není tento přípravek DIR8VV napájen ze sériového portu, protože předpokládaný odběr zařízení připojeného k jeho výstupům může být poměrně značný! Napájení je tedy nutno přivést z vnějšího zdroje s napětím zhruba 9 V (pokud vynecháte stabilizátor IO4, lze použít přímo napájecí napětí 5 V).

Pro převod z úrovní RS-232 C na úrovně TTL a opačně je použit známý obvod MAX232 (IO3) v klasickém zapojení. Jeho popis nebudu uvádět, protože se jedná o dobře známý obvod.

Linky DTR a RTS procházejí obvodem IO3 a ovládají hodinové a strobovací vstupy posuvných registrů IO1 a IO2. Připomeňme, že IO3 pracuje jako invertor, takže např. pro programové nastavení RTS = 1 bude na lince RTS (kontakt 7 konektoru KON) napětí zhruba +12 V, ale na vývodu R2O obvodu IO3 bude úroveň „log. 0“ (pro RTS = 0 zase úroveň „log. 1“). Tato skutečnost



Obr. 9.3. Schéma přípravku DIR8VV

musí být pochopitelně zohledněna ovládacím programem.

Pro řízení posuvných registrů byl nutný ještě jeden datový výstup, ten je získán z linky TxD pomocí omezovače s R1, D2, D3 (signál na TxD tedy není negován!).

Posuvný registr **74HCT595** pracuje jako osmibitový výstup a je vyveden na konektor P1. Sériový vstup dat odpovídá lince TxD, hodinový vstup odpovídá lince DTR (negované) a strobovací vstup odpovídá lince RTS (negované). Vývod SCLRnon je neaktivní a OEnon je připojen trvale na úroveň „log. 0" (třístavové výstupy budiče jsou tedy aktivovány). Výstup SO je zaveden na datový vstup obvodu IO2 (působí jako zpětná vazba, pomocí které lze zjistit, zda je přípravek skutečně připojen na zvolený sériový port).

Posuvný registr **4021** (IO2) je použit jako osmibitový vstup a přivádějí se na něj binární signály z konektoru P2. Hodinový vstup je ovládán linkou DTR (negováno), strobovací vstup je ovládán linkou RTS (negováno). Sériový vstup je použit pro zpětnovazební připojení IO1 (jak bylo popsáno dříve). Datový výstup je přes IO3 připojen na linku CTS.

Na lince CTS lze tedy číst nejdříve bity odpovídající stavu jednotlivých vstupů P2 a potom další bity posílané přes zpětnovazební smyčku IO1. Připomeňme, že CTS je průchodem převodníkem IO3 negován!

Všechny součástky přípravku DIR8VV jsou umístěné na desce s jednostrannými plošnými spoji.

Na obr. 9.4 je obrazec plošných spojů a na obr. 9.5 je rozmístění součástek na desce. Zapojení konektorů P1 a P2 je na obr. 9.6.

Pro IO1 až IO3 je vhodné použít objímky, aby je bylo možné přemístit i do dalších konstrukcí.

Fotografie přípravku DIR8VV je na obálce tohoto časopisu.

Seznam součástek

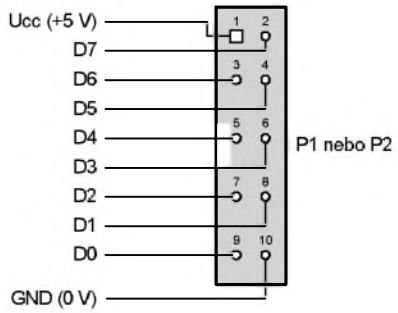
(cena asi 120 Kč)

R1	5,6 kΩ	1 ks
C1, C11	470 µF/16 V	2 ks
C2 až C6	100 nF	5 ks
C7 až C10	22 µF/16 V	4 ks
D1	1N4007	1 ks
D2, D3	1N4148	2 ks
IO1	74HCT595	1 ks
IO2	4021	1 ks
IO3	MAX232	1 ks
IO4	7805	1 ks
KON	CAN 9 Z 90	1 ks
P1, P2	PSL10	2 ks

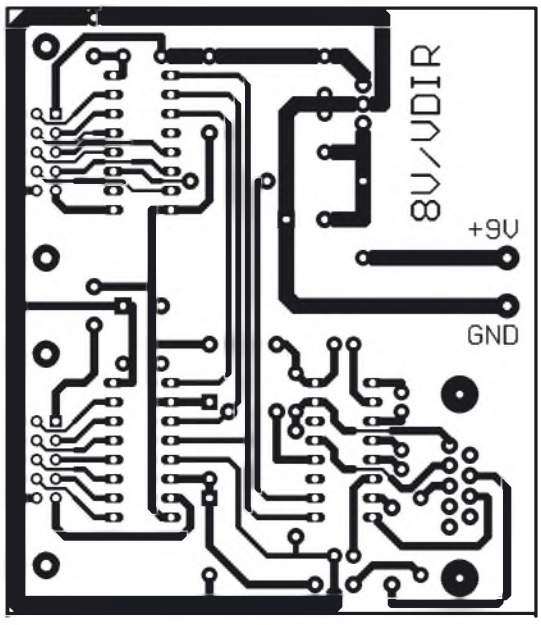
deska s plošnými spoji 8V/VDIR

Přípravek AT8LED

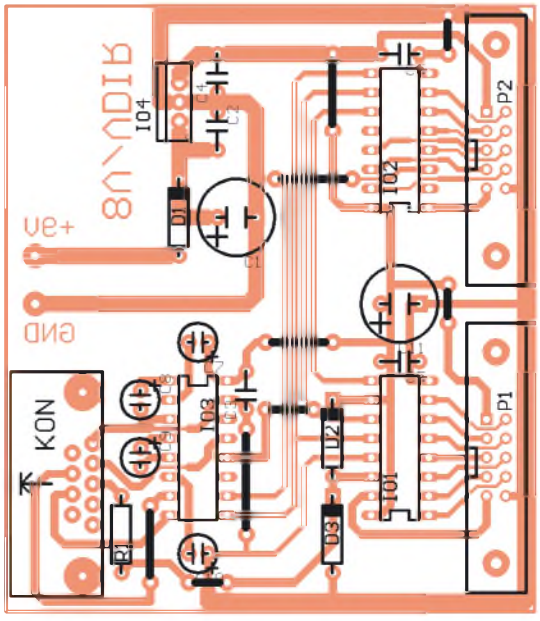
Přípravek **AT8LED** umožňuje připojit k přípravku **DIR8VV** osm LED pro indikaci výstupních dat. Přípravek AT8LED byl poprvé publikován v [4].



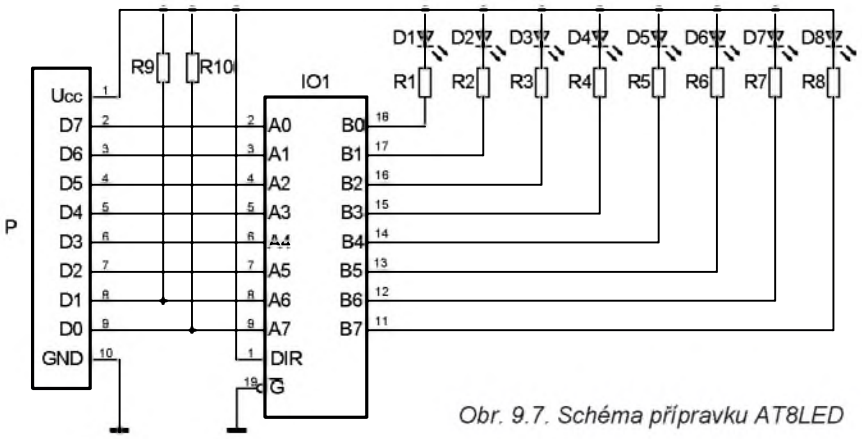
Obr. 9.6. Zapojení konektorů P1 a P2



Obr. 9.4. Obrazec plošných spojů přípravku DIR8VV (měř.: 1 : 1, kratší rozměr desky je 70 mm)



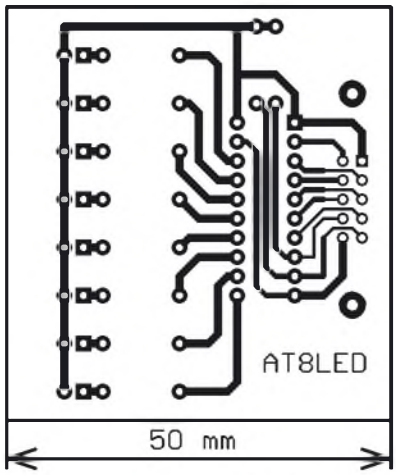
Obr. 9.5. Rozmístění součástek na desce přípravku DIR8VV



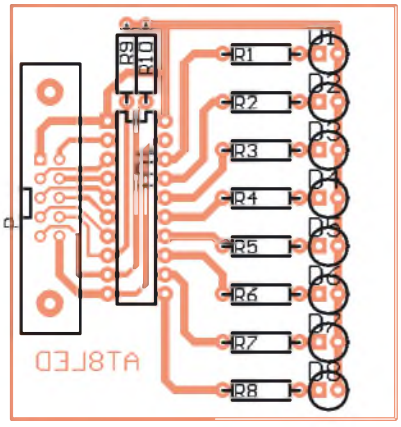
Obr. 9.7. Schéma přípravku AT8LED

Schéma přípravku AT8LED je na obr. 9.7. Do cesty binárních signálů je vřazen budič sběrnice **74HCT245** (IO1), který slouží jako oddělovač, takže LED svítí při úrovních „log. 0" na výstupech DIR8VV. Zatížení výstupů DIR8VV je zanedbatelné.

Přípravek AT8LED se připojuje k výstupnímu konektoru P1 přípravku DIR8VV. LED D1 je řízena nejvýznamnějším výstupním bitem D7, LED D2 je řízena bitem D6 atd. Proud diodami LED určují rezistory R1 až R8. Rezistory R9 a R10 není nutné osazovat (byly nutné pro původní použití v souvislosti s mikrořadičem AT89C2051).



Obr. 9.8. Obrazec plošných spojů přípravku AT8LED (měř.: 1 : 1)



Obr. 9.9. Rozmístění součástek na desce přípravku AT8LED

Součástky přípravku AT8LED jsou umístěné na desce s jednostrannými plošnými spoji.

Na obr. 9.8 je obrazec plošných spojů a na obr. 9.9 je rozmístění součástek na desce.

Pro IO1 je vhodné použít objímku. LED jsou červené o průměru 5 mm nejlevnějšího typu v ceně asi 1 Kč za kus.

Fotografie přípravku AT8LED je na obálce tohoto časopisu.

Pro vzájemné propojení obou přípravků si musíte zhotovit kablík z kousku plochého desetižilového vodiče a dvou konektorů PFL10.

Seznam součástek

(cena asi 40 Kč)

R1 až R8	330 Ω	8 ks
R9, R10	4,7 kΩ	2 ks
D1 až D8	LED, R, 5 mm	8 ks
IO1	74HCT245 (74HC245, T4LS245)	1 ks
P	PSL10	1 ks
deska s plošnými spoji AT8LED		

Ovládací program DIR8VV

Konfigurace aplikace (zvolený sériový port a rychlost snímání) je určena obsahem inicializačního souboru **DIR8VV.INI**.

Po startu se nejprve otestuje připojení přípravku **DIR8VV**. Nejdřív se linkou TxD pošle šestnáct bitů s hodnotou „log. 0" (strobují se hodinami ovládanými signálem DTR). Po této operaci musí být linka CTS (odpovídá výstupu kaskádního spojení obou posuvných registrů) ve stavu „log. 1" (je v negaci). Podobný test se provede s bity o hodnotě „log. 1" posílanými na vývodu TxD. Pokud test selže, není spojení aktivní a patrně to znamená, že přípravek není připojen (nebo je vadný). Program je pak předčasně ukončen navozením výjimečného stavu (**throw**).

Zápis i četní musí být řešeno současně (je provedeno pomocí časovače), protože oba posuvné registry mají společné hodinové i strobovací signály.

Nejdříve se přijme jeden bit z obvodu IO2 a potom se zapíše jeden bit na obvod IO1. Vysílané bity se před odesláním sestaví do podoby bajtu **Vystup**, přijímané bity se postupně skládají do bajtu **Vstup**. Po příjmu všech osmi bitů se provede rozklad na jednotlivé bity (zohlední se zpřeházení linek zvolené s ohledem na jednodušší návrh desky s plošnými spoji přípravku.

Chod aplikace ilustruje obr. 9.10.

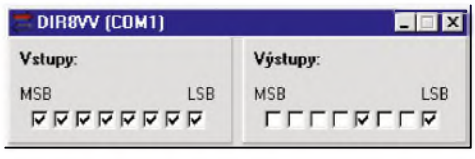
DIR8VV.INI:

```
[PORT]
Port=1
[TIMER]
Interval=55
```

Aplikace DIR8VV

HLFORM.CPP:

```
#include <vcl.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "HlForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormular *Formular;
//---------------------------------------
__fastcall TFormular::TFormular
        (TComponent* Owner)
        : TForm(Owner)
{
  //načtení konfigurace:
  int CisloPortu;
  TIniFile *ini=
    new TIniFile
        (GetCurrentDir()+"\\DIR8VV.INI");
  CisloPortu=ini->ReadInteger
        ("PORT","Port",1);
  Port=new TSPort(CisloPortu);
  Casovac->Interval=
    ini->ReadInteger
("TIMER","Interval",Casovac->Interval);
  delete ini;

  //zobrazení vybraného portu v titulku
        //okna:
  Caption=AnsiString
        ("DIR8VV (COM")+CisloPortu+")";
  Application->Title=Caption;

  //test připojení přípravku (1):
  Port->RTS=1;
  for(int i=0;i<16;i++){
    Port->TxD=0;
    Sleep(1);
    Port->DTR=1;
    Sleep(1);
    Port->DTR=0;
  }
  Sleep(1);
  if(Port->CTS==0)
    throw Exception(
    "Přípravek není připojen");

  //test připojení přípravku (2):
  for(int i=0;i<16;i++){
    Port->TxD=1;
    Sleep(1);
    Port->DTR=1;
    Sleep(1);
    Port->DTR=0;
  }
  Sleep(1);
  if(Port->CTS==1)
    throw Exception(
    "Přípravek není připojen");
```



Obr. 9.10. Aplikace v akci

```
}
//---------------------------------------
__fastcall TFormular::~TFormular()
{
  delete Port;
}
//---------------------------------------
void __fastcall TFormular::CasovacPretekl
       (TObject *Sender)
{
  //obsluha je řešena časovačem:

  //sestaví výstupní bajt:
  BYTE Vystup=Vystup7->Checked*128
    +Vystup6->Checked*64
    +Vystup5->Checked*32
    +Vystup4->Checked*16
    +Vystup3->Checked*8
    +Vystup2->Checked*4
    +Vystup1->Checked*2
    +Vystup0->Checked;
  BYTE Maska=0x80;

  //posílá výstupní bajt
  //a současně přijímá vstupní bajt:
  Port->RTS=0;
  Sleep(1);
  Port->RTS=1;
  Sleep(1);
  BYTE Vstup=0;
  for(int i=0;i<8;i++){
    //čte vstup:
    Vstup|=Port->CTS;
    if(i<7)
      Vstup<<=1;
    //CLK=0
    Port->DTR=1;
    Sleep(1);
    //zapíše výstup:
    Port->TxD=!(Vystup&Maska);
    Maska>>=1;
    //CLK=1
    Port->DTR=0;
    Sleep(1);
  }
  //STB=1
  Port->RTS=0;

  //dekódování přijatého stavu:
  Vstup7->Checked=Vstup&0x01;
  Vstup6->Checked=Vstup&0x02;
  Vstup5->Checked=Vstup&0x04;
  Vstup4->Checked=Vstup&0x10;
  Vstup3->Checked=Vstup&0x08;
  Vstup2->Checked=Vstup&0x20;
  Vstup1->Checked=Vstup&0x40;
  Vstup0->Checked=Vstup&0x80;
}
```

10. Obvody se sběrnicí I²C

Především ve spojení s mikrořadiči se velmi často používají obvody ovládané sběrnicí I²C. Dále si také ukážeme použití obvodů **PCF8591** a **24CXX** (které využívají tuto sběrnici) ve spojitosti s osobním počítačem.

Charakteristiky sběrnice I²C

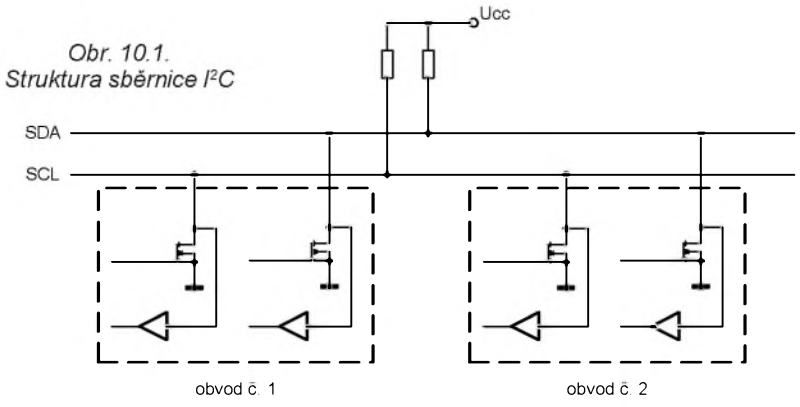
I²C je sběrnice vytvořená firmou Philips Semiconductors a je původně určena pro komunikaci jednočipových procesorů s podřízenými obvody. Proto většinu integrovaných obvodů pracujících s touto sběrnicí vyrábí právě Philips.

I²C je dvouvodičová obousměrná sériová sběrnice pro komunikaci mezi různými integrovanými obvody nebo moduly. I²C je zkratkou označení Inter Integrated Circuit Bus, odpovídající český překlad je sběrnice pro komunikaci mezi integrovanými obvody (mezi-obvodová sběrnice).

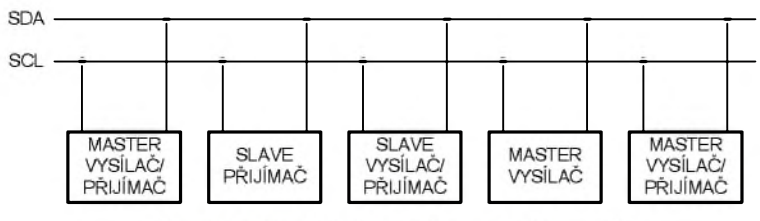
Linky sběrnice I²C se se nazývají SDA a SCL (třetím vodičem sběrnice je společný zemní vodič GND). Linka SDA přenáší data v sériové formě a linka SCL hodinový (taktovací) signál.

Obě linky musí být připojeny na kladný pól napájecího napětí prostřednictvím tzv. zdvihacích rezistorů (pull-up, jedná se vlastně o výstup typu otevřený kolektor). Tím je zajištěna práce linek SDA a SCL v obou směrech. Pokud by nastala kolize (kdyby chtělo vysílat současně více obvodů), poškodí se pouze úrovně signálů a nikoli vysílající obvody.

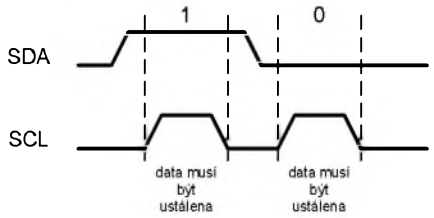
Zpětnou vazbou je zajištěno, že každý obvod může pracovat jako vysílač i jako přijímač (obr. 10.1).



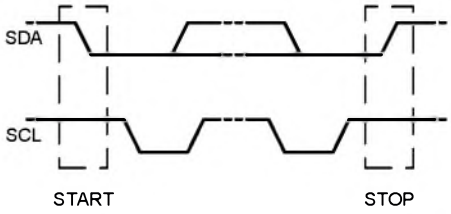
Obr. 10.1. Struktura sběrnice I²C



Obr. 10.4. Uspořádání systému se sběrnicí I²C



Obr. 10.2. Přenos bitů dat sběrnicí I²C



Obr. 10.3. START a STOP na sběrnici I²C

Přenos bitů

V průběhu jednoho hodinového cyklu SCL je přenesen právě jeden datový bit (obr. 10.2). Data přivedená na linku SDA musí zůstat neměnná po celou dobu trvání kladného impulsu hodin SCL. Při SCL = 1 jsou totiž změny SDA chápány jako řídicí signál.

START a STOP

K označení začátku a konce přenosu nejsou používány přídavné řídicí linky, ale dva speciální stavy sběrnice (obr. 10.3).

Start přenosu (START nebo S) je oznámen sestupnou hranou SDA při SCL = 1.

Konec přenosu (STOP nebo P) je definován náběžnou hranou SDA při SCL = 1. Pokud je sběrnice v neaktivním stavu, jsou signály SDA a SCL ve stavu „log. 1".

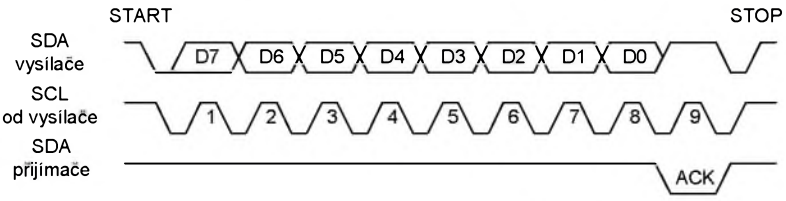
Uspořádání systému

Zařízení, které vysílá zprávy, se označuje jako **vysílač** (transmitter). Zařízení přijímající zprávy je **přijímač** (receiver). Tyto intuitivní pojmy jsou dobře známy.

V terminologii sběrnice I²C se objevují ještě další dvě označení (viz obr. 10.4):

- **Master** (řídicí obvod) je zařízení, které řídí tok zpráv.
- **Slave** (řízený obvod) je zařízení řízené masterem.

Jak plyne z obr. 10.4, funkce zařízení na sběrnici I²C se může měnit podle potřeby. Jednou pracuje řídicí obvod (master) jako vysílač a jindy jako přijímač. Podobně se může chovat i řízený obvod (slave). Některé obvody pracují pouze jako vysílače nebo přijímače (směr toku dat se v nich nemění).



Obr. 10.5. Kvitování na sběrnici I²C (musí být zajištěn předstih a přesah SDA vůči SCL)

Obr. 10.6. Příklad protokolu pro přijímač slave

Obr. 10.7. Adresování obvodů PCF8574, PCF8591 a TDA8444

	A7	A6	A5	A4	A3	A2	A1	R/$\overline{W}$
PCF8574	0	1	0	0	v	v	v	0/1
PCF8591	1	0	0	1	v	v	v	0/1
TDA8444	0	1	0	0	v	v	v	0

Tab 10.1. Elektrické standardy sběrnice I²C

Parametr	Standard		Fast		
	min	max	min	max	jednotka
Rozsah vstupních úrovní log. 0	–0,5	1,5	–0,5	1,5	V
Rozsah vstupních úrovní log. 1	3	—	3	—	V
Rozsah výstupních úrovní log. 0	—	0,4	—	0,4	V
Vstupní odběr	–10	10	–10	10	µA
Kmitočet SCL	0	100	0	400	kHz
Doba trvání log. 0 SCL	4,7	—	1,3	—	µs
Doba trvání log. 1 SCL	4,0	—	0,6	—	µs
Doba náběhu	—	1000	—	300	ns
Doba poklesu	—	300	—	300	ns

Kvitování (ACK)

Počet datových bitů přenesených z vysílače do přijímače mezi stavy START a STOP není omezen. Každý datový bajt (8 bitů) je následován jedním kvitovacím (potvrzovacím) bitem (ACK nebo A).

ACK představuje bit o hodnotě „log. 0", který je vložený přijímačem na linku SDA a je potvrzený hodinovým impulsem na lince SCL (obr. 10.5). Přijímač, který je adresován, musí generovat ACK po přijetí každého bajtu.

Jak je vidět z obr. 10.5, datové bity se vysílají počínaje nejvyšším.

Protokol sběrnice I²C

Sběrnice I²C není vybavena adresovou sběrnicí. Z toho důvodu musí být adresa vysílána stejným způsobem jako data.

Příklad protokolu jednoduchého obvodu připojeného ke sběrnici I²C je na obr. 10.6. Jedná se o přijímač slave.

Po vyvolání START je nejdříve třeba vyslat platnou adresu oslovovaného obvodu. Poslední bit této adresy určuje směr přenosu (viz dále). Po adrese musí oslovovaný obvod vytvořit ACK a pak následuje bajt dat zakončený opět ACK. Přenos uzavírá STOP.

Konkrétní protokol je vždy závislý na typu obvodu. Proto bude tato informace konkretizována v následujícím textu.

Adresy zařízení I²C

Adresy zařízení I²C se skládají z pevné a volitelné části (obr. 10.7).

Pevná část adresy je hadwarově vestavěna v integrovaném obvodu, který pracuje se sběrnicí I²C. Např. obvod PCF8574 má čtyři nejvyšší bity adresy vždy rovny 0100.

Volitelná část adresy umožňuje připojit ke sběrnici I²C několik stejných obvodů. Obvykle má volitelná část adresy 3 bity. Pak lze připojit až osm obvodů stejného typu na sdílenou sběrnici. Tato část adresy je v obr. 10.7 je označena symbolem **v** (jako volitelná). Volitelná část adresy se nastavuje připojením adresovacích vstupů k úrovním „log. 1" nebo „log. 0".

Nejnižší bit **R/Wnon** adresy určuje **směr přenosu**. Pro R/Wnon = 1 je obvod vysílačem (čteme z něj data), pro R/Wnon = 0 pracuje obvod jako přijímač (zapisujeme do něj data). Obvody, které pracují v obou směrech, tedy rozlišují dvě adresy (adresa pro vysílání je o jedničku vyšší než pro příjem). Obvod TDA8444 může data pouze přijímat, proto je na obr. 10.7 ve sloupci R/Wnon jen hodnota 0.

Parametry sběrnice I²C

Pro praktickou práci s obvody vybavenými sběrnicí I²C je třeba znát statické a dynamické parametry této sběrnice. Je to nutné z toho důvodu, že je např. omezena maximální přenosová rychlost. Při rychlejší komunikaci by vznikaly chyby přenosu.

Elektrické standardy sběrnice I²C jsou v tab. 10.1.

Z tabulky vyplývá, že pro běžně dostupné provedení Standard může mít SCL nejvyšší kmitočet 100 kHz, přičemž záporná půlvlna hodin musí trvat alespoň 4,7 µs a kladná 4,0 µs.

11. Přípravek PCF8591 - čtyřkanálový osmibitový A/D převodník

Základem přípravku je integrovaný obvod PCF8591. Je to A/D a D/A převodník pracující s jediným napájecím napětím. Obvod je vybaven čtyřmi analogovými vstupy a jedním analogovým výstupem. Blokové schéma obvodu je na obr. 11.1.

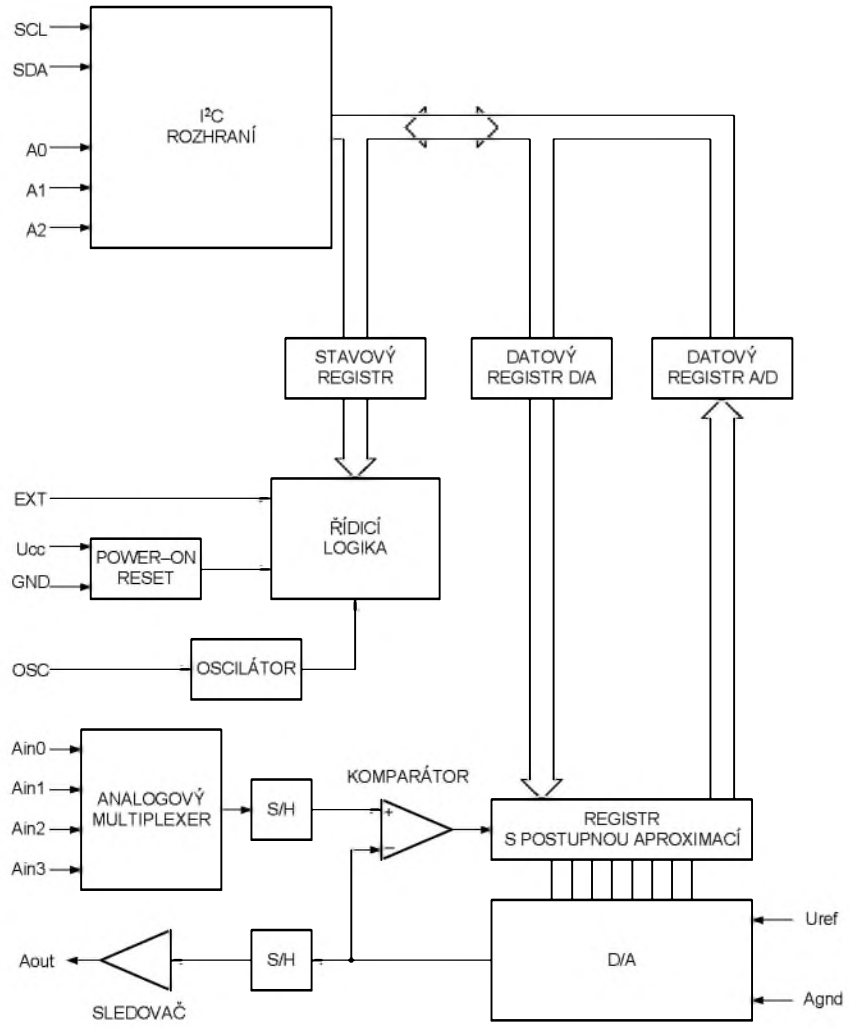
IO PCF8591 obsahuje čtyři analogové vstupy **Ain0** až **Ain3**, které mohou měřit napětí proti zemi (SE - single-ended) nebo diferenčně (mezi sebou). Vstupy se volí čtyřkanálovým analogovým multiplexerem, který je připojuje k osmibitovému A/D převodníku s postupnou aproximací. Kromě analogových vstupů obsahuje obvod i jeden analogový výstup **Aout**.

Analogové vstupy a výstupy používají jako srovnávací potenciál vstup **Agnd** (analogovou zem).

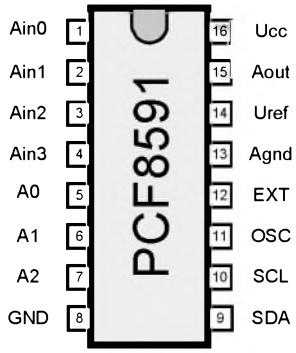
Referenční napětí pro D/A převodníku se zavádí do vstupu **Uref** .

Vývody **EXT** a **OSC** ovládají pracovní kmitočet A/D převodníku. Je-li vstup EXT připojen na napájecí napětí Ucc, vytváří obvod pracovní kmitočet sám a jeho výstup je k dispozici na vývodu OSC. Je-li EXT připojen na GND, slouží OSC jako vstup pracovního kmitočtu.

Zapojení vývodů obvodu PCF8591 v pouzdru DIP16 je na obr. 11.2. Význam signálů na jednotlivých vývodech je uveden v tab. 11.1.



Obr. 11.1. Blokové schéma obvodu PCF8591



Obr. 11.2. Zapojení vývodů obvodu PCF8591 v pouzdru DIP 16

Tab 11.1. Význam signálů na vývodech obvodu PCF8591

Vývod	Význam
Ucc	napájecí napětí (2,5 až 6 V)
GND	zem (0 V)
SDA	datový vstup/výstup sběrnice I²C
SCL	hodinový signál I²C (od mastera)
Ax	adresové vstupy obvodu
Ainx	analogové vstupy
Aout	analogový výstup
Agnd	analogová zem
Uref	referenční napětí analogové části
OSC	vstup/výstup oscilátoru
EXT	přepínač oscilátoru

Adresování

Jako ostatní obvody, pracující se sběrnicí I²C, je i obvod PCF8591 aktivován po příjmu platné adresy.

Adresovací bajt je znázorněn na obr. 11.3. Čtyři nejvyšší bity nesou pevnou část adresy (1001). Za nimi následují bity A2, A1 a A0 a bit R/Wnon, který určuje směr komunikace.

Řídicí bajt

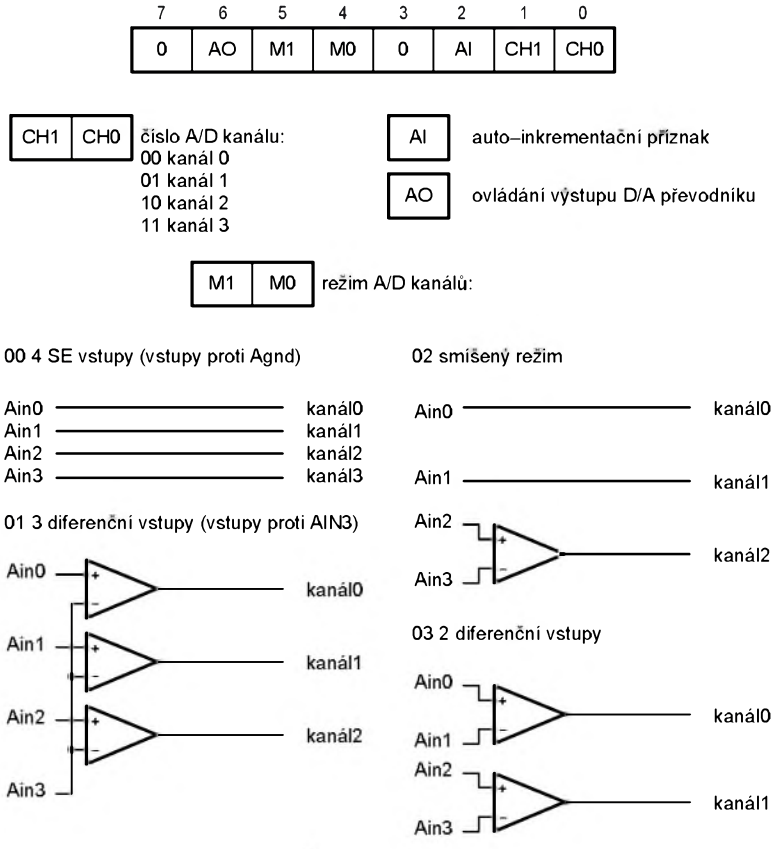
Po vyslání adresovacího bajtu je třeba určit funkci obvodu - musí se vyslat řídicí bajt. Význam jednotlivých bitů řídicího bajtu je zřejmý z obr. 11.4.

Je-li příznak auto-inkrementace nastaven, je číslo A/D vstupu zvětšeno po každém převodu. Je-li auto-inkrementace použita v režimu s interním oscilátorem, měl by být analogový výstup aktivní (bit 6), aby interní oscilátor kmital nepřetržitě a tím se zabránilo chybám vzniklým v důsledku existence prodlevy nutné pro spuštění oscilátoru.

V ostatních případech (pokud není třeba použít analogový výstup), se doporučuje bit 6 vynulovat. Tím se dosáhne výrazného zmenšení příkonu obvodu.

1	0	0	1	A2	A1	A0	R/W

Obr. 11.3. Adresování obvodu PCF8591



Obr. 11.4. Řídicí bajt obvodu PCF8591

D7	D6	D5	D4	D3	D2	D1	D0

Obr. 11.5. Bajt dat pro IO PCF8591

Výběr neexistujícího kanálu (například výběr kanálu 3 v režimu se dvěma diferenčními vstupy) způsobí výběr posledního dostupného kanálu. V auto-inkrementačním režimu je tento problém odstraněn, vybere se kanál 0 (cyklické adresování).

D/A převodník

Třetí bajt při přenosu dat (obr. 11.5) je uložen do registru D/A převodníku a převeden na odpovídající analogové napětí.

D/A převodník obsahuje odporový dělič s 256 odbočkami a výběrovými spínači. Dělič je připojen mezi vývody Uref a Agnd. Výstupní napětí je proudově posíleno v automaticky nulovaném zesilovači s jednotkovým zesílením. Tento zesilovač lze vyřadit z funkce bitem 6 řídicího slova (viz obr. 11.4) a tím výrazně zmenšit příkon obvodu.

Jak vyplývá z obr. 11.1, je D/A převodník použit rovněž pro převod A/D (s využitím algoritmu postupné aproximace). Proto je k výstupu D/A převodníku napřed připojen obvod S/H (Sample & Hold, vzorkovač) a teprve potom výstupní zesilovač.

Vstup Agnd nemusí být nutně spojen s vývodem GND. Pak může D/A převodník pracovat v rozsahu napětí od *Agnd* až zhruba do *Uref*.

A/D převodník

Cyklus A/D převodu začíná vždy po vyslání platné čtecí adresy (R/Wnon = = 1). Cyklus je spuštěn sestupnou hranou ACK hodinového impulsu SCL a probíhá současně s přenosem výsledku předchozího převodu. To má za následek, že při čtení výsledku A/D převodu dostaneme nejprve předchozí hodnotu (obr. 11.6). Teprve následujícím čtením dostaneme hodnotu převedenou v přechozím cyklu! Po vynulování (resetu) má první načtený bajt vždy hodnotu 1000 0000_2.

Pokud je použito zapojení vstupů SE (měřené napětí se přivádí mezi vstup Ain a Agnd), používá se jednoduchý osmibitový kód (minimu odpovídá binární číslo 0000 0000_2 a maximu 1111 1111_2) - obr. 11.7.

Při diferenčním zapojení vstupů (měřené napětí se přivádí mezi dva vstupy Ain+ a Ain-) se používá druhý doplněk (minimu odpovídá 1000 0000_2 a maximu 0111 1111_2) - obr. 11.8.

Maximální rychlost A/D převodu je určena aktuální rychlostí sběrnice I^2C.

Referenční napětí

Pro A/D a D/A převod je třeba zajistit stabilní vnější referenční napětí připojené mezi vstupy Uref a Agnd.

Vývod Agnd může být napěťově posunut vůči systémové zemi GND, tím je dosaženo posunutí (offset).

Mezi vstupy Uref a Agnd může být připojen nízkofrekvenční signál, D/A převodník se pak chová jako jednokvadrantová násobička a A/D převodník jako jednokvadrantová nebo dvoukvadrantová dělička.

Mezní a charakteristické údaje

Mezní a charakteristické údaje obvodu PCF8591 jsou v tab. 11.2 a tab. 11.3.

Tab 11.2. Mezní parametry obvodu PCF8591

Parametr	min	max
Napájecí napětí *Ucc* [V]	-0,5	+8,0
Napájecí proud *Icc* [mA]	-	±50
Ztrátový výkon Ptot [mW]	-	300
Vstupní napětí [V]	-0,5	*Ucc* + 0,5
Výstupní proud *Io* [mW]	-	±20
Provozní teplota *ta* [°C]	-40	+85

Popis přípravku PCF8591

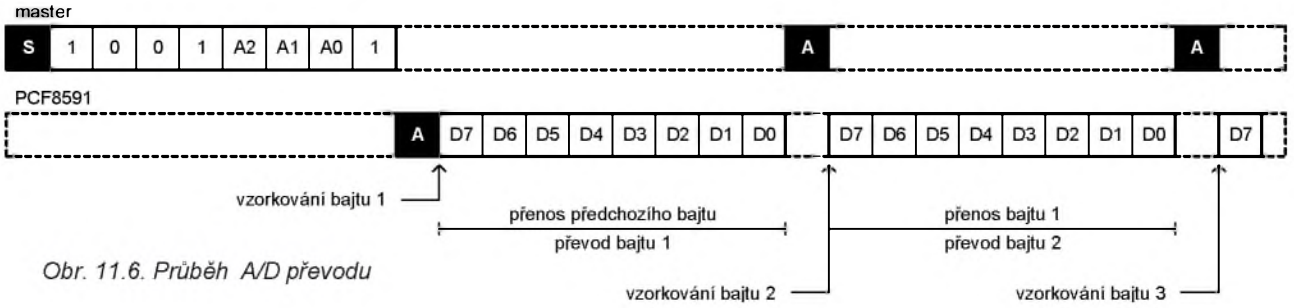
Pro praktické vyzkoušení práce s obvodem PCF8591 byl navržen přípravek PCF5891. Schéma přípravku je na obr. 11.9.

Přípravek je napájen ze symetrického zdroje napětím ±15 V, přepólování brání diody D1 a D2. Napájecí napětí musí být dobře vyfiltrované, protože přípravek neobsahuje filtrační kondenzátory. Číslicová část obvodu PCF8591 je napájena ze stabilizátoru **78L05** (IO4).

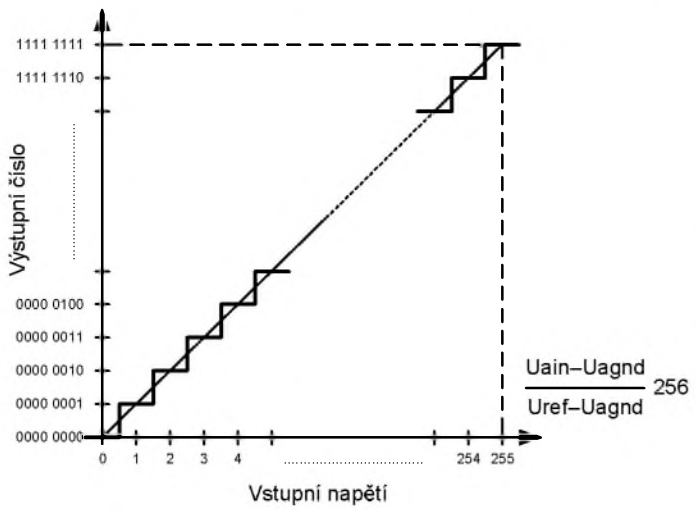
Referenční napětí zhruba 2,5 V je získáváno v referenčním zdroji s obvodem **TL431** (IO2) a pracovním rezistorem R17.

Vývod **EXT** obvodu PCF8591(IO1) je připojen na úroveň „log. 0" (uzemněn), takže pracovní kmitočet pro A/D převodník je vytvářen pomocí vnitřního *RC* generátoru v IO1. Adresovací vstupy A0, A1, A2 jsou spojeny s úrovní „log. 0", adresa obvodu tedy je 144 resp. 145.

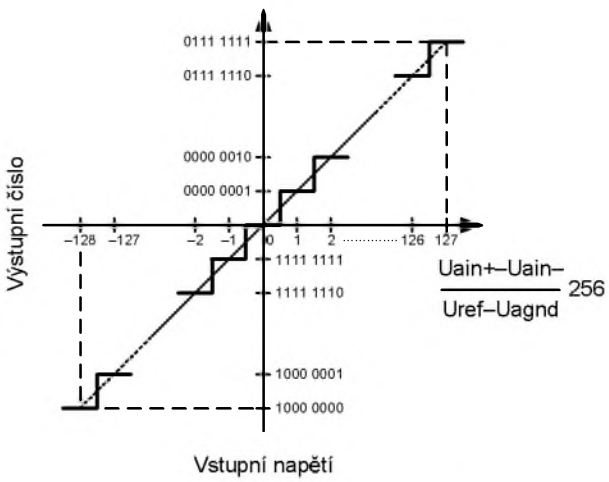
IO1 je připojen k sériovému portu PC přes rezistory R18 a R19, které pracují jako omezovače proudu. V tomto



Obr. 11.6. Průběh A/D převodu



Obr. 11.7. Převodní charakteristika vstupů SE.



Obr. 11.8. Převodní charakteristika diferenčních vstupů

Tab 11.3. Charakteristické údaje obvodu PCF8591 (CMRR[1] je činitel potlačení souhlasného signálu)

Parametr	Podmínky	min	typ	max	jednotka
VSTUPY REFERENČNÍHO NAPĚTÍ					
Refereční napětí Uref	Uref > Uagnd	Ugnd+1,6	—	Ucc	V
Napětí Uagnd	Uref > Uagnd	Ugnd	—	Ucc–0,8	V
Odběr z referečního zdroje Iref		–250	—	+250	nA
VSTUPY OSCILÁTORU					
Odběr		—	—	250	nA
Kmitočet oscilací	vnitřní oscilátor	0,75	—	1,25	MHz
D/A PŘEVODNÍK					
Výstupní napětí	naprázdno	Ugnd	—	Ucc	V
	Rz = 10 kΩ	Ugnd	—	0,9Ucc	V
Chyba nuly	ta = 25 °C	—	—	50	mV
Chyba linearity		—	—	±1,5	LSB
Chyba zesílení	naprázdno	—	—	1	%
Doba ustálení	s chybou výstupu do 0,5LSB	—	—	90	µs
Kmitočet převodu		—	—	11,1	kHz
Činitel potlačení kolísání Ucc	ΔUcc = 0,1Ucc	—	40	—	dB
A/D PŘEVODNÍK					
Rozsah vstupního napětí		Ugnd	—	Ucc	V
Vstupní proud		—	—	100	nA
Vstupní kapacita		—	10	—	pF
Měřicí rozsah SE vstupů	měřicí rozsah	Uagnd	—	Uref	V
Měřicí rozsah dif. vstupů	měřicí rozsah, Ufs = Uref – Uagnd	–0,5Ufs	—	+0,5Ufs	V
Chyba nuly	ta = 25 °C	—	—	20	mV
Chyba linearity		—	—	±1,5	LSB
Chyba zesílení		—	—	1	%
CMRR[1]		—	60	—	dB
Činitel potlačení kolísání Ucc	ΔUcc = 0,1Ucc	—	40	—	dB
Doba ustálení		—	—	90	µs
Kmitočet převodu		—	—	11,1	kHz

zapojení jsem vynechal omezovací diody, využil jsem totiž skutečnosti, že většina integrovaných obvodů má takové (ochranné) diody itegrované u každého vývodu ve své vnitřní struktuře (je to i případ obvodu PCF8591).

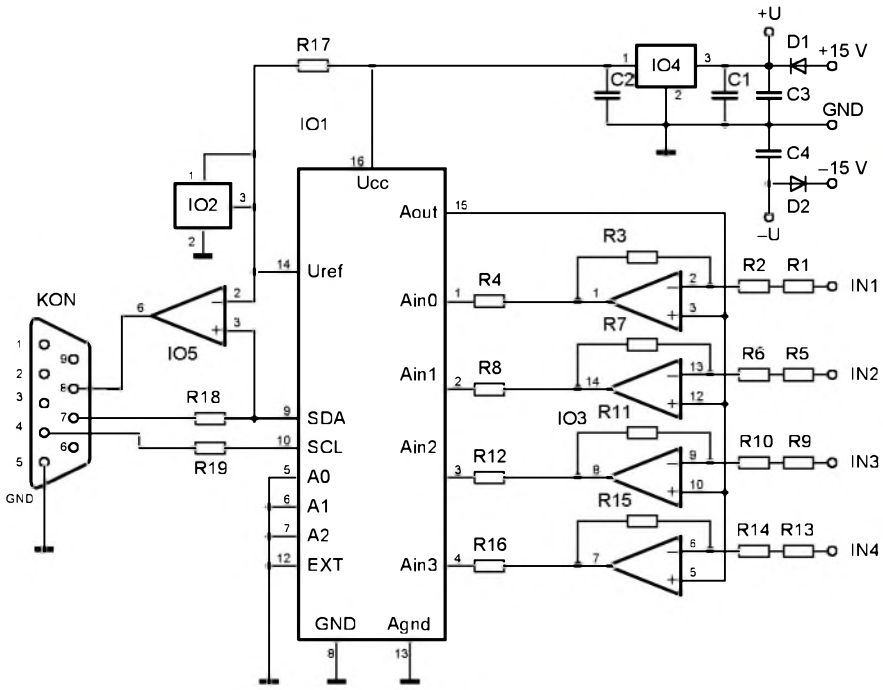
Vývody **SDA** a **SCL** jsou připojeny na linky RTS, DTR a CTS. SCL je připojen přímo na DTR. Výstupní linka SDA odpovídá RTS. Čtení SDA probíhá linkou CTS.

Pro převod z úrovní TTL na RS-232C je použit operační zesilovač (OZ) TL071 (IO5), který je zapojen jako komparátor. V tomto přípravku není nutné šetřit napájecím proudem, proto nebyl použit OZ TL061 s malým příkonem, ale levnější TL071. Komparátor je napájen přímo z vnějšího zdroje. Signál není komparátorem invertován, úrovni „log. 0" na neinvertujícím vstupu OZ odpovídá výstupní napětí +15 V a úrovni „log. 1" napětí -15 V.

Signály ze vstupů **IN1** až **IN4** jsou přivedeny na vstupy **Ain0** až **Ain3** IO1 přes operační zesilovače **TL074** (IO3). Realizovaná operační síť je analyzována na obr. 11.10. Napětí *U2* je generováno vnitřním D/A převodníkem a lze je nastavovat v rozsahu 0 až 2,5 V.

Bude-li *U2* = 1 V, odpovídá rozsahu vstupního napětí -5 až +5 V rozsah výstupního napětí 2,5 až 0 V a je tedy k dispozici bipolární vstup -5 až +5 V. Pro *U2* = 2 V odpovídá vstupnímu napětí v rozsahu 0 až 10 V rozsah výstupního napětí 2,5 až 0 V a je tedy k dispozici unipolární vstup 0 až 10 V.

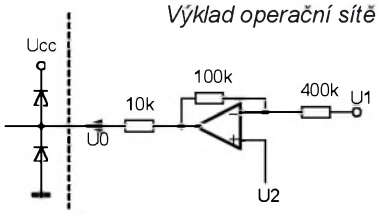
Operační zesilovače pochopitelně nevylučují nebezpečí poškození vstupů Ain0 až Ain3 IO1 nevhodným vstupním napětím. Proto jsou vstupy chráněny omezovacími rezistory R4, R8, R12, R16 o odporu 10 kΩ. Opět se využívají vnitřní ochranné diody.



Obr. 11.9. Schéma přípravku PCF8591

PCF8591

Obr. 11.10. Výklad operační sítě



Součástky přípravku PCF8591 jsou umístěné na desce s jednostrannými plošnými spoji. Na obr. 11.11 je obrazec plošných spojů a na obr. 11.12 je rozmístění součástek na desce.

Pro IO1, IO3 a IO5 je vhodné použít objímky.

Fotografie přípravku PCF8591 je na obálce tohoto časopisu.

Seznam součástek

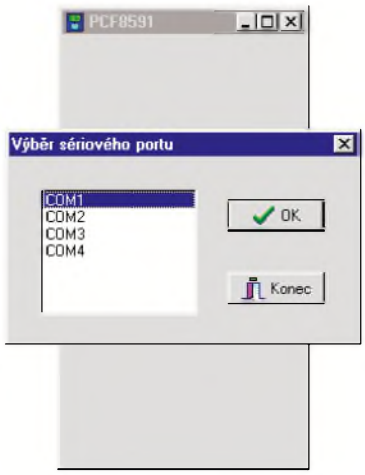
(cena asi 200 Kč)

R1, R2, R5, R6, R9, R10, R13, R14	200 kΩ	8 ks
R3, R7, R11, R15	100 kΩ	4 ks
R4, R8, R12, R16, R18, R19	10 kΩ	4 ks
R17	100 Ω	1 ks
C1 až C4	100 n	4 ks
D1, D2	1N4007	2 ks
IO1	PCF8591	1 ks
IO2	TL431	1 ks
IO3	TL074	1 ks
IO4	78L05	1 ks
IO5	TL071	1 ks
KON	CAN 9 Z 90	1 ks
deska s plošnými spoji PCF8591		

Ovládací program

Na tomto místě nebude uveden výpis ovládacího programu (je příliš dlouhý). Spíše bude stručně popsána jeho realizace a důraz bude kladen na vystvětlení jeho obsluhy.

Po spuštění musí uživatel zvolit port, ke kterému je přípravek připojen. Volba se provede pomocí dialogu



Obr. 11.13. Volba portu při startu aplikace

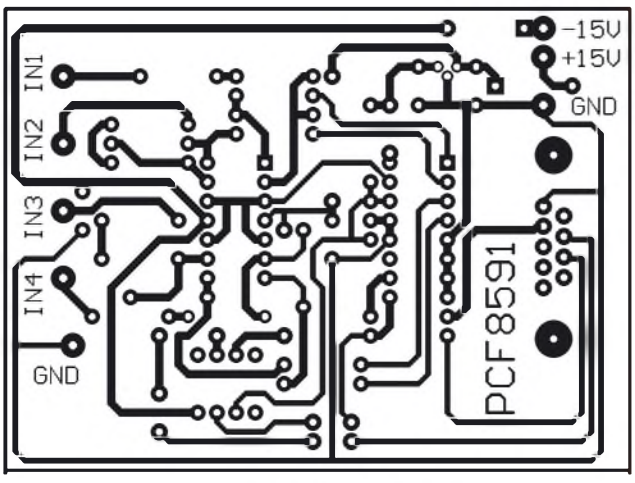
z obr. 11.13, na kterém je zobrazen seznam dostupných portů.

Potom řídicí program ověří (testováním ACK při komunikaci), zda je přípravek skutečně připojen. Pokud ano, zobrazí se hlavní panel podle obr. 11.14.

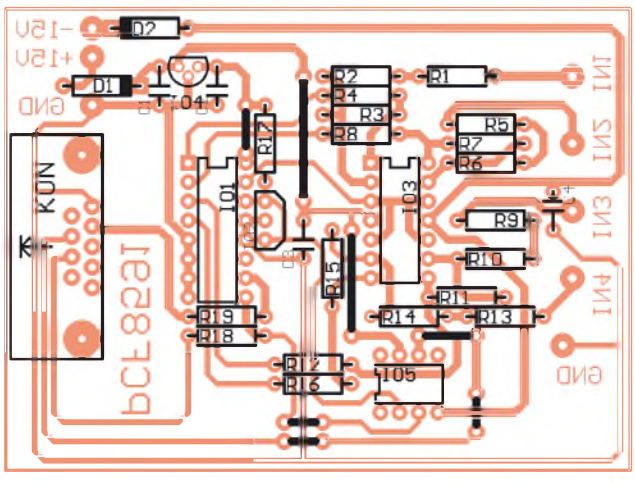
V hlavním panelu je možno volit bipolární nebo unipolární rozsah, režim práce a kanál, který se snímá.

Vlastní ovládání je řešeno pomocí specializovaných tříd, jejich popis přesahuje rámec tohoto článku (viz [6]).

Obr. 11 14 Ovládací panel



Obr. 11.11. Obrazec plošných spojů přípravku PCF8591 (měř.: 1 : 1, delší rozměr desky je 82,5 mm)



Obr. 11.12. Rozmístění součástek na desce přípravku PCF8591

12. Přípravek PROG24 - programátor pamětí 24CXX

Ve spojení s mikrořadiči se velmi často používají paměti E^{2}PROM. Jedná se o paměť, která je mazatelná elektricky a udržuje svůj obsah i po odpojení napájecího napětí.

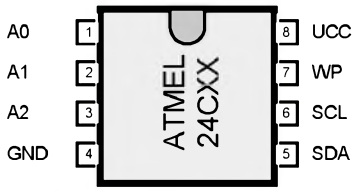
Zaměříme se na paměti typu 24CXX, které se ovládají sběrnicí I^{2}C.

Stručný popis pamětí 24C01A, 24C02, 24C04, 24C08, 24C16

Tyto obvody jsou sériové paměti E^{2}PROM o kapacitách 1024, 2048, 4096, 8192 a 16384 bitů s organizací 128x 8, 256x 8, 512x 8, 1024x 8 a 2048x 8 bitů.

Paměti mají tyto klíčové vlastnosti:

- Provedení standardní nebo s malým příkonem s napájecím napětím 2,7 až 5,5 V nebo 1,8 až 5,5 V.
- Organizace od 128x 8 do 2048x 8 bitů.
- Vstupy se Schmittovými klopnými obvody, které filtrují rušení.
- Hodinový kmitočet 100 kHz při napájení 1,8, 2,5 a 2,7 V nebo 400 kHz při napájení 5 V.
- Vývod WP pro hardwarovou zábranu zápisu.
- Osmibajtové (u kapacit 1Kb a 2 Kb) nebo šestnáctibajtové (u kapacit 4 Kb až 16 Kb) stránkové zapisovací režimy.
- Dovolený částečný zápis stránky.
- Automatické časování zapisovacího cyklu (doba zápisu maximálně 10 ms).



Obr. 12.1. Zapojení vývodů pamětí 24CXX v pouzdru DIP 8

- 1 000 000 programovacích cyklů, trvanlivost obsahu 100 let.

Zapojení vývodů pamětí

Zapojení vývodů pamětí 24CXX v pouzdru DIP 8 je na obr. 12.1.

Adresovací vstupy **A2**, **A1**, **A0** slouží k adresování I^{2}C zařízení podobně jako u jiných I^{2}C obvodů. Úloha se však liší podle typu paměti:

- Paměti **24C01A** a **24C02** se adresují sedmi/osmibitovou adresou, v tomto případě lze na stejné vodiče SDA a SCL připojit až 8 pamětí tohoto typu a adresovat je vývody A2, A1, A0.
- U paměti **24C04** není vývod A0 použit, lze tedy adresovat 4 paměti (volný bit se používá pro stránkování).
- U paměti **24C08** nejsou použity vývody A1 a A0, lze tedy adresovat 2 paměti (volné bity se používají pro stránkování),
- Paměť **24C16** nelze adresovat (volné bity se používají pro stránkování).

Signál na vývodu **WP** blokuje programování. Pokud je WP = 0, paměť lze programovat, při WP = 1 je programování blokováno (tab. 12.1).

Vývody **SDA** a **SCL** jsou linky sběrnice I^{2}C.

GND je společný vývod napájení, na vývod **Ucc** se přivádí kladné napájecí napětí.

Organizace paměti

Organizace jednotlivých typů pamětí je uvedena v tab. 12.2.

Adresování obvodů

Podobně jako jiné I^{2}C obvody musí být paměť po vyslání **START** adresována, aby se povolila operace čtení nebo zápisu.

Bajt adresy obvodu se skládá z pevné a volitelné části. Pevná část je tvořena povinnou sekvencí jedniček a nul pro čtyři nejvýznamnější bity. Ve volitelné části určuje nejnižší bit směr toku dat (jako v ostatních případech) a zbývající tři bity mají význam závislý na typu paměti.

Pro paměti **24C01A** a **24C02** (o kapacitě 1 Kb a 2 Kb) představují další tři bity adresu, která musí odpovídat stavu vývodů A2, A1, A0.

Pro paměť **24C04** (o kapacitě 4 Kb) se používají pouze vývody A2, A1 (A0 není připojen), volný bit umožňuje stránkování.

Pro paměť **24C08** (o kapacitě 8 Kb) se používá pouze vývod A2 (A1, A0 nejsou připojeny), volné bity umožňují stránkování.

Paměť 24C16 (o kapacitě 16 Kb) nelze adresovat (vývody A2, A1, A0 nejsou připojeny), volné bity umožňují stránkování.

Zápis do paměti

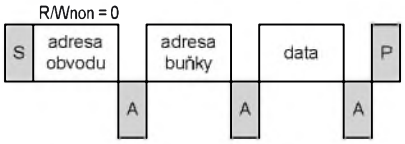
Do paměti lze zapisovat data dvěma způsoby:

- **Zápis bajtu** (obr. 12.3): Tento typ zápisu vyžaduje osmibitovou adresu buňky, kam se má zapsat bajt dat (předchází ji pochopitelně START a adresa obvodu). STOP je nutný, aby se provedl zápis (trvá méně než 10 ms). V průběhu zápisu se nesmí měnit stav žádného vstupu!
- **Zápis stránky** (obr. 12.4): Paměti o kapacitě 1 Kb a 2 Kb podporují osmibajtové stránkování. Paměti o kapacitě 4 Kb, 8 Kb a 16 Kb podporují šestnáctibajtové stránkování. V tomto režimu nevysílá master (například mikrořadič) po zápisu prvního datového bajtu STOP a místo toho pokračuje vysláním dat pro následující buňku paměti. Takto lze vyslat 8 bajtů dat (u pamětí s kapacitou 1 nebo 2 Kb) nebo 16 bajtů dat (u pamětí s kapacitou 4, 8 nebo 16 Kb) odpovídajících jedné stránce.

Dolní 3 bity (u pamětí s kapacitou 1 nebo 2 Kb) nebo 4 bity (u pamětí s kapacitou 4, 8 nebo16 Kb) adresy jsou vnitřně inkrementovány po příjmu dalšího bajtu dat (horní část adresy se nemění, zachovává zvolenou stránku).

Tab 12.1. Význam signálu WP u jednotlivých typů pamětí

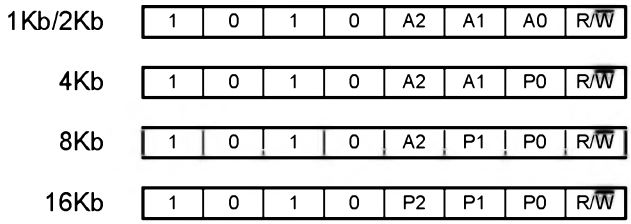
WP	Chráněná část paměťového pole				
	24C01A	24C02	24C04	24C08	24C16
0	celý prostor	celý prostor	celý prostor	normální operace čtení/zápisu	horní polovina pole (8Kb)
1	normální operace čtení/zápisu				



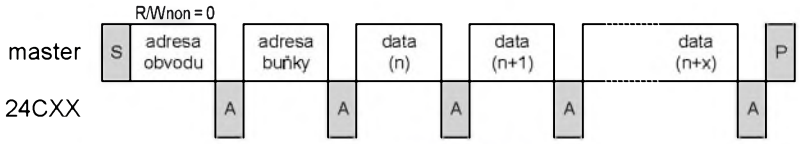
Obr. 12.3. Zápis bajtu. Nad střední vodorovnou linkou je signál, který je vysílán masterem (např. mikrořadičem), pod linkou je signál z paměti 24CXX

Tab 12.2. Oraganizace jednotlivých typů pamětí

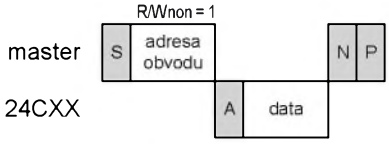
Typ	Vnitřní organizace	Délka adresy pro libovolný přístup
24C01A	16 stránek po 8B	7 bitů
24C02	32 stránek po 8B	8 bitů
24C04	32 stránek po 16B	9 bitů
24C08	64 stránek po 16B	10 bitů
24C16	128 stránek po 16B	11 bitů



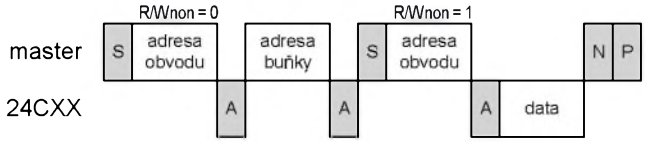
Obr. 12.2. Adresy pamětí



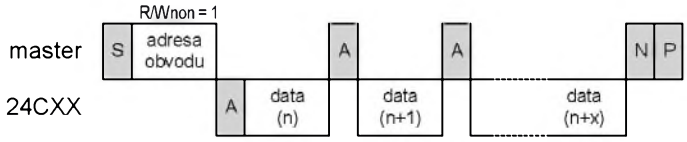
Obr. 12.4. Zápis stránky



Obr. 12.5. Čtení z aktuální adresy



Obr. 12.6. Čtení z libovolné adresy



Obr. 12.7. Sekvenční čtení

Je-li přeneseno více než 8 bajtů dat (u pamětí s kapacitou 1 nebo 2 Kb) nebo 16 bajtů dat (u pamětí s kapacitou 4, 8 nebo16 Kb), přepíšou se příchozí data (která byla uložena v dočasném registru) do paměti.

Čtení z paměti

Čtení je iniciováno stejným způsobem jako zápis, pouze se nastaví R/Wnon = 1. Existují 3 varianty čtení:

• **Čtení z aktuální adresy** (obr. 12.5): Vnitřní čítač adres udržuje poslední použitou adresu při zápisu/čtení zvýšenou o 1.

Tato adresa zůstává platná tak dlouho, dokud je zachováno napájení. To dává možnost přečíst data bez jejich opětovného adresování.

• **Čtení z libovolné adresy** (obr. 12.6): Pro čtení z libovolné adresy se musí vyvolat neúplný („dummy") zápis dat.

Tím se určí adresa buňky, kterou chceme číst. STOP je vynechán a místo něj se vloží START následovaný adresou obvodu (R/Wnon = 1) a lze číst data.

• **Sekvenční čtení** (obr. 12.7): Sekvenční čtení umožňuje číst obsah celé stránky, tedy 8/16 bajtů najednou.

Sekvenční čtení lze vyvolat po čtení z aktuální adresy nebo po čtení z libovolné adresy. Dokud není přenos ukončen (N), lze číst další bajty z dané stránky.

Popis přípravku PROG24

Pro programování pamětí typu 24CXX počítačem PC byl navržen přípravek PROG24. Připojuje se k sériovému portu PC a je z něj přímo napájen (nepotřebuje vnější napájecí zdroj).

Napájecí napětí pro přípravek je získáváno ze všech výstupních linek sériového portu PC. Problém byl hlavně u kladného napětí, proto jsou diody D1, D2 a D9 zapojeny tak, aby kondenzátor C1 byl nabíjen vždy alespoň jednou z linek. Stabilizátor IO1 typu LM317L s malým příkonem zmenšuje a stabilizuje napětí z C1 na velikost přibližně 5 V. Výstupní napětí stabilizátoru je definováno odpory rezistorů R1 a R2. Velikost stabilizovaného napětí není kritická a může pohybovat v širokém rozmezí 2,7 až 5,5 V. Záporné napájecí napětí se sbírá kondenzátorem C2 z linky RTS.

Linky SDA a SCL jsou řízeny signály DTR a TxD. Konverze na úrovně logických signálů provádí diodové omezovače připojené k rezistorům R4 a R3. Výstupní směr SDA je zajištěn komparátorem IO2 s operačním zesilovačem TL061 s malým příkonem. Odporový dělič R5, R6 definuje rozhodovací úroveň komparátoru jako polovinu napájecího napětí. Stavu SDA se čte linkou RLSD.

Součástky přípravku PROG24 jsou umístěné na desce s jednostrannými plošnými spoji.

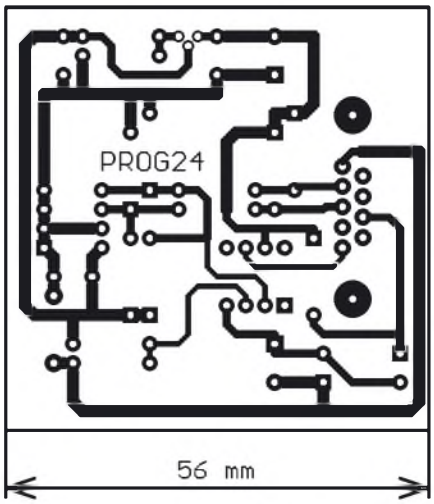
Na obr. 12.9 je obrazec plošných spojů a na obr. 12.10 je rozmístění součástek na desce. Pro vkládání IO3 je použita precizní objímka.

Fotografie přípravku PROG24 je na obálce tohoto časopisu.

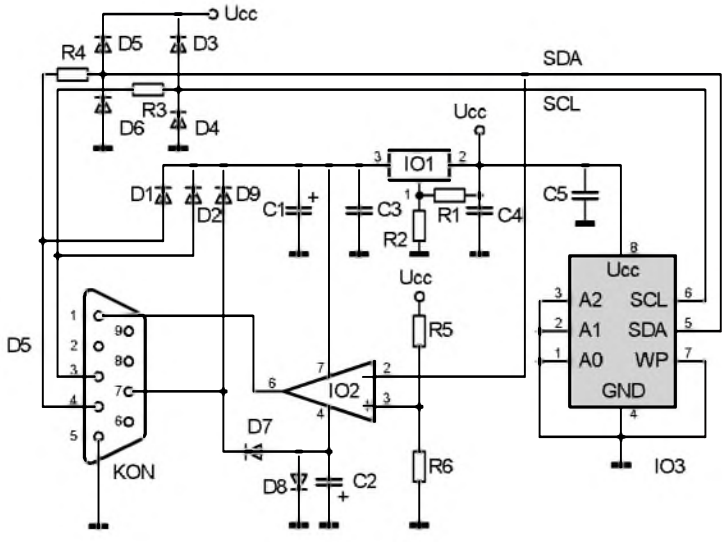
Seznam součástek

(cena asi 70 Kč)

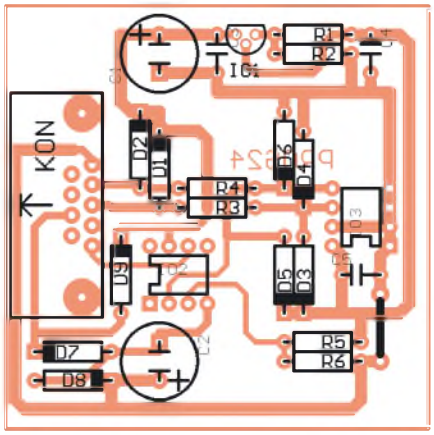
R1	12 kΩ	1 ks
R2 až R4	18 kΩ	3 ks
R5, R6	100 kΩ	2 ks
C1, C2	470 µF/16 V	2 ks
C3 až C5	100 nF	3 ks
D1 až D9	1N4148	9 ks



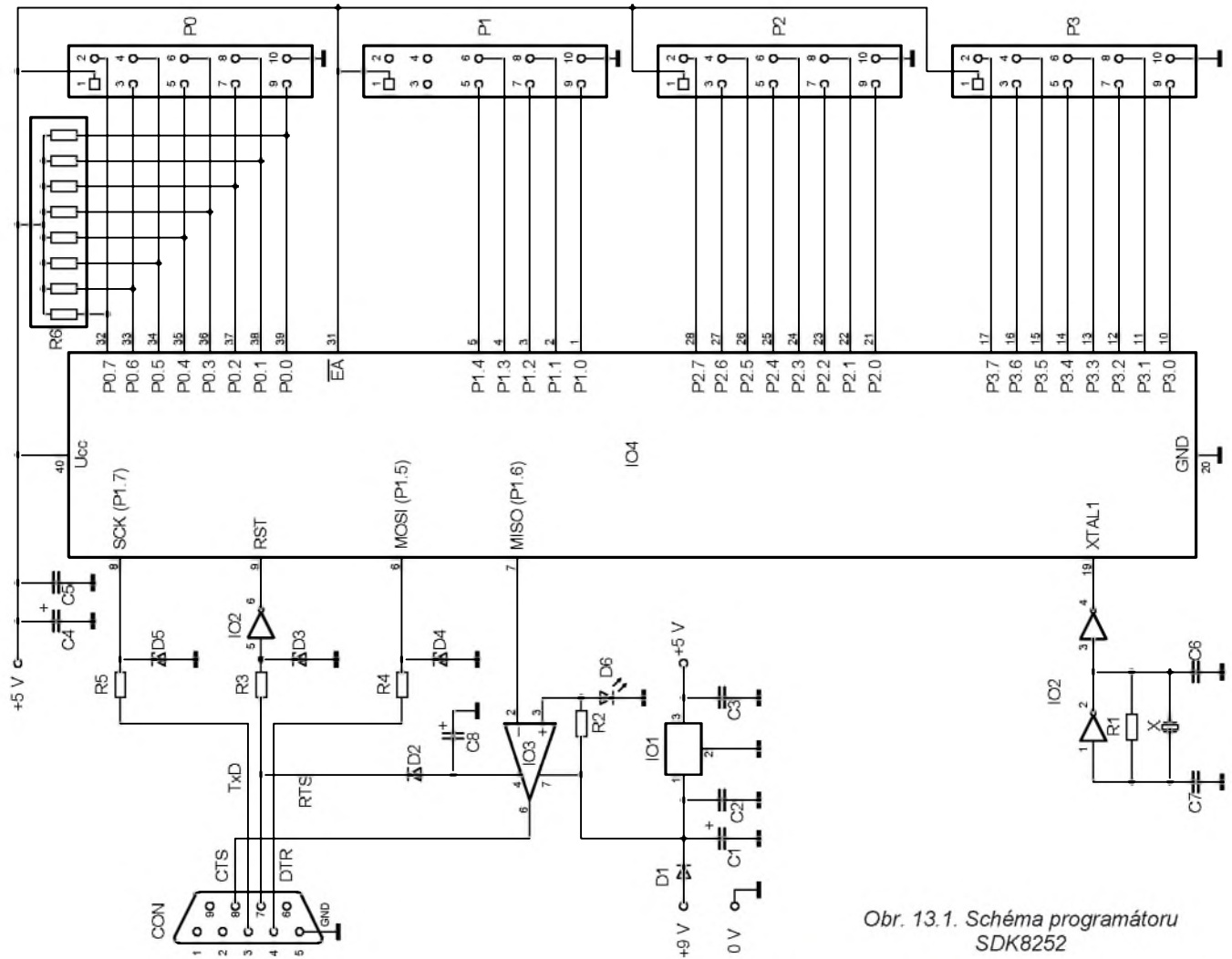
Obr. 12.9. Obrazec plošných spojů přípravku PROG24 (měř.: 1 : 1)



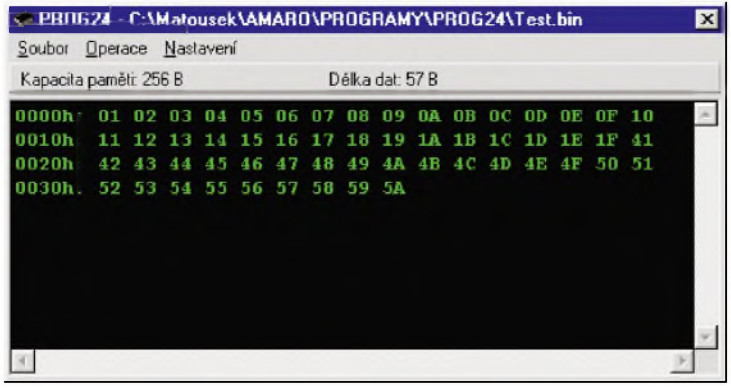
Obr. 12.8. Schéma přípravku PROG24



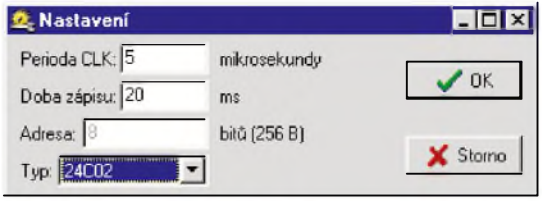
Obr. 12.10. Rozmístění součástek na desce přípravku PROG24



Obr. 13.1. Schéma programátoru SDK8252



Obr. 12.11. Hlavní okno aplikace



Obr. 12.12. Konfigurační dialog

IO1	LM317L	1 ks
IO2	TL061	1 ks
IO3	precizní objímka DIP8	1 ks
KON	CAN 9 Z 90	1 ks

deska s plošnými spoji PROG24

Ovládací program PROG24

Ovládací program poskytuje běžné funkce jako je čtení/zápis do paměti (obr. 12.11).

Volba sériového portu, na který je přípravek připojen, se provádí pomocí inicializačního souboru **PROG24.INI**.

Důležitější je dialog vyvolaný po volbě položky menu **Nastavení|Časování a typ paměti** (obr. 12.12).

Tento dialog obsahuje časovací parametry při práci s pamětí (je možné, že někteří výrobci budou požadovat delší časy přístupů) a vybírá typ programované paměti. Je možné volit:

- Dobu trvání periody SCL v µs.
- Dobu zápisu v ms (postačí 10 ms a méně).
- Typ programované paměti (podle typu se poněkud mění algoritmus programování).

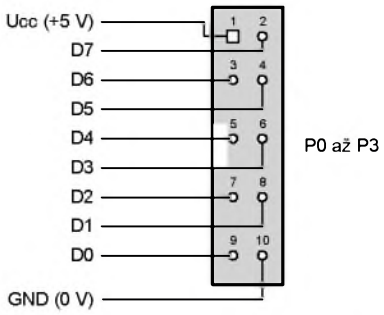
Ovládací program je k dispozici na webových stránkách autora, podrobnosti jsou uvedené v závěru tohoto článku.

13. SDK8252 - programátor a vývojový KIT pro AT89S8252

V této kapitole je popsán programátor mikrořadiče AT89S8252, který lze současně použít i jako vývojový kit.

Mikrořadič AT89S8252 mě zaujal svojí podporou sériového downloadu (sériového zavádění programu). To umožňuje vytvořit programátor a vývojový kit jako jedinou desku.

Programátor SDK8252, jehož schéma je na obr. 13.1, je připojen k sériovému portu počítače PC a je ovládán pomocí přímého řízení sériového portu.



Obr. 13.2. Zapojení konektorů P0 až P3

Popis zapojení programátoru

Napájecí část programátoru je tvořená především stabilizátorem IO1 s ochrannou diodou D1. Na výstupu stabilizátoru je napětí 5 V, kterým se napájí mikrořadič a připojené obvody. Doporučuji vybavit stabilizátor chladičem přiměřených rozměrů. Možná by bylo vhodné umístit stabilizátor mimo desku na hliníkový plech tloušťky 1 až 2 mm, který by byl k desce přišroubován zespodu. Vnější napájecí napětí by nemělo být větší než 9 V, aby nevznikala na stabilizátoru příliš velká výkonová ztráta.

Krystal je zapojen do oscilátoru, který je tvořen dvěma invertory, obsaženými v obvodu IO2. Toto zapojení vždy bez problémů kmitá i s krystaly běžně vypájenými např. z počítačového „šrotu" (krystal o kmitočtu 24 MHz najdeme obvykle na přídavné desce portů).

Řídicí signály pro ovládání vstupů SCK, MOSI a RST jsou získány pomocí Zenerových diod D3 až D5 přímo z linek TxD, DTR a RTS sériového kanálu na konektoru CON. Zenerovy diody omezují původní rozkmit napětí linek RS-232 na rozsah -0,7 až +4,7 V. Z linky RTS se zároveň získává záporné napájecí napětí pro IO3.

Operační zesilovač IO3 typu TL061 je použit jako převodník z úrovní TTL na RS-232. Je to mnohem levnější varianta, než použít klasický obvod MAX232. Obvod TL061 se vyznačuje velmi malými nároky na napájecí proud (jeho klidový odběr je menší než 0,2 mA), a tak je přes diodu D2 a kondenzátor C8 napájen přímo z linky RTS sériového kanálu.

Protože během programování musí být na vstupu RST mikrořadiče IO4 úroveň „log. 1" a přitom musí být na lince RTS záporné napětí pro napájení IO3, je mezi linku RTS a vstup RST zapojen invertor z IO2.

Dioda D6 indikuje přítomnost kladného napájecího napětí a zároveň definuje rozhodovací úroveň pro převodník úrovní IO3.

Velmi důležitou úlohu mají kondenzátory C4 a C5. Blokují napájecí napětí mikrořadiče a připojených přípravků. Bez těchto kondenzátorů docházelo velmi často k chybám zápisu a byly i velké problémy s připojením přípravků (proudový náraz při zapnutí vedl k chybnému rozběhu mikrořadiče).

Porty P0 až P3 jsou vyvedeny na konektory označené P0 až P3. Pro port P0 jsou použity vnější zdvihací (pull-up) rezistory R6, protože je tento port nemá integrované. Bity P1.5, P1.6 a P1.7 nejsou na konektor P1 vyvedeny, neboť se používají pro sériové zavádění programu (download) a mohly by nastávat nepříjemné kolize. Zapojení vývodů konektorů P0 až P3 je obr. 13.2.

Pokud by bylo třeba využít integrované rozhraní SPI pro ovládání připojených obvodů, musí se použít jiná deska.

Propojovací šňůru mezi počítač PC a programátor lze realizovat pomocí devítižilového plochého kabelu o délce asi 1,5 m a dvou samořezných konektorů CANON. Vodiče nejsou nijak kříženy. Postačí i pětižilový kabel pro signály TxD, DTR, GND, RTS, CTS, výroba takového kabelu je však pracnější.

Všechny součástky programátoru SDK8252 jsou umístěné na desce s plošnými spoji. Deska je navržena tak, aby mohla být velmi snadno vyrobena i v amatérských podmínkách. Proto jsou spoje jednostranné a druhá strana spojů je nahrazena drátovými propojkami na straně součástek.

Na obr. 13.3 je obrazec plošných spojů a na obr. 13.4 je rozmístění součástek na desce.

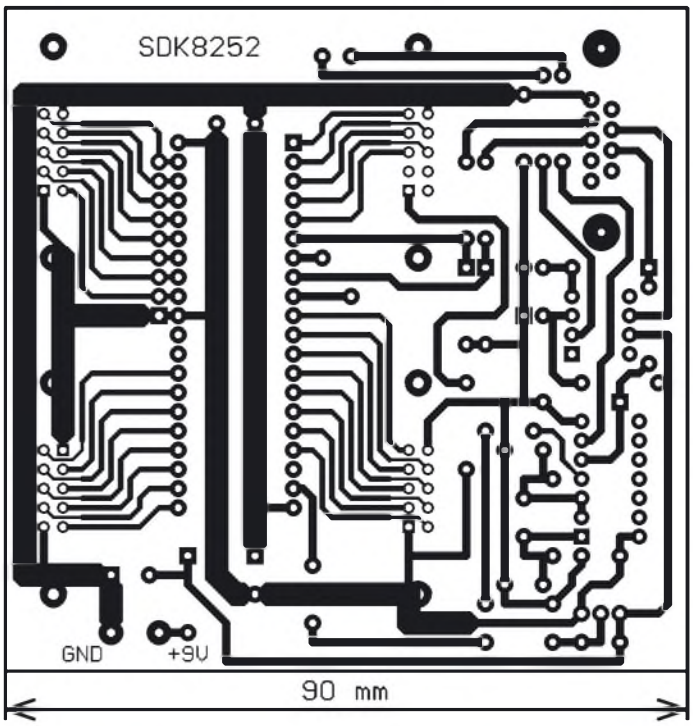
Pro IO2 až IO4 je vhodné použít objímky, aby je bylo možné přemístit i do dalších konstrukcí. Stabilizátor IO1 je opatřen chladičem.

Fotografie programátoru SDK8252 je na obálce tohoto časopisu.

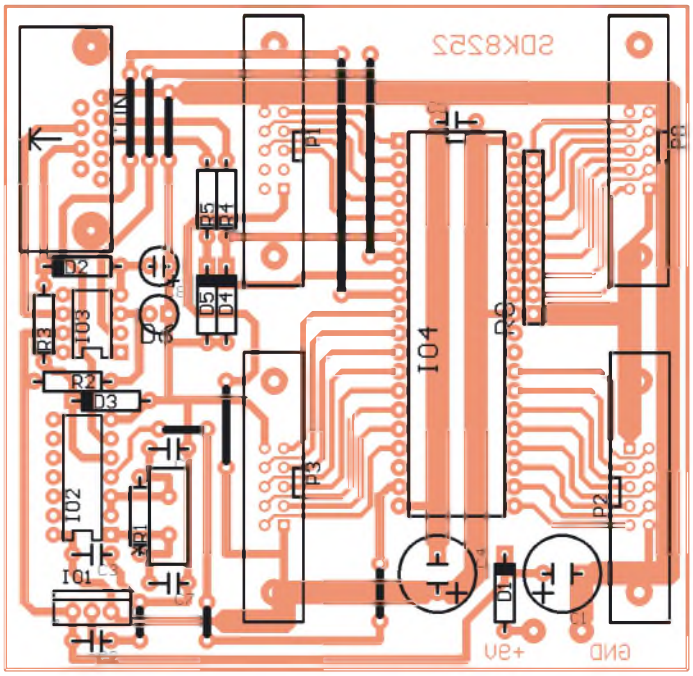
Seznam součástek

(cena asi 200 Kč bez mikrořadiče)

R1	1 MΩ	1 ks
R2	680 Ω	1 ks
R3	10 kΩ	1 ks



Obr. 13.3. Obrazec plošných spojů programátoru SDK8252 (měř.: 1 : 1)



Obr. 13.4. Rozmístění součástek na desce programátoru SDK8252

R4, R5	1,5 kΩ	2 ks
R6	RR 8x10 kΩ, odporová síť	1 ks
C1, C4	470 µF/16 V	2 ks
C2, C3, C5	100 nF	3 ks
C6, C7	33 pF	2 ks
C8	47 µF/16 V	1 ks
D1	1N4001	1 ks
D2	1N4148	9 ks
D3 až D5	BZX83V004.7, Zenerova dioda 4,7 V	3 ks
D6	LED, R, 5 mm	1 ks
IO1	7805	1 ks
IO2	74HCT04 (74HC04)	1 ks
IO3	TL061	1 ks
IO4	AT89S252-24PI	1 ks
X	krystal 24 MHz	1 ks
CON	CAN 9 Z 90	1 ks
P0 až P3	PSL10	4 ks
deska s plošnými spoji SDK8252		

Testovací a oživovací program SDK8252.EXE

Program byl zvlášť vytvořen pro test a oživení programátoru. Dokáže zavést do mikrořadiče AT89S8252 čtyři standardní programy, umí mikrořadič vynulovat (reset) a čte zpětně obsah paměti Flash (pokud nejsou aplikovány zámky). Také vydatně pomůže při oživení desky programátoru.

Testovací program je velmi snadno ovladatelný, zmíním se pouze o několika drobnostech:

INI soubor

Pro inicalizaci programu se používá inicializační soubor AT89S8252.INI, který musí být umístěn v adresáři, do kterého jste překopírovali SDK8252.EXE.

Význam inicializačního souboru se názorně osvětlí příkladem jeho obsahu:

```
[PORT]          ;sekce nastavení parametrů
                ;komunikace
Port=2          ;číslo sériového kanálu
                ;(COM1=1, COM2=2)
Zpozdeni=100    ;zpoždění při komunikaci
                ;v mikrosekundách(100 ob-
                ;vykle stačí)

[OKNO]          ;rozměry a umístění okna
                ;při posledním spuštění
X=78
Y=147
W=570
H=259

[NASTAVENI]
Verifikace=1    ;verifikace po zápisu
                ;(1-ano, 0-ne)
Zamek=4         ;režim zámku (1 až 4)
```

Nejzávažnější jsou klíče v sekci **PORT**. Klíč **Port** udává číslo sériového portu PC, na který je připojen programátor (SDK8252.EXE neumožňuje měnit komunikační kanál po svém spuštění). Klíč **Zpozdeni** udává čekací dobu mezi dvěma přístupy na sériový port v mikrosekundách. Obvyklé nastavení 100 postačuje (programátor pracoval i při zpoždění 70 µs). Pokud by nastávaly problémy při komunikaci (v případě rychlejších počítačů se to může stát), nastavte delší prodlevu.

Ostatní parametry lze nastavit přes menu. Do inicializačního souboru se ukládají proto, aby po novém startu dostal uživatel předchozí nastavení.

Ovládání aplikace

Ovládání této aplikace je snadné, proto jej není nutno obšírně popisovat. Zde je pouze krátký popis (viz obr. 13.5 až obr. 13.7):

• Položka menu **Soubor|Přímé řízení** vyvolá dialog podle obr. 13.6. Tento dialog slouží pro oživení samotné desky programátoru s mikrořadičem vyjmutým z objímky Pomocí zaškrtávacích políček (checkboxů) nastavujeme logické úrovně signálů RST, MOSI a SCK a voltmetrem kontrolujeme napětí na odpovídajících kontaktech objímky pro mikrořadič. Úroveň „log. 0" nesmí být zápornější než -0,7 V, úroveň „log. 1" by měla být větší než 4,5 V, avšak musí být menší než 5,7 V.

• Tlačítko **Smazání** smaže paměť programu (Flash) i paměť dat (E^2PROM). Tato operace je nutná před novým programováním v případě, že jsme použili zámek v režimu 2. Pokud používáme režim zámku 1 (zámek není použit), nemusí být paměť před programováním smazána.

• Tlačítkem **Reset** se nuluje (resetuje) mikrořadič. Přípravek SDK8252 nemá nulovací tlačítko (zdálo se mi zbytečné), k resetu je třeba spustit program SDK8252.EXE a kliknout na tomto tlačítku.

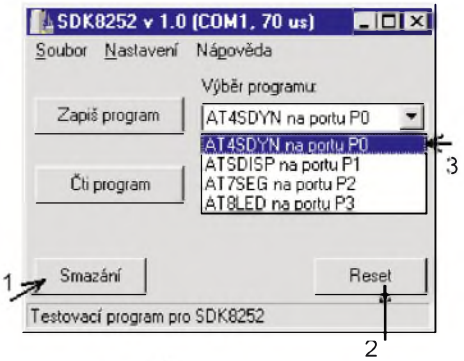
• Rozbalitelný seznam **Výběr programu** umožní zvolit, který z programů chceme nahrát do mikrořadiče. Jedná se o programy ovládající prostřednictvím portů P0 až P3 přípravky AT4DYN, ATSDISP, AT7SEG a AT8LED, které byly popsány v [4].

• Tlačítko **Zapiš program** zapíše program vybraný pomocí **Výběr programu** do mikrořadiče. Je-li zatržena položka menu **Nastavení|Zápis s verifikací**, provede se po zápisu verifikace (zpětné čtení), kterým je ověřena úspěšnost zápisu.

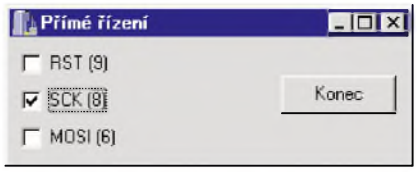
• Tlačítko **Čti program** přečte počet bajtů udaných rozbalovacím seznamem **Počet čtených bajtů** do diskového souboru READ.BIN (tento soubor vznikne v adresáři, ze kterého byl program SDK8252.EXE spuštěn).

• Položkou menu **Nastavení|Zápis s verifikací** se volí, zda se po zápisu vykoná kontrolní čtení (zaškrtnuto) nebo ne (není zaškrtnuto),

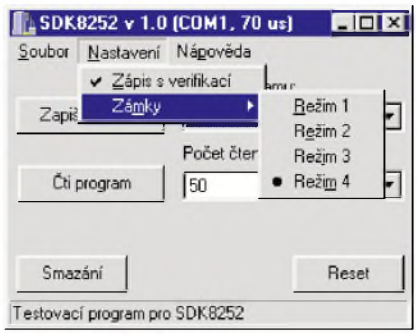
• Podpoložky menu **Nastavení|Zámky** umožňují volit režim zámku Flash. Např. režim 1 značí nepoužití zámku (obvod lze libovolně číst a přeprogramovat bez mazání) a režim 4 značí použití všech bitů zámku (obvod nelze zpětně číst ani přeprogramovat).



Obr. 13.5. Výběr souboru (programu), který má být zaveden do mikrořadiče. 1 - vymaže paměť Flash a EEPROM, 2 - nulování mikrořadiče, 3 - výběr programu pro zavedení (download) do mikrořadiče



Obr. 13.6. Dialog pro oživení desky programátoru



Obr. 13.7. Výběr zámku

Oživení programátoru SDK8252

Přestože je stavba programátoru poměrně jednoduchá a při pečlivé montáži pracuje napoprvé, je důležité uvést postup oživení pro případ, že nastanou problémy.

• Pečlivě osaďte desku, dbejte zejména na správnou polaritu všech diod a kondenzátorů.

• Zatím neosazujte mikrořadič IO4 (pro IO4 je nejvhodnější je použít čtyřicetivývodovou precizní objímku). Připojte zdroj 9 V, oscilátor musí kmitat a na kontaktu *19* objímky pro IO4 musí být kmitočet 24 MHz.

• Připojte programátor ke zvolenému sériovému portu PC a zkontrolujte, zda je tento port zapsán v inicializačním souboru.

• Spuťte na PC program SDK8252. Vyberte položku menu **Soubor|Přímé řízení**. Zaškrtávejte v dialogu **Přímé řízení** jednotlivá políčka a voltmetrem sledujte napětí na kontaktech *6*, *8* a *9* objímky pro IO4. Je-li políčko prázdné, jedná se o stav „log. 0" a napětí na kontaktu objímky musí být v rozsahu -0,7 až +0,5 V. Je-li políčko zaškrtnuté, jed-

ná se o stav „log. 1“ a napětí musí být v rozsahu 4,5 až 5,7 V. Pro zaškrtnuté políčko RST navíc platí, že napětí na vývodu *4* obvodu IO3 musí být alespoň -7 V (čím je toto záporné napětí větší, tím lépe; u některých počítačů může být až -12 V). Pokud nejsou tyto podmínky splněny, nebude patrně fungovat sériové zavádění programu do mikrořadiče (download)!!! Problémy se zaváděním mohou nastat u některých notebooků (zkoušel jsem programátor s notebookem Armada 1500 c a zde problémy nebyly).

• Vypněte zdroj, vložte mikrořadič do objímky, zapněte zdroj a zkuste zavést nějaký program. Verifikace musí být aktivována a zámek vypnut (režim 1).

• První zápis po připojení zdroje může selhat. Pokud tedy po prvním zápisu dostanete chybové hlášení, zkuste operaci opakovat. Obvykle je pak již vše v pořádku.

• Nedaří-li se zápis, zkuste zmenšit odpor rezistorů R3 až R5.

Vývojové prostředí AT8252

Vývojové prostředí AT8252 (viz obr. 13.8) umožňuje komplexní práci s mikrořadičem AT89S8252, vloženým do programátoru (vývojového kitu) SDK8252.

Hlavními znaky tohoto prostředí jsou:

• Snadný překlad programů (integrovaný překladač ASM51, hledání chyb na úrovni zdrojového textu).

• Snadné zavedení (download) programu do mikrořadiče bez opuštění vývojového prostředí (po úspěšném překladu je možno program zavést výběrem položky menu).

• Download již přeložených binárních souborů.

• Možnost programovat E^{2}PROM.

• Možnost nastavit zámky chránící obsah mikrořadiče proti zpětnému čtení.

• Možnost smazat obsah Flash a E^{2}PROM a vynulovat (resetovat) mikrořadič.

Inicializační soubor:

• Má stejný název (AT89S8252.INI) a položky jako inicializační soubor pro program SDK8252.EXE.

• Oba programy mohou tento soubor sdílet.

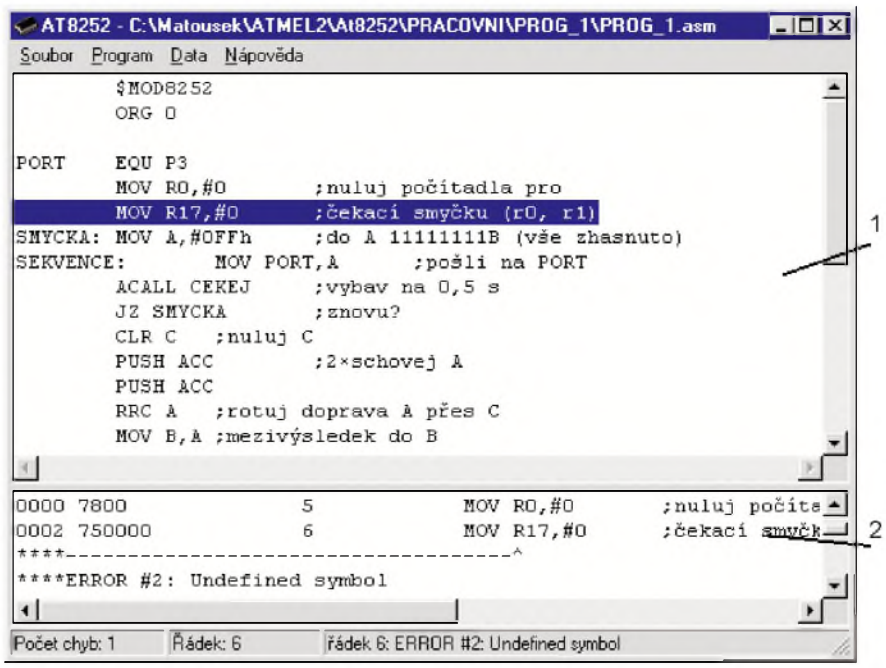
Práce se soubory:

• **Soubor|Nový Ctrl + N** - vytvoří prázdný zdrojový soubor, který obsahuje pouze direktivu $MOD8252,

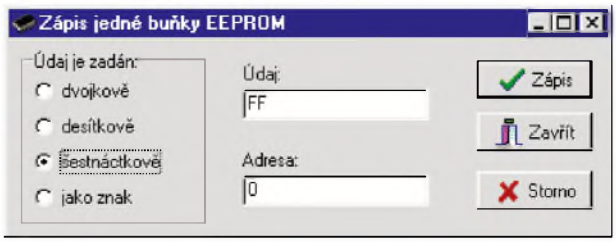
• **Soubor|Otevřít... Ctrl + O** - zobrazí dialog pro otevření zdrojových souborů s příponou ASM,

• **Soubor|Uložit Ctrl + S** - uloží dříve vytvořený soubor,

• **Soubor|Uložit jako...** - uloží aktuální soubor pod novým jménem,



Obr. 13.8. Vývojové prostředí AT8252. 1 - editor, 2 - hlášení chyb překladu



Obr. 13.9. Dialog pro programování jedné buňky E^{2}PROM

• **Soubor|Konec Alt + X (Alt + F4)** - ukončí vývojové prostředí.

Editace textu:

• **Editace|Zpět Ctrl + Z** - vratí editaci textu o jednu operaci zpět,

• **Editace|Vyjmout Shift + Del** - vyjme označený text do schránky,

• **Editace|Kopírovat Ctrl + Insert** - zkopíruje označený text do schránky,

• **Editace|Vložit Shift + Insert** - vloží text ze schránky do editoru,

• **Editace|Vybrat vše Ctrl + A** - označí celý text,

• **Editace|Hledat... Ctrl + F** - zobrazí dialog pro hledání slova.

Ovládání SDK8252:

• **Program|Přeložit Ctrl + F9** - přeloží **program (generuje soubory HEX a** BIN),

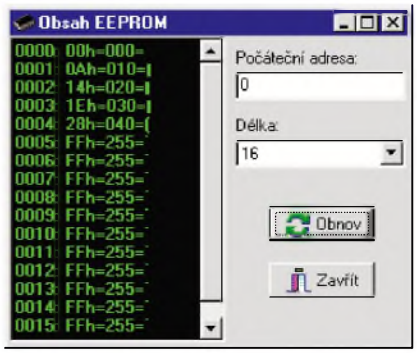
• **Program|Hledej chybu F3** - najde řádek s následující chybou přímo ve zdrojovém textu (viz obr. 13.8),

• **Program|Reset Ctrl + F2** - vynuluje (resetuje) mikrořadič,

• **Program|Vymaž** - vymaže obsah Flash i E^{2}PROM mikrořadiče,

• **Program|Zapsat program F9** - provede download dříve přeloženého binárního souboru,

• **Program|Zapsat program ze souboru...** - zobrazí dialog pro výběr binárního souboru a provede jeho download,



Obr. 13.10. Dialog pro prohlížení obsahu E^{2}PROM

• **Program|Verifikace** - zaškrtnuto značí, že po downloadu je provedena verifikace (ověření úspěšnosti zápisu),

• **Program|Zámek** - volba zámku, který se aktivuje po zápisu.

Programování E^{2}PROM:

• **Data|Zapsat jednu buňku...** - zobrazí dialog pro volbu hodnoty (je možno použít dvojkový, desítkový nebo šestnáctkový zápis čísla nebo zadat znak), kterou chceme zapsat na vybranou buňku v E^{2}PROM (viz obr. 13.9),

• **Data|Dump...** - zobrazí dialog pro prohlížení obsahu E^{2}PROM (viz obr. 13.10).

Nápověda:

• **Nápověda|O aplikaci...** - zobrazí krátkou informaci o aplikaci AT8252.

14. SDK2313 - programátor a vývojový KIT pro AT90S2313

V poslední době se i u nás začínají prosazovat mikrokontroléry s redukovaným instrukčním souborem. Typickým příkladem z dílny firmy ATMEL jsou procesory AVR.

Podrobný popis vybraných zástupců řady AVR je uveden v [9].

Popis programátoru SDK2313

Schéma programátoru je na obr. 14.1. Napájecí část zapojení je tvořena především stabilizátorem IO2 a ochrannou diodou D6. Na výstupu stabilizátoru je napětí 5 V, kterým se napájí mikrokontrolér a připojené obvody. Doporučuji vybavit IO2 chladičem přiměřených rozměrů (chladič by se měl vejít na desku, je pro něj vynechán dostatek místa). Vnější napájecí napětí by nemělo být větší než 9 V, aby se stabilizátor zbytečně nezahříval.

Krystal X je připojen přímo k mikrokontroléru IO3. Dále jsou připojeny obvyklé pomocné kondenzátory C1 a C2.

Řídicí signály pro ovládání vstupů SCK, MOSI a RESETnon jsou vyvedeny přímo z linek sériového kanálu (konektor CON) TxD, DTR a RTS pomocí Zenerovy diody D1 (pro signál RESETnon) nebo tranzistorů T1 a T2 (pro signály SCK a MOSI). Linka RTS je zároveň použita jako zdroj záporného napájecího napětí pro IO1.

Operační zesilovač (OZ) IO1 (TL061) je zapojen jako komparátor a slouží k převodu úrovně signálu MOSI z TTL na RS-232. Je to mnohem levnější řešení, než použít klasický obvod MAX232. OZ TL061 se vyznačuje velmi malým napájecím proudem (jeho klidový odběr je menší než 0,2 mA), a tak je přes diodu D4 a kondenzátor C3 napájen přímo z linky RTS sériového kanálu.

Dioda D5 indikuje přítomnost kladného napájecího napětí a zároveň definuje rozhodovací úroveň pro komparátor s IO1. Dioda D7 chrání IO1 v době, kdy na lince RTS není záporné napětí.

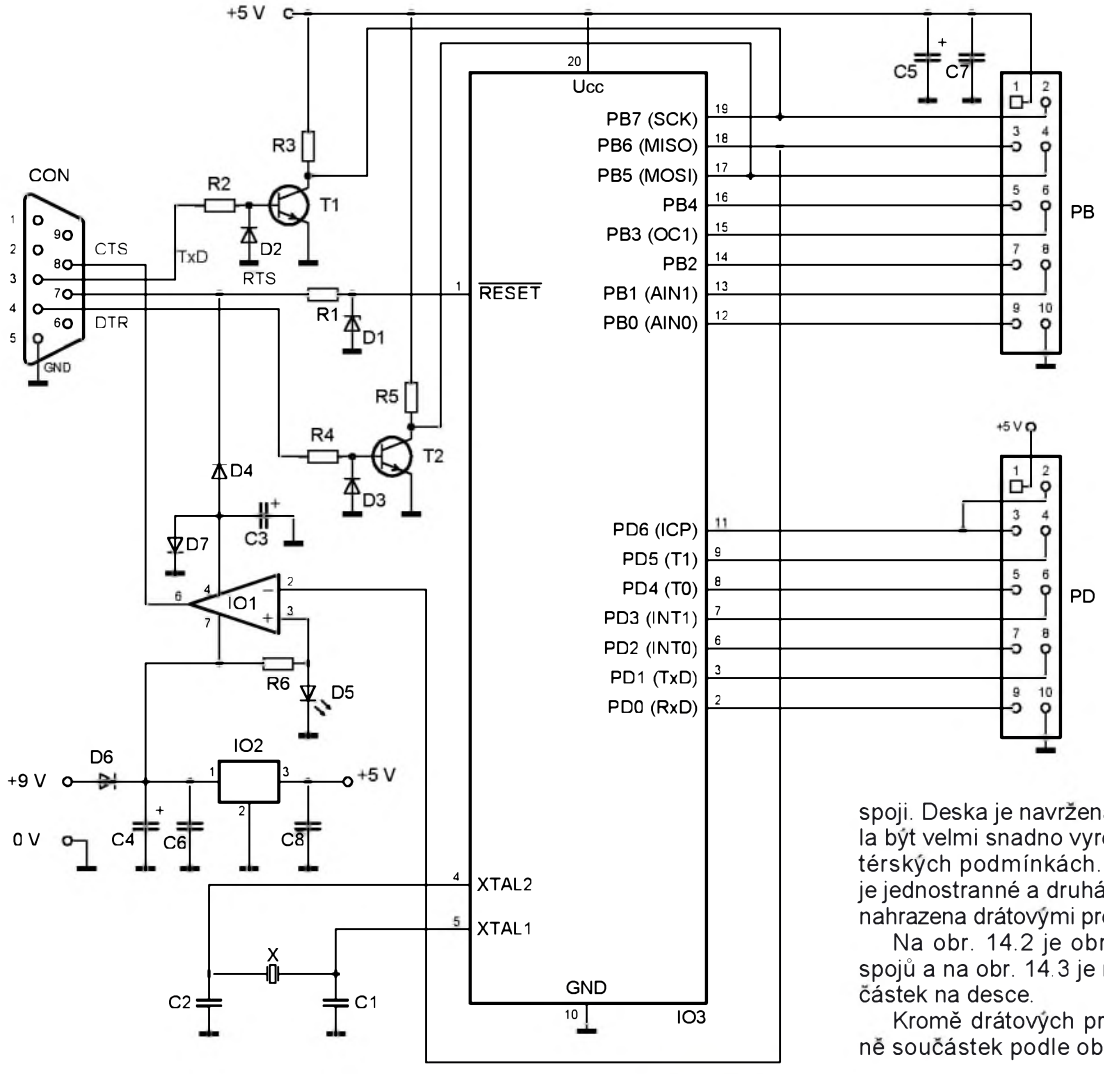
Velmi důležitou úlohu mají kondenzátory C5 a C7. Blokují napájecí napětí mikrokontroléru a připojených přípravků. Bez těchto kondenzátorů by mohly nastávat chyby při zavádění programu do mikrokontroléru a byly by i velké problémy s připojením přípravků (proudový náraz při zapnutí vedl k chybnému rozběhu mikrokontroléru).

Porty PB a PD mikrokontroléru jsou vyvedeny na konektory PB a PD. Určité problémy může způsobovat skutečnost, že linky PB7, PB6 a PB5 se používají při zavádění programu do mikrokontroléru. Tyto linky jsou totiž na konektor PB také vyvedené.

Problém není ani tak v tom, že by se mikrokontrolér a obvody připojené k linkám mohly poškodit (úroveň „log. 1" je dosahována zdvíhacími (pull-up) rezistory R3 a R5, úroveň „log. 0" odpovídá sepnutému tranzistoru T1 nebo T2), může však nastat kolize signálů a zavádění programu nebude úspěšné. Proto je třeba dát pozor na to, aby během zavádění programu nebyly k těmto třem linkám připojeny žádné výstupy přípravků (periferií). Většinou to není problém, tam kde to vadí, musí být přípravek připojen ke konektoru PD.

Zapojení konektorů PB a PD (PSL 10) je stejné jako u předchozího programátoru a je zřejmé z obr. 13.2. Jediným rozdílem je, že vývod PD6 z IO3 je připojen na kontakty *3* a *2* konektoru PD, které odpovídají signálům D6 a D7. Vývod PD6 z IO3 je využit pro oba signály D6 i D7 kvůli tomu, že port PD mikrokontroléru AT90S2313 nemá vývod PD7.

Součástky programátoru SDK2313 jsou umístěné na desce s plošnými

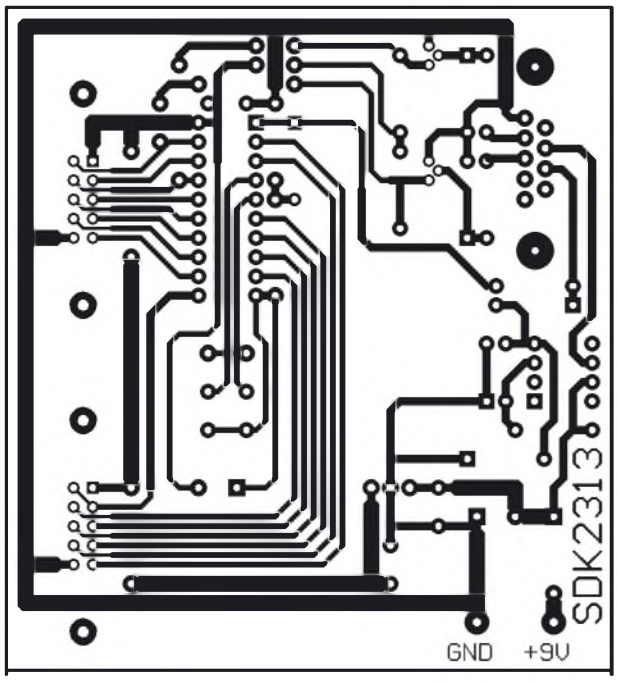


Obr. 14.1. Schéma programátoru SDK2313

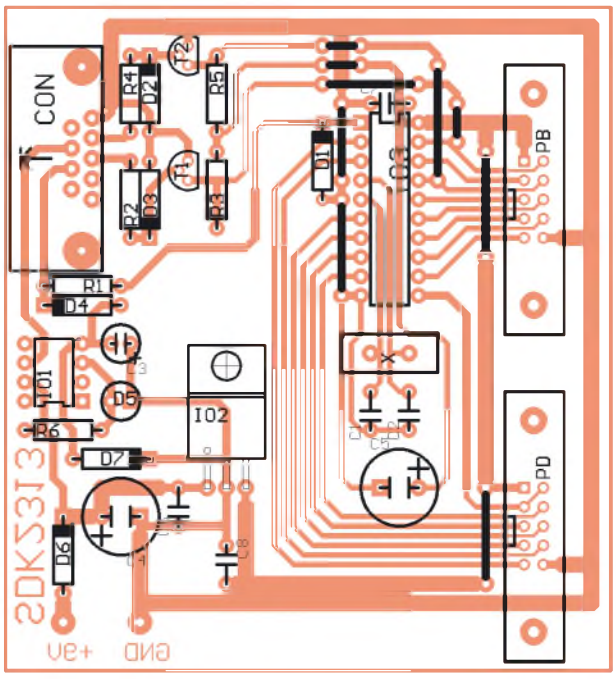
spoji. Deska je navržena tak, aby mohla být velmi snadno vyrobena i v amatérských podmínkách. Proto jsou spoje jednostranné a druhá strana spojů je nahrazena drátovými propojkami.

Na obr. 14.2 je obrazec plošných spojů a na obr. 14.3 je rozmístění součástek na desce.

Kromě drátových propojek na straně součástek podle obr. 14.3 je jedna



Obr. 14.2. Obrazec plošných spojů programátoru SDK2313 (měř.: 1 : 1, kratší vodorovný rozměr desky je 80 mm)



Obr. 14.3. Rozmístění součástek na desce programátoru SDK2313

Obr. 14.4. Ovládací program pro SDK2313

propojka vedená izolovaným drátem i na straně pájení (spojů) a propojuje signálem MISO vývod *2* IO1 s vývodem *18* IO3 (tato propojka není na obr. 14.3 znázorněna a nesmíme na ni zapomenout!). Pro IO1 a IO3 je vhodné použít objímky, aby je bylo možné přemístit i do dalších konstrukcí (a také kvůli oživování). Stabilizátor IO2 je opatřen chladičem.

Fotografie programátoru SDK2313 je na obálce tohoto časopisu.

Seznam součástek

(cena asi 200 Kč bez procesoru)

R1	2,2 kΩ	1 ks
R3, R5	10 kΩ	2 ks
R2, R4	1,5 kΩ	2 ks
R6	680 Ω	1 ks
C1, C2	33 pF	2 ks
C3	47 µF/16 V	1 ks
C4, C5	470 µF/16 V	2 ks
C6 až C8	100 nF	3 ks
D1	BZX83V004.7, Zenerova dioda 4,7 V	1 ks
D2 až D4, D7	1N4148	4 ks
D5	LED, R, 5 mm	1 ks
D6	1N4001	1 ks
T1, T2	BC548	2 ks
IO1	TL061	1 ks
IO2	7805	1 ks
IO3	AT90CS2313-10PI	1 ks
X	krystal 10 MHz	1 ks
CON	CAN 9 Z 90	1 ks
PB, PD	PSL10	2 ks
deska s plošnými spoji SDK2313		

Stručný popis ovládacího programu

Ovládací program **SDKAVR** je vytvořen značně univerzálně a podporuje programování dalších čtyř typů mikrořadičů AVR, jako jsou např. AT90S2313, AT90S2343, AT90S4433, AT90S8515 a AT90S8535.

Lze programovat paměť programu (**Flash**) nebo paměť dat (**E^{2}PROM**), a také lze volit zámky pro ochranu obsahu mikrořadiče proti neoprávněnému čtení (**Volby**). Port PC pro připojení kitu se vybírá pomocí inicializačního souboru SDKAVR.INI.

Program SDKAVR je k dispozici na webových stránkách autora, podrobnější informace jsou v závěru.

15. Přípravek AT8VV - osmibitová vstupně/výstupní deska

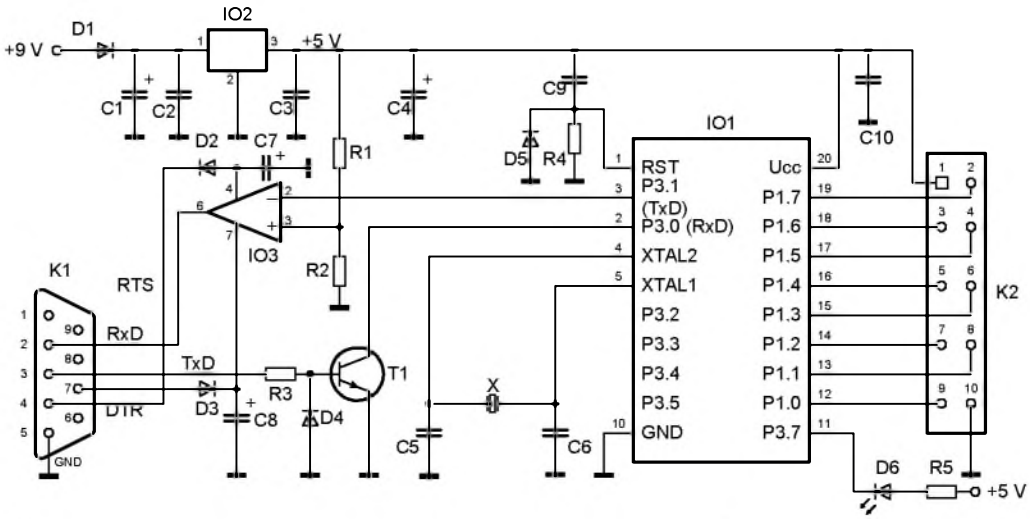
Popisovaný přípravek AT8VV je nejjednodušší aplikací mikrokontroléru AT89C2051 ovládaného sériovou linkou, jehož port P1 je využit jako osmibitový vstup/výstup. Schéma přípravku je na obr. 15.1.

V napájecí části nalezneme obvyklou ochrannou diodu D1 (zabraňuje zničení přípravku při přepólování vnějšího napájecího napětí), filtrační kondenzátor C1 (vyhlazuje napájecí napětí) a stabilizátor IO2 typu 7805. Pokud máte k dispozici stabilizovaný pětivoltový zdroj, lze všechny tyto součástky vynechat a přivést vnější napájecí napětí +5 V přímo na vývod *3* nezapojeného stabilizátoru.

Vnitřní hodinový oscilátor mikrokontroléru IO1 je doplněn vnějším krystalem X a kondenzátory C5 a C6.

Port P1 mikrokontroléru je vyveden na konektor K2 (PSL 10), na který lze připojovat různé periferie. Z portu P3 je použit pouze vývod P3.7, který budí LED D6. LED D6 je zapojena tak, že svítí při úrovni „log. 0" na vývodu P3.7. Připomeňme, že ve stavu „log. 0" může téci do jednotlivých vývodů portů P1 a P3 proud až 20 mA.

Zajímavější je řešení nulovacího obvodu. Při zapnutí napájecího zdroje se derivačním článkem *C9*, *R4* vytvoří krátký kladný impuls na vývod RST mikrokontroléru, čímž se mikrokontrolér vynuluje. Po nabití kondenzátoru C9 je napětí na vývodu RST rovné nule, takže se mikrokontrolér může rozběhnout. Po vypnutí zdroje se kondenzátor C9 rychle vybije přes diodu D5, čímž je vývod RST chráněn proti zápornému přepětí.



Obr. 15.1. Schéma přípravku AT8VV

Nejdůležitější je připojení mikrokontroléru k sériovému portu PC.

V této konstrukci není k převodu z TTL na RS232C použit obvyklý obvod MAX232, ale komparátor s operačním zesilovačem (OZ) IO3 typu TL061 s malým příkonem, který je napájen přímo z portu PC. K napájení slouží linky RTS (kladný pól) a DTR (záporný pól).

K převodu z RS232C do TTL slouží tranzistor T1, který je doplněn rezistorem R3, určujícím proud báze, a diodou D4, chránící přechod báze-emitor při závěrné polarizaci.

Oba převodníky úrovní jsou invertující, což povaha linek sériového kanálu vyžaduje.

Součástky přípravku AT8VV jsou umístěné na desce s jednostrannými plošnými spoji. Na obr. 15.2 je obrazec plošných spojů a na obr. 15.3 je rozmístění součástek na desce.

Pro IO1 a IO3 je vhodné použít objímky.

Fotografie přípravku AT8VV je na obálce tohoto časopisu.

Seznam součástek

(cena asi 50 Kč bez mikrokontroléru)

R1 až R4	5,6 kΩ	4 ks
R5	330 Ω	1 ks
C1, C4	470 µF/16 V	2 ks
C2, C3, C9, C10	100 nF	4 ks
C5, C6	33 pF	2 ks
C7, C8	47 µF/16 V	2 ks
D1	1N4001	1 ks
D2 až D5	1N4148	4 ks
D6	LED, R, 5 mm	1 ks
T1	BC548	1 ks
IO1	AT89C2051	1 ks
IO2	7805	1 ks
IO3	TL061	1 ks
X	krystal 11,059 MHz	1 ks
K1	CAN 9 Z 90	1 ks
K2	PSL10	1 ks

deska s plošnými spoji AT8VV

Program pro AT89C2051

Než zapíšeme programy pro mikrokontrolér a PC, bude vhodné rozmyslet si, jakým způsobem bude komunikace mezi oběma zařízeními probíhat.

Při přenosu dat je třeba rozlišit, zda jsou data vstupní či výstupní. Takže první bajt přenosu vyslaný z počítače PC bude obsahovat buď hodnotu (dekadicky) 1 (čtení) nebo 2 (zápis).

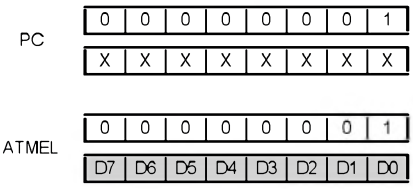
Druhý bajt vysílaný z PC nese data, která se mají zapsat do mikrokontroléru. Aby se komunikace zjednodušila a formát měl délku dvou bajtů při zápisu i čtení, posílá se druhý bajt i při čtení. Tento bajt je však mikrokontrolérem ignorován (může mít libovolnou hodnotu).

Mikrokontrolér (ATMEL) vrací do PC hodnotu prvního bajtu beze změny (tak se pozná, že bajt byl správně přijat) a připojí druhý bajt.

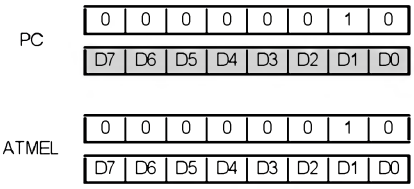
Při čtení je to bajt dat z mikrokontroléru, při zápisu je to kopie dat posílaných z PC. Vracení prvního bajtu zpět umožňuje rozpoznat úspěšné připojení přípravku.

Přenos dat ilustrují obr. 15.4a a obr. 15.4b.

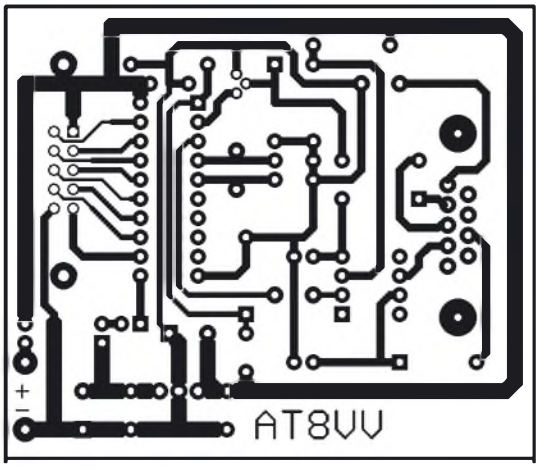
Při realizaci programu pro mikrokontrolér si předně musíme zvolit přenosovou rychlost a formát. Obvyklá přenosová rychlost je **9600 Bd**, formát zvolíme **osmibitový** s **jedním stop-bitem** a **bez parity** (opět obvyklé hodnoty). Přenos bude zabezpečen kontrolním čtení prvního bajtu přijatého zpět. Mikrokontrolér první bajt vždy zopakuje.



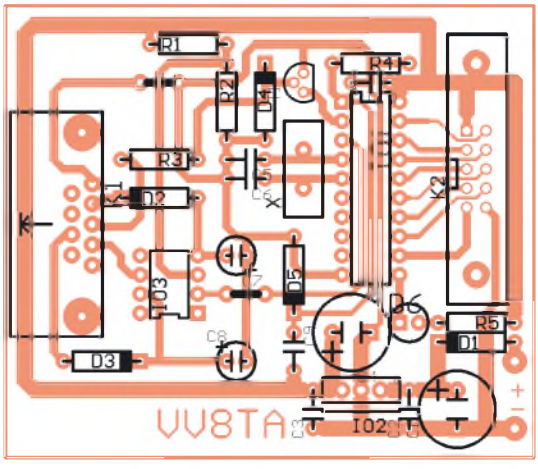
Obr. 15.4a. Formát přenosu dat při čtení z mikrokontroléru ATMEL



Obr. 15.4b. Formát přenosu dat při zápisu do mikrokontroléru ATMEL



Obr. 15.2. Obrazec plošných spojů přípravku AT8VV (měř.: 1 : 1, delší rozměr desky je 70 mm)



Obr. 15.3. Rozmístění součástek na desce přípravku AT8VV

Stačí tedy porovnat prvně poslaný bajt z PC s prvně přijatým bajtem, který zopakoval mikrokontrolér.

V realizovaném programu pro mikrokontrolér nalezneme dva vektory **RESET** (vynulování = reset procesoru) a **SERIAL** (příjem nebo odvysílání bajtu sériovým kanálem).

Při resetu je třeba nastavit přenosovou rychlost (**TH1**, **SMOD**, **TMOD**) a formát přenosu (**SCON**). Dále musíme aktivovat časovač 1 (**TR1**), který je použit jako synchronizační zdroj sériového přenosu a povolit přerušení od sériového kanálu. Nakonec se instrukcí **SJMP $** program „zacyklí", bude tedy čekat na přerušení od sériového kanálu (příjem/odvysílání znaku).

Rutina **SERIAL** obsluhuje přerušení od sériového kanálu. Nejdříve se testuje bit **TI**, abychom zjistili, zda se jedná o příjem nebo vysílání. Je-li **TI = 0**, jedná se o příjem (jinak jde o vysílání).

Dále se musí rozlišit režim přenosu (čtení nebo zápis). Výchozí hodnota proměnné **REZIM** je **0**, což značí, že režim zatím není určen. Je-li **REZIM = 0**, určuje přijatý bajt režim přenosu. Hodnota se tedy uloží do **REZIM** a odvysílá zpět. Pokud je **REZIM = CTENI**, přijatá hodnota se „zahodí" a zpět se odešle načtený stav P1. Pokud je **REZIM = ZAPIS**, je přijatá hodnota zapsána na P1 a vrácena zpět.

Důležitý je závěr rutiny **SERIAL**. Zde se příznaky **RI** a **TI** nulují. Pokud by nebyly vynulovány, aktivovalo by se přerušení znovu! Příznaky **RI** a **TI** se nenulují automaticky po aktivaci obslužné rutiny, protože je nutno rozlišit, proč aktivace nastala.

Po každé aktivaci **SERIAL** se stav LED mění na opačný. LED bliká v rytmu komunikace a indikuje tak činnost zařízení.

Výpis programu je v tab. 15.1.

Tab. 15.1. Výpis zdrojového programu pro mikrokontrolér AT89C2051-24PI v přípravku AT8VV

```
AT8VV.ASM:
                $MODxx51



PORT            EQU P1                  ;vstupně/výstupní port
LED             EQU P3.7                ;indikátor komunikace
BAUD            EQU 250                 ;přenosová rychlost
HPCON           EQU 10000000B           ;SMOD nastaven
CTENI           EQU 1                   ;režim čtění
ZAPIS           EQU 2                   ;režim zápisu

                DSEG AT 30H
REZIM:          DS 1                    ;proměnná indikující
                                        ;režim
                CSEG
                AJMP RESET
                ORG 0023H
                AJMP SERIAL

RESET:          MOV REZIM,#0            ;žádný režim
                MOV TH1,#BAUD           ;9600 Bd
                MOV TMOD,#00100000B     ;č/č 1 8bitový
                MOV SCON,#01010000B     ;8bitový přenos dat
                MOV PCON,#HPCON         ;SMOD=1
                SETB TR1                ;č/č 1 spuštěn
                SETB ES                 ;povolení přerušení
                SETB EA                 ;od sériového kanálu
                SJMP $                  ;vyčkává na přerušení

;obsluha sériového kanálu:
SERIAL:         PUSH ACC                ;uložení registrů
                PUSH B
                PUSH PSW
                JB TI,SERIAK            ;test vysílání/příjem
;příjem bajtu:
                MOV A,REZIM             ;do A režim
                MOV B,SBUF              ;přijatý bajt do B
                JZ SERIAR               ;test režimu
                CJNE A,#CTENI,SERIAZ
;jedná se o CTENI:
SERIAC:         MOV SBUF,PORT           ;vyšli hodnotu kanálem
                MOV REZIM,#0            ;nuluj režim
                CPL LED                 ;zneguj indikační LED
                SJMP SERIAK
;jedná se o ZAPIS:
SERIAZ:         MOV PORT,B              ;zapiš přijatou hodnotu
                MOV SBUF,B              ;pošli ji zpět
                MOV REZIM,#0            ;znuluj režim
                CPL LED                 ;zneguj indikační LED
                SJMP SERIAK
;jedná se o bajt indikující režim:
SERIAR:         MOV REZIM,B             ;ulož režim
                MOV SBUF,REZIM          ;pošli zpět
SERIAK:         CLR TI                  ;znuluj příznaky
                CLR RI
                POP PSW                 ;obnov registry
                POP B
                POP ACC
                RETI

                END
```

Program pro PC

Ovládací program se skládá ze dvou formulářů. První formulář (**HlForm**) obsahuje prvky pro nastavení a sledování stavu portu P1 a komunikaci s AT8VV. Druhý formulář (**Nastaveni**) umožňuje pohodlně nastavovat parametry komunikace.

Parametry komunikace jsou uloženy v inicializačním souboru **AT8VV.INI**. Jedná se o číslo použitého sériového portu a interval (v ms), ve kterém se periodicky čte stav portu P1. Tyto parametry lze nastavit editací souboru **AT8VV.INI**, avšak umožňuje to i dialog **Nastaveni**. Tím se zjednodušuje obsluha programu.

```
AT8VV.INI:
[PORT]          ;zvolen port
Port=1          ;COM1
[TIMER]
Interval=100  ;interval 100 ms
```

Hlavní formulář zajišťuje komunikaci a umožňuje zobrazit dialog pro nastavení parametrů komunikace.

```
HLFORM.CPP:
//---------------------------------------
#include <vcl.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "HlForm.h"
#include "NastForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TMainForm *MainForm;
//---------------------------------------
__fastcall TMainForm::TMainForm
        (TComponent* Owner)
        : TForm(Owner)
{
  //čtení konfigurace z AT8VV.INI:
  TIniFile *ini=new TIniFile
        (GetCurrentDir()
    +"\\AT8VV.INI");
  int Cislo=ini->ReadInteger
        ("PORT","Port",1);
  int Interval=ini->ReadInteger("TIMER",
    "Interval",100);
  if(Interval<=0)
    Interval=100;
  delete ini;
  //pokus o otevření portu:
  Port=NULL;
  try{
    Port=new TSerial(Cislo);
    Port->BaudRate=br9600;
```

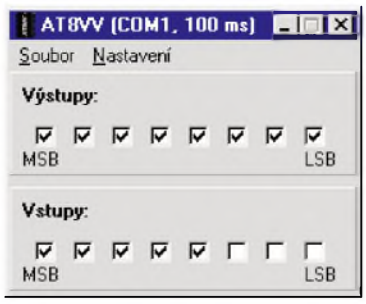
```
    Port->Parity=pNo;
    Port->StopBits=sbl0;
    Port->ByteSize=bs8;
    Port->SetupComm(16000,50);
    Port->ReadIntervalTimeout=0;
    Port->ReadTotalTimeoutMultiplier=0;
    Port->ReadTotalTimeoutConstant=1000;
    Port->WriteTotalTimeoutMultiplier=2;
    Port->WriteTotalTimeoutConstant=0;
    Port->RTS=1;
    Port->DTR=0;
    //sestavení titulku:
    Casovac->Interval=Interval;
    Caption=AnsiString("AT8VV (COM")
      +Cislo
      +AnsiString(", ")
      +Interval
      +(" ms)");
  }

  catch(...){
    //port se nepodařilo otevřít,
    //uživatel může zvolit jiný port
    if(!Nastav)
      Application->CreateForm
        (__classid(TNastav),
        &Nastav);
    Nastav->CisloPortu=1;
    Nastav->IntervalCasovace=Casovac->Interval;
    Nastav->Show();
  }
}
//---------------------------------------
__fastcall TMainForm::~TMainForm()
{
  if(Port)
    delete Port;
}
//---------------------------------------
void __fastcall TMainForm::OUTClick
        (TObject *Sender)
{
 //zápis dat:
 if(Port){
    Byte Zapis,Cteni;

    //pošle první bajt (indikace režimu):
    Zapis=2;
    Port->WriteByte(Zapis);
    Port->ReadByte(&Cteni);
    //test chyby komunikace:
    if(Zapis!=Cteni){
      MessageBox(Handle,
        "Komunikace selhala!",
        "Chyba",
        MB_ICONHAND);
      return;
    }

    //sestaví druhý bajt:
    Zapis=128*OUT7->Checked
      +64*OUT6->Checked
      +32*OUT5->Checked
      +16*OUT4->Checked
      +8*OUT3->Checked
      +4*OUT2->Checked
      +2*OUT1->Checked
      +OUT0->Checked;

    //pošle druhý bajt:
    Port->WriteByte(Zapis);
    Port->ReadByte(&Cteni);
    if(Zapis!=Cteni){
      MessageBox(Handle,
        "Komunikace selhala!",
        "Chyba",
        MB_ICONHAND);
      return;
    }
  }
}
//---------------------------------------
void __fastcall
        TMainForm::AkvivaceCasovace
        (TObject *Sender)
{
  //obsluha časovače, který čte vstup:
  if(Port){
    Byte Zapis,Cteni=0;

    //pošle první bajt (indikace režimu):
    Zapis=1;
    Port->WriteByte(Zapis);
    Port->ReadByte(&Cteni);
    //test chyby komunikace:
    if(Zapis!=Cteni){
      Casovac->Enabled=false;
      MessageBox(Handle,
        "Komunikace selhala!",
        "Chyba",
        MB_ICONHAND);
      //zobrazí dialog pro volbu portu:
      Nastav->CisloPortu=Port->Number;
      Nastav->IntervalCasovace=Casovac->Interval;
      Nastav->ShowModal();
      Application->Terminate();
      return;
    }
    //čtení druhého bajtu:
    Port->WriteByte(0);

    //zobrazení hodnot:
    Port->ReadByte(&Cteni);
    IN7->Checked=Cteni&0x80;
    IN6->Checked=Cteni&0x40;
    IN5->Checked=Cteni&0x20;
    IN4->Checked=Cteni&0x10;
    IN3->Checked=Cteni&0x08;
    IN2->Checked=Cteni&0x04;
    IN1->Checked=Cteni&0x02;
    IN0->Checked=Cteni&0x01;
  }
}
//---------------------------------------
void __fastcall
        TMainForm::NastaveniClick
        (TObject *Sender)
{
  //zobrazí dialog pro výběr portu:
  Nastav->CisloPortu=Port->Number;
  Nastav->IntervalCasovace=Casovac->Interval;
  Nastav->ShowModal();
}
//---------------------------------------
void __fastcall
        TMainForm::SouborKonecClick
        (TObject *Sender)
{
  Close();
}
```

Dialog **Nastaveni** slouží pro pohodlné zadávání parametrů komunikace bez nutnosti editovat inicializační soubor **AT8VV.INI**. Pokud nastavení potvrdíme tlačítkem OK, bude uloženo do inicializačního soubrou **AT8VV.INI**. Nastavení se však akceptuje až při novém spuštění programu.

Tento dialog se také zobrazí při startu programu, pokud není vybraný port k dispozici. Pokud dojde k chybě komunikace, zobrazí se rovněž tento dialog.

NASTFORM.CPP:

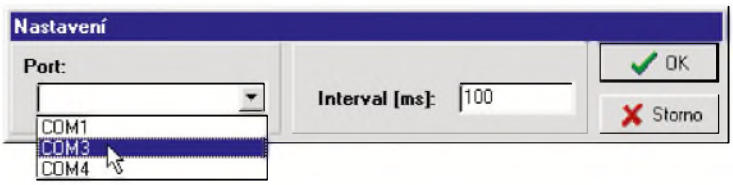
```
//---------------------------------------
#include <vcl.h>
#include <stdio.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "NastForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TNastav *Nastav;
//---------------------------------------
__fastcall TNastav::TNastav
        (TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------
void __fastcall TNastav::PortyDropDown
        (TObject *Sender)
{
  //reakce na rozbalení seznamu Porty:
  Porty->Items->Clear(); //smaže seznam
  HANDLE h;
  char jmeno[8];

  //naplní seznam dostupnými porty:
  for(int i=1;i<257;i++){
    sprintf(jmeno,"COM%i",i);
    h=CreateFile
    (jmeno,0,0,NULL,OPEN_EXISTING,0,NULL);
    if(h!=INVALID_HANDLE_VALUE||CisloPortu==i){
      Porty->Items->AddObject
        (jmeno,(TObject*)i);
      CloseHandle(h);
    }
  }
}
//---------------------------------------
void __fastcall TNastav::FormShow
        (TObject *Sender)
{
  //aktualizace seznamu Porty při aktivaci formuláře:
  Porty->Items->Clear(); //smaže seznam
  HANDLE h;
  char jmeno[8];

  //naplní seznam dostupnými porty:
  for(int i=1;i<257;i++){
    sprintf(jmeno,"COM%i",i);
    h=CreateFile(jmeno,0,0,NULL,OPEN_EXISTING,0,NULL);
    if(h!=INVALID_HANDLE_VALUE||CisloPortu==i){
      Porty->Items->AddObject(jmeno,(TObject*)i);
      CloseHandle(h);
      if(CisloPortu==i){
        //označí aktuální port:
        Porty->ItemIndex=Porty->Items->Count-1;
        Porty->Text=Porty->Items->
          Strings[Porty->Items->Count-1];
      }
    }
  }
  Interval->Text=IntervalCasovace;
}
//---------------------------------------
void __fastcall TNastav::OKClick(TObject *Sender)
{
  //reakce na stisk tlačítka OK:
  int i;
  try{
    //převod Interval s testem chyby:
    i=Interval->Text.ToInt();
    if(i<=0)
      throw Exception("");

    //uložení hodnot:
    TIniFile *ini=new TIniFile(GetCurrentDir()
      +"\\AT8VV.INI");
    ini->WriteInteger("PORT","Port",
      int(Porty->Items->Objects[Porty->ItemIndex]));
    ini->WriteInteger("TIMER","Interval",i);
    MessageBox(Handle,
      "Nové nastavení se akceptuje"
        " až při novém spuštění programu!",
      "AT8VV",
      MB_ICONINFORMATION);
    Application->Terminate();
  }
  catch(...){
    //ošetření chyby zadání intervalu:
    MessageBox(Handle,
      "Zvolte port a zadejte interval"
        " jako kladné celé číslo!",
```



Obr. 15.5. Ovládací program v akci

```
    "AT8VV",
    MB_ICONHAND);
  }
}
//---------------------------------------
void __fastcall TNastav::StornoClick
      (TObject *Sender)
{
  //reakce na stisk tlačítka Storno:
  //zavře aplikaci, když je toto hlavní
  //formulář:
  if(Application->MainForm==this)
    Application->Terminate();
}
```

Zobrazování dialogu **Nastaveni** při chybových stavech si vynutilo i drobnou úpravu projektového souboru.



Obr. 15.6. Výběr portu přes dialog Nastavení (port COM2 je obsazen jinou aplikací)

AT8VV.CPP:

```
//---------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("AT8VV.res");
USEFORM("Hlform.cpp", MainForm);
USEUNIT("Serial.cpp");
USEFORM("NastForm.cpp", Nastav);
//---------------------------------------
WINAPI WinMain
        (HINSTANCE, HINSTANCE, LPSTR, int)
{
  Nastav=NULL;
  try
  {
    Application->Initialize();
    Application->Title = "AT8VV";
    Application->CreateForm(
      __classid(TMainForm), &MainForm);
    //vytvoří formulář Nastav, jen když
    //neexistuje:
    if(!Nastav)
      Application->CreateForm(
        __classid(TNastav), &Nastav);
    Application->Run();
  }
  catch (Exception &exception)
  {
    Application->ShowException(&exception);
  }
  return 0;
}
```

Na obr. 15.5 a obr.15.6 jsou oba formuláře programu **AT8VV**. Stav výstupů se zadává v panelu **Výstupy**, stav vstupů lze sledovat v panelu **Vstupy**. Připomeňme, že výstupy mají vliv na chování vstupů (vývod se chová jako vstup, pokud je na něj zapsána „log. 1“).

Dialog **Nastavení** umožňuje volit port ze seznamu dostupných portů (nezobrazí se porty, které jsou obsazeny jinými aplikacemi) a čtecí interval. Po stisku tlačítka OK se změny uloží a program ohlásí, že tyto změny budou akceptovány až po novém spuštění programu.

16. Čítač do 16 MHz

Popis zapojení

Schéma čítače je na obr. 16.1. Jak je ze schématu vidět, je to relativně jednoduchý přístroj.

Jádrem zapojení je mikrokontrolér AT89C2051 (IO2), který je připojen k sériovému portu počítače PC přes konvertor MAX232 (IO1).

Krystal vytvářející hodinový (taktovací) signál je v zájmu lepší kmitočtové stability připojen do oscilátoru, který je vytvořen ze dvou invertorů 74HCT04 (IO3).

Nulovací obvod mikrokontroléru je tvořen součástkami R2, C8 a D1. Tímto obvodem se mikrokontrolér vynuluje při každém zapnutí napájecího napětí.

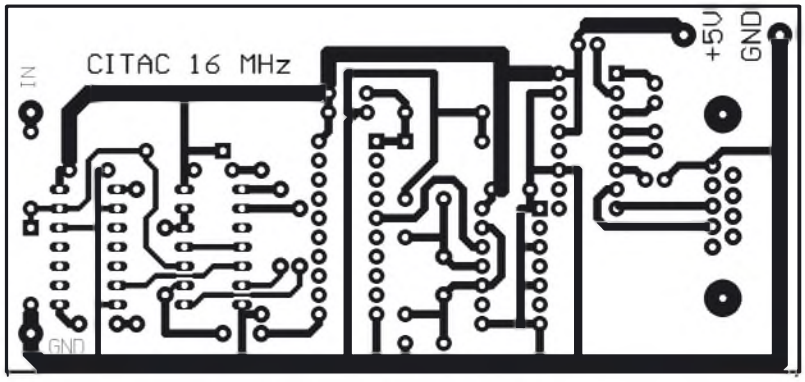
Signál, jehož kmitočet měříme, se přivádí na vstup T0 (P3.4) mikrokontroléru buď přímo, nebo přes děličku s IO5, která měřený kmitočet dělí 256x. Dělička je tvořena dvojnásobným čtyřbitovým čítačem 74HCT393 (IO5). Použití děličky má smysl při měření vyšších kmitočtů.

Cesta měřeného signálu (přímá nebo přes děličku) je určována multiplexerem z hradel 74HCT00 (IO3). Multiplexer je ovládán portem P3.7 mikrokontroléru. Je-li P3.7 = 0, měří se kmitočet přímo, při P3.7 = 1 se kmitočet měří až po průchodu děličkou 1/256.

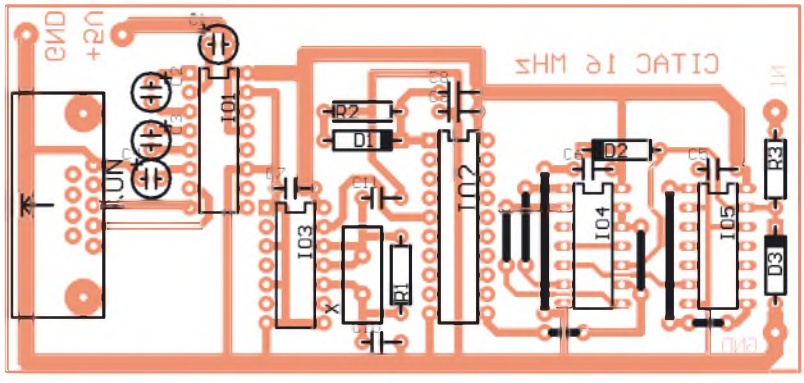
Vstup čítače je chráněn proti příliš velkému nebo zápornému vstupnímu napětí diodovým omezovačem se součástkami R3, D2 a D3.

Zvláštností čítače je skutečnost, že se napájí přímo napětím 5 V ze stabilizovaného zdroje. Původně jsem chtěl použít obvyklý napájecí obvod se stabilizátorem 7805 (jako v předchozích konstrukcích), ale deska s plošnými spoji mi připadala příliš rozměrná. Z toho důvodu jsem tedy stabilizátor, filtrační kondenzátor a ochrannou diodu vypustil!

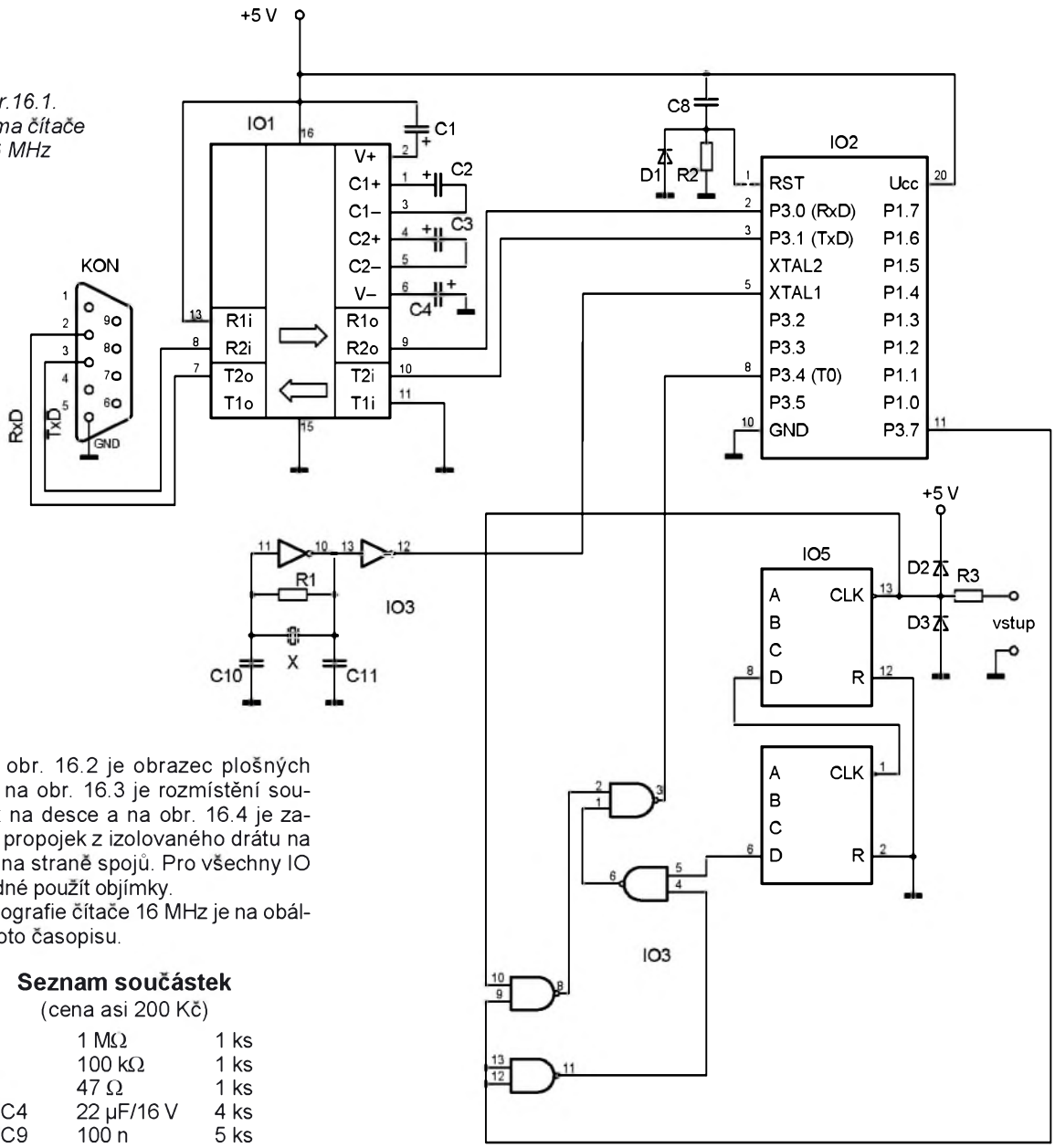
Součástky čítače 16 MHz jsou umístěné na desce s jednostrannými plošnými spoji.



Obr. 16.2. Obrazec plošných spojů čítače 16 MHz (měř.: 1 : 1, delší rozměr desky je 105 mm)



Obr. 16.3. Rozmístění součástek na desce čítače 16 MHz



Obr.16.1. Schéma čítače 16 MHz

Na obr. 16.2 je obrazec plošných spojů, na obr. 16.3 je rozmístění součástek na desce a na obr. 16.4 je zapojení propojek z izolovaného drátu na desce na straně spojů. Pro všechny IO je vhodné použít objímky.

Fotografie čítače 16 MHz je na obálce tohoto časopisu.

Seznam součástek

(cena asi 200 Kč)

R1	1 MΩ	1 ks
R2	100 kΩ	1 ks
R3	47 Ω	1 ks
C1 až C4	22 µF/16 V	4 ks
C5 až C9	100 n	5 ks
C10, C11	33 pF	2 ks
D1	1N4148	1 ks
D2, D3	1N4001	2 ks
IO1	MAX232	1 ks
IO2	AT89C2051-24PI (naprogramovaný)	1 ks
IO3	74HCT04 (74HC04)	1 ks
IO4	74HCT00 (74LS00)	1 ks
IO5	74HCT393 (74LS93)	1 ks
X	krystal 24 MHz	1 ks
KON	CAN 9 Z 90	1 ks

deska s plošnými spoji CITAC 16 MHz

CITAC.ASM - program pro mikrokontrolér AT89C2051

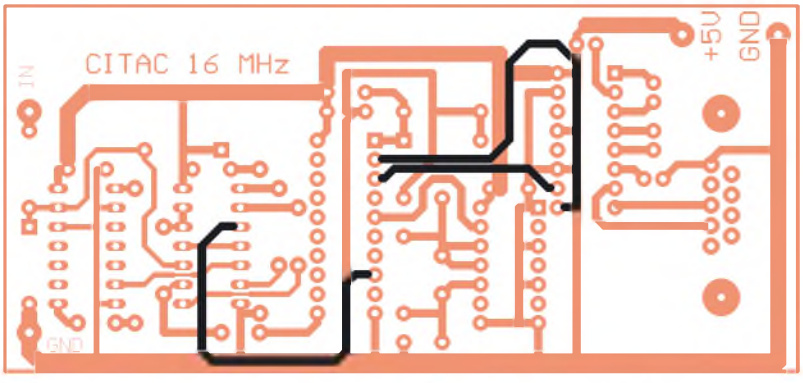
Program je zapsán podle obvyklých konvencí.

Komunikace mezi PC a mikrokontrolérem je řešena tak, že ovládací program v PC vyšle bajt definující způsob měření (použití předděličky, doba měření). Pak se provede odměr kmitočtu a do PC se odešle nazpět řídicí bajt (pro kontrolu komunikace; nejvyšší bit indikuje případné přetečení čítače v průběhu měření), za kterým následují dva datové bajty, nesoucí informaci o změřeném kmitočtu (šestnáctibitové číslo).

Řídicí program umožňuje měřit kmitočet po dobu 0,1 s, 1 s a 10 s. Kratší doby měření jsou vhodné pro vyšší kmitočty, delší pro nižší kmitočty.

Např. pro vstupní kmitočet 100 kHz (bez použití předděličky) vede použití doby měření 1 s k přetečení čítače (za 1 s se načítá 100 000 impulsů, délka čítače je však pouze 65 535). Při době měření 0,1 s dostaneme 10 000 impulzů (údaj se pak programem vynásobí 10x). Při měření kmitočtu 100 Hz po dobu 1 s dostaneme 100 impulsů, měření po dobu 10 s pak bude 10x přesnější (údaj se programově vydělí 10x).

Výpis zdrojového programu pro mikrokontrolér AT89C2051-24PI je v tab. 16.1.



Obr. 16.4. Drátové propojky na straně spojů na desce čítače 16 MHz

Tab. 16.1. Výpis programu pro mikrokontrolér AT89C2051-24PI v čítači 16 MHz

```
CITAC.ASM:
                $MODxx51
BAUD            EQU 243                 ;9600 Bd (SMOD=1)
DELIC           EQU P3.7                ;DELIC=0 (dělí 256)
                                        ;DELIC=1 (přímo)
                DSEG AT 20H
BUFFER:         DS 3                    ;3bajtový buffer
                                        ;pro sériový kanál
BUFPOS:         DS 1                    ;pozice v bufferu

                CSEG AT 0
                AJMP RESET
                ORG 0023H
                AJMP SERIAL

;inicializace:
RESET:          MOV TH1,#BAUD           ;9600 Bd
                MOV TMOD,#00100101B     ;8bitový časovač 1
                                        ;16bitový čítač 0
                SETB TR1                ;spuštěn časovač 1
                MOV SCON,#01010000B     ;8bitový přenos
                MOV PCON,#10000000B     ;SMOD=1
                SETB ES                 ;konfigurace
                SETB EA                 ;systému
                SJMP $

;obsluha sériového kanálu:
SERIAL:         PUSH ACC                ;uložení registrů
                PUSH B
                PUSH PSW
                PUSH DPL
                PUSH DPH
                CLR TR0                 ;zastavení č/č 0
                JB TI,SERIAV            ;test vysílání
                CLR RI                  ;nulování příznaku příjmu
                MOV BUFFER,SBUF         ;příjem bajtu
                MOV BUFPOS,#0           ;příprava na vysílání
                MOV C,BUFFER.2          ;nastavení předděličky
                MOV DELIC,C             ;podle přijatého bajtu
                MOV DPTR,#CASTAB        ;DPTR=adresa
                MOV A,BUFFER            ;hodnota
                ANL A,#00000011B        ;pouze dolní 2 bity
                ADD A,ACC               ;A=A*2
                MOV B,A                 ;kopie do B
                MOVC A,@A+DPTR          ;čti horní
                                        ;bajt doby měření
                XCH A,B                 ;B=horní bajt
                INC A                   ;na další adresu
                MOVC A,@A+DPTR          ;čti dolní
                                        ;bajt doby měření
                MOV DPL,A               ;nastavení doby
                MOV DPH,B               ;měření do DPTR
                CLR TF0                 ;vynulování TF0
                MOV TH0,#0              ;nulování č/č 2
                MOV TL0,#0

TMER:           CLR BUFFER.7            ;nulování
                                        ;příznaku přetečení
                SETB TR0                ;spustí č/č 0 jako
                                        ;volně běžící čítač
TMERS:          MOV A,#196
                DJNZ ACC,$
                NOP                     ;197 (s
                INC DPTR                ;zvyš DPTR o 1
                MOV A,DPH
                ORL A,DPL
                JNZ TMERS               ;A=0 značí přetečení
                CLR TR0                 ;zastav čítač 0
                MOV BUFFER+1,TL0        ;ulož hodnoty
                MOV BUFFER+2,TH0        ;do bufferu
                MOV C,TF0
                MOV BUFFER.7,C
                MOV SBUF,BUFFER         ;vyšli první bajt bufferu
                AJMP SERIAK             ;a konec
SERIAV:         CLR TI                  ;nuluj příznak příjmu
                MOV A,BUFPOS            ;A=rel. pozice v bufferu
                JB ACC.1,SERIAK         ;test konce
                INC A
                XCH A,R0                ;prohoď A, R0
                MOV SBUF,@R0            ;pošli bajt
                                        ;bufferu podle R0
                XCH A,R0                ;obnov R0
                INC BUFPOS              ;zvyš ofset v bufferu
SERIAK:         POP DPH
                POP DPL
                POP PSW
                POP ACC                 ;obnovení regsitrů
                RETI

;doby měření:
CASTAB:         DW 65535-500+1          ;0,1 s
                DW 65535-5000+1         ;1 s
                DW 65535-50000+1        ;10 s
                DW 65535-5000+1         ;1 s
                END
```

CITAC.EXE - program pro PC

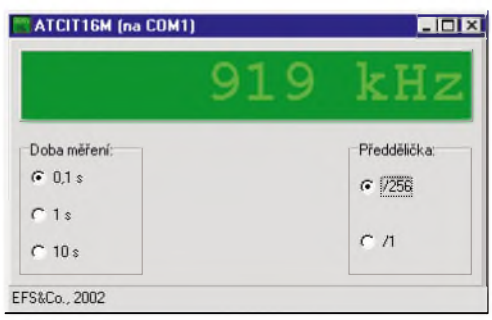
Ovládací program pouze sestaví informaci definující podmínky měření (použití předděličky a doba měření) a odešle čítači. Čítač provede odměr a tuto informaci vrací zpět (se změnou, že nejvyšší bit bude nastaven, když při měření údaj přetekl). Dále se přijmou dva bajty odpovídající změřenému kmitočtu.

Před výpisem změřeného kmitočtu se musí zohlednit nastavení předděličky (je-li aktivována, násobí se přijatý údaj kmitočtu číslem 256) a doba měření (pro dobu 0,1 s se údaj kmitočtu vynásobí číslem 10, pro dobu 10 s se údaj vydělí číslem 10).

Pohled na okno čítače na monitoru PC je na obr. 16.5.

Dále je uveden pouze výpis obsluhy časovače, který spuští odměry:

```
void __fastcall TTestForm::TimerTimer
      (TObject *Sender)
{
  Timer->Enabled=false;
  int DobaMereni=pow10(rgGate->ItemIndex+2);

  //sestavení konfiguračního bajtu:
  Byte PrvniBajt=rgGate->ItemIndex
    |(rgDelicka->ItemIndex<<2);
  Byte Precteno=~PrvniBajt;
  double Hodnota;
  int i;
  //vyprázdní vstupní buffer sériového kanálu:
  Serial->PurgeInput();

  //spusť měření:
  Serial->WriteByte(PrvniBajt);
  ProgressBar->Position=0;
  for(i=0;i<10;i++){
    Sleep(DobaMereni/10);
    ProgressBar->Position=ProgressBar->Position+1;
  }

  //test úspěšnosti komunikace:
  Serial->ReadByte(&Precteno);
  if((PrvniBajt&0x07)!=(Precteno&0x07)){
    MessageBox(Handle,
      "Chyba komunikace",
      Application->Title.c_str(),
      MB_ICONHAND);
    return;
  }

  //test úspěšnosti komunikace:
  if(!(Precteno&0x80)){
    TCitData Data;
    if(!Serial->ReadByte(&Data.Bajt.Nizsi)||
      !Serial->ReadByte(&Data.Bajt.Vyssi)){
      MessageBox(Handle,
        "Chyba komunikace",
        Application->Title.c_str(),
        MB_ICONHAND);
      return;
    }
```



Obr. 16.5. Aplikace v akci

```
  //příprava dat na zobrazení:
  Hodnota=(rgDelicka->ItemIndex==0)
    ?Data.Celek*256:Data.Celek;
  Hodnota=Hodnota/pow10(rgGate->ItemIndex-1);

  AnsiString Jednotky="Hz";
  if(Hodnota>1e6){
    Hodnota/=1e6;
    Jednotky="MHz";
  }
  else if(Hodnota>1000){
    Hodnota/=1000;
    Jednotky="kHz";
  }

  AnsiString Maska;
  if(Jednotky!="Hz"){
    if(Hodnota>100)
      Maska="##0.00";
    else if(Hodnota>10)
      Maska="#0.000";
    else
      Maska="0.0000";
    switch(rgGate->ItemIndex){
      case 0:Maska.Delete(6,1);break;
      case 2:Maska.Insert("0",6);break;
    }
    if(rgDelicka->ItemIndex==0)
      Maska.Delete(Maska.Length()-2,2);
  }
  else{
    switch(rgGate->ItemIndex){
      case 0:
      case 1:Maska="##0";break;
      case 2:Maska=("##0.0");break;
    }
  }
  AnsiString pom=FormatFloat(Maska,Hodnota);
  if(Jednotky=="Hz")
    pom=pom+" ";
  pom=pom+" "+Jednotky;
  for(i=pom.Length();i<12;i++)
    pom.Insert(" ",1);
  Kmitocet->Caption=pom;
 }
 else  //údaj přetekl
  Kmitocet->Caption="------";
 Timer->Enabled=true;
}
```

Volby provedené v aplikaci se ukládají do inicializačního souboru **CITAC.INI**. Je v něm zapsána i volba sériového kanálu.

```
CITAC.INI:
[NASTAVENI]
Port=1        ;číslo portu
Gate=0        ;doba měření
              ;(0 odpovídá 0,1 s)
Divider=0     ;použití předděličky
              ;(0 značí nepoužita)
```

17. Závěr

Na závěr uvedu několik důležitých informací.

Programy pro ovládání zařízení popsaných v tomto časopise a další soubory lze stáhnout ze stránky: *http://www.mujweb.cz/www/efs-prodej/PE2003/PROGRAMY.html*

Informace jsou přehledně rozděleny do skupin po jednotlivých kapitolách. Pro snažší stažení jsou soubory komprimovány programem WinZip 8.0.

V současnosti nevyrábím desky plošných spojů na zakázku, lze si však objednat již hotové přístroje. V případě většího zájmu budou do „výrobního programu" zařazeny další konstrukce.

Seznam dostupných přístrojů včetně jejich cen je uveden na stránce (viz obr. 17.1): *http://www.mujweb.cz/www/efs-prodej/PE2003/PE2003.html*

Objednávky přístrojů a případné další dotazy zasílejte na e-mailovou adresu autora: *matousekd@quick.cz*

Písemný kontakt na autora je na adrese: Ing. David Matoušek
Vyšší odborná škola
Tolstého 16
586 01 JIHLAVA

Obr. 17.1. Stránka: http://www.mujweb.cz/www/efs-prodej/PE2003/PE2003.html

18. Literatura

[1] *Matoušek, D.*: C++ Builder - 1. díl. BEN - technická literatura, Praha 2002 (3. vydání).

[2] *Matoušek, D.*: C++ Builder - 2. díl. BEN - technická literatura, Praha 2001.

[3] *Matoušek, D.*: C++ Builder - 3. díl. BEN - technická literatura, Praha 2003.

[4] *Matoušek, D.*: Práce s mikrokontroléry AT89C2051. BEN - technická literatura, Praha 2002.

[5] *Matoušek, D.*: Udělejte si z PC - 2. díl. BEN - technická literatura, Praha 2002.

[6] *Matoušek, D.*: Udělejte si z PC - 1. díl. BEN - technická literatura, Praha 2001.

[7] *Kainka, B.*: Využití rozhraní PC. HEL, Ostrava 1998.

[8] *Matoušek, D.*: Práce s mikrokontroléry AT89S8252. BEN - technická literatura, Praha 2002.

[9] *Matoušek, D.*: Práce s mikrokontroléry AVR. BEN - technická literatura, Praha 2003.

Z dějin vědy a techniky

(dokončení ze str. 2)

Shockley zemřel 12. 8. 1989 ve věku 79 let.

Ne nadarmo se druhé polovině dvacátého století říká „polovodičový věk". Vývoj v této oblasti jde nepřetržitě kupředu rychlostí, která nemá obdoby v jiných odvětvích. Stejně tak nemá obdobu ani pokles cen finálních výrobků. Na konci 50. let stály první hrotové tranzistory firmy RCA kolem 50 dolarů. Když mi jej poslala teta z USA, byl u něj lístek, že stejný kousek zlata by byl lacinější - jenže z toho bychom asi stěží postavili možná první tranzistorový vysílač u nás v Poděbradech v radioklubu OK1KKJ.

Dnešní moderní mikroprocesor Pentium III (psáno v roce 2001) obsahuje asi 9 milionů tranzistorů - a cena celého mikroprocesoru se pohybuje přibližně ve stejné oblasti, jako u prvních tranzistorů.

Literatura

[1] *Belkind, L. D.:* Tomas Alva Edison. Znanie, Moskva 1957.

[2] *Vassjor, Ž. P.*: Schemy na poluprovodnikovych priborach. Sovětskoje radio, Moskva 1956.

[3] *Shockley, W.*: Electrons and Holes in Semiconductors. D. Van Nostrand Company, New York - Toronto - London 1959.

[4] *Seger, J.*: Jak se lidé dorozumívali. Albatros, Praha 1987.

[5] *Josephson, M.*: Edison. McGraw -Hill Book Company Inc., New York - Toronto - London 1959.

[6] Antique Radio č. 33, Mose Edizioni, Maser 1999.

[7] *Mayer, D.*: Pohledy do minulosti elektrotechniky. KOPP, České Budějovice 1999.

QX